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,

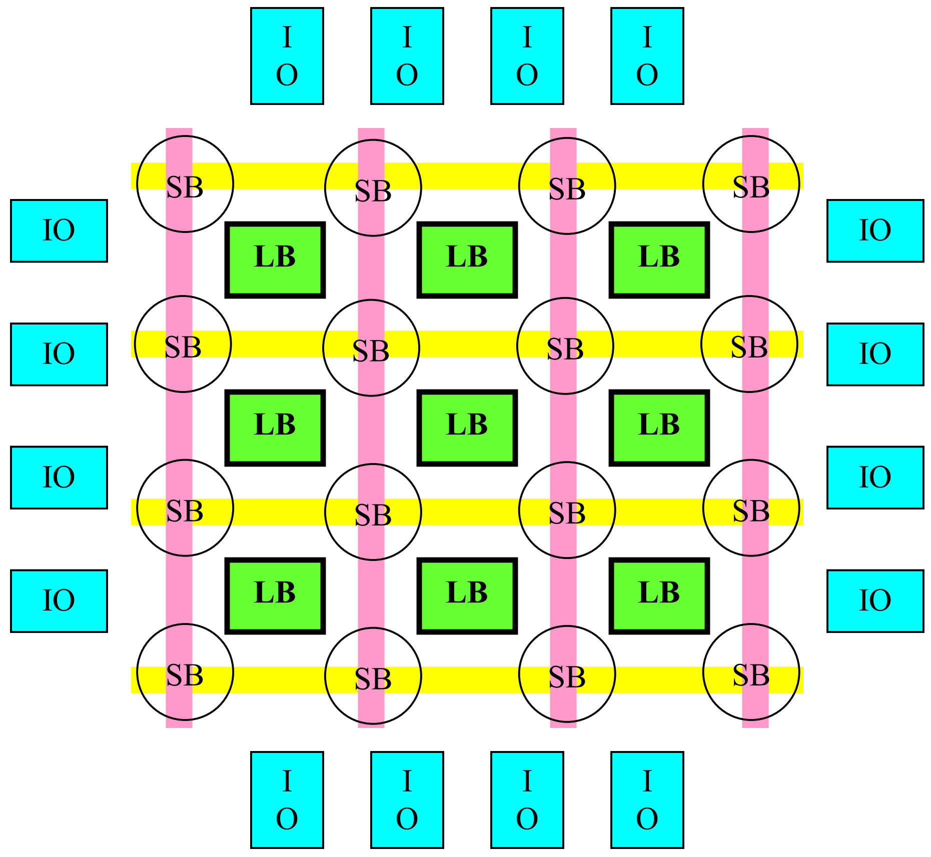


Figure 1. FPGA Structure

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 work involved 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. A study conducted by the makers of VPR showed that for homogeneous FPGAs it is optimal for all channels to be the same size [4], however their study did not involve any hard macros. We plan to conduct our investigation 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 modified VPR to handle these architectures.

Another related study was conducted in 2001 by P. Hallschmid and S. Wilton [5]. 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.

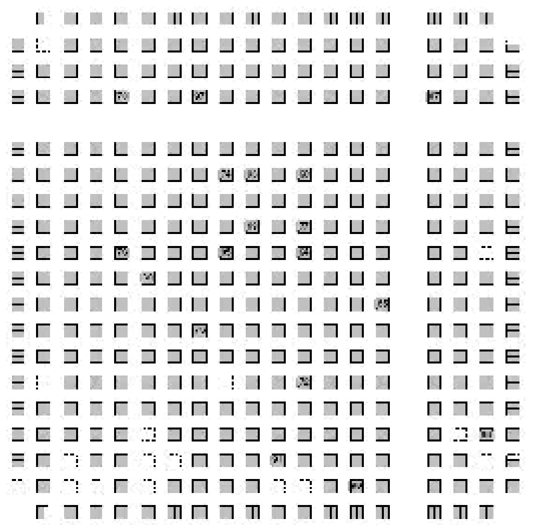


Figure 2. VPR Display of an FPGA with Two Very Wide Channels

2) Next I integrated some code written by a former student working with my mentor into my modified version of VPR. The MCNC Benchmark circuits we are placing and routing in our study do not contain any hard macros – when we place and route a circuit, we want to select a group of logic blocks from that design to model a hard macro in our experiment. The former student's code chooses logic blocks to represent the hard macro, places them together in a square at the FPGA's center, and colors these blocks and wires that connect to them pink in VPR's graphical user interface (GUI) display (see Figure 3). His code ensures that

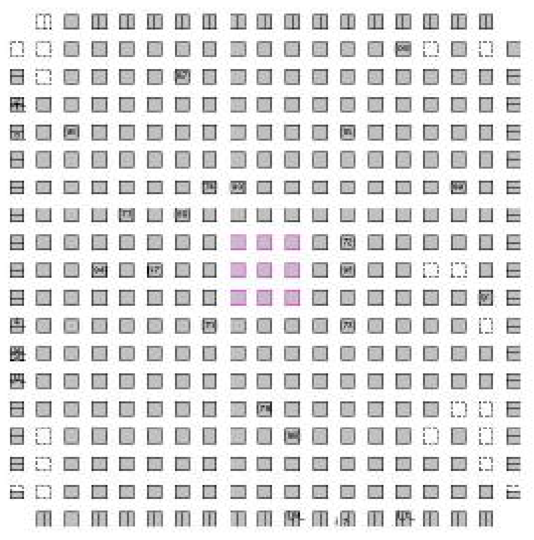


Figure 3. VPR Display of an FPGA with A Hard Macro at its Center

the blocks are chosen at random (i.e. not the blocks that happen to be adjacent after placement) to ensure high connectivity to logic blocks that are not also in the hard macro. I integrated this portion of his code into my copy of VPR rather than initially adding my modifications to his version of the program because other sections of his code are incomplete and do not work properly.

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.

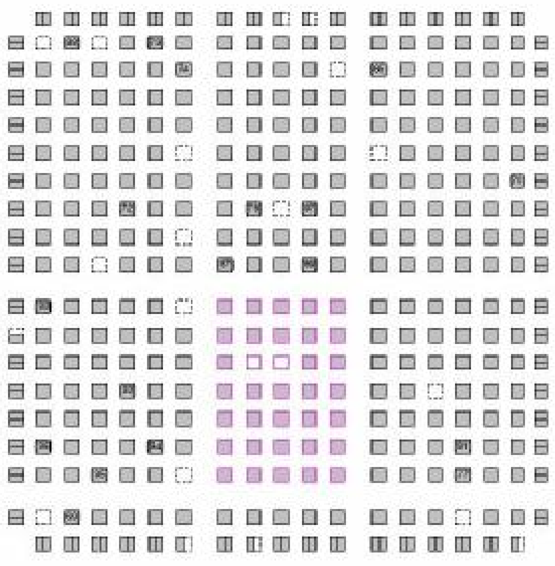
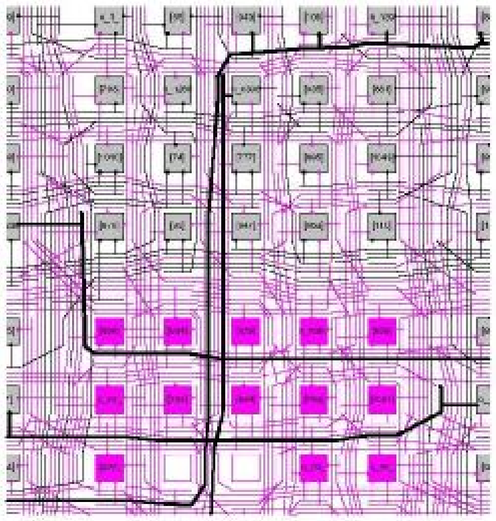


Figure 4. VPR Display of an FPGA with a Hard Macro Surrounded by Wide Channels

4) The next stage of my work – which was planned but not completed by my mentor's former student – was to change VPR's routing behavior so that only pink wires, wires that connect to a block inside the hard macro are allowed to route through the channels inside the fixed macro. This makes our model hard macro similar to a real one, which would allow signals connecting to the hard macro to enter it though its pins, but occupy a space containing no channels for other signals to route through. Figure 5 shows a VPR routing display before and after this modification. Wires that are not colored pink are drawn in black.

5) At this point in my work with VPR, there was an unexpected challenge. VPR had been modified enough to run a preliminary experiment on ten of the twenty Benchmark circuits, and in attempting to run these trials I discovered that VPR consistently failed to route the five largest Benchmarks I was working with.

Analysis of the problem showed that very large hard macros interfered with VPR's routing algorithm. Figure 6 shows an FPGA with yellow squares representing logic blocks. Consider VPR attempting to route a connection between the two pink squares. The rectangle formed


a.

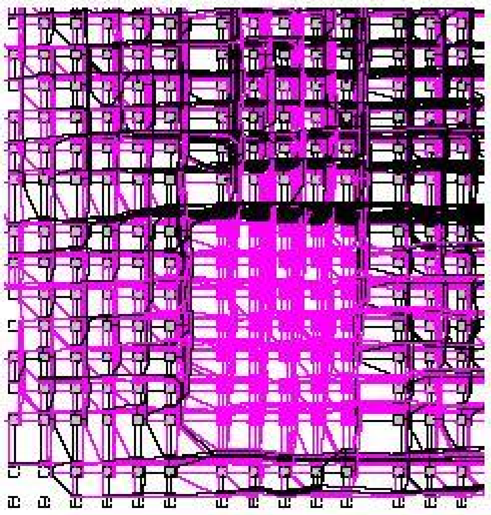

b.

Figure 5. Display of VPR's Routing With Black Wires (thickened for clarity) in the Pink Region (a), and With No Black Wires in the Pink Region (b)

with these squares as its corners, drawn in pink, is called the bounding-box of these two logic blocks. To reduce the time it takes for VPR to make each route, it will not consider a route between two points that goes more than three channels beyond the points' bounding-box. The large blue box in Figure 6 shows the entire region VPR will consider using to route a connection between the pink blocks. If no connection can be made within this area, even if a connection is possible that goes outside the blue box, VPR will consider the desired route impossible.

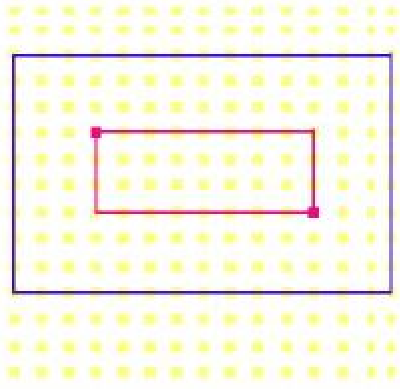


Figure 6. Bounding-Box

Figure 7 shows the problem caused by a very large hard macro. If the hard macro cuts the blue box in two, with one end of a net on either side of the hard macro, VPR will not be able to form the connection needed between them without leaving the blue box, and the route will fail.

VPR has a command line option that allows a user to determine how far outside the bounding-box it should look in building routes. The number three is a default, not built in to the algorithm itself. By setting this number to three plus the size of the hard macro, I was able to get the Benchmarks to route that would not route previously.

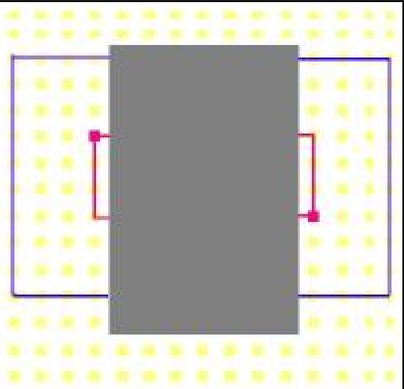


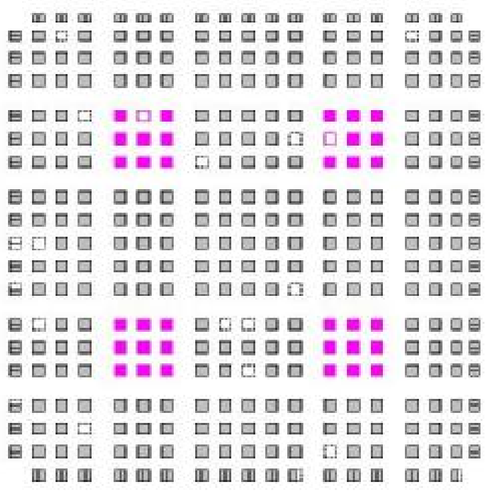
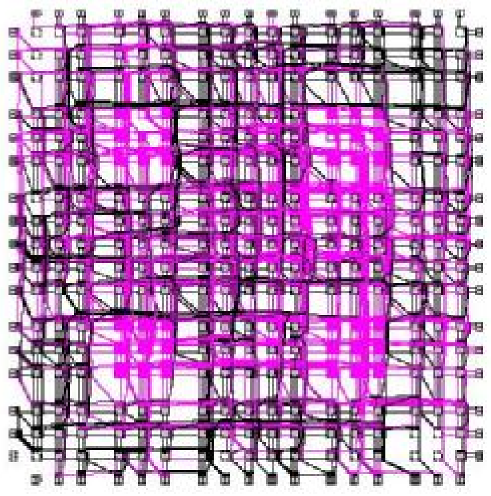
Figure 7. Bounding-Box With Large Hard Macro

6) While working on the bounding-box problem, I also improved the mechanism ensuring that only hard-macro-related (pink) wires were allowed to route through the pink region. The original approach had been created by a former student, although it had not been fully implemented before I worked with VPR, and in working on the bounding-box issue, I found a flaw in this student's approach.

When VPR attempts to create a route, it builds it one net at a time. For each net, VPR creates a heap data structure containing one node for each channel that might be used in routing the net, each with a cost assigned, reflecting how full that channel was in previous routing attempts. VPR then routes the net, creating a path through FPGA channels that joins all of its endpoints, by choosing the lowest cost channels it can find in the heap.

During this process, VPR keeps track of only the cost of channels, not whether they have been filled to capacity, so it is common for VPR to create routes in which some channels are overfilled. Thus, after VPR has routed all nets, it checks to see if any channels are overfilled, and if so discards the route, starting it over with the costs of the overfilled channels increased. Because, if channels are too small, there might be no way to create a route that does not overfill channels, VPR will only make thirty routing attempts at a given channel size before declaring this size too small and giving up.

The former student's approach to keeping non-hard-macro (black) wires from being routed through hard macro (pink) channels was to make the router think that all of the hard macro channels were full whenever a non-hard-macro net was being routed. Since VPR will route nets through full channels, only checking for this problem after the route is completed, this was allowing some of the thirty allowed routing attempts to be wasted on routes that put non-

a. **b.**

Figure 8. VPR Placement (a) and Routing (b) of an FPGA with Four Hard Macros

hard-macro wires in hard macro channels. In some cases, all thirty attempts were being wasted in this fashion causing VPR to declare routing impossible. I modified the code so that when VPR is routing an individual net, if it is a non-hard-macro net, no hard macro channels will ever be put on the heap. This way, no illegal arrangements waste router iterations.

7) The final modification I made to VPR was to enable it to handle more than one hard macro. I had originally intended to create a format for a new input file a user could give VPR that would describe the location and size of each hard macro and the size of each channel surrounding the hard macros. While this is a very general input format, it would not have been very practical for the work my mentor wants to do – which is placing and routing a

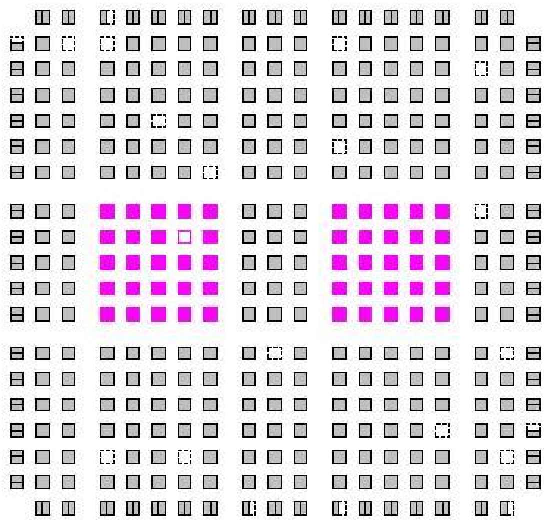


Figure 9. VPR Display of an FPGA with Two Hard Macros

large number of Benchmark circuits of very different sizes with groups of identically sized hard macros in symmetric arrangements. For this purpose, it seemed more useful for a user to input what the arrangement would be with some kind of keyword, and have routines that would figure out the exact coordinates of each hard macro based on the size of the FPGA being used. I wrote routines to arrange two or four hard macros symmetrically around the FPGA's center, and it should be easy to add more routines as needed in the future. Figure 8 shows a placement and routing for an FPGA with four hard macros, and Figure 9 shows a placement for an FPGA with two hard macros.

8) Figure 10, below, traces all of the modifications I have made to VPR:

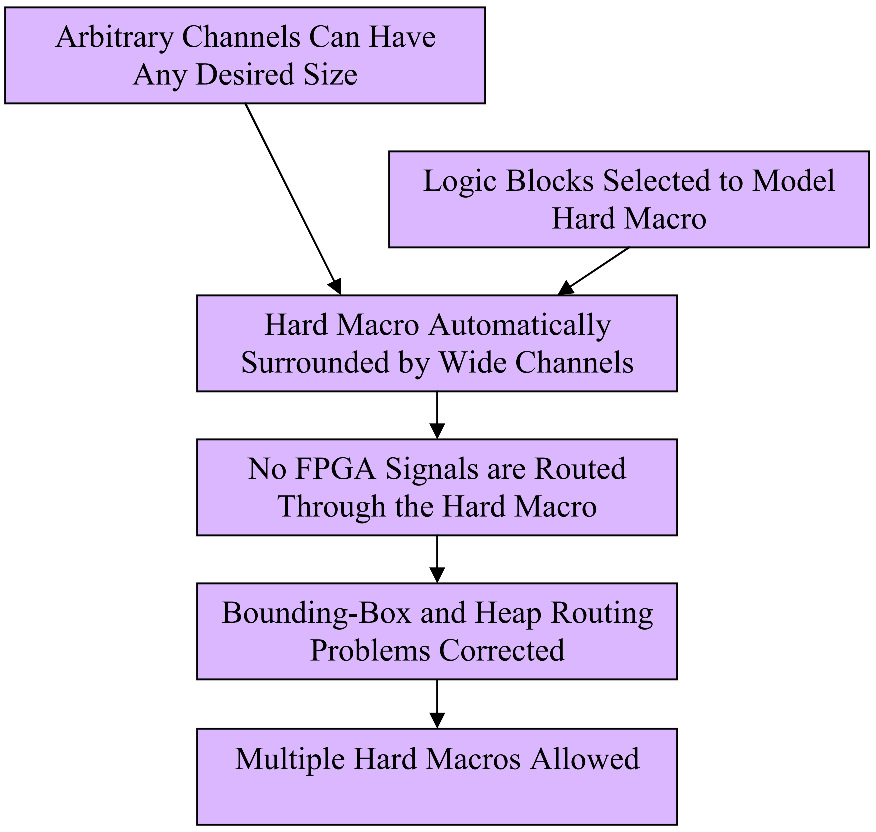
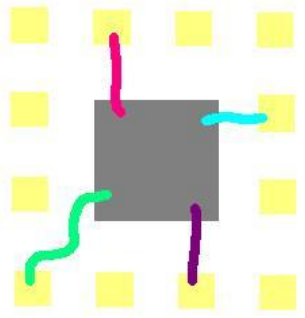

Arbitrary Channels Can Have Any Desired Size
Logic Blocks Selected to Model Hard Macro
Hard Macro Automatically Surrounded by Wide Channels
No FPGA Signals are Routed Through the Hard Macro
Bounding-Box and Heap Routing Problems Corrected
Multiple Hard Macros Allowed

Figure 10. VPR Modifications

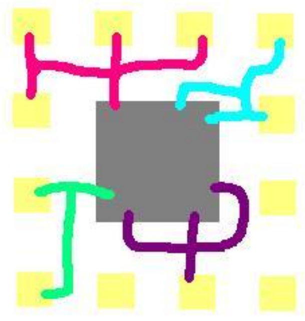
Experimental Results:

When the modifications to VPR were completed, I ran a preliminary experiment on ten of the twenty MCNC Benchmark circuits: Alu4, Apex4, Diffeq, Ex5p, Spla, Elliptic, Frisc, S 38417, S 38584, and Clma. In these experiments, a single hard macro, occupying 4% of the FPGA's area was placed in the center of the FPGA. Each trial was run once with the channels around the hard macro the same size as the FPGA's other channels, and once with these channels twice as large. VPR then determined the smallest number of routing tracks needed to route the circuit in both configurations.

A random number generator is involved in determining which blocks will be a part of the hard macro. Because the particular blocks that are chosen will affect placement and routing, for each Benchmark, I ran three different trials, each one giving a different seed to the random number generator so that different hard macros would be created. Thus, for each of the ten Benchmark circuits used, six data points were generated, one for each seed with normal sized channels, and one for each seed with double sized channels.



a. Eight Pins



b. Sixteen Pins

Figure 11. Two Different Hard Macros With Four Nets

One of the most significant differences between hard macros made from different logic blocks is that they will have different numbers of connections to blocks outside the hard macro, potentially affecting how congested routing tracks around the hard macro are. Figure 11 shows two different methods of computing the connectivity of a hard macro. Both hard macros in Figure 11 connect to four nets, four pieces of wiring connecting them to the rest of the FPGA. In Figure 11 a, however each net has only two pins, one inside the hard macro and one outside, while in 11 b each net has many pins. When describing the connectivity of a hard macro, we list how many pins it has rather than how many nets it has so that circuits like the one in 11 b will be distinguished from the one in 11 a. The term "special pins" in the experimental results refers to the total number of pins in all of the nets connecting to the FPGA's "special regions," the groups of logic blocks representing hard macros.

1) Preliminary Experiment:

Figures 12 a - j, below, show the results of the preliminary experiment on each of the ten Benchmark circuits that were used. In each graph, the smallest number of routing tracks that was needed for the circuit to route successfully is plotted versus the number of special pins in the circuit's hard macro. For each circuit there are data-points for three different numbers of special pins, reflecting the three different hard macros that were created with three different seeds. For each of these three hard macros, there is a data-point in blue representing the result with uniformly sized channels and a data-point in pink showing the result with doubly wide channels around the hard macro.

A low data-point represents a successful route – one that required few routing tracks. Thus, whenever a pink point is below its corresponding blue point, the double-wide channels made routing more successful, but whenever the pink point is above its corresponding blue point, the double-wide channels made routing less successful.

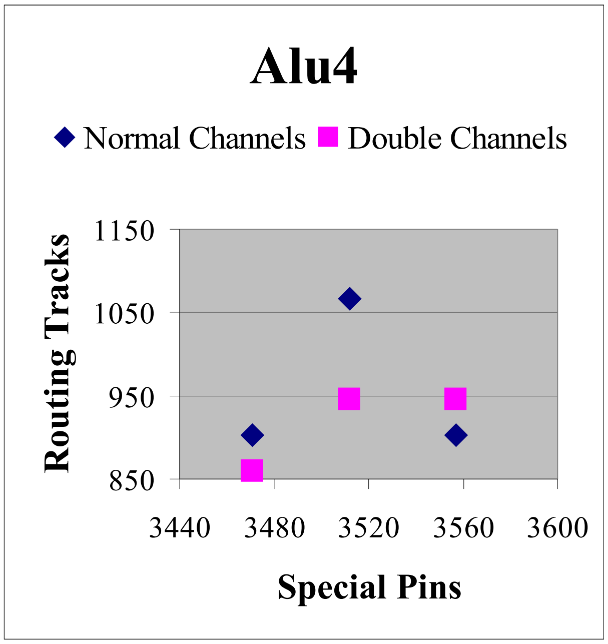


Figure 12 a

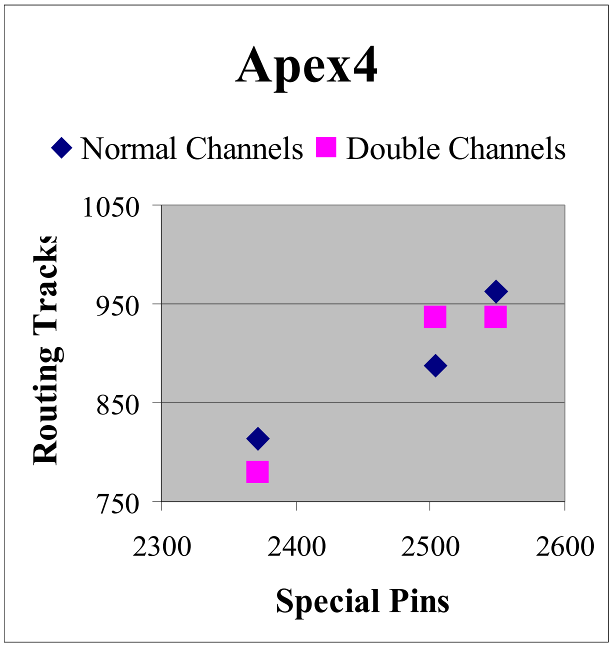


Figure 12 b

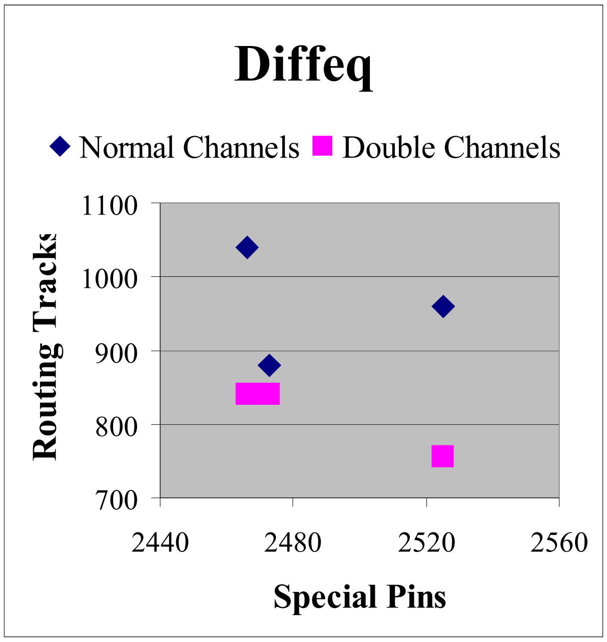


Figure 12 c

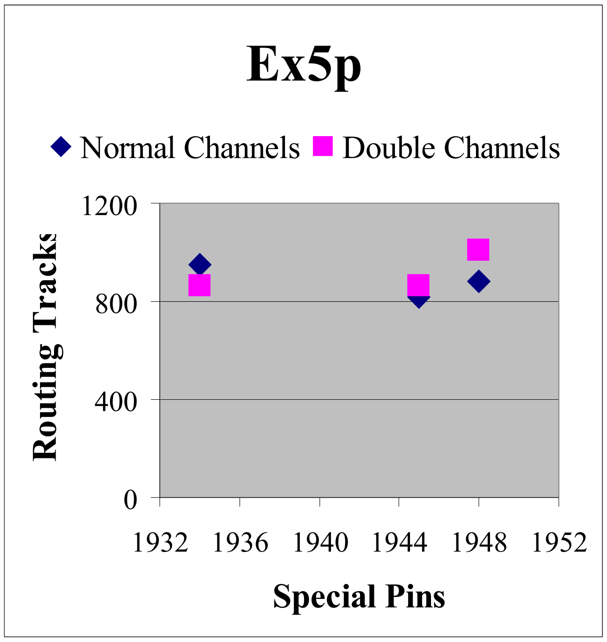


Figure 12 d

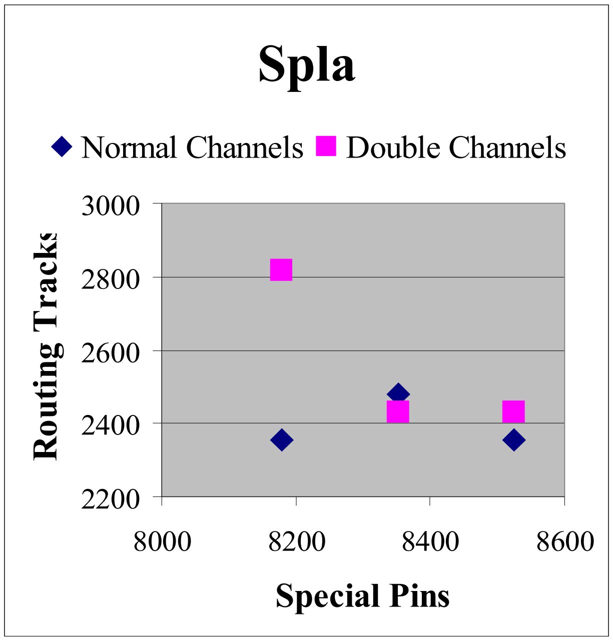


Figure 12 e

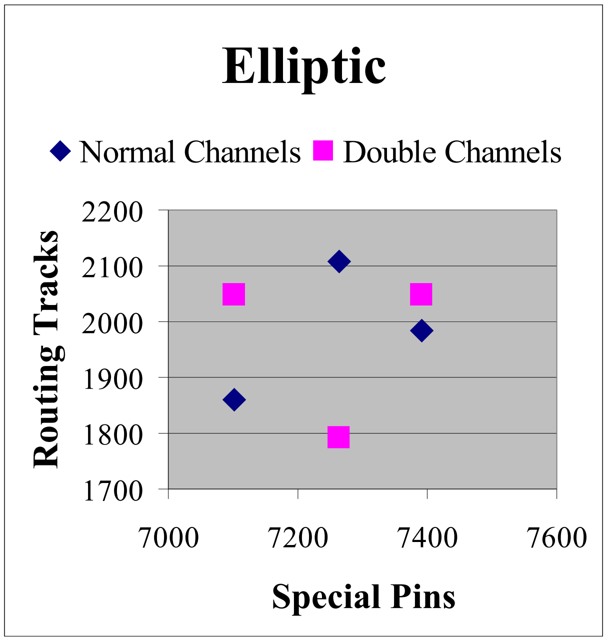


Figure 12 f

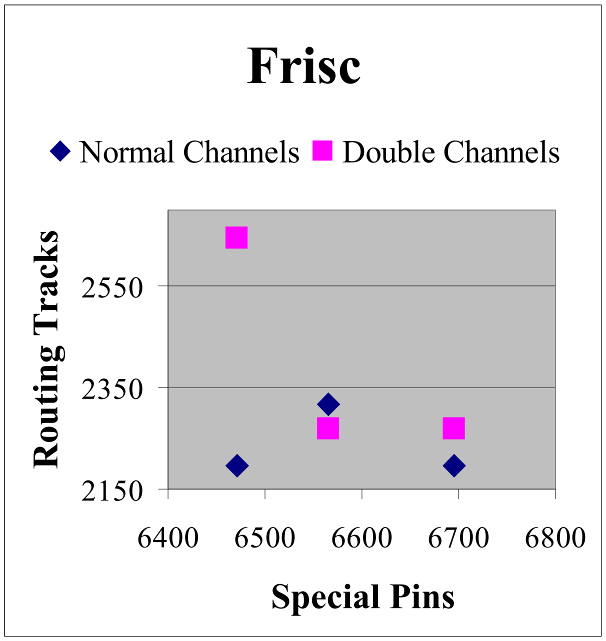


Figure 12 g

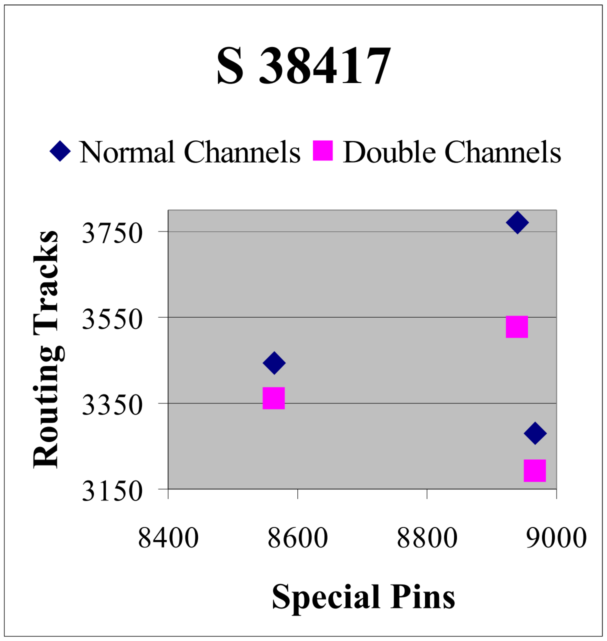


Figure 12 h

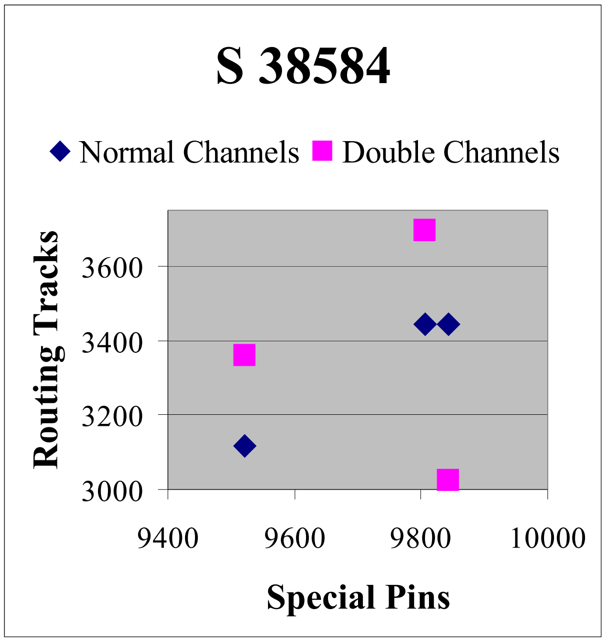


Figure 12 i

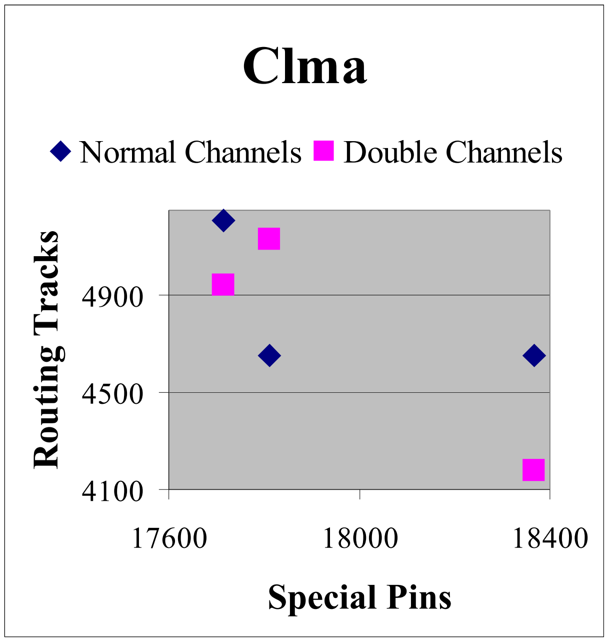


Figure 12 j

Comparing all of the data in Figures 12 a – j, there were two circuits, Diffeq (c) and S 38417 (h), for which the double-wide channels improved routing with all three hard macro seeds, and there was no circuit for which double-wide channels always made routing worse. Nonetheless, overall the double-wide channels seem as likely to worsen the routability as to make it better. Additionally, there does not appear to be any attribute of the data that correlates with the number of special pins.

Figure 13 a brings together the data from Figures 12 a – j, showing the average number of routing tracks needed with normal and double-wide channels for each circuit. This graph shows that about half the time more tracks are needed with the double-wide channels, and the other half the time, less. Figure 13 b shows the data from Figures 12 a – j in terms of routing track reduction, the number of routing tracks needed with normal channels minus the number of tracks needed with double-width channels present. The values in Figure 13 b are positive, indicating improvement with the double-wide channels, half the time and negative the other half.

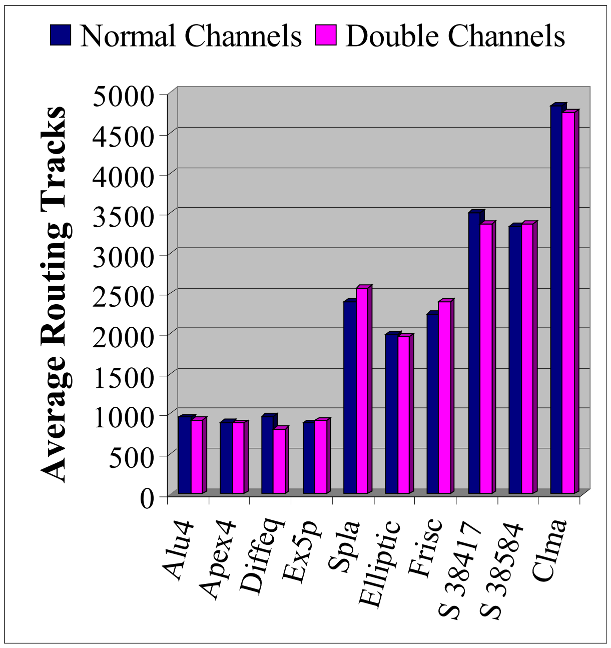


Figure 13 a

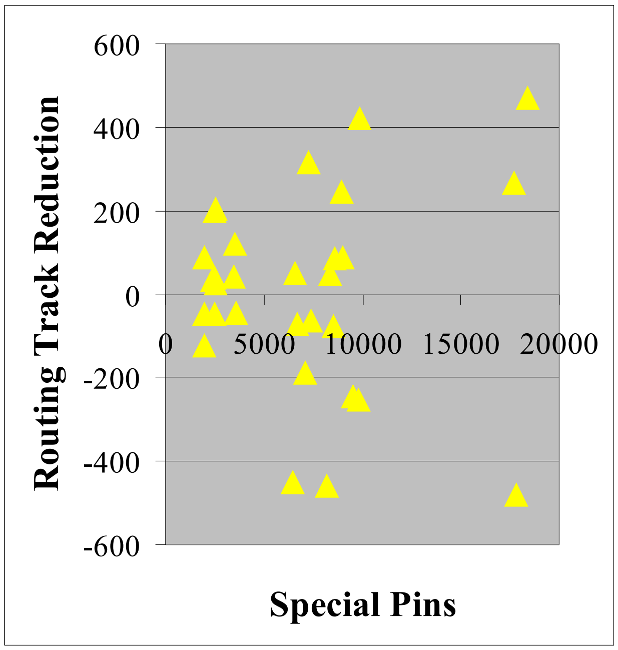


Figure 13 b

Figure 14 a shows the amount of congestion in the channels surrounding the hard macro, showing the fraction of the tracks in these channels that are occupied when the channels are normal size and when they are double-wide. In all cases, the congestion is less with double-wide channels than it was with normal size channels, but the amount of initial congestion and the amount of change varies. The values in this graph are averages across all three random number generator seeds.

Figure 14 b shows routing track reduction as a function of congestion reduction. While Figure 13 b showed three data-points for each circuit, Figure 14 b shows one point per circuit, the average across all three seeds. With the exception of the rightmost point, all of the positive – successful – points occur when congestion is lessened by 5 to 20 %. Below 5 percent congestion relief, the routing track reduction is near zero – doubling these channels did not make them big enough to remove congestion problems, so routing was no easier than before. Above 20 %, routing track reduction is extremely negative – congestion was made so low that there is a great deal of wasted space in the FPGA. This suggests that rather than doubling the size of the channels around the hard macro regardless of the size of the FPGA being used, results might be better if the channels were increased by an amount that would relieve congestion by 5 – 20 % for that size FPGA.

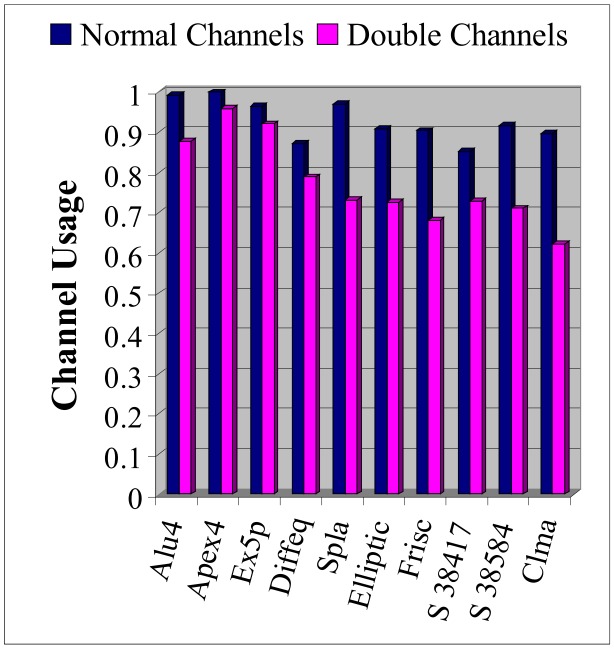


Figure 14 a

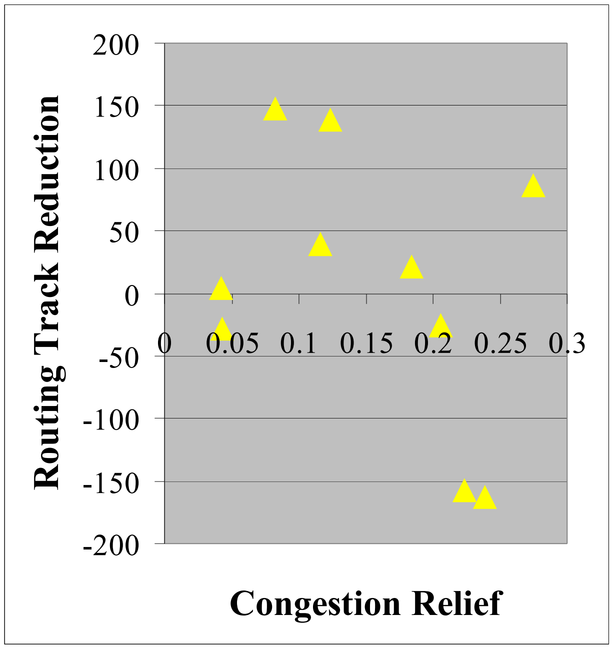


Figure 14 b

<u>2) Second Experiment – Adding Rather than Doubling:</u>

The data in Figure 14 seems to indicate small congestion relief and small change in routing for the smallest Benchmark circuits, large congestion relief, usually accompanied by wasted space for large Benchmarks, and the best results for intermediately sized Benchmark circuits. It seemed that making channels around the hard macro more than twice as large as other channels for the small Benchmarks, and less than twice as large for the large Benchmarks might give better results than were obtained in the preliminary experiment.

To explore this, I repeated the preliminary experiment using only a single seed, adding ten tracks to the channels around the hard macro rather than doubling them. The results of this are shown in Figures 15 a – b. Although ten tracks are about how many tracks were added to the intermediately sized circuits, more than were added to the small circuits and less than were added to the large circuits, the data from this experiment is quite similar to that in the preliminary experiment, about half improvements with the wider channels and half worsenings. My estimate of ten extra tracks was very rough – its failure indicates that much more careful analysis of FPGA size, hard macro size, and hard macro connectivity will be necessary to determine the exact size widened channels need to be to be useful.

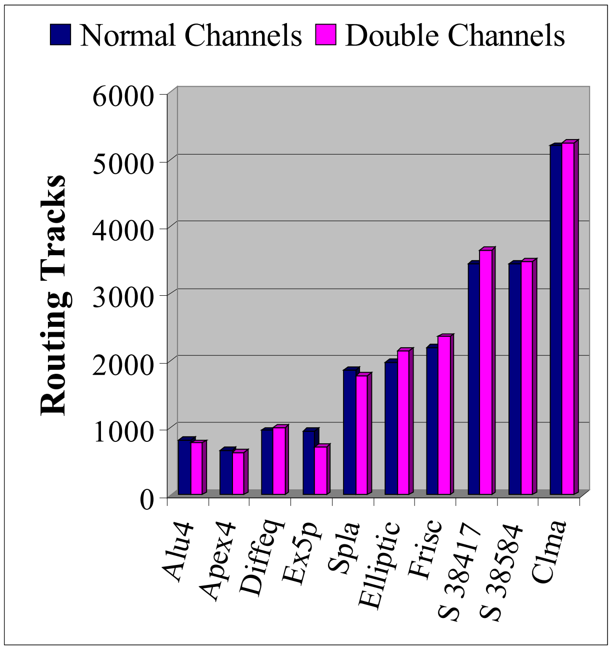


Figure 15 a

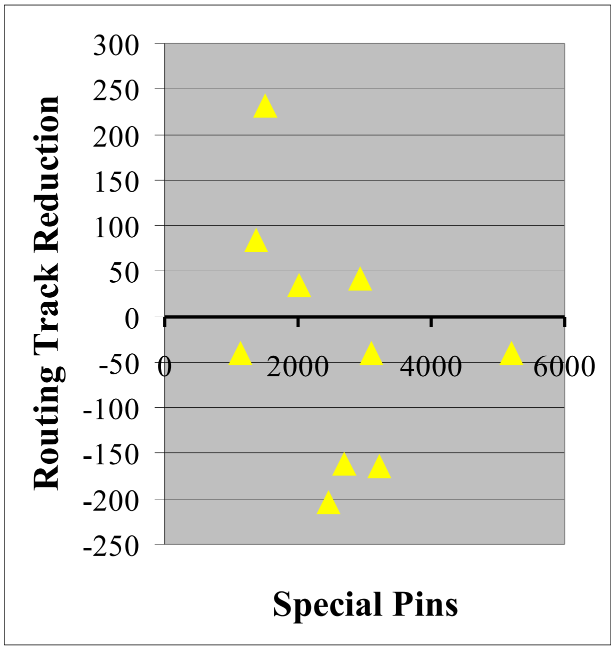


Figure 15 b

3) Experiments with Multiple Hard Macros:

In addition to my experiments with a single hard macro at the FPGA's center, I ran several trials with four hard macros arranged symmetrically around the center of the FPGA, mainly to demonstrate that the multiple hard macro features added to VPR worked correctly. Detailed analysis is needed even to run proper experiments with a single hard macro, so at this stage we are certainly not ready to run extensive experiments with multiple hard macros. I ran trials using a single seed on just three Benchmark circuits: Alu4, Apex4 and Spla. Each hard macro occupied 1% of the FPGA's area, so together, the four hard macros filled 4 % of the FPGA, as in previous experiments. As in the preliminary experiment, channels around the hard macros were made twice as wide as the FPGA's other channels. The results, shown in Figures 16 a – b, are similar to previous data, half improvements, half worsenings, again because the decision to make wide channels doubly wide is arbitrary and not chosen to be optimal for this hard macro arrangement.

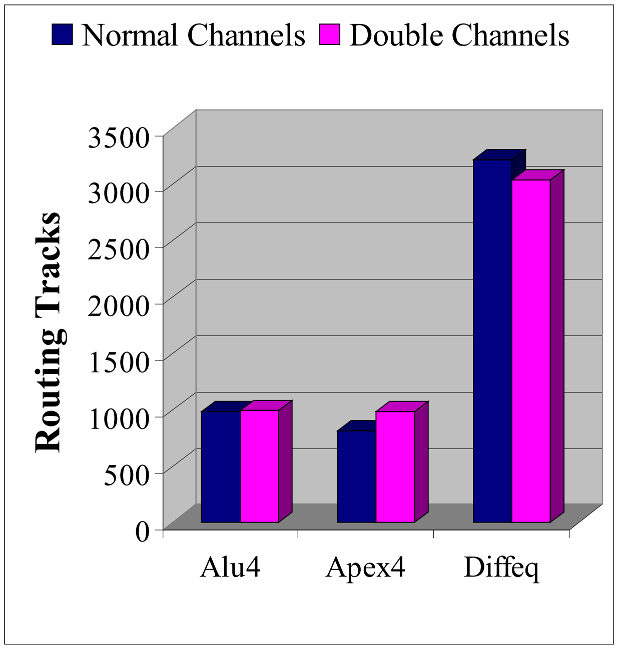


Figure 16 a

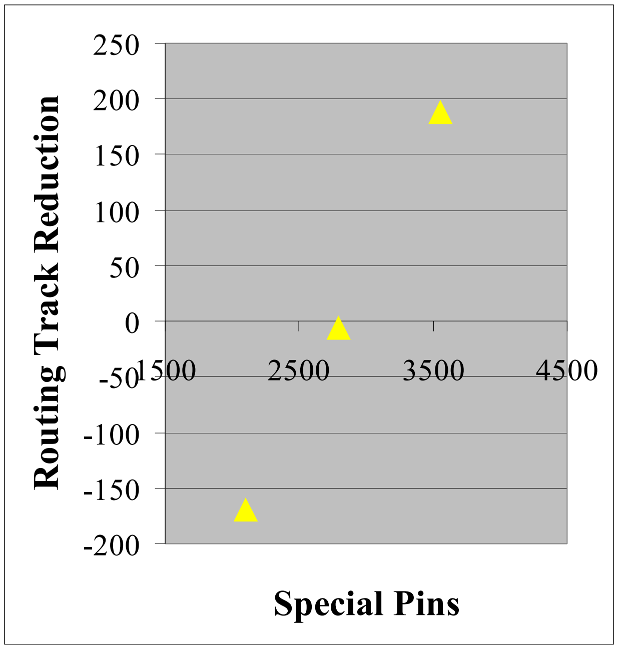


Figure 16 b

Possible Future Work:

My primary contribution to this project was preparing the VPR program for experimentation. I ran several experiments to gather some preliminary data, but these mainly just demonstrate the

CAD tool's effectiveness and do not lead to many useful conclusions. Thorough analysis of what size channels around a hard macro should be, based on FPGA size, and hard macro size and

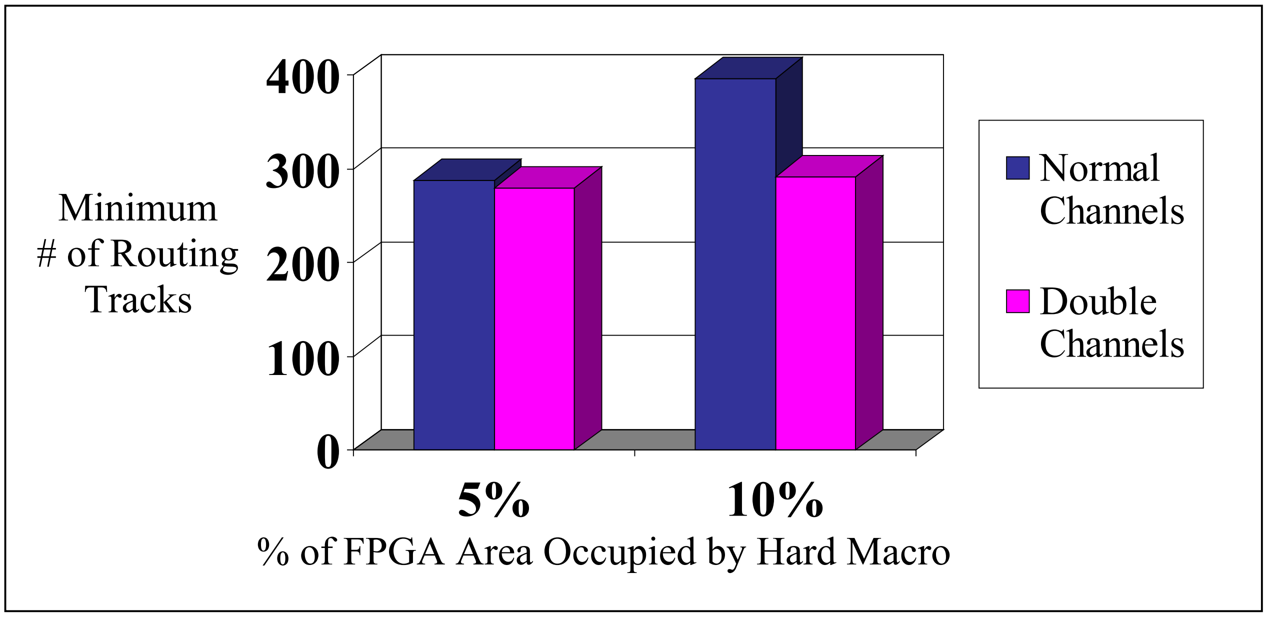


Figure 17

connectivity will be necessary to run more productive experiments on arrangements of single and multiple hard macros. In future work, we may explore increasing the size of not just the channels immediately around hard macros, but of the first three or four rows of channels around them. We may also investigate whether heterogeneous channel structures might be more effective when hard macros occupy a greater fraction of the FPGA's area. This was the case, as shown in Figure 17, in the only trial I ran that routed the same circuit (the e64 circuit) with two hard macros of different sizes. Larger hard macros should affect more of the FPGA's routing and might produce more problems that wide channels could alleviate. The version of VPR I have modified is now capable of performing these experiments.

Summary:

We want to investigate whether a heterogeneous channel structure, particularly one with widened channels around the edges of hard macros could improve routing in FPGAs containing hard macros. I have modified the VPR placer router tool to create arrangements of hard macros and widened channels, allowing placement and routing to be performed on these structures.

Preliminary experiments I have performed with VPR indicate that doubling the width of channels along the edges of a hard macro does not improve routability and further analysis must be done to determine how much these channels should be widened to be useful. The VPR tool is ready for future experimentation based on this analysis.

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.

[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.

[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.