University of California
Santa Cruz

Objective-Based Routing For Physical Design-For-Test

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

by

Richard McGowen

June 1995

The dissertation of Richard McGowen is approved:

F. Joel Ferguson

John M. Acken

Tracy Larrabee

Dean of Graduate Studies and Research

Contents

List of Figures

Objective-Based Routing For Physical Design-For-Test

Richard McGowen

ABSTRACT

This dissertation describes how objective-based routing can be used to create circuits that are more testable. It presents techniques that allow the importance of routing time, routing area, and testing difficulty to be weighed against one another. These techniques vary the strength of an objective function to take the relative importance of differing goals into account. These techniques achieve significant improvements in testability with minimal time and area penalties. This dissertation presents methods for using the routing techniques to efficiently improve the testability of circuits under the static-voltage and pseudo-exhaustive testing methodologies and gives results for sample circuits. It discusses the relevancy of varying strength objectives to routing in general.

Keywords: Routing, Channel Routing, CAD, VLSI Layout, DFT, P-DFT, Testability, Design For Test, Goal-Oriented Routing

Acknowledgments

I would like to thank Sequoia Semiconductor and Silicon Engineering for providing the motivation and funding for my original implementation of Yoeli's router. I would like to thank the NSF and the SRC for their support of my later work.

I would like to thank Dr. F. Joel Ferguson for his support. His help and insights were much appreciated. (He also plays a pretty mean game of 1830.)

I would like to thank John M. Acken and Tracy Larrabee for agreeing to be on my committee and for their useful suggestions. I would like to thank Jon Udell for providing a copy of his segmentation software.

Finally, I would like to thank my wife, Janet McGowen, for encouraging me to finish writing my dissertation and for her general support.

1. Introduction

During the fabrication of an integrated circuit, errors may occur that cause two wires to be shorted together. Unintentional shorts are undesirable as they may cause a circuit to fail or may decrease its long-term reliability. In order to detect such shorts, the circuit must be tested. However, some shorts are more difficult to detect than others. Some may be undetectable. Increasing quality requirements drive the need for Physical Design-For-Testability. The goal of this research is to define and demonstrate how an objective-based channel router improves testability.

To ease the testing problem and increase the likelihood of shorts being detected, the probability of an undetectable or hard-to-test short occurring should be reduced. This can be accomplished by generating circuit layouts that have increased spacing, or additional wires, between wire pairs that could potentially cause undetectable shorts. When improving testability, care must be taken to ensure that this is not detrimental to the achievement of other, more important, goals. Since the relative importance of goals varies from design to design, it is important that the router be able to vary the strength with which it increases testability. The importance of a goal may range from that of an *objective*, a goal that should be met, to a constraint, a goal that must be met. Although increased testability is important, it is not necessary and can be considered an objective. To allow trade-offs to be made with other goals, such as routing time or routing area, the router should be capable of varying the relative strength with which it increases testability. This can be done by using a carefully designed weighting scheme.

This dissertation describes a channel router that makes use of the above ideas to generate more testable circuits. First, previous work on improving the testability of circuits through layout modification is presented. Next, an overview of channel routing and goal-oriented routing is given. The development of a channel router that improves testability is then described and its performance measured. It is then shown how the routing techniques can be used to generate more testable circuits. The applicability of the underlying ideas to routing in general is then discussed. Finally, areas of future work and conclusions are presented.

2. Physical Design-For-Test Overview

Physical Design-For-Test (P-DFT) is the process of changing the layout of a circuit to make it more testable. It is a subset of Physical Design-For-Manufacturability (P-DFM), the process of modifying the layout of a circuit to produce and ship a larger number of good die. Some of the work presented in this chapter provides both P-DFT and P-DFM techniques. The P-DFM techniques are included because they also improve the testability by relaxing the importance of testing for specific faults, even though their primary focus is improving other manufacturability aspects, such as reliability.

2.1 Cell Modifications

One way in which the testability of circuits has been improved has been to change the layout of cells so that the cells themselves are more testable. One reason for doing this is to take advantage of the reuse of cells within a circuit and across many designs. By investing effort in cell design, a large improvement in testability can be gained since the cells will be used many times.

2.1.1 Targeting Stuck-Open Faults

In 1987, Koeppe provided layout rules for decreasing the likelihood of occurrence of hard-to-test stuck-open faults and for improving their testability should they occur [Koe87]. Stuck-open faults present particular problems when they occur in parallel paths in CMOS gates as they can introduce state into the circuit and require a sequence of patterns to test a single fault[Wad78]. To ease this problem, Koeppe suggested three modifications that can be made to cells during the layout process.

The first modification targets opens in paths to power or ground. Parallel paths are replaced with a ring-shaped LOCOS structure. Moving the parallel paths into the diffusion layer provides two advantages. The diffusion layer is less likely to experience a break fault

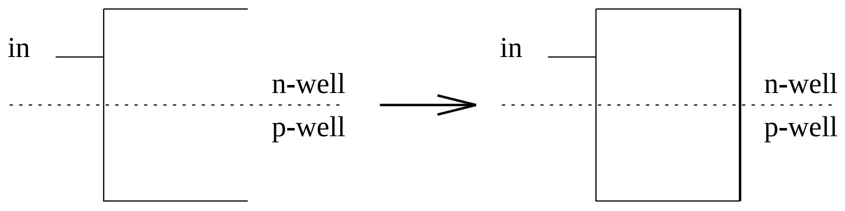

Figure 2.1: Koeppe's second suggested modification.

(DFM), A break in the metal layer or a contact will now disconnect the entire parallel network; thus the fault can usually be detected with a simple stuck-at test (DFT).

The second modification targets the susceptibility of gate inputs to opens. It applies to cells created with a vertical-well design style which are defined by Koeppe as one in which "the n-wells and p-wells are arranged in parallel with the direction of the data flow". For these cells, he suggests closing metal input line branches to form a single loop as shown in Figure 2.1. Doing this provides redundancy that can overcome a single break in the input line.

The third modification targets the testability of gate input to opens for the horizontal-well design style. In this case, Koeppe suggests that input lines be branchless and connect to gates in parallel paths before gates in serial paths. By wiring inputs up in this manner, testing is eased since any break in an input line will disconnect the serial path transistor(s), which, like the first modification, causes an easily detectable stuck-open fault. The fault is easily detectable since either the pull-up or pull-down path (whichever is serial) will be totally disconnected and once a parallel path has precharged the output, the fault will behave as a stuck-at fault.

2.1.2 Local Transformations

Levitt and Abraham also considered the problem of enhancing testability through the modification of cell layouts [LA90]. They devised a method whereby local transformations can be performed on an extended switch-graph representation of a circuit to increase its testability. The first transformation they proposed, shown in Figure 2.2, is equivalent to

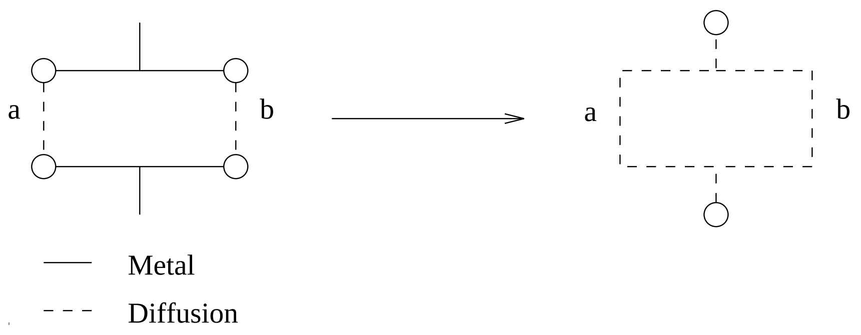

Figure 2.2: Levitt's and Abraham's transform 1.

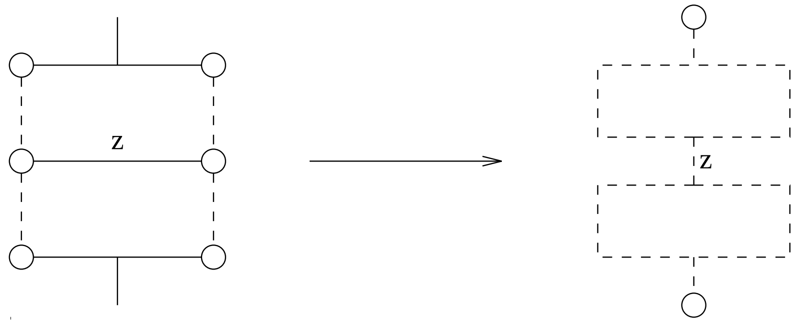

Figure 2.3: Levitt's and Abraham's transform 2.

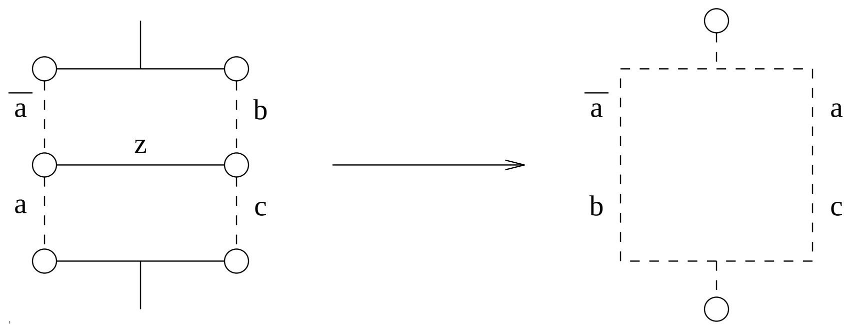

Figure 2.4: Levitt's and Abraham's transform 3.

Koeppe's LOCOS modification. The second transformation, shown in Figure 2.3, combines the first transformation with an additional replacement of any intermediate metal line with diffusion. In addition to having the benefits of the first transformation, this transformation increases reliability by removing a metal line and a couple of contacts, all of which are more likely to break than diffusion. This in turn leads to easier testing, as one can, for practical purposes, now ignore a break in line **z**. Before the transformation, a break in **z** would be relatively difficult to detect since come complete single-stuck-at test sets will not detect this fault. After the transformation, a break in z causes a stuck-at fault by disconnecting the circuit output from the power rail. The third transformation, shown in Figure 2.4, is similar to the second except that now line **z** is removed altogether. This is a specialized case as it requires a variable to be present in both parallel networks, being complemented in one and uncomplemented in the other.

2.2 Routing Modifications

Another means of improving the testability of circuits is to modify the routing. In general, this is accomplished by separating lines to reduce the likelihood of undesirable shorts occuring. Several papers advocate improving testability through routing [Ack88, TTA+91, SCS+92, FKM92, Fer93], but little actual work has been done [MF94b, MF94a].

2.2.1 Routing and Placement Suggestions

In his Ph.D. work, John M. Acken discussed reducing the likelihood of occurrence of shorts and increasing the testability of the shorts that do occur with routing [Ack88]. He suggests that outputs of multi-input gates should be adjacent to as few other lines as possible. The more inputs a gate has, the harder it is to control. If a short occurs between two lines that are difficult to set to different values, the short may be hard to detect. Acken also suggests that segment crossing shorts be avoided. Under the pseudo-exhaustive testing techniques, a circuit is divided into segments and each segment is exhaustively tested. If a short occurs between two lines in the same segment, pseudo-exhaustive testing is guaranteed to detect the short if it can be detected with a single input pattern. If a short occurs between two lines in different segments, detection is not guaranteed.

Other routing suggestions include giving preference of use to routing layers that are less susceptible to shorts and giving preference of avoidance to either adjacencies or cross-overs, depending on which is more susceptible to shorts. Since the routing is in large part determined by the placement, he recommended that testability should be improved during cell placement and discussed how to do this for circuits tested with the pseudo-exhaustive testing methodology. By modifying the placement of cells so that cells in the same segment are grouped together within the layout, testability can be enhanced because adjacent wires are now more likely to be in the same segment.

2.2.2 Classifying Faults for P-DFT

Teixeira *et al.* suggested dividing realistic faults into fault classes based upon their resistance to stuck-at test sets [TTA+91]. The different fault classes can then be used to focus DFT attention where it is most needed, namely on fault classes that are hard to detect and contain large numbers of faults. Once an important fault class has been identified, modifications can be guided by tracing the fault class back to its Physical Failure Mode(s) (PFM). An understanding of the PFM can be used to guide the derivation of DFT rules for the fault class.

The work of Teixeira *et al.* was further developed in a later paper [SCS+92]. In this paper, Saraiva *et al.* increased the granularity of the bridging fault category and gave DFT rules. A metric, called *Rule Violation Severity* (RVS) was given to help guide the application of DFT rules. The RVS metric helps formalize the importance of avoiding a particular fault-type based on how often it occurs and how likely it is to be detected. They stressed the importance of judicious application of DFT rules to avoid large area penalties. They gave the following DFT strategies:

- Cell versioning: Identify difficult-to-test areas and use test-based layouts of cells, rather than the performance-based layouts of cells, in these areas.
- Testability constrained routing: Identify difficult-to-test routing areas and increase the distance between lines in these areas.
- Selective decompaction: Identify difficult-to-test areas where routing lines or cells are close together and spread them out into sparse neighboring areas.

2.2.3 P-DFT Based Upon Testability and Likelihood

Along similar lines, Khare and Maly suggested that routing and placement can improve testability by taking a testing based objective function into account [FKM92]. They suggest using a measure of overall testability to choose between different solutions. They recommend a testability measure based on probability of detection and probability of occurrence. For probability of detection, they suggest using the Sandia Controllability and Observability Analysis Program (SCOAP) measures. For probability of occurrence, they suggest using critical area measures for each failure mode. By then weighting each fault's probability of detection by its probability of occurrence, and summing these numbers for all the faults in the circuit, they feel that a measure of circuit testability can be achieved. One can then use this measure to compare the testability of different solutions and, by using an iterative process, maximize the testability.

2.2.4 Targeting Fanout-Free Regions

In 1993 Ferguson said that routing could be used to avoid placing lines adjacent to each other if a short between them would cause feedback [Fer93]. Since separating all such lines is potentially too expensive, he said that the criteria could be relaxed. A router could target only lines where there is no fanout in the region between the two lines and the line closer to the output is more strongly driven.

2.3 Inductive Fault Analysis

In order to develop P-DFT techniques and to measure their improvements, it is important to know which faults can actually occur in a given layout. Inductive Fault Analysis (IFA) techniques provide a means of determining this [MFS84, Fer87, FS88]. Carafe is an example of an IFA tool [JF93].

Carafe is a realistic fault extractor that, given a circuit's layout, process technology information, and fabrication defect characteristics, calculates which shorts and opens can occur if a spot defect modifies a circuit during the manufacturing process. Carafe reports which faults can occur and reports *critical area* totals for each fault. The critical area for a given fault is the total amount of area in which the center of a spot defect could land and cause the fault to occur. Critical area totals provide a means of measuring the likelihood of a given short occurring and thus how important it is that it be covered. Critical area coverage relates more closely to quality levels than fault coverage. Critical area totals also allow the impact of changes that reduce, but do not eliminate, the likelihood of a short occurring to be measured.

3. Channel Routing Overview

This chapter presents a sampling of previous work on channel routing and discusses previous work on goal-driven channel routing. This chapter provides an understanding of general and goal-oriented channel routing so that the work presented later for improving testability may be put in perspective.

3.1 Channel Routing in General

Channel routing was developed as an efficient, automated method of generating wiring for two-layer printed circuit boards [HS71]. It has since been adapted to efficiently generate the interconnect for integrated circuits. The strength of channel routing is that it constrains and divides the wiring problem, making the overall interconnect generation problem less difficult.

3.1.1 Terminology

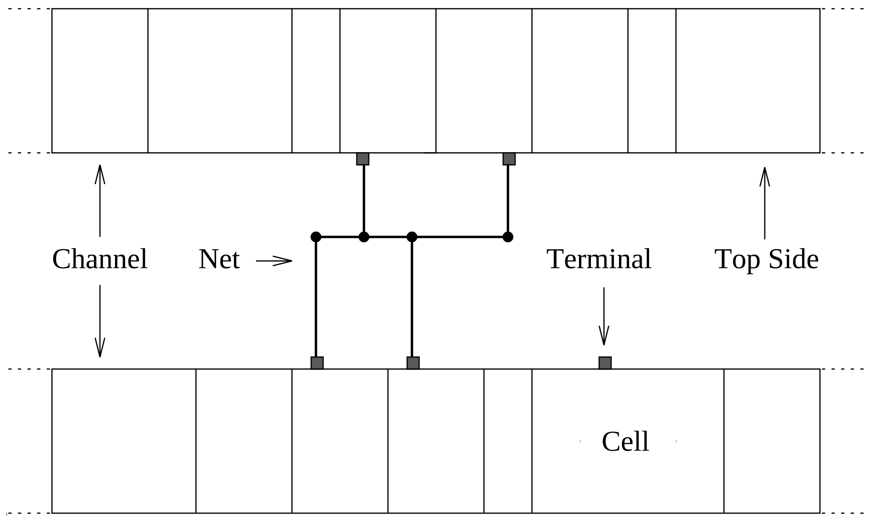

Figure 3.1: Basic channel terminology

Some definitions of channel routing terms follow. Refer to Figure 3.1 for examples. The *channel* is the rectangular region between two parallel rows of cells in which channel routing is performed. Since it is customary to place cells into horizontal rows, the top side of the channel is the side closest to the upper row of cells.

A *terminal* is a point on a channel edge to which a connection must be made. Terminals represent the inputs and outputs of cells along the channel and connections to other channels above and below the channel being considered. In channel routing, only the locations of terminals along the top and bottom sides are fixed. If the locations of the side terminals are fixed, the routing problem is not considered channel routing, but rather switch-box routing.

A *net* represents an electrical connection between two or more terminals. Each net becomes a set of connected wire segments when the chip is fabricated.

A *wiring model* specifies how the wire segments will be physically represented. It specifies how many layers are available for wiring, the directions in which wires may run, and how they may overlap.

Using the above terms, the channel routing problem can be defined as: Given a channel, a list of nets, and a wiring model, generate an efficient physical representation of the nets that obeys the rules of the wiring model.

3.1.2 Distinctions Between Channel Routers

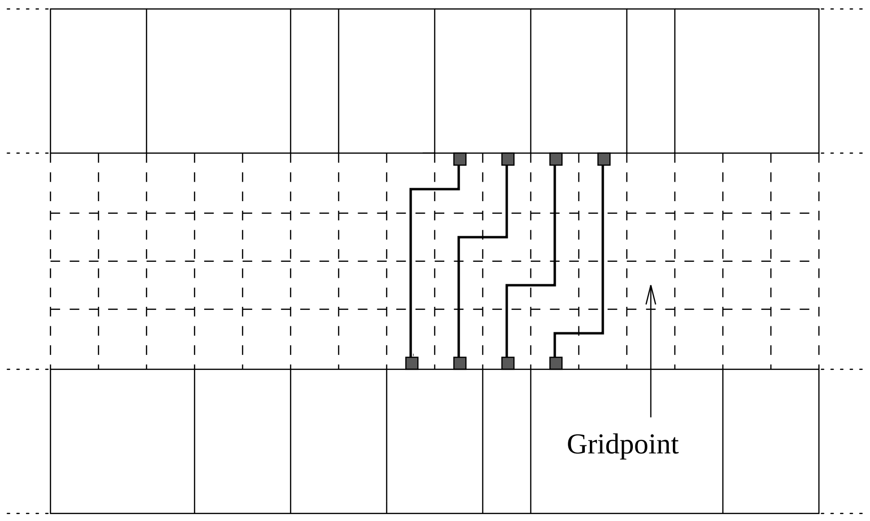

Figure 3.2: Routing grid

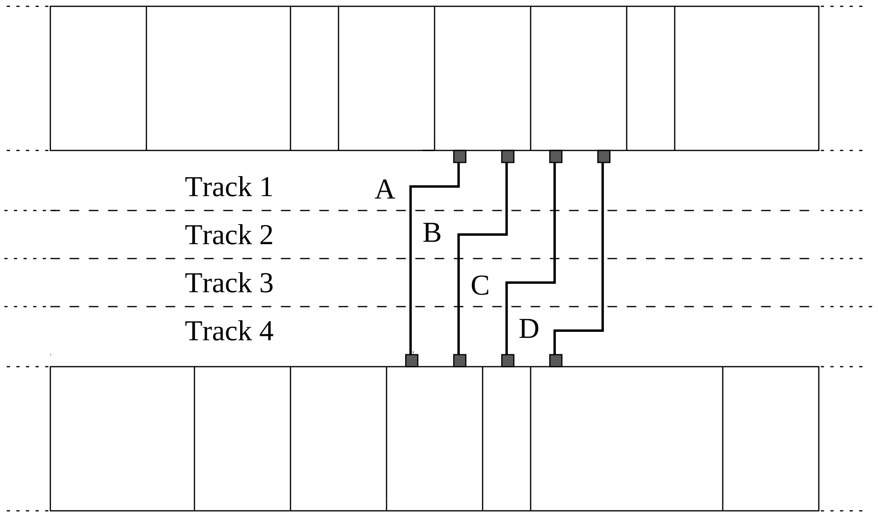

Figure 3.3: Channel with 4 tracks

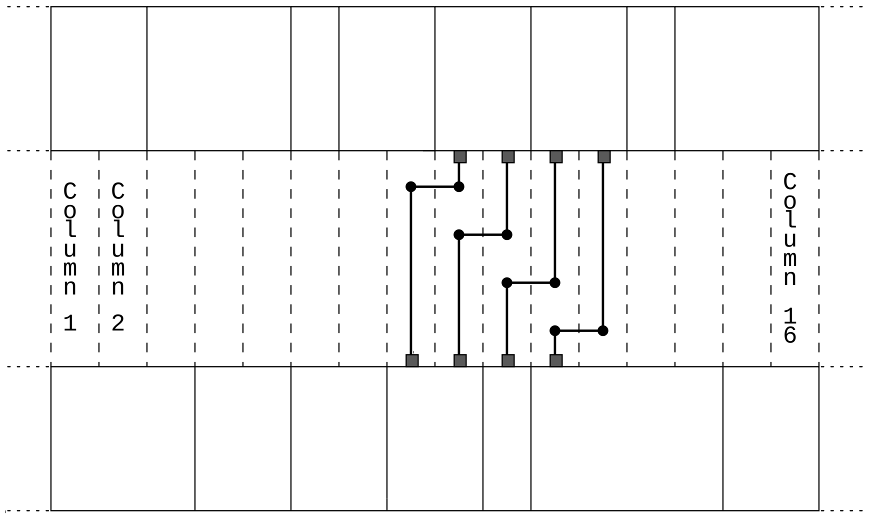

Figure 3.4: Channel with 16 columns

Channel routers can be distinguished by differences in their wiring models. One such distinction is whether a routing grid is used. Gridless channel routers constrain nets based upon the design rules. Gridded channel routers constrain nets to adhere to a *routing grid*, usually determined by the minimum size and spacing of *vias* (connections between wires in different layers). Figure 3.2 gives an example of a routing grid. The rows of the routing grid are called *rows* or *tracks* as shown in Figure 3.3. The *columns* are shown in Figure 3.4. The intersection of a row and a column yields a *gridpoint.* Gridded channel routing is easier than gridless channel routing due to the added constraint of the grid. Gridless channel routing has the advantage of being more flexible and area efficient.

Wiring models differ in the number of routing layers that can be used. Initially, channel routers used two layers. As production technology has advanced, fabrication processes with up to five layers of metal have been introduced and routers using three or more layers have been developed.

Wiring models also differ in the directions in which wires may run and the ways in which they are allowed to overlap. Overlap occurs when a wire in a higher layer is immediately above a wire in a lower layer. The directional flexibility may be physically constrained by the capabilities of the fabrication technology or artificially constrained by the layout tools in order to reduce complexity. Two common directional models are the rectilinear and octilinear models. The first allows wires to run in four directions, north, south, east, and west, and the second allows wires to run in four additional directions, northwest, northeast, southwest, and southeast.

If two or more routing layers are available, restrictions are placed on how wires in the different layers may overlap. It is useful to limit the amount of overlap in order to reduce capacitive and inductive coupling between signals in different layers. The four main overlap models are the *Manhattan, knock-knee, limited-overlap,* and *general-overlap* models.

In the Manhattan model, which is sometimes called the directional model, wire segments are restricted to running either horizontally or vertically. Additionally, each layer is restricted to carrying only horizontal or vertical segments. For instance, in a two-layer Manhattan model, the first layer might contain vertical wire segments and the second layer horizontal segments. Adjacent layers contain segments running in different directions. In three-layer channel routing, this model is further divided into the Horizontal-Vertical-Horizontal (HVH) and Vertical-Horizontal-Vertical (VHV) models. These models simply state which directions the wire segments run, with the middle layer containing either vertical (HVH) or horizontal (VHV) segments. Since this model forms a mesh-like structure, wires in adjacent layers only overlap where they cross.

The other three models do not restrict a layer to contain only vertical or horizontal segments. The only difference between the three is the amount of overlap allowed. The knock-knee model allows wires in different layers to overlap only where they cross or where they each have a corner. The limited-overlap model allows nets in different layers to overlap for short runs. The general-overlap model does not restrict the amount of overlap between nets in different layers.

An additional wiring model that has started to gain use is the *segmented* channel model [GRKG90]. Under this model, the channel has predefined segments that can be joined together, usually through programmable connections, to route signals. This model is primarily used for Field Programmable Gate Arrays (FPGAs).

3.1.3 Comparing Channel Routers

Channel routing is an NP-complete problem [LP90]. As such, no channel router can generate an optimal route for all problems in a practical amount of time. Some routers will produce better routes than others when routing the same channel. There are certain characteristics of routes that are used to rate and compare routers. One characteristic is the area (height) of the generated routes. For gridded routers, sizes can be compared by the number of tracks required. For non-gridded routers the actual channel heights must be compared. Area traditionally has been the most important comparison criteria because it has a high impact on production costs.

In order to accurately judge a router's performance in terms of area, it is useful to know how close to optimal is its performance. In two-layer Manhattan routing there are two measures that provide lower bounds on the height of a channel. One reflects horizontal constraints and the other reflects vertical constraints.

The first measure, which considers horizontal constraints, is channel density. The *local density* at any point in the channel is a measure of the total height of the routing resources

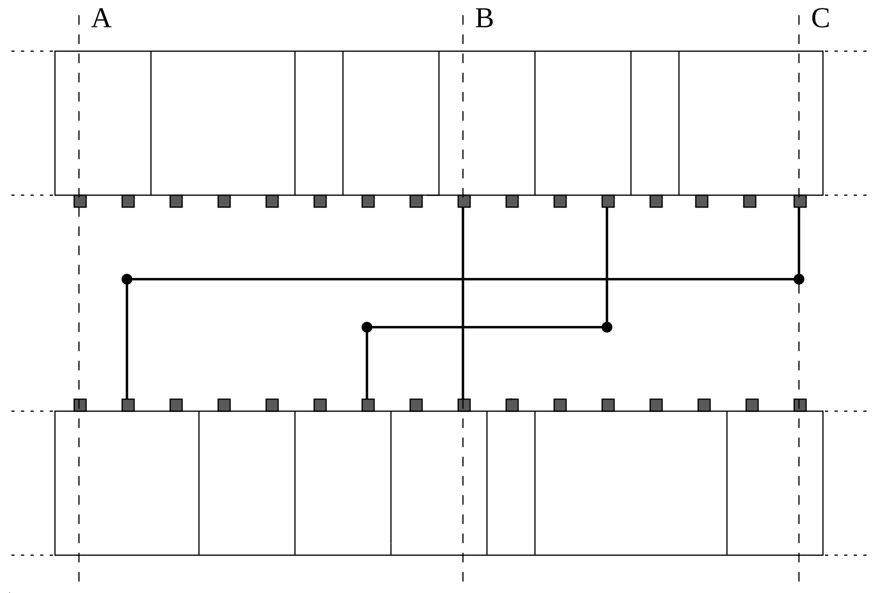

Figure 3.5: Examples of density. The density at A is 0, the density at B is 2, and the density at C is 1.

required at that point. In gridded-channel routing, local density is the number of nets that start at or cross a given column. Nets that only connect terminals at the top and bottom of the same column are not included as they require no horizontal routing resources. (These nets are called trivial or single-column nets.) In non-gridded channel routing, the local density is the sum of the heights of the nets and the sum of the required spacings between the nets that start at or cross a vertical slice of the channel. The *channel density* is defined as the maximum value of all local densities within the channel.

For a density example, see Figure 3.5. The local density at A is 0, the local density at B is 2, and the local density at C is 1. Note that the density at B is 2, not 3, since one of the three nets there is a trivial net. The channel density is therefore 2, the maximum value of all local densities.

The reason that channel density provides a lower bound is that for the point in the channel where the local density, X, is equal to the channel density, X signals will have to pass through a vertical cross-section of the channel at that point. Since those X signals can not overlap because of horizontal constraints between them, there is no way that the channel could legally be routed and have a height that is less than the density at that point. Channel density reflects a lower bound arising from horizontal constraints.

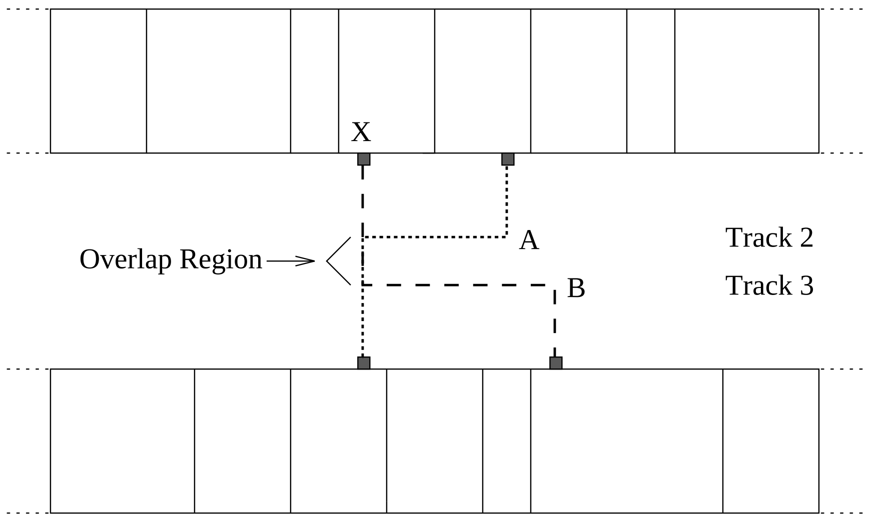

Figure 3.6: Example VCV

A lower bound reflecting vertical constraints also exists. If nets run vertically in the same layer there can be *vertical constraint violations* (VCVs). Figure 3.6 shows an example VCV. In this figure, net A has been assigned to track 2 and net B has been assigned to track 3. However, in column X net B has a connection to the top of the column and net A has a connection to the bottom of the column. If the nets were routed this way, they would be shorted together in the region between tracks 2 and 3. As such, a vertical constraint exists between nets A and B and net B must be assigned to a track that is above the track to which net A is assigned.

As all vertical constraints must be met in order to generate a valid route, the lower bound arising from these constraints can be found by generating a *vertical constraint graph* (VCG). A VCG is a directed graph in which the nodes represent nets and the edges represent vertical constraints between nets. The VCG is constructed by considering each column in which the upper and lower terminals are connected to different nets. For each such column, a node is created (if one does not already exist) for each net and a directed edge is added from the node representing the net connected to the top of the column to the node representing the net connected to the bottom of the column. Once the VCG is constructed, the number of nodes on the longest path in the graph provides a lower bound on the number of tracks that will be required for gridded routers. If any cycles exist, a route cannot be generated unless the cycles are broken. If each edge is given a weight that reflects the spacing between the center lines of the two nets it connects, the longest weighted path gives a lower bound on the channel height for gridless routers. This also requires that a source and a sink node are added to reflect the required spacing between nets and the channel edges.

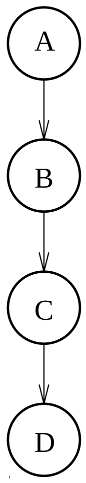

Figure 3.7: Vertical constraint graph.

Figure 3.7, which shows the vertical constraint graph for Figure 3.3, presents an example of a vertical-constraint-driven lower bound. Notice that although the channel in Figure 3.3 has a channel density of two, it requires four tracks due to the channel's vertical constraints. Figure 3.3 shows that four tracks were used and Figure 3.7 shows that the vertical constraint graph has four nodes on the longest path, A-B-C-D.

Most routers are able to meet or come within a track or two of meeting these lower bounds. In cases where routers create routes of equal height, additional characteristics of the routes are used for comparison purposes. Typically total net length and total number of vias have been used. Routes that use shorter wire segments and require fewer vias are better as they are less prone to fabrication errors and less susceptible to parasitic resistances and capacitances.

3.2 General Purpose Channel Routers

This section provides a sampling of two-layer Manhattan channel routers. These routers generate routes without directly considering the effects of yield, coupling, or testing, although they may try to minimize the number of vias and the length of wires. While the routers presented are by no means all that exist, they are representative of most channel routing approaches.

3.2.1 Left-Edge Algorithm

Hashimoto and Stevens are credited with introducing channel routing in 1971 [HS71]. They devised channel routing as a method for routing two-layer printed circuit boards quickly and efficiently. Since its introduction the terminology has changed somewhat; for instance, what they call a *channel* is what we now call a *track* and what they call a *space* is what we now call a *channel*. The basic idea of connecting rows of parallel terminals remains the same. In order to simplify the discussion of their algorithm the terms presented earlier are used.

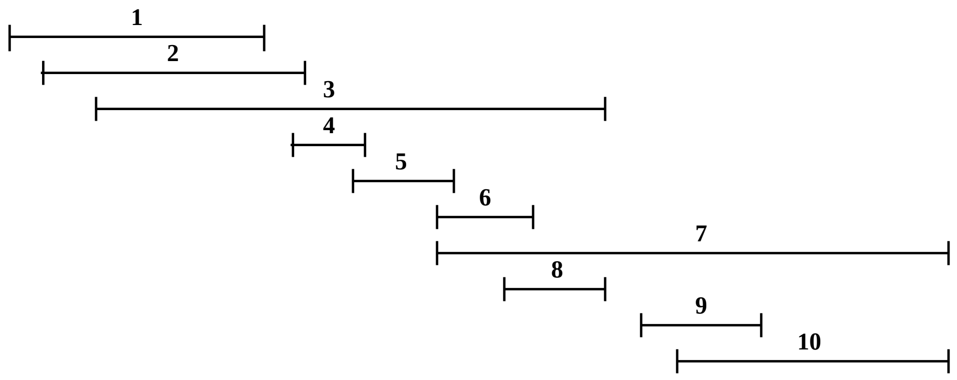

Figure 3.8: Horizontal net segments represented as intervals.

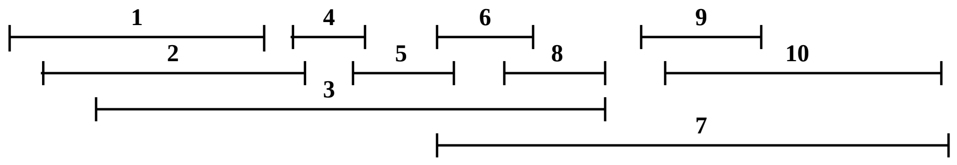

Figure 3.9: Sample net assignment.

If a net is restricted to a single horizontal segment within a channel, it can be represented as an interval with left and right bounds. If the nets are represented as such, an optimal assignment of nets to tracks can be found by performing the following steps:

1. Identify the leftmost-bounded, unassigned net and assign it to the current track.
2. Identify the leftmost-bounded, unassigned net that doesn't overlap with any nets assigned to the current track and assign it to the current track.
3. Repeat Step 2 until no more nets can be placed in the current track.
4. Add a track and return to Step 1 until all nets have been assigned.

Performing the above steps on the nets in Figure 3.8 would assign nets 1, 4, 6, and 9 to the first track, assign nets 2, 5, 8, and 10 to the second track, assign net 3 to the third track, and net 7 to the last track. The resulting assignment is shown in Figure 3.9. Because this algorithm starts at the left and always selects the left-most net, it is often referred to as the *left-edge algorithm* (LEA).

The advantage of this algorithm is that it assigns nets to a number of tracks that is equal to density. Unfortunately, it does not consider VCVs. Hashimoto and Stevens chose to resolve VCVs by rerouting one of the connections involved in each VCV as a separate stage. Because they were routing printed circuit boards, this was always possible. Another way to handle VCVs is to avoid them by not selecting a wire with unassigned ancestors in the vertical constraint graph during net selection. Enforcing vertical constraints during track assignment may generate routes that require more tracks than are necessary.

3.2.2 “Greedy” Channel Router

In 1982 Rivest and Fiduccia described their greedy channel router [RF82]. They presented a column-oriented method of routing two-layer gridded channels. Their basic idea was to start at the left edge of the channel and completely wire each column before moving on to the next in order to avoid the creation of VCVs.

Their algorithm specifies that connections in a column should be made in the following order:

1. Bring nets with connections to the top or the bottom of the column into the channel by assigning them to the nearest track that is either empty or already contains a segment of the same net. Surprisingly, they favor a closer empty track over a further track that already contains the net. This is eventually corrected in Step 2, but is not done immediately in order to provide as much freedom as possible for the following steps.
2. Free as many tracks as possible by joining as many split nets, where a split net is a net assigned to multiple tracks, in the column as possible.
3. Add jogs to bring split net segments closer together to improve the chance of joining them later.
4. Add jogs to move nets closer to the channel side to which they have their next connection.
5. Insert a track close to the center to allow any connection that couldn’t be made in the first step to be made.
6. Extend the horizontal wires that continue to the next column and start again at Step 1.

As can be seen from the above, this is a heuristic algorithm that frees up as much space as possible at each step while still ensuring that all necessary connections are made. An interesting feature of this algorithm is that it can require additional routing columns on the right to join any split nets that remain. Drawbacks of this algorithm are that it generates many vias because of the jogs it adds and that it generates routes that require more tracks than later routers.

3.2.3 Yoshimura and Kuh

In 1982, Yoshimura and Kuh presented two channel routing algorithms [YK82]. Both algorithms work by using the vertical constraint graph to generate routes. The general idea is to first generate a VCG and then merge nodes in the graph by considering horizontal constraints. This merging assigns nets to the same track, assuming that the nets do not overlap. In both algorithms, nodes are merged to minimize the length of the longest path in the VCG. The difference in Yoshimura and Kuh's two algorithms is that the first algorithm makes merging decisions in a "local" manner. It considers only the information represented by the current VCG when selecting between possible merges. This presents a problem since a choice may be made that prevents a better merge at a later step. Their second algorithm addresses this problem by postponing merges as long as possible. It does this by constructing a bipartite graph where the nodes on the left represent nets whose right ends have been reached and the nodes on the right represent nets whose left, but not right ends, have been reached in a left to right scan of the channel. An edge exists between a node on the left and the right if the nets represented by the nodes can coexist in the same track. At this point, any maximum matching that obeys the VCG represents the current best merging. Only when the right end of a net is reached will a node be moved from the right side of the graph to the left. When it moves, the node is merged with the corresponding node on the left if it is part of the maximum matching. When nodes in the bipartite graph are merged, the corresponding nodes in the VCG are also merged. When both algorithms complete, the merged VCG represents a valid route of the channel with each node corresponding to a track.

3.2.4 Yet Another Channel Router 2

Reed, Sangiovanni-Vincentelli and Santomauro presented “Yet Another Channel Router 2” (YACR2) in 1985 [RSVS85]. YACR2 uses a modified LEA to place nets in tracks in such a manner that the number of VCVs created is minimized while the area available to fix unavoidable VCVs is maximized. Once all nets have been assigned to tracks, a series of maze routers is used to correct any VCVs that exist.

Their algorithm starts by finding a column of maximum density and assigning all the nets in that column to tracks. Next, it assigns nets to the right of the column to tracks and then nets to the left of the column to tracks. Even though YACR2 proceeds in a net by net rather than track-by-track manner, this algorithm is LEA-like in that it still optimally fills the tracks by constraining each net to a single horizontal segment and it fills each track as full as possible.

In order to increase the likelihood of generating a route, YACR2 uses an assignment procedure that considers the VCG in order to determine in which track a net should be placed. It selects the track that will leave the most flexibility for later assignments, and, if possible, it selects a track that will not generate any VCVs.

If VCV creation cannot be avoided, YACR2 tries to place the net so that the overlap, in number of tracks, of any VCVs will be kept small. By minimizing the overlap, the ability to fix the VCV should be increased. To further increase the likelihood of correcting VCVs, they avoid placing the net in the top or bottom track.

Once all nets have been assigned to tracks, vertical wire segments are added in columns without VCVs and maze routing is used to make the connections in columns with VCVs. During the maze routing stage, columns containing unresolved VCVs are considered off-limits for fixing other VCVs. The maze routing stage contains three maze routers called *maze1, maze2, and maze3.*

The first router, maze1, tries to fix the violation without adding any additional vias. In order to do this, it must violate the single direction per layer rule and use the vertical layer to make horizontal connections. Since allowing the vertical layer to run horizontally

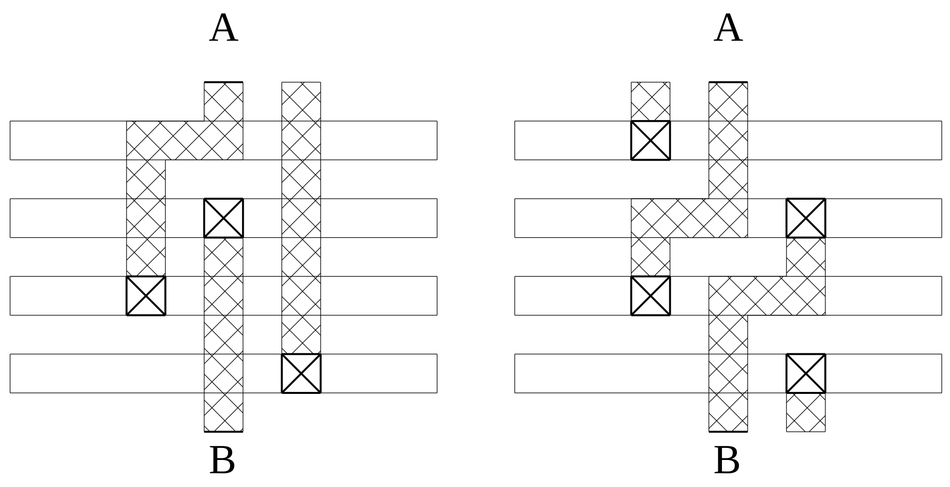

Figure 3.10: Examples of maze1.

can cause lines to overlap for long distances, maze1 restricts any such horizontal segments to span no more than two columns. Maze1 jogs either one or both of the nets only a single column in order to resolve the VCV.

An example of a maze1 route is shown in Figure 3.10. In this figure, there is a vertical constraint between nets A and B. On the left side, the constraint has been met by jogging A to the left. On the right side, the constraint has been met by jogging both nets.

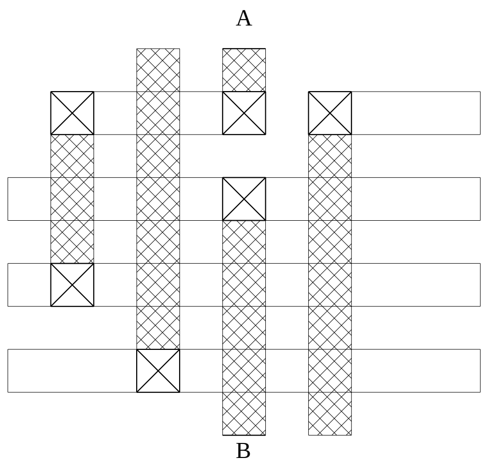

Figure 3.11: Example of maze2.

The second router, maze2, tries to resolve the remaining VCVs by adding only two vias. In order to do this, maze2 only allows one of the nets to jog, but it allows the net to jog for more than 1 column.

An example of a maze2 route is shown in Figure 3.11. In this figure, there is a vertical constraint between nets A and B. Net A has been jogged left two columns so that it can then be connected vertically to its assigned track.

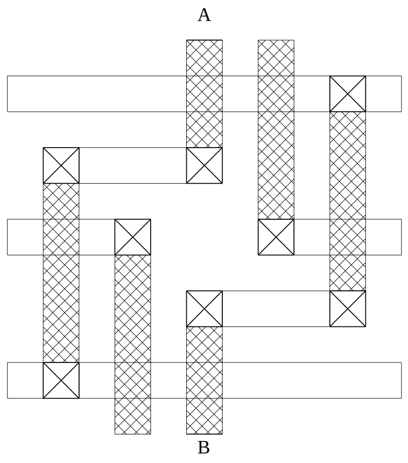

Figure 3.12: Example of maze3.

The final router, maze3, tries to resolve the remaining VCVs by adding four vias. This stage allows both nets to jog for several columns. An example maze3 route is shown in Figure 3.12.

If YACR2 is unable to generate a route after trying all three maze routers, it adds a track that may allow maze2 to fix all the remaining violations. (Adding a single track doesn't help maze1. Maze3 requires the addition of two tracks.) If the channel is still unroutable, YACR2 starts over with the additional track since reassigning the nets to tracks with the additional track may generate fewer VCVs.

3.2.5 Glitter

Chen and Kuh presented Glitter, a gridless, variable-width channel router, in 1986 [CK86]. Glitter uses an elegant, yet simple, algorithm. Chen and Kuh represented vertical, horizontal, and spacing constraints with a single graph, the *Weighted Constraint Graph* (WCG).

Like previous algorithms, nets are restricted to a single horizontal segment, unless division is necessary to break vertical constraint cycles. The WCG is a superset of the VCG. Each node represents a horizontal segment and a directed edge between nodes represents a vertical constraint between the two nodes. However, there is also an undirected edge between two nodes if there is a horizontal constraint between the two nets. Each edge, directed and undirected, is given a weight representing the spacing required between the center lines of the two nets represented by the nodes. For instance, if net A has a width of 2, net B has a width of 6, and the required spacing between the two nets is 5, the weight of an edge between the two nodes representing A and B would have a weight of 9 since $2/2 + 6/2 + 5 = 9$. A source and sink node representing the channel edges are also added.

Once the WCG has been constructed, Glitter generates a correct route by directing the undirected edges in the graph since this will cause all vertical and horizontal constraints to be met. To keep the channel height low, these edges are directed in a manner that minimizes the longest directed path through the graph. This implies that the undirected edges are directed in such a way that no cycles are created. Since guaranteeing a path of minimum length is itself NP-Complete, Glitter uses heuristics for directing the edges.

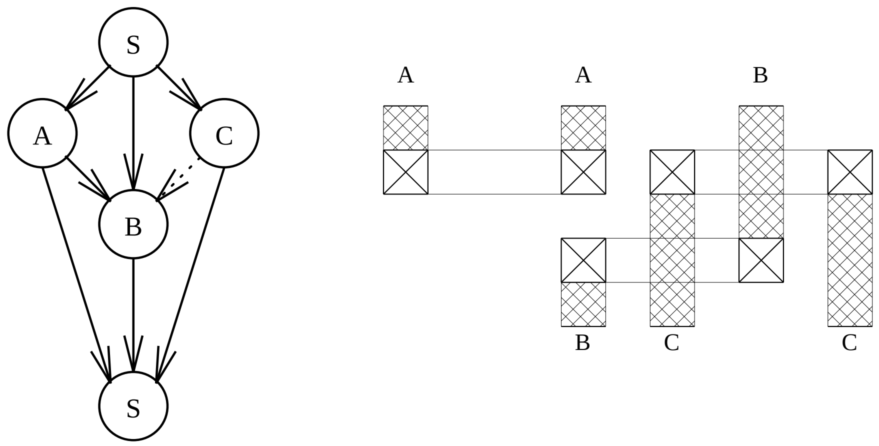

Figure 3.13: Undirected edge directed from C to B.

Figures 3.13 and 3.14 illustrate edge direction. In both figures, there is a directed edge (solid line) from A to B representing a vertical constraint between A and B and an

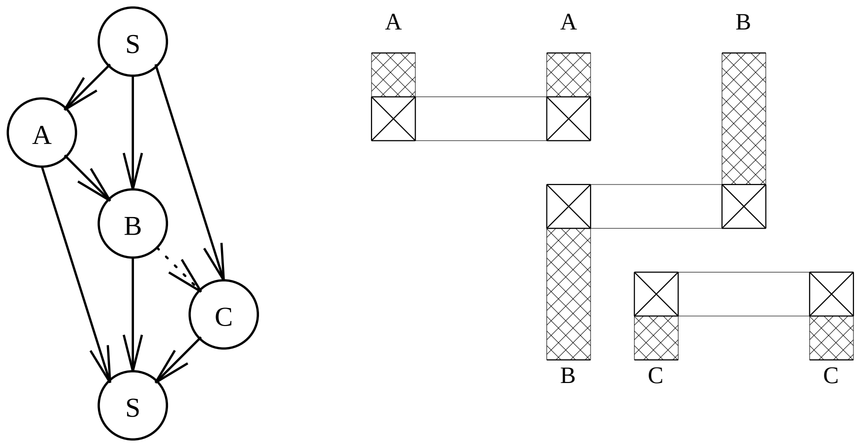

Figure 3.14: Undirected edge directed from B to C.

undirected edge (dotted line) between B and C representing a horizontal constraint between B and C. In Figure 3.13 the undirected edge between B and C has been directed from C to B. If each edge has a weight of 4, the longest path is of length 12, which represents either S-A-B-S or S-C-B-S. In Figure 3.14 the undirected edge has been directed the other way. The longest path now has length 16, from S-A-B-C-S. The case in which the edge is directed to give a shorter longest path gives better results.

One feature of this algorithm is that it can handle nets of different sizes since the weights reflect the spacing between center lines of the nets, not simply the spacing between the nets. Another feature is that it can handle irregular channel boundaries by setting the weight of an edge between the source or sink and a node to represent how far away from the respective boundary it must be.

3.3 Goal Driven Channel Routers

Since a sufficient body of research exists for the basic channel routing problem of generating an area and time efficient route, channel routing research has migrated to achieving other goals. Channel routing has been used to reduce the effects of capacitive coupling, such as delay and crosstalk, to improve the yield of circuits, and to improve the testability of circuits. The work on improving testability was presented in the previous chapter. One point of interest is that while the motivation and approach may differ, much of the goal-oriented work has been accomplished by increasing the spacing between adjacent wires.

3.3.1 Channel Routing With Analog Considerations

The motivation for considering analog effects is to ensure that circuits function correctly and quickly. If the capacitive coupling between lines in a circuit is too large, the circuit may fail to function because of delay or crosstalk. This concern applies equally to analog circuits and high performance digital circuits.

Coupling problems are becoming worse as increases in fabrication technology lead to smaller feature sizes and faster operating speeds. Smaller feature sizes cause the line-line capacitance to dominate the line-substrate capacitance as wires come closer together [Bak90]. Additionally, as circuits begin to switch faster, the cross-talk effects rise. As a result, researchers are developing channel routing algorithms to reduce the effects of these behaviors.

In 1985 Kimble *et al.* [KDG^{+}85] proposed a method for reducing coupling by placing standard-cells such that nets sensitive to capacitive coupling, such as analog signals, are routed in different channels than non-sensitive nets, such as digital signals. This eases the routing problem without requiring any changes in routing algorithms. This method unfortunately suffers from two drawbacks. It has a large area penalty and it does not address the problem of interaction between nets to a granularity of more than two groups. In particular, sensitive nets may still have an undesirable coupling affect upon each other.

In 1989, Gyurcsik and Jeen [Gn89] devised a method of reducing horizontal and cross-over coupling capacitances. They explained how a WCG based channel router can be modified to eliminate such problems. Their method for reducing coupling capacitances between adjacent horizontal nets is to change the weight on an edge between two nodes to reflect the minimum required spacing between the nets in order to meet the capacitive coupling constraints.

They show how the WCG can be modified to eliminate cross-over between two nets by forcing a vertical ordering on the edges through direction of undirected edges. To understand this, consider Figure 3.13. In this figure, B and C cross over each other once, which creates two cross-overs. The reason for this is that B and C have a common horizontal span in which B has a connection to the top of the channel, C has a connection to the bottom of the channel, and B has been assigned to a lower track than C. This means that in the WCG, the edge between B and C was directed from C to B. However, if the edge were directed the other way, the two cross-overs could be eliminated, as in Figure 3.14. Gyurcsik and Jeen noticed this and presented methods for determining when an edge could be so directed to avoid cross-overs. These methods are excellent for meeting hard-constraints, but they have a large area penalty.

In 1990, Harada, Kitazawa, and Kaneko [HKtK90] described their global and detail routing methods for reducing crosstalk by reducing the number of cross-overs that occur. Their channel router uses Gyurcsik and Jeen's method to eliminate cross-over during channel routing. As a post-processing step, it shields unavoidable crossovers by temporarily dropping the line in the lower metal layer to the poly-Si layer where it crosses under the other line. It then places a metal shielding line between them where the cross-over occurs. The weaknesses of this method are that it adds more vias, temporarily runs a line in poly, and requires that a shield line be present.

In 1990 Choudhury and Sangiovanni-Vincentelli [CSV90] eased the analog routing problem by presenting a thorough method for constraint generation. Previous work used constraints during channel routing, but did not address the constraint generation issue. In their work they show how to generate bounding and matching constraints that, given a set of performance functions and a set of parasitics, maximize the routing flexibility while still meeting performance constraints. They rate the importance of each parasitic in respect to each performance function and call these ratings *sensitivities.* They then use the sensitivity information to generate bounding constraints that meet the performance constraints while keeping the overall constraint cost low in order to maximize routing flexibility. If the constraint on a highly sensitive parasitic is increased by a small amount, the constraint on a highly insensitive parasitic can be decreased by a large amount. By keeping the overall cost of the constraint set small, any router using the constraint set should require less area to generate a route than if it were given a constraint set of higher cost. While this method provides an excellent means of weighting constraints against each other, it generates hard constraints that then must be met by the router.

In 1993 Choudhury and Sangiovanni-Vincentelli [CSV93] furthered their work by combining their techniques with both the work of Gyurcsik and Jeen and the work of Harada, Kitazawa, and Kaneko. Their work was unique in that they used shield nets to shield adjacent lines on the same layer and showed how the weighted constraint graph can be modified to do so. In addition, they also described how to modify the weighted constraint graph to keep vertical net segments that begin on opposite sides of the channel from being adjacent by adding a directed edge from the net whose vertical segment begins at the top to the net whose vertical segment begins at the bottom. Looking back at Figures 3.13 and 3.14, it can be seen that directing the undirected edge from B to C also causes the rightmost vertical segments of B and C to no longer be adjacent in addition to removing cross-overs. In 1994 Kirkpatrick and Sangiovanni-Vincentelli [KSV94] further refined this work by extending their analog sensitivity concept to include digital sensitivity.

In 1992, Mitra, Nag, Rutenbar, and Carley [MNRC92] described their global and detail mixed-signal routing system. Their system helps analog nets meet user defined Signal-to-Noise Ratios (SNRs). The channel routing part uses linear programming to budget the parasitics and then uses simulated annealing to generate a route. Simulated annealing is used to select different paths in the weighted constraint graph. The annealing cost function includes meeting of parasitic constraints as a primary objective. This method comes close to performing as a variable strength objective but closer inspection proves it to be constraint driven with an objective function for choosing between different constrained solutions.

In 1993 Chaudhary, Onozawa, and Kuh [COK93] took a different approach by fixing problems in the channel as a post-processing step. They showed how to reduce capacitive coupling by adapting compaction techniques to separate wires by adding repulsion constraints. The repulsion constraints reduce capacitive coupling between target wires by pushing them apart.

In the same year, Gao and Liu [GL93] presented a post-processing method for minimizing coupling in gridded routing solutions by reordering the routing tracks. They generated a mixed integer linear programming formulation of the track-permutation problem. They considered both vertical and horizontal adjacency in their equations and simplified the coupling problem to consider lines as coupled only if they are in immediately adjacent routing tracks. The following year they extended the work to switch-box routing [GL94]. This algorithm never causes an increase in area, but it does not allow tradeoffs to be made in area or time and it has a large time penalty.

3.3.2 Channel Routing for the Improvement of Yield

The motivation for considering yield during channel routing is simple. Chip manufacturers want to increase the percentage of good die on each wafer they produce. The way channel routing has been used to improve yield is to make circuits more resistant to shorts in the routing layers and resistant to opens in the vias. These problems are most often solved by increasing the spacing between wires or by reducing the number of vias.

Removing vias to improve manufacturability has been a topic since the introduction of channel routing when Hashimoto and Stevens presented an algorithm to minimize the number of vias. Since then, much work has been done. One such example is the research performed by The, Wong, and Cong [TWC89]. Their algorithm works as a post-processing step and eliminates vias by violating the directional model once a route has been generated. They try to remove vias by shifting vias and re-routing connections. Shifting a via can be thought of as sliding it in one direction and replacing the span over which it slid with the layer it drags behind it. Simple via shifting can eliminate vias if two vias bump into each other during shifting. This method can increase wire lengths, which in turn can cause additional adjacencies and increase the chance of shorts and therefore fail to meet its goal of improving yield.

In 1989, Pitaksanonkul, Thanawastien, Lursinsap, and Gandhi presented "DTR: A Defect-Tolerant Routing Algorithm" [PTLG89]. DTR reduces the probability of shorts occuring between horizontal wires by separating them through the imposition of an ordering on the tracks. This work is similar to the analog work of Gao and Liu except that it performs the separation during routing itself. DTR uses a modified version of Yoshimura and Kuh's algorithm to assign nets to tracks. It modifies the merging part to consider not only path length in the VCG, but horizontal adjacency. When two candidate merges are equal in terms of path length in the VCG, they select the one that minimizes adjacencies between tracks. They do this by making some tracks as full as possible and others as empty as possible. This way a full track and an empty track can hopefully be placed adjacent to each other later, thereby minimizing the adjacencies between the two tracks. Once all possible merges have been made, the algorithm then orders the tracks within the channel. It does this by adding undirected edges to the graph so that all node-pairs have an edge between them (the VCG is made complete). Each edge is given a weight that reflects the amount of horizontal adjacency between the wires in the two nodes it connects. At this point, a Hamiltonian path that obeys the relationships of the directed edges gives a solution that best optimizes yield by minimizing the total amount of horizontal adjacency. They use heuristics to find a minimal, or near minimal, path. Finding a path determines an ordering of tracks within the channel. Balachandran, Bhaskaran, Ganesan, and Lursinsap furthered this work in 1992 with a post-processing method for separating both horizontal and vertical wires in available space with doglegs [BBGL92], and in 1994 Tyagi, Bayoumi, and Manthravadi improved on the underlying algorithm itself [TBM94].

In 1993, Kuo introduced "YOR: A Yield-Optimizing Routing Algorithm" that reduces critical areas and vias [Kuo93]. Kuo's work is similar to the work of The, Wong, and Cong except that it focuses on reducing critical area rather than minimizing vias. YOR works as a post-processing step by performing *net floating*, *net burying*, *net bumping*, and *via shifting*. Net floating and net burying involve moving a net to another layer, a higher layer for floating and a lower layer for burying, in order to make adjacent wires run in different layers. Net bumping involves jogging a net in available space to move it further away from adjacent nets in the same layer. Via shifting is the process of partially changing the layer of a net segment by shifting the via along the segment. Since these techniques work as a post-processing step, they can be used with other algorithms. The techniques jointly consider via removal and adjacency reduction, so they don not improve one at the expense of the other.

Xue, Huijbregts, and Jess developed yield-based cost functions for a sea-of-gates router [XHJ94]. They consider extra and missing material for both single-layer and inter-layer defects and calculate an overall cost for each net segment based on its susceptibility to the above defects. They then use a Steiner tree based router to minimize the overall cost of each net based on the costs of its segments.

4. A Testability Enhancing Router

This chapter describes my channel router, which I will refer to as MCR, that generates more testable circuits. The underlying channel routing algorithm is an improved version [McG92] of Uzi Yoeli's robust channel routing algorithm [Yoe91], which is in turn based on YACR2[RSVS85]. MCR's basic strategy is to first place nets in tracks using a modified Left-Edge Algorithm (LEA) and then resolve VCVs with a maze router.

4.1 MCR

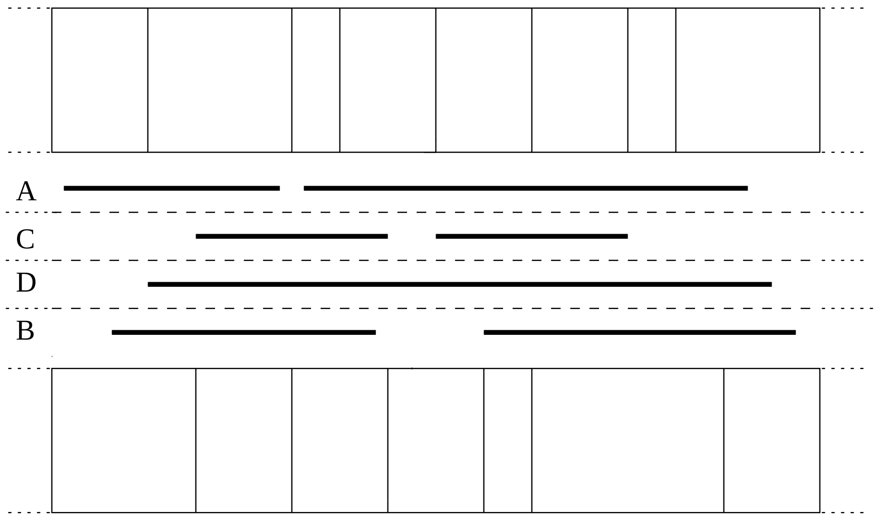

Figure 4.1: Nets are placed in tracks, alternating between the top-most and bottom-most free track. For example, first A would be filled, then B, then C, and, finally D.

In order to describe how MCR was modified to generate more testable routes, the underlying algorithm will first be described. MCR's algorithm can be assumed to be equivalent to Yoeli's algorithm except where otherwise stated.

MCR begins by filling tracks with a maximal set of non-overlapping nets. It alternates between filling the top-most and bottom-most free track. This process is illustrated in Figure 4.1. MCR starts with an initial number of tracks that is equal to the channel

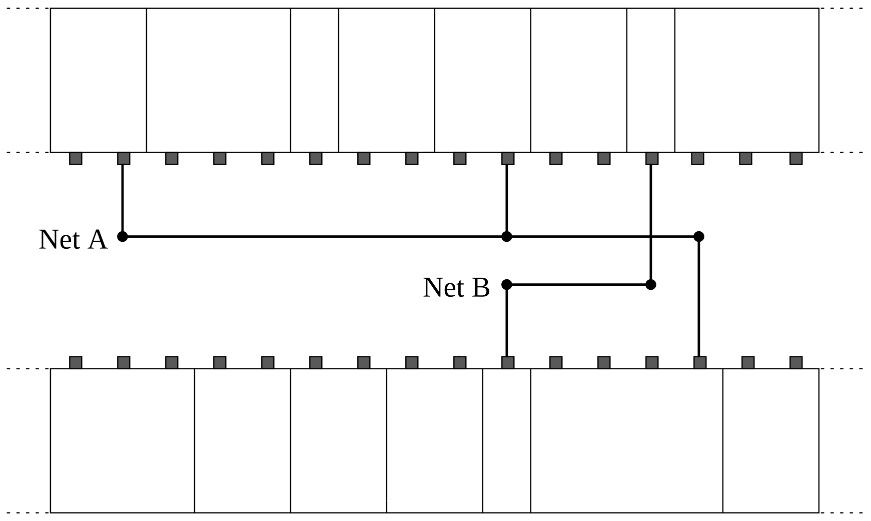

Figure 4.2: Illustration of net weights when placing nets in the topmost track. Step 1 would set the weight of A to 0 and B to 0. Step 2 would add 2 to the weight of A for the column in which A has a connection to the top and B has a connection to the bottom. Step 3 would subtract 64 $(2 \cdot 32)$ from the weight of B for the same column since placing B above A would create a VCV. Step 4 would add 120,000 $(30,000 \cdot 4)$ to both A and B for the four column span where they coexist. The final weight of A would therefore be 120,002 and the final weight of B would be 119,936.

density. When selecting nets for a given track, the desirability, which reflects the creation and resolution of VCVs, of placing each unassigned net into the given track is taken into consideration. To quantify desirability, a weighting scheme is used. Each time a net is being considered for assignment to a track, it is given a weight as follows:

1. Set the weight of the net equal to zero.
2. For each column in which the net has a connection to the channel edge that is closer and another net has a connection to the opposite edge, add the density of the column to the weight of the net.

   Consider Figure 4.2, assuming that nets are being assigned to the topmost track and that the topmost track is the first track to which nets are assigned. Net A has two connections to the closest (upper) side. Since the left-most connection is in a column where A is the only net with a connection, this connection will not contribute to A's weight. The second connection to the upper edge is in a column where Net B has a connection to the lower edge. Because another net, B, has a connection to the opposite side of the channel, Net A will have the density of the column, 2, added to its weight. Since Net B only has one connection to the upper side, and that connection is in a column where no other nets have connections, its weight will not change.

3. For each column in which assigning the net to the current track will create a VCV, subtract the quantity COLUMN DENSITY · VCVWEIGHT from its weight. The subtraction is performed only if placing the net in the track under consideration will cause a VCV. If a VCV already exists in the column, the net is not penalized. VCVWEIGHT is equal to 32 by default.

   Once again consider Figure 4.2, again assuming that nets are being assigned to the topmost track and that the topmost track is the first track to which nets are assigned. This step will penalize Net B's weight by 64 ($2 \cdot 32$) since placing Net B in the topmost track would create a VCV between it and Net A. Net A will not be penalized since placing it in the top track will not cause any VCVs.

4. For each column that the net crosses in which the number of unassigned nets originating at or crossing the column is equal to the number of remaining tracks, add DENSECOL, to the weight of the net. DENSECOL is equal to 30,000 by default.

   Once again consider Figure 4.2. The weights of Net A and Net B will both be increased by 120,000 ($30{,}000 \cdot 4$) for the four columns in which they coexist since the number of remaining tracks is two (including the topmost track) and the number of unassigned nets for those four columns is 2. The initial number of available tracks is set to the channel density.

Before continuing, let me first discuss the purpose of the above steps. Step 1 is obviously an initialization step. The purpose of Step 2 is to encourage the early placement of a net if it can potentially create a VCV. By adding the density of the column to encourage the early placement, the congestion of the area in which the VCV would be created is taken into account. Yoeli always added to the weight of a net, regardless of whether or not another net had a connection to the opposite side. I found that by only adding to a net's weight if another net has a connection in the same column that fewer VCVs would be created to begin with and therefore fewer VCVs would need to be resolved. As a result, smaller routes can be generated [McG92].

The purpose of Step 3 is to discourage the placement of a net in a track if doing so will create a VCV. By multiplying the density of the column by VCVWEIGHT, disallowing the actual creation of a VCV, particularly one in a congested area, is given preference over the possible avoidance of a VCV, the weight added in Step 1. Where MCR's VCVWEIGHT default is 32, Yoeli set it to 8. This discrepancy arises since the designs MCR routes are larger than Yoeli's test designs. As a result, Step 2 could give a net a large enough weight that Step 3 would not have a great enough effect if VCVWEIGHT were only 8.

The purpose of Step 4 is to always ensure an assignment of nets to tracks. By adding a large enough number, at least one net in each dense column will be assigned to a track at each iteration. A dense column is a column in which the number of unassigned nets that cross or originate at the column is equal to the number of tracks that have not been assigned nets. By always assigning at least one of the nets in each dense column to a track, an assignment of nets to tracks can always be found, Yoeli proved this. The caveat here is that DENSECOL needs to be large enough to always outweigh the added bonuses or subtracted penalties of Steps 2 and 3. Since MCR is tuned for larger designs, the default value of DENSECOL was increased from 10,000 to 30,000. The value of DENSECOL needs to reflect the expected number of connections per net and the expected channel densities.

Once all nets have been given assigned weights, a optimal non-overlapping set of greatest weight is selected with a dynamic programming algorithm. The pseudocode that Yoeli gave is:

```
for (col = 1; col $\leq$ channel_length; col = col + 1) {
  for each net that ends at column col {
```

```
            compute $\text{weight}_{net} + \text{total}_{leftcol_{net}-1}$
            $\text{total}_{col} = \textbf{max}(\text{total}_{col-1}, \text{total}_{col}, \text{weight}_{net} + \text{total}_{leftcol_{net}-1})$
        }
    }
```

In the pseudocode, the quantity weight_{net} is the weight of a net. The quantity total_{col} is the sum of the weight_{net}s of the optimal set of nets that end on or before the column. The quantity $\text{total}_{leftcol_{net}-1}$ is the total_{col} for the column that immediately precedes the leftmost column of *net*. The above code steps through the columns in the channel from left to right. For each column, it calculates the maximum total weight for any group of non-overlapping nets ending on or before the column. This quantity is termed the Maximum Accumulated Weight (MAW). When the MAW has been computed for the rightmost column, we know the maximum total weight for the set of nets that can fit in the track. The MAW is stored for each column as it is computed, so that the optimal set of nets can be retrieved at the end of the algorithm.

Before showing how to retrieve the set of nets, it is worthwhile to briefly discuss why the algorithm works. Consider what it means for the MAW to be greater at column n than it was at the previous column, $n - 1$. If the MAW is greater at n, there must be a net contributing to the MAW at n that does not contribute to the MAW at $n - 1$. Since the MAW at a column represents the maximum sum of weights of any set of nets ending on or before a column, the net contributing to the MAW at n that does not contribute to the sum at $n - 1$ must end at column n. Therefore to calculate the MAW at each column we need only consider the effect of nets that end at that column. If no net ends at the column, the MAW at the column is the same as that of the previous column. If a net ends at the current column, a check needs to be made to see if the weight of the net is larger the the sum of the weights of the nets that it would displace. This check can be made by calculating the effect of including the net. Since including the net will displace any nets that overlap with it, the MAW achievable by including the net must simply be the weight of the net plus the MAW of the column immediately before the leftmost end of the net. If including the net increases the MAW, the new MAW and the net that contributes to it are stored.

Once the MAW has been calculated for each column, the set of nets responsible for the final MAW can be retrieved by scanning through the columns from right to left. If at some point the MAW at column n is greater than the MAW at column $n-1$, this means that a net that is part of the optimal set ends at column n. Remember, the MAW at a column can only be greater than the MAW of the previous column if a net that contributes to the higher MAW ends at the column. Since the net contributing to the increase in MAW was stored, that net can be assigned to the current track. The algorithm then skips to the column immediately preceding the left end of the net it just found and continues scanning left until the first column is reached.

This process of selecting a track, calculating the weights of all unrouted nets, and then selecting the set of nets with the greatest total MAW continues until all nets are assigned tracks. Once all the tracks have been filled, a maze router, maze2 from YACR2, is used to resolve any VCVs. If the maze router is unable to resolve any VCVs, a track is added and the nets are reassigned to tracks. This process continues until a successful route is generated.

4.1.1 Efficiency of Track Assignment

Since track assignment is at the heart of the routing algorithm, an estimation of its efficiency is useful. Let N specify the number of nets in a channel, C specify the number of columns in the channel, and D the channel density.

For each track, all unassigned nets have to be assigned weights. For each net, each column it crosses must be considered to determine if the total number of unassigned nets crossing the column is equal to the remaining density of the column. Since the number of unassigned nets is bounded by N and the span of a net is bounded by C, weight assignment is $O(CN)$. Since most nets span only a few columns and the number of unassigned nets drops for each track, in practice the algorithm performs much better.

Once all nets have been assigned weights, a non-overlapping set of unassigned nets must be selected for the track. This requires that each column be considered at most twice, once for calculating the MAW and once for selecting the set of nets, so this step is $O(C)$. Therefore, each track requires an amount of computation no greater than $O(CN + C)$, or $O(CN)$.

This must be done for each track and the number of tracks varies depending on the number of times an extra track must be added because of unresolvable VCVs. Usually, few additional tracks are added and the number of tracks required is very close to density. This means that in practice, it performs close to $O(CND)$. The true upper bound is $O(CND^2)$ since a hard limit of $5D$ has been programmed into the router.

4.2 Wire Separation

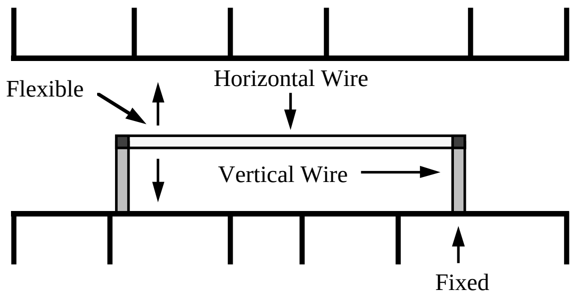

Figure 4.3: Illustrations of horizontal/vertical segments and their flexibility in being moved.

Testability can be improved by decreasing the likelihood of occurrence of hard-to-detect and undetectable shorts in the routing channels. This can be accomplished by not placing nets adjacent to each other if a short between them will be difficult or impossible to detect. However, testability is not the only goal that a testability improving router should consider. Care must be taken to keep area and time penalties within an acceptable range while maximizing the testability improvements. Additionally, since the allowable size of time and area penalties may vary, the strength with which improved testability is approached should be flexible. Finally, in order to avoid wasted effort, maximize the ability of the router to reduce the probability of occurrence of hard-to-test shorts, and reduce area penalties, only specific wire-pairs, rather than all wire-pairs, should be targeted. The remainder of this chapter describes the modifications that were made to MCR to allow it to efficiently separate wires and the resulting performance in terms of time, area, and separation of wires that arose from the modifications.

4.2.1 Horizontal Wires

Most routers restrict wires in routing channels to have segments that run either horizontally or vertically. For simplicity, the term horizontal wire will be used to refer to any segment of a net that runs in a track and the term vertical wire will refer to any segment of a net that runs in a column as shown in Figure 4.3. Horizontal and vertical wires have different characteristics that affect how MCR separates them.

One such characteristic is that one of the endpoints of a vertical wire is fixed, whereas a horizontal wire can be placed in almost any track in the channel. (See Figure 4.3.) Additionally, vertical wires tend to run for much shorter distances than do horizontal segments. This combination of fixed positions and short wire lengths means that vertical wires do not offer as much opportunity for separation as do horizontal wires.

The fact that horizontal wires run adjacent to each other for long distances is a mixed blessing. On one hand, it means that large improvements can be made by separating them. On the other hand, it means that separating them can have a large impact on circuit area if not handled properly. In order to control the increase in time and area when generating more testable results, MCR was carefully modified with these goals in mind.

Two features of MCR lend themselves to efficient separation of horizontal wires. The first is that nets are placed on a track-by-track basis. Because of this, when MCR is considering placing nets into a given track, it can determine which nets were placed in the preceding track. Since this information is available, an efficient check can be made to see if placing a net in the current track should be disallowed to avoid a potentially hard-to-detect short.

If this is the case, the placement of the net can be deferred. This helps keep the router run-times small because it must only consider a small subset of all previously placed nets when considering the desirability of assigning a net to a track.

The other feature of the algorithm that lends itself well to efficiently separating nets is the weighting scheme. Because the assignment of nets to tracks is based on weights, assigning a net to an undesirable track can be discouraged by penalizing its weight. By making this penalty small, separation can be treated as an weak objective, thus minimizing the area penalty at the expense of separation. By making the penalty large, separation can be made a strong objective, thus maximizing the separation at the expense of area. Since the size of the penalty can vary, the relative importance of area efficiency and improved testability can be adjusted.

4.2.2 Vertical Wires

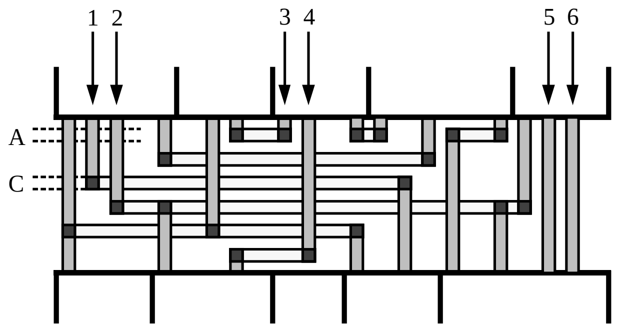

Figure 4.4: Example of three types of potentially undetectable shorts between vertical wires.

Vertical wires also provide a mixed blessing. On one hand, it is harder to separate vertical wire pairs because the column in which a wire enters the channel is dictated by the location of the pin to which it is connected. On the other hand, it is less important that vertical wires be separated since they tend to be adjacent for much shorter distances than horizontal wires.

Vertical wire pairs can be partially separated after entering the channel and before the shorter of the two nets reaches its target track. Consider the two wires originating at Pins 1 and 2 in Figure 4.4. The only space available for separating these nets is the area between Tracks A and C. Since the design rules do not allow horizontal wires closer to the cells than the highest track, the vertical wires of Nets 3 and 4 cannot be separated since Net 3 extends no further than Track A. Nets 5 and 6 are *single-column* nets and can be separated in the area between the top and bottom tracks.

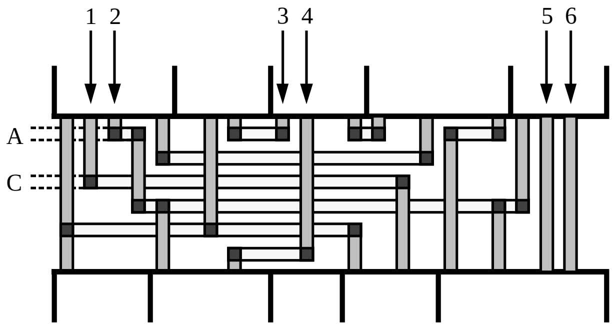

Figure 4.5: Example of how to separate Nets 1 and 2.

Consider Figure 4.4 and assume that this figure represents a route that has been completed but has not yet had any attempts made to separate vertical wires. Notice that there is some free space to the right of Net 2 in which the net-pair 1-2 can be separated. If Net 2 is jogged, the spacing between Nets 1 and 2 can be increased. Figure 4.5 shows how the resulting wiring might look.

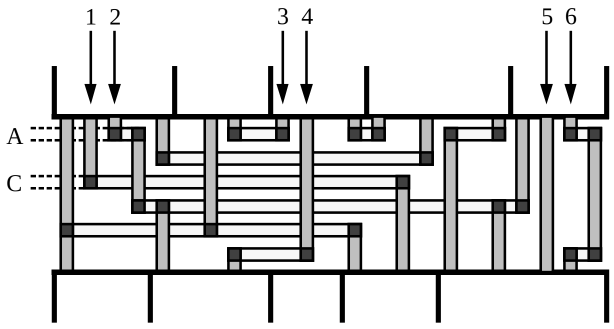

Figure 4.6: Example of how to increase spacing between Nets 5 and 6.

If a double jog is added to one of the nets in a single-column net-pair 5-6, the spacing between the two nets can be increased. The resultant wiring might look something like

Figure 4.6.

Adding these jogs during the routing process itself will greatly increase the difficulty of generating a successful route and therefore have a negative impact on time and perhaps area. Therefore, they should be added as a post-processing step. (These techniques are similar to those of Balachandran, *et al.* [BBGL92] and Gao and Liu [GL93, GL94].)

If the jogs are added as a post-processing step, there will not be an area penalty. However, there may be other disadvantages. Adding the jogs can add vias, increase wire lengths, and possibly increase undesirable adjacency if jogging the wire creates new undesirable adjacencies, either horizontal or vertical.

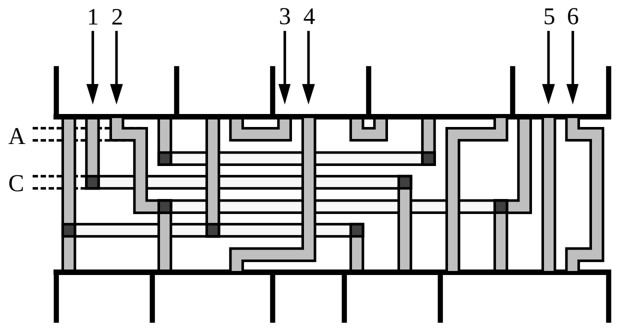

Figure 4.7: What channel looks like after removing unnecessary vias.

Fortunately, many of the added vias can be removed using a simple technique that Yoeli mentioned of moving all horizontal segments that aren't crossed by vertical segments to the vertical layer [Yoe91]. For the example in Figure 4.6, this would remove all the added vias, plus a few others, giving the channel shown in Figure 4.7. If a wire is only allowed to jog one column, any added vias are guaranteed to be removed. (It would not be able to jog if there were a vertical wire in the adjacent column.) Since tradeoffs between number of vias, wire lengths and vertical separation may be necessary, the number of columns a net is allowed to jog should be made a controllable parameter. If increases in routing time are not a consideration, explicit checks can be made to see if any added vias could be removed before a jog is introduced. The issue of increasing the length of adjacencies can be addressed by performing explicit checks to determine if jogging the wire should be allowed.

4.3 Experimental Procedure

To determine the effectiveness of modifying MCR to separate selective wire pairs, ten different placements of each of the 5 largest ISCAS combinational test circuits [BF85] were routed with and without the testability modifications. The placements were generated by TimberWolf [SSV85] using MCNC's Oasis cell library. In order to make the placements differ as much as possible, each circuit input and circuit output was randomly assigned to one of the four sides of the chip for each different placement. In addition, TimberWolf was started with a different seed for each placement.

For each placement, the modified router targeted differing percentages of randomly chosen net pairs. In order to avoid enumerating all n^2 combinations of adjacencies and then choosing $X\%$ of those, for each placement the nets were randomly ordered and assigned numbers. Then during routing, MCR tried to separate wire pairs whose associated numbers fell into the same bin (using the modulo operation). Experiments were run with the number of bins set to 100, 20, 10, 7, 5, 4, 3, 2, and 1. In the results that are presented later, these are converted to percentages for easier comprehension. A bin size of 100 corresponds to 1%, 20 to 5%, and so on.

To determine adjacencies after routing, Carafe [JF93], an inductive fault analysis tool, was used to extract the set of likely shorts from the cell interconnect. Carafe reads a description of a physical layout and determines which wires could be shorted together if a spot defect were to fall on the chip. For my experiments, the defect size is limited to being almost large enough to short three minimally spaced wires. Carafe reports the potential shorts and their critical area totals for this limit. Once again, critical area totals are useful as they not only reflect which shorts can occur but also their relative likelihood of occurrence. A short with twice the critical area of another short is twice as likely to occur. This allows the presentation of results as reductions in the probability of an undesirable short occurring.

4.4 Experiments and Results

This section discusses the modifications that were made to the weighting scheme and presents results on how successfully the different modifications performed. It gives emphasis to the trade-offs in adjacency reduction, routing area, and routing time.

4.4.1 Penalizing Nets 500 Weighting Units

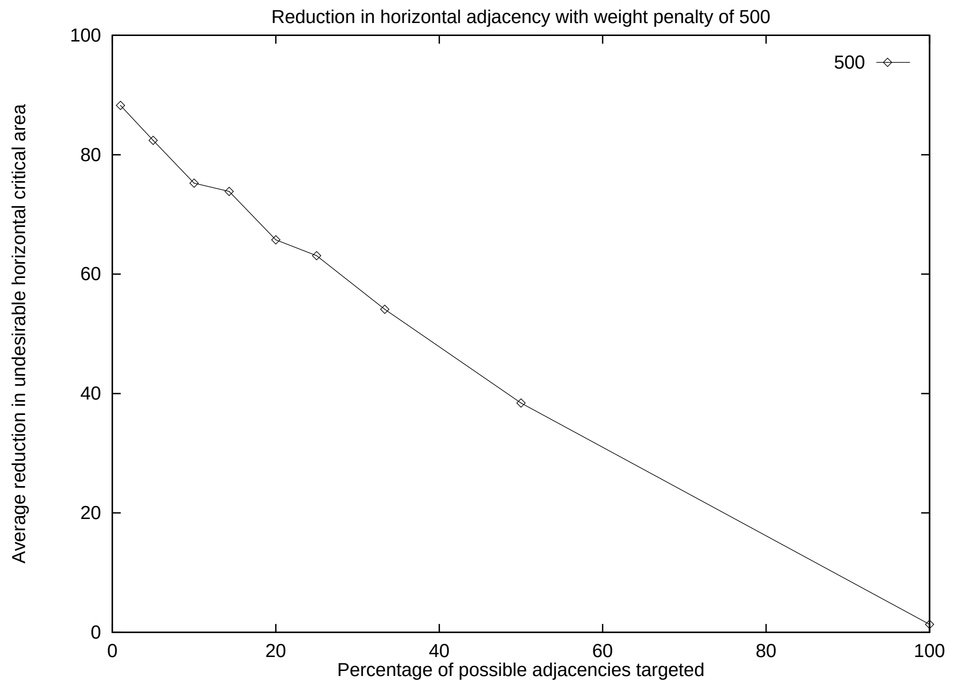

Figure 4.8: Reduction in undesirable horizontal adjacencies for a weight penalty of 500. Example: targeting 50% of adjacencies yielded a 40% reduction.

The first modification to MCR for separating horizontal wires was to penalize a net 500 weighting units for each undesirable net to which it would be placed adjacent. Figure 4.8 shows the percent reduction in undesirable horizontal adjacencies when differing percentages of potential adjacencies are targeted. From this figure it can be seen that a substantial reduction in horizontal adjacency can be achieved. The reduction shown is the average overall reduction in undesirable horizontal critical area totals. This means that when 1%

of the possible adjacencies were targeted as undesirable that there was an average decrease in critical area total of 88.3% for those 1% of adjacencies across all 50 layouts.

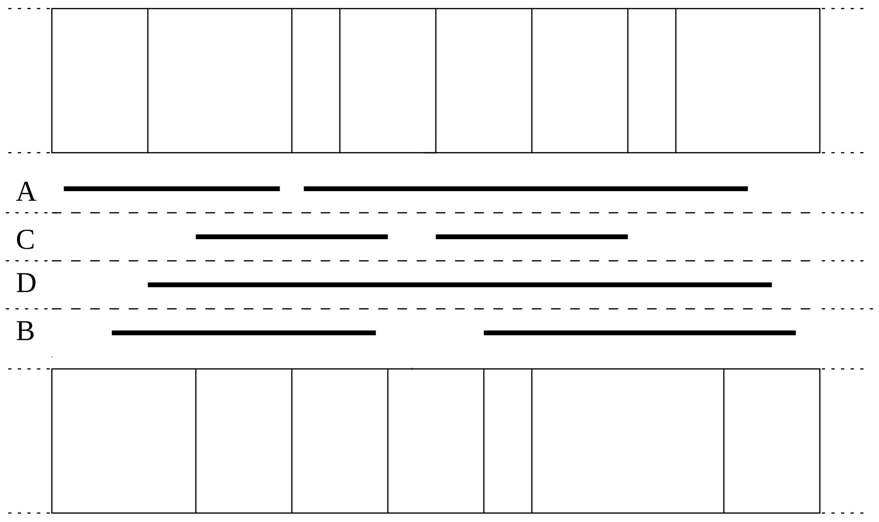

Figure 4.9: Since nets are placed in tracks, alternating between the top-most and bottom-most free track, when track D is being filled, the track assignment of a net can no longer be deferred to a later track, even if a short between the net and a net in track B or track C would be undesirable.

While the improvements are significant, they never reach a 100% reduction and they decrease as larger percentages of possible adjacencies are targeted. There are two reasons for this. The first is that nets are penalized only 500 weighting units, so separation is only encouraged, not enforced. This means that some nets may have a large enough weight that the penalty does not postpone their placement. It also means that when a large percentage of adjacencies are targeted that their weights do not move as much with respect to one another since nearly all of them will be penalized. The second reason is an artifact of the track assignment procedure. Since the track assignment procedure alternates between placing nets in upper and lower tracks and converges upon the final track, once the final track is reached, the placement of a net can no longer be delayed. As an example, consider Figure 4.9, when nets are being assigned to track D, their placement can no longer be deferred.

4.4.2 Improvement By Looking Ahead

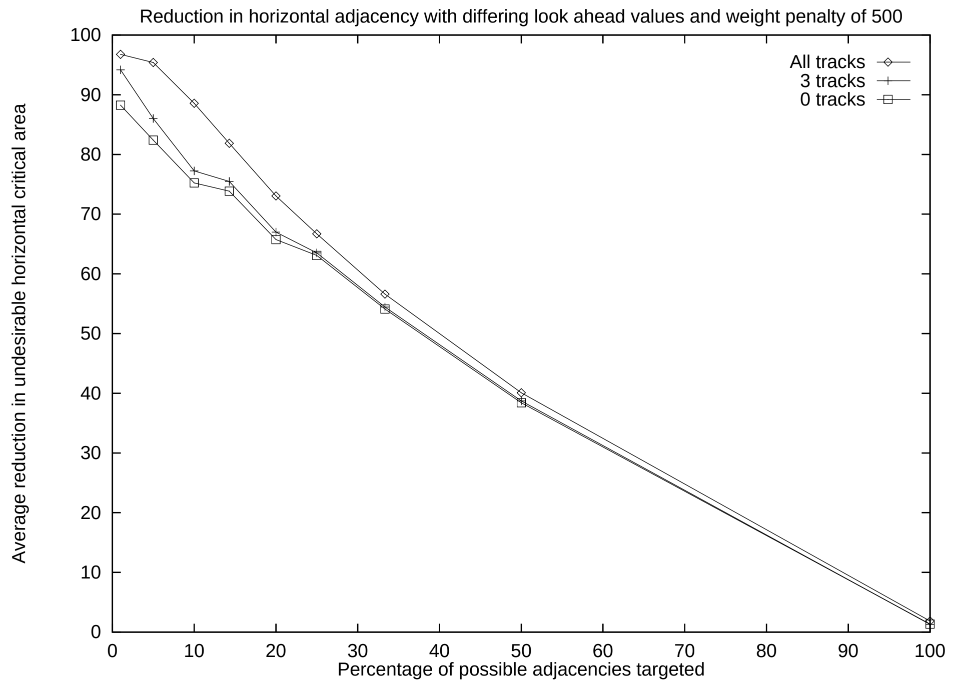

Figure 4.10: Reduction in undesirable horizontal adjacencies when look-ahead encouragement is used.

In order to address the final track issue, the early assignment of a net can be encouraged when there is at least one unassigned net with which it could create an undesirable adjacency. However, its early assignment should only be encouraged if doing so won't create an undesirable adjacency with a previously placed net. Because early assignment requires that adjacency checks be made against all unassigned nets, instead of just the nets assigned to the previous track, it incurs a larger time penalty. Figure 4.10 shows the improvements that can be gained by looking ahead 0, 3, and all tracks. In each case 50 weighting units are added to the weight of a net to encourage its early assignment if there are greater than 3 tracks remaining, at least 1 unassigned net to which it should not be placed adjacent, and no previously assigned nets to which it should not be placed adjacent.

Once 3 or fewer tracks remain, early assignment becomes more important. Looking at Figure 4.9 again, if Tracks B, C, and D have not been assigned nets, the only way to separate an unassigned net pair is to place one in Track B and the other in Track C. If either of them is placed in Track D, they will still be adjacent since Track D is adjacent to both B and C. Therefore, if exactly three tracks remain, 500 weighting units are added to encourage early assignment if the net should not be placed adjacent to any unassigned nets. If only 2 tracks remain, nets can no longer be separated since the nets will be adjacent regardless of ordering (C and D are always adjacent). When only 2 tracks remain, 500 units are added to a net's weight for each undesirable net it will be placed adjacent to in the third to the last filled track if it is placed in the last track. In Figure 4.9, when track C is being filled, a net will receive 500 weighting units for each undesirable adjacency that would occur between it and a net in track B if it were placed in track D.

As can be seen from Figure 4.10, differing amounts of look-ahead provide differing amounts of improvement. As would be expected, looking ahead more tracks produces better results. It is interesting to note that when large percentages of all adjacencies are targeted that encouraging early placement has a smaller affect than when small percentages of all adjacencies are targeted. This is not surprising. Since a larger percentage of adjacencies will be undesirable when a larger percentage of all adjacencies is targeted, early placement in not as likely to be encouraged since the encouragement factor is only added if assigning the net to a given track will not create an undesirable adjacency. Even when it is met, it is less likely to help since more undesirable adjacencies exist.

Because it adversely affects both time and area, the number of tracks of look-ahead should be a user controllable variable . Figure 4.11 shows the time penalty. From Figure 4.11 it can be seen that a look-ahead of 3 tracks has only a slightly larger time penalty than no look-ahead. The reason the time penalty for looking ahead for all tracks decreases is due to early assignment being encouraged as soon as at least one unassigned target net is found, at which point the process of checking for more undesirable adjacencies for the target net is halted. When only 1% of possible adjacencies are targeted, all or most of the unassigned nets may have to be scanned before an undesirable net is found, if one is found at all. When 100% of possible adjacencies are targeted, the first unassigned net checked will be found to

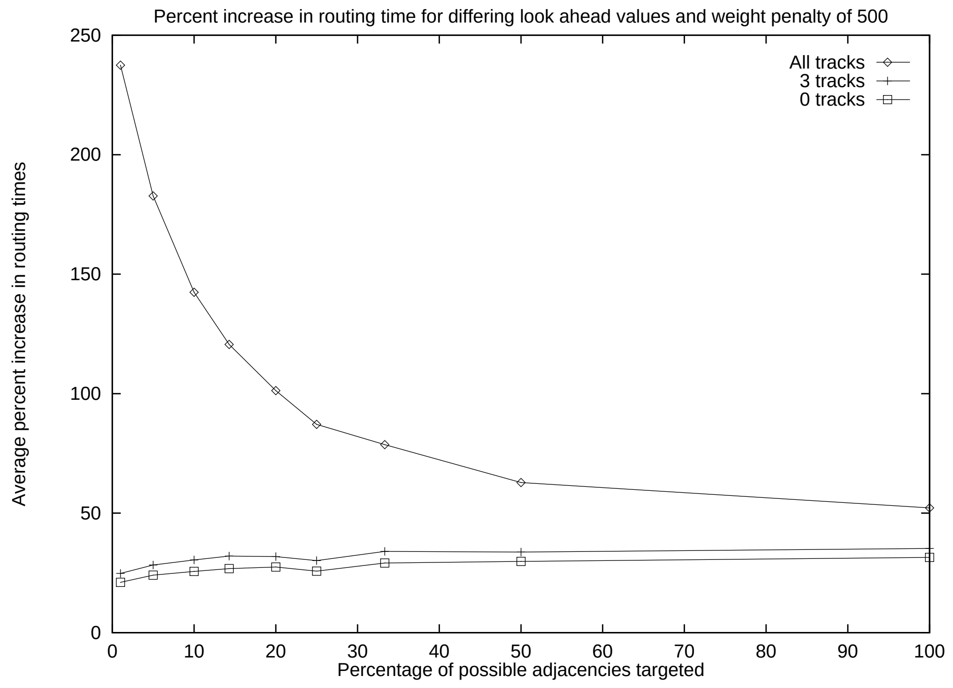

Figure 4.11: Increase in routing time when look-ahead encouragement is used.

be undesirable. As a result, fewer checks have to be performed when more adjacencies are targeted.

Figure 4.12 shows the area penalty for looking ahead. The area penalty doesn't vary as much as the time penalty. In general the area penalty is fairly low. The changes in area aren't as smooth as the changes in time for two reasons. First, changes in net assignment can cause perturbations in the number of tracks required to route a circuit. Different assignments can change the number of VCVs created and the router's ability to resolve them. Secondly, the changes in percentages are very small. The difference between the "0 tracks" and "3 tracks" points when targeting 33% of all possible shorts is .12% (.0012), which is a difference of 14 tracks out of roughly 12850.

The conclusion to draw from Figure 4.11 and Figure 4.12 is that look-ahead causes time and area penalties. When time efficiency is important, the amount of look-ahead should be kept low, perhaps to only a few tracks. When area efficiency is important, a slightly higher amount of look-ahead can be used. When separation is important, full look-ahead should

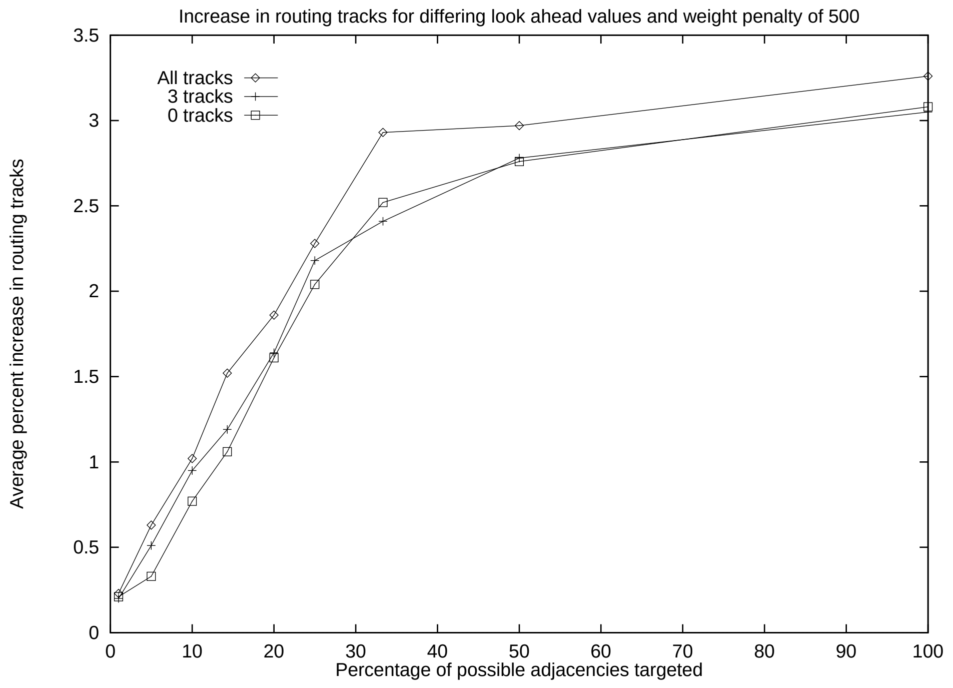

Figure 4.12: Increase in routing tracks when look-ahead encouragement is used.

be used. In all cases, the number of tracks of look-ahead should be configurable.

Efficiency of Looking Ahead

When look-ahead is being performed for a given track, for each unassigned net, a check is made against all other unassigned nets to see if their horizontal spans overlap. Overlap means that a column exists in which both nets have horizontal segments. If the nets overlap, a check is made to see if they should be allowed to be adjacent. If they should not be allowed to be adjacent, an encouragement weight is added and the algorithm short-circuits and moves on to the next unassigned net.

Let N be the number of nets, N_U be the number of unassigned nets, T be the number of tracks of look ahead, D the channel density, and C_A be the average number of columns a net spans. In the worst case $O(N_U^2)$ overlap checks are made. T is limited to $5D$, the track limit encoded into the router. The maximum number of tracks of look-ahead is therefore $O(T^2)$. In practice, the number of required tracks is closer to $O(T)$ unless large penalty weights force the addition of a large number of tracks. Since overlap checks are made for each track, the upper bound on the efficiency of looking ahead is $O(N^2T^2)$. Checking for overlap among unassigned net pairs can be made more efficient if N_U is greater than C. Instead of checking each unassigned net pair for overlap, each of the columns an unassigned net crosses can be checked to see if another unassigned net crosses it.

4.4.3 Variations in Penalty Weight

The next experiment measures the router's performance when the strength of the testability objective is varied. To accomplish this, the penalty weight was set to several different values. To avoid variations in improvement due to the final track problem, for these experiments the look-ahead factor was set to all tracks. Five different experiments were run with the penalty weight set to 50, 500, 5,000, 50,000, and 500,000,000. The way in which the penalty is handled is different for the 500,000,000 case, which will be referred to as the forced case. The weight is simply set to -500,000,000 if there is an undesirable adjacency. This is done in order to avoid integer overflow and is acceptable since the number is large enough (in magnitude) to always force separation. To put these numbers in perspective, let us consider the case where a net has four connections to the top side of a channel in columns where the density is 15. If MCR is assigning nets to the top track in the channel, there are no dense columns, and no VCVs will be created, the net would have a weight of 60 $(4 \cdot 15)$. If the net only had 2 connections, the weight would be 30. This means that a penalty of 50 is comparable to the size of net weights for small nets and may or may not force the weight of a net to go negative. To force the weight of a net to go negative (assuming DENSECOL has not been added to the weight), for all but the largest nets, a penalty of 500 should be sufficient. To override DENSECOL, at least 30,000 has to be subtracted for each dense column of a net.

When the experiments were first run for the forced case, the router failed. It was unable to successfully generate routes. This led to the discovery of an important aspect of the weighting system. If there is a cycle in the VCG of length C, the nets in that cycle may not be assigned to tracks until the last C tracks are reached. The reason for this is that the cycle can cause all the nets that are part of the cycle to have a negative weight until the last C tracks are reached and the effects of DENSECOL take over. If adjacency is not allowed, C nets will require at least 2C - 1 tracks. Since the first of the C nets may not be assigned to a track until C tracks remain, a successful assignment may never be found.

In order to force separation, a way of separating the nets without violating the integrity of the router had to be found. This meant that the vertical constraint cycles needed to be broken. To break the cycles, another stage was added to the weighting scheme. After the assignment of net weights, but before the penalty/encouragement stage, MCR scans all the unassigned nets to see if the largest weight is negative. If so, MCR sets the largest weight to 1 and adjusts all the other weights accordingly so that their previous relation to each other is maintained. This process ensures that at least one net in a cycle has a positive weight and can therefore be assigned to a track. The drawback of this method is that it can increase the number of tracks required by a large amount since it essentially doesn't break a cycle until all nets that are not part of the cycle have been assigned to tracks.

One final problem with the forced case occurred. Since nets are placed in an alternating manner, when the last track is reached, adjacency checks must be made against two tracks rather than just one. Looking at Figure 4.9 again, when assigning nets to Track C, only Track A needs to be examined to see if previously assigned nets could create an undesirable adjacency. However, when assigning nets to Track D, the nets already assigned to Tracks B and C need to be considered. Normally, when separation is only a weak objective, double adjacency checks don't need to be made; when separation is forced, they must be made.

Figure 4.13 shows what happens when the weight penalty is varied. Note that the difference between a penalty of 50 and 500 is much larger than the difference between a penalty of 500 and 5,000. The average weight of a net is responsible for this. A weight of 50 is not quite big enough to always have a large influence on the weight of a net whereas 500 and 5000 are. The amount of improvement diverges again at 50,000 because the penalty weight begins to be large enough to overcome the effect of DENSECOL, which is only 30,000.

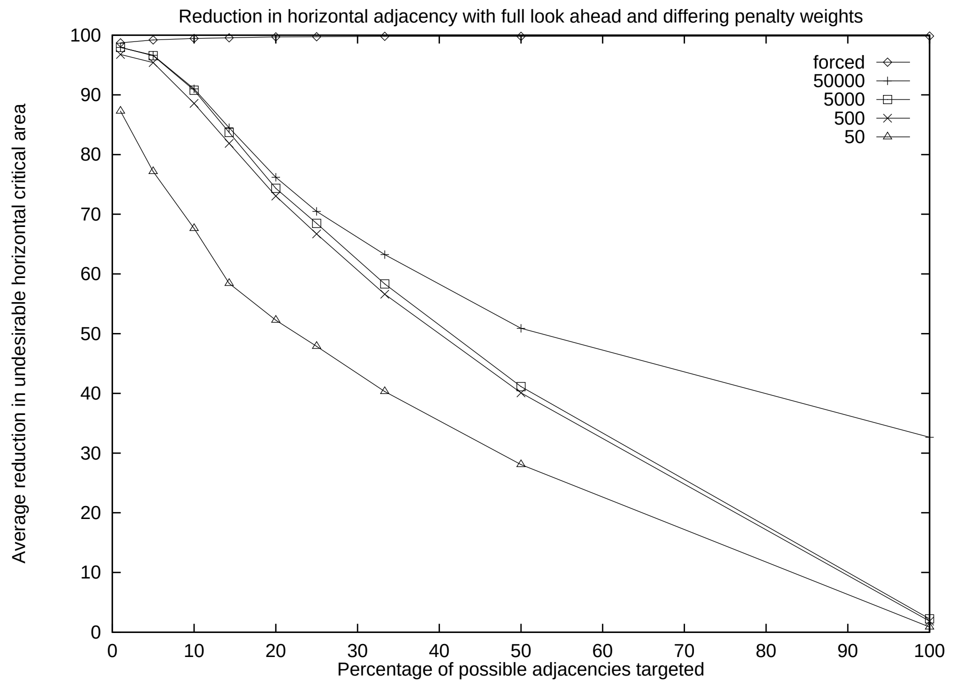

Figure 4.13: Decrease in undesirable horizontal adjacency with full look-ahead and differing weights.

The reason that 100% separation is not always achieved in the forced case is because of the maze router. The maze router may add new horizontal segments that create undesirable adjacencies. The maze router simply tries to generate a valid route and doesn't check for undesirable adjacencies since generating a correct route takes precedence over separation. If separation is critical, an improved maze router should be used.

Figure 4.14 shows the increase in routing tracks for differing penalty weights. The figure shows that there is very little increase until a penalty weight of 50,000 is reached, at which point the penalty weight can begin to force separation. The smaller amounts only indirectly cause an increase in routing tracks by creating more VCVs. By comparing Figure 4.13 and Figure 4.14 it can be seen that the increased improvement for a weight of 50,000 is gained at a substantial cost of increased area.

When considering the forced case, one would expect a maximum two-fold increase in area. If a channel can be routed in X tracks, one could force horizontal separation by simply

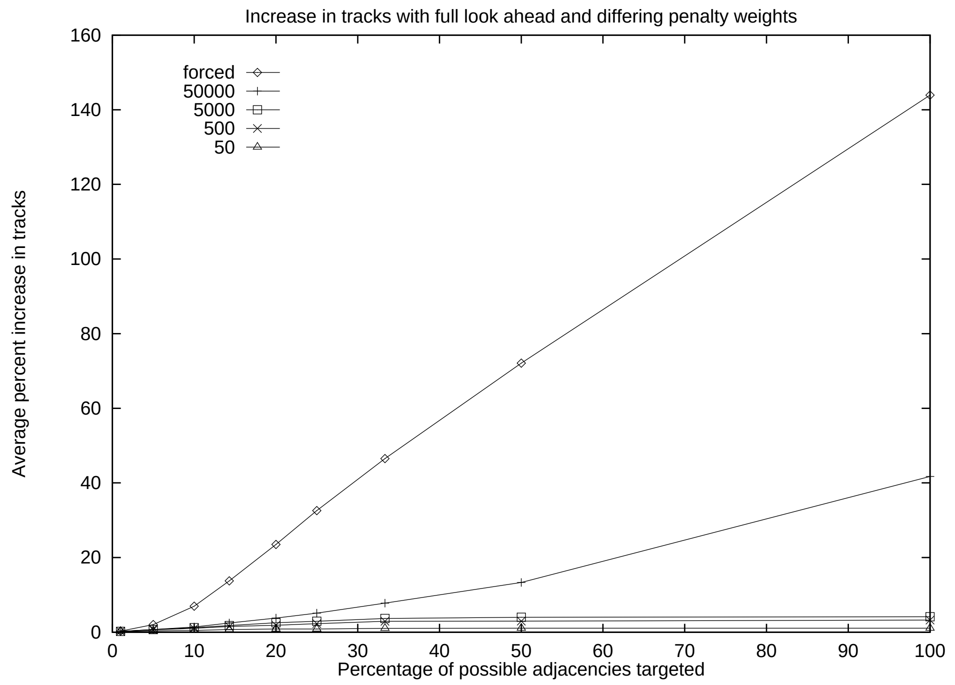

Figure 4.14: Increase in routing tracks with full look-ahead and differing penalty weights.

inserting a track between each pair of tracks for an increase in tracks of X - 1. (One would expect the channel size to roughly double.) Unfortunately, the problem with cycles in the vertical constraint graph causes a larger than expected increase.

Figure 4.15 shows the time penalties for the differing weights. Remember that since a full look-ahead is being used, the time penalty can decrease as the number of targeted adjacencies increases. The reason that the time grows for the forced and 50,000 cases is because of the addition of tracks. Since the router only adds a track at a time, and reassigns all nets when it does so, it has many times more work than it would normally have because of the addition of tracks.

So far, only results for reductions in horizontal adjacency have been presented. Electrical shorts can also be caused by vertical adjacencies and cross-over adjacencies. A cross-over occurs when a metal1 and a metal2 net cross each other. By deferring the placement of a net, we would expect that vertical wires would be made longer thus increasing the amount of

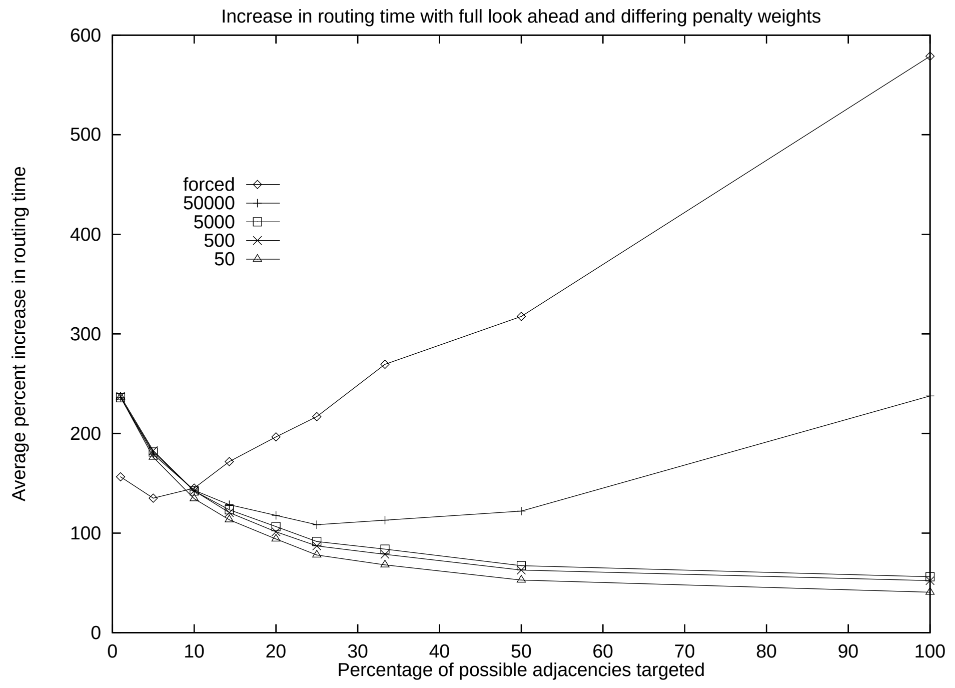

Figure 4.15: Increase in run time with full look-ahead and varying penalty weights.

undesirable vertical and cross-over adjacencies. When the number of tracks is increased, we would again expect longer vertical net lengths and larger increases in vertical adjacencies. Figure 4.16 shows that horizontal adjacencies account for the largest fraction, followed next by vertical adjacencies, and finally by cross-over adjacencies in routes generated by the unmodified version of MCR.

This distribution of adjacencies is advantageous. Since horizontal adjacencies are the easiest to separate, it is encouraging that they are the most plentiful. Since cross-over adjacencies are the hardest to fix, it is encouraging that they are the least plentiful.

Figure 4.17 shows the increase in vertical adjacencies due to the targeting of horizontal adjacencies. An interesting note is that when 1% of the adjacencies are targeted, a reduction in total vertical adjacency is experienced. The increases in the number of routing tracks cause fairly large increases in vertical adjacency as can be seen by looking at the 50,000 and forced case.

Since a decrease in horizontal adjacency is achieved at the expense of an increase in

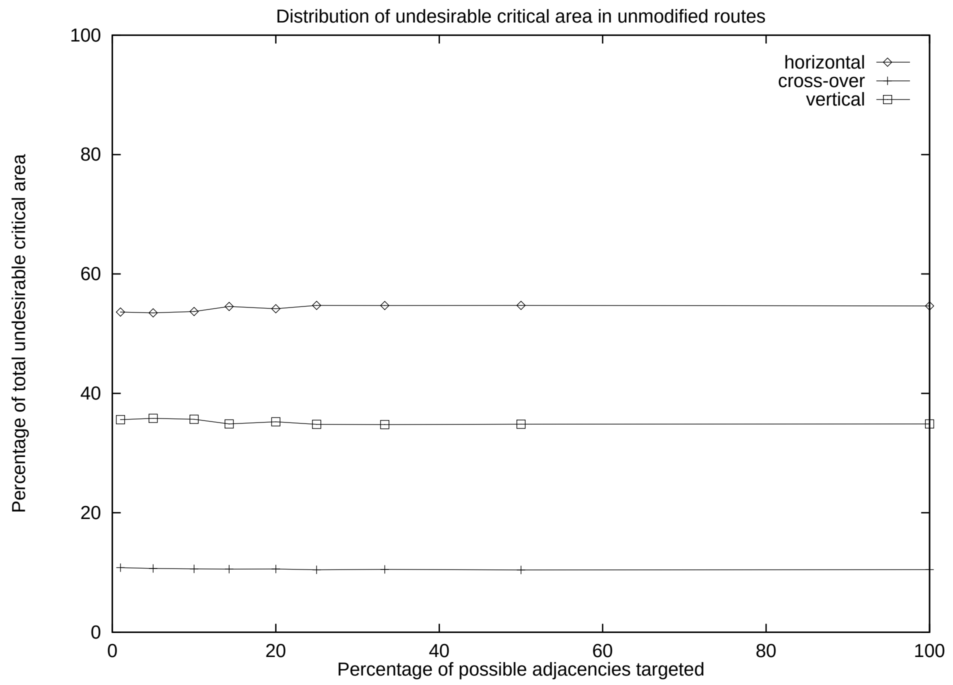

Figure 4.16: Distribution of undesirable adjacencies for differing target percentages.

vertical adjacency for large target percentages, one would expect the decrease in overall undesirable adjacency to be lower than the decrease in horizontal adjacency. Since horizontal adjacencies account for a larger percentage of overall adjacencies, one would hope that an overall decrease can be achieved. Figure 4.18 shows that this is indeed true until 100% of adjacencies are targeted. When 100% of adjacencies are targeted, the penalty weights that never force separation actually increase the amount of undesirable adjacency. This is because vertical adjacencies and cross-over adjacencies are increased without a reduction in horizontal adjacency. However, in most cases, an overall improvement is experienced, particularly when 50% or less of the adjacencies are targeted.

4.4.4 Vertical Adjacency

Since vertical adjacency contributes to overall adjacency and since it increases when methods for reducing horizontal adjacency are used, it is desirable to reduce vertical adja-

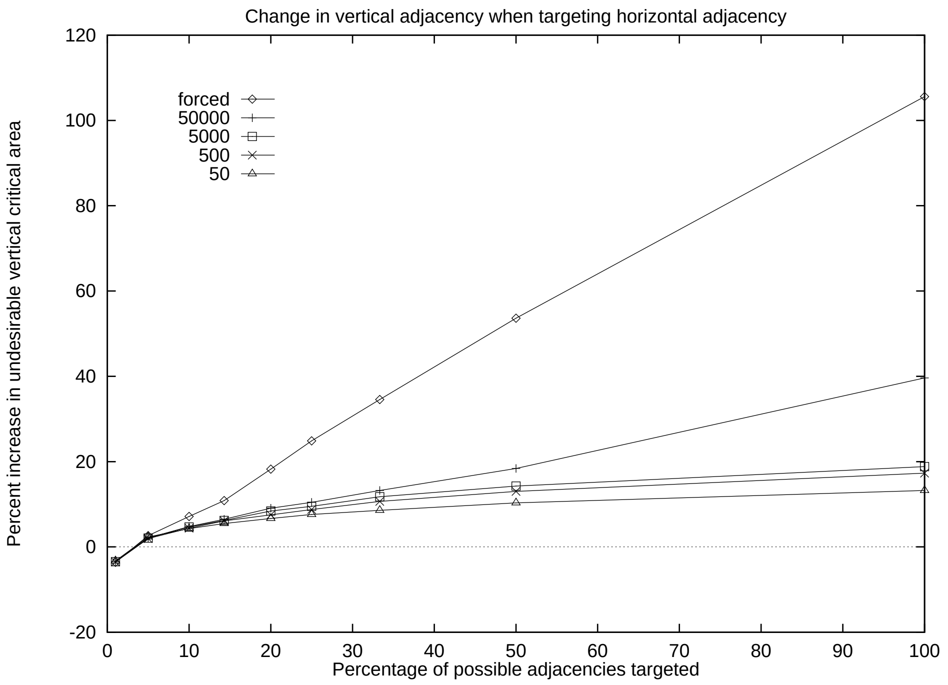

Figure 4.17: Changes in undesirable vertical adjacencies when targeting undesirable horizontal adjacencies with differing weights and full look-ahead.

cencies. Reductions in vertical adjacencies are not as easily achieved.

MCR reduces vertical adjacencies as a post-processing step. In order to ensure that any added vias can be removed, only jogs of a single column are allowed. By adding a jog of only a single column, any introduced vias can be removed by moving the horizontal section of the jog to the vertical layer. This also ensures that new horizontal adjacencies will not be created since the jog is solely in the vertical layer. This is very similar to maze1 (Figure 3.10) from YACR2, except that MCR first adds the vias and then removes them during a via-removal step.

To ensure that jogs do not increase vertical adjacencies, checks are made to see if jogging a net will cause an undesirable adjacency of greater length than the adjacency the jog it is trying to avoid.

Figure 4.19 shows the decrease in vertical adjacency that can be obtained from using the above method. Overall, the method does not perform as well as techniques for reducing

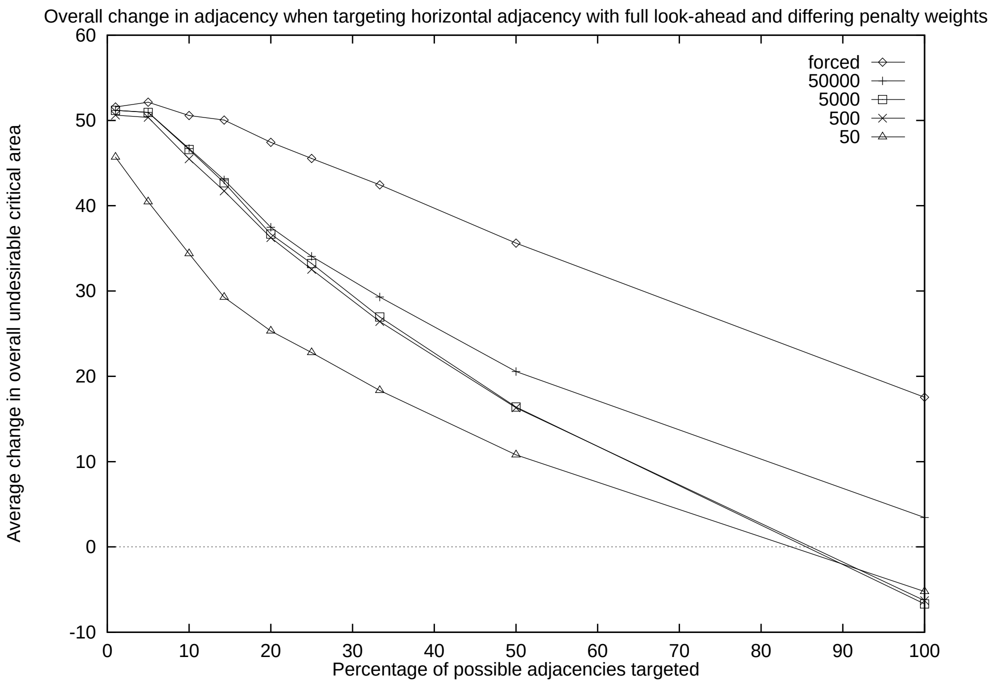

Figure 4.18: Overall decrease in undesirable adjacency when all types of adjacency are considered with differing penalty weights and full look-ahead.

horizontal adjacency. There are many reasons that the techniques do not perform as well. The first is that vertical adjacencies simply do not allow as much room for improvement. Because reductions in adjacency can only be achieved in the area between where the nets enter the channel and before the shorter net has reached its assigned track, not much can be done during routing. However, placement algorithms, as mentioned by Acken [Ack88], might be able to reduce undesirable vertical adjacencies. The second reason is the restriction that is placed on only allowing a net to jog a single column. Allowing a net to jog for more than one column would allow more separations to be made but would raise the likelihood of increases in vias and horizontal adjacencies. The third reason is that MCR only targets vertical adjacencies between nets that start on the same side of the channel. Nets can still be adjacent vertically if they are in adjacent columns, but enter the channel on opposite sides.

The advantage of the techniques for reducing vertical adjacencies is that they do not

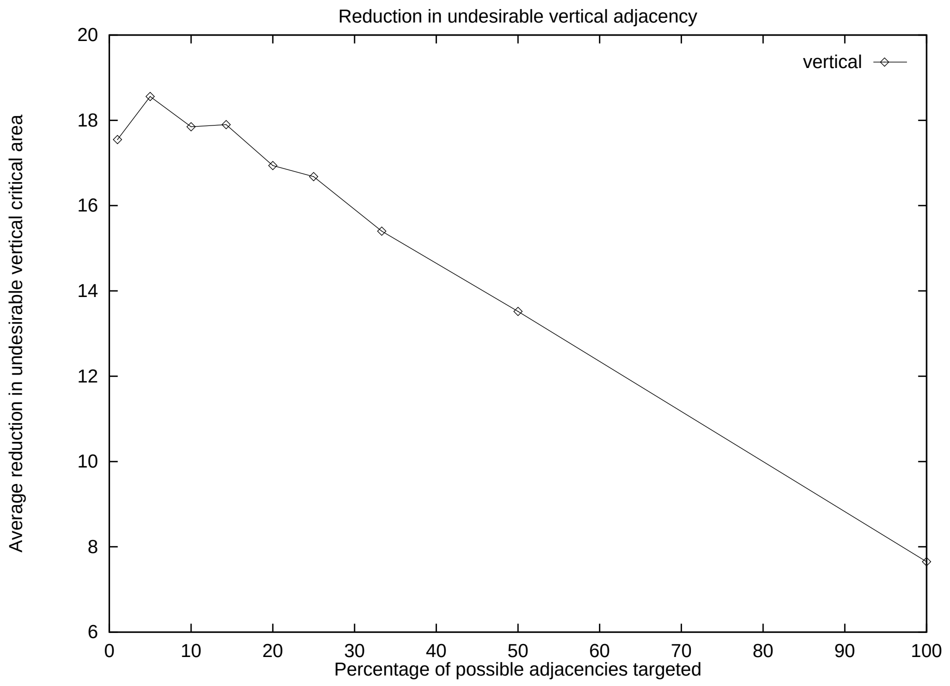

Figure 4.19: Decrease in vertical adjacency.

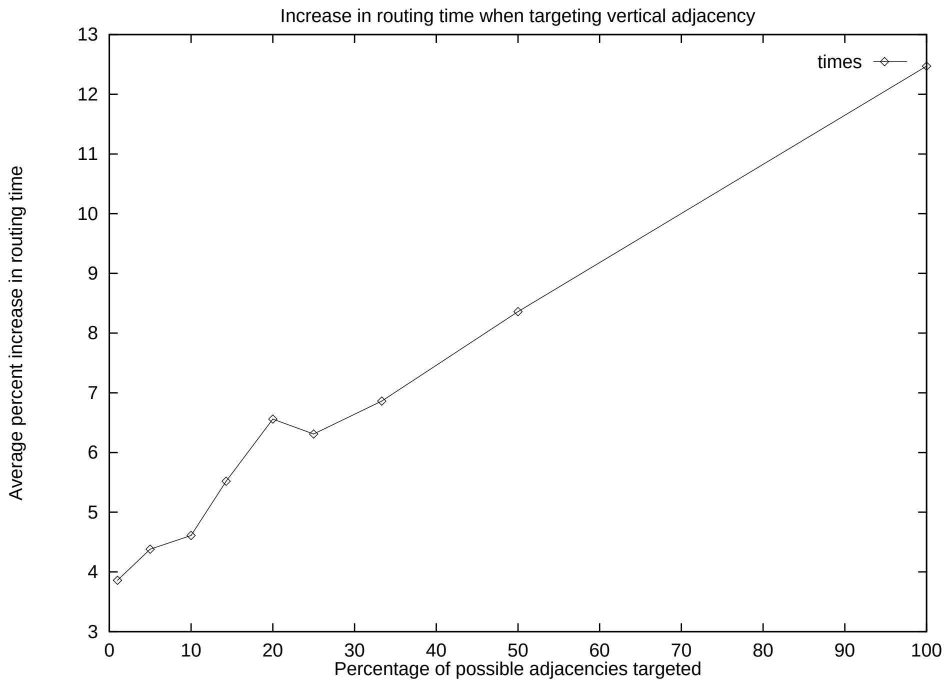

Figure 4.20: Increase in routing time when targeting vertical adjacencies.

increase the channel area. As Figure 4.20 shows, they also have little impact on routing times. The anomalies in the graph can be attributed to error in the calculation of the running times. The original running time of the router for all 50 placements was approximately 210 CPU seconds, so a difference of .5% is on the order of a second.

4.4.5 Combined Reductions

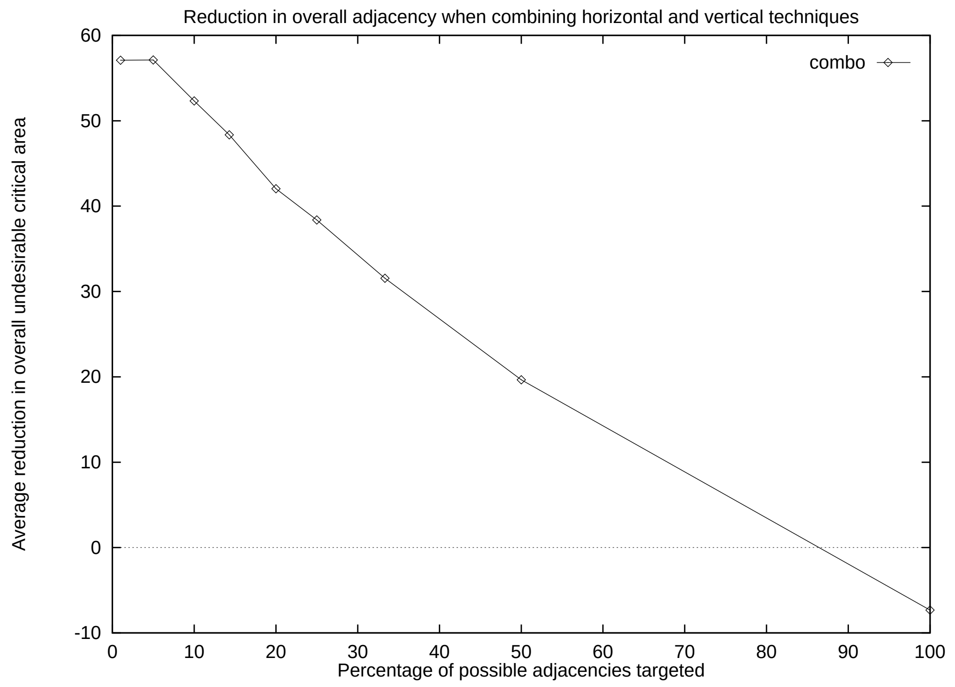

Figure 4.21: Overall decrease in undesirable adjacency when horizontal and vertical techniques are combined. The weight penalty is set to 5000 and a full look-ahead is used.

Since the primary goal is to reduce undesirable adjacencies, MCR combines the horizontal and vertical techniques. Since previous experiments found a weight penalty of 5,000 to provide a good trade off between reduction in adjacency and increase in routing tracks, it was chosen as the penalty weight. Figure 4.21 shows the overall reduction in adjacency when the two techniques are combined and a full look-ahead is used. The results are very encouraging as they show that significant improvements can be made with little penalty.

As a point of reference, let us consider the case when 10% of all possible adjacencies are targeted. The modified router required 437 cpu seconds to route all 50 designs on a a DEC Alpha 3000/600. The original version of the router required only 210 cpu seconds to route all 50 designs. The average increase in routing tracks was 1.2%. This means that with excellent running times and a 1% increase in area that a 52% decrease in undesirable adjacencies was achieved.

4.5 Conclusions

This chapter showed how a router can be created that varies the strength with which it enforces separation. This was done by penalizing adjacency with a tunable weighting scheme and encouraging early placement with a tunable look-ahead factor. It was shown that significant reductions in adjacency could be obtained with reasonable time and area penalties. It was also shown that trade-offs in reduction, time, and area could be made by changing the size of the penalty weight and the number of tracks of look-ahead.

5. Reductions in Undetectable Shorts

This chapter describes how the techniques of the previous chapter can be used to make circuits more testable. In particular, the chapter focuses on reducing the occurrence of shorts that are undetectable under the static-voltage testing methodology and potentially undetected under the pseudo-exhaustive segmentation testing methodology.

5.1 Static-Voltage Testing

The following definitions apply to this chapter. A *short* occurs when two or more lines are unintentionally connected during fabrication. The term short, rather than the term *bridge fault*, is used to distinguish between the physical manifestation of a defect and the functional description that is often implied by the term bridge fault. The shorts that are considered are interconnect shorts—shorts between two lines that are gate outputs or circuit inputs. If the graph representing the unfaulted circuit contains a directed path between the two shorted lines, the short is a *feedback short*. If no such path exists, the short is a *non-feedback short*. Only non-feedback shorts are considered. A short is *detectable* if there exists an input vector that will cause at least one of the circuit outputs to have a different logic value than it would in the defect free circuit. Likewise, a short is *undetectable* if no input vector can be found that distinguishes the faulty and defect-free circuits. Some of the shorts categorized as undetectable may be detectable, but that the test pattern generator aborted. Some of them may also be detectable with I_{DDQ} [Ack83] or delay testing. However, few companies do systematic I_{DDQ} and delay fault test pattern generation. Furthermore, the techniques shown here are applicable to shorts that are undetectable under I_{DDQ} or delay testing methodologies, with only a change in the separation criteria.

5.1.1 Motivation

Even if an undetectable short does not change the logic function of a circuit, it can change other functional aspects. An undetected short can cause reliability problems by creating unexpected delay, noise, power consumption, and heat. These may introduce intermittent errors or eventually cause a catastrophic failure. The elimination of undetectable shorts would provide three benefits:

1. It would increase the long-term reliability of a circuit by reducing power consumption, heat, and average current density caused by undetectable shorts.
2. It would reduce the number of intermittent errors by eliminating electrical noise and circuit delay caused by undetectable shorts.
3. It would reduce fault simulation and test generation costs by decreasing the amount of time spent proving shorts undetectable.

As mentioned before, one should be careful when deciding how a goal is best achieved. When the goal is reducing the likelihood of undetectable shorts, care should be taken to avoid excessive run-times. A straight-forward way of eliminating undetectable shorts, is an iterative process that performs ATPG to determine which shorts are undetectable and then re-routing the circuit to avoid the shorts found to be undetectable. This is expensive. If the goal of reduction in undetectable shorts is more important than the goal of router performance, this method is fine and should perform quite well. However, if router execution times are more important, something else must be done.

Ideally, a method for predicting a large number of undetectable shorts could be found that is not as costly as performing full ATPG. Researchers have previously found characteristics that predict the difficulty of detecting a short, but had not found easily obtainable characteristics that distinguish undetectable shorts from hard-to-test shorts. Kapur *et al.* [KBRM91] studied the relationship of controllability and observability to detectability. Teixeira *et al.* [TTA+91] and Saraiva *et al.* [SCS+92] divided faults into categories based on their resistance to stuck at tests and suggested using the "hardness" of a category to drive DFT efforts.

Although MCR could be used with these probabilistic methods, the percentage of targeted adjacencies would be high. By targeting a large class of shorts, a decrease in undetectable shorts might be achieved, but a large area and run time penalty would be incurred. By targeting actual undetectable shorts, a smaller number of potential shorts will be targeted and the router will do a better job with less of an area and run time penalty.

Undetectable shorts were analyzed to find local characteristics that can be used to efficiently identify a large number of undetectable shorts. By looking for these characteristics, a router could reduce the likelihood of occurrence of undetectable shorts without requiring as much time as full ATPG.

5.1.2 Methodology

In order to find such characteristics, A study of the shorts that were likely to occur in circuit layouts was performed. The shorts studied were all the non-feedback shorts that are physically realizable by small spots of metal in the metal layers of cell interconnect or small spot defects in the oxide separating these layers. Shorts in the metal layers are the most common fault type in many CMOS technologies [MTCC87].

Carafe [JF93] was used to extract the set of possible shorts from ten different layouts of each of the five largest ISCAS [BF85] combinational test circuits. These MCNC standard cell layouts were generated by TimberWolf [SSV85], a placement and global routing package, and the unmodified version of MCR. Once the likely shorts were extracted from the circuit layouts, the Nemesis [Lar92] ATPG system was used to generate tests for each short [FL91].

5.1.3 Undetectable Shorts

First this section covers the characteristics of undetectable shorts in general. Then it covers the characteristics that efficiently identify some, but not all, undetectable shorts. For a short to be detectable, it must be both excitable and propagatable. One must first be able to set the shorted lines to different values in the short-free circuit, *excitation.* One must then be able to get the resulting discrepancy to a circuit output, *propagation.* Therefore, for a short to be undetectable, it must either exhibit the characteristic of being non-excitable, non-propagatable, or both.

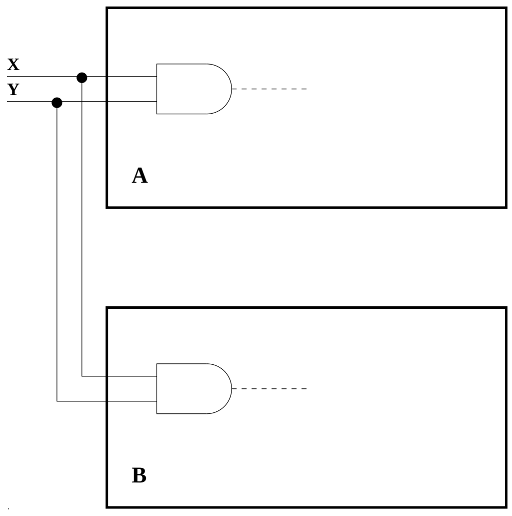

Figure 5.1: Example of how unintentional redundancy may be introduced.

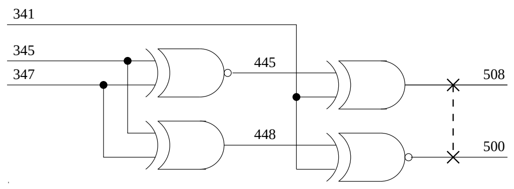

Figure 5.2: Example of redundant circuitry taken from the 1908.

For a bridge between two lines to be non-excitable, each node must implement the same logic function since every combination of circuit inputs places the same value on both lines. Hence, one of the two lines is redundant and would appear to be unnecessary. Unfortunately it is easy to unintentionally introduce this type of redundancy. For instance, consider Figure 5.1. If a circuit designer is using a hierarchical design style, he may have two blocks, A and B, that both require signals X and Y. While designing block A the designer may determine that he needs to **AND** X and Y for use within block A. Later, while designing block B he may determine that he needs to **AND** X and Y for use within block B. The designer may not notice that the **AND** could have been performed at the higher level.

Redundancies can also be introduced when two equivalent lines are generated by a combination of gates—the equivalence may simply be overlooked. An example taken from the 1908 benchmark circuit, Figure 5.2, illustrates this. In the example, lines 500 and 508

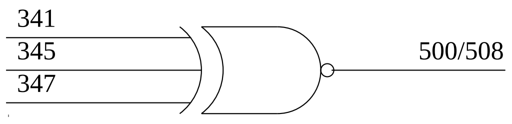

Figure 5.3: Single gate that performs the same function as the logic in Figure 2.

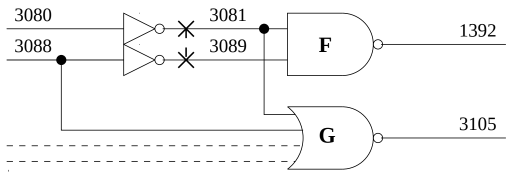

Figure 5.4: Example taken from 2670 that shows two types of masking that can occur.

are equivalent. Lines 341, 345, and 347 do not go to any other gates so the entire circuit in Figure 5.2 could be replaced by the circuit in Figure 5.3.

Although 500 and 508 may be logically equivalent, the proper timing of the circuit may require that they be separate. Line 500 might have a greater load than line 508 and line 508 may be on a critical path. If these two lines are unintentionally shorted, the load will be shared and the delay on the critical path will be greater. This can be detrimental as the corresponding alteration in delay may cause the circuit to fail for some input sequences. Unless this circuit path is checked with an appropriate delay test, the short will not be detected.

The second category of undetectable non-feedback shorts, non-propagatable shorts, is more subtle. All inputs that stimulate a non-propagatable short also mask the generated error.

Fault masking can occur when the short performs the same function as a portion of its propagation path. Consider Figure 5.4, an example taken from the 2670, in which the shorted lines 3081 and 3089 are driven by CMOS inverters whose n-channel transistors are stronger than their p-channel transistors as is typical in CMOS technologies. This results in a logic 0 being present on both lines (wired-AND) when there is a conflict between the two

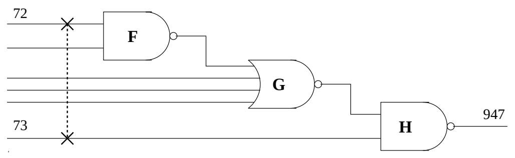

Figure 5.5: Additional example of masking through function equivalence.

driving gates[1]. Therefore, even when the short is excited, and 3081 and 3089 have different values, the output of Gate F will not show a discrepancy since a 0 on either 3081 or 3089 automatically forces F's output to be a logic 1. The discrepancy cannot be propagated through 1392 because Gate F performs the same logic function (before the inversion) as the short. In general, an error can be masked whenever a gate on the immediate propagation path from the short performs the same function as the short. A short that is masked in this manner is referred to as a function-masked short. Another example of a function-masked short is taken from the 880 benchmark circuit and is shown in Figure 5.5. In this figure, the undetectable short involving lines 72 and 73 acts as a wired-AND. Because the combination of F, G, and H ANDs 72 with 73, for the input combinations that excite the short, the short is masked.

In addition to function-masked shorts, excitation-masked shorts can occur. Excitation masking occurs when the values placed on lines to excite the short mask the propagation of the short. In Figure 5.4, gate G provides an example of this type of masking for a short between lines 3081 and 3089, since the short cannot be propagated through gate F. For a discrepancy to be propagated through G, and out line 3105, a discrepancy must be placed on line 3081. Placing a discrepancy on line 3081 requires that it be a 1 and line 3089 be a 0. (The fault acts a wired-AND.) However, for 3089 to be a 0, line 3088 must be a 1 which forces the output of G to 0 and, in turn, blocks the propagation of a discrepancy on line 3081 through G.

Non-propagatable shorts are undesirable for the same reasons as non-excitable shorts:

[1]Although wired-AND/wired-OR models do not accurately represent all shorts, of the 308 short types we encountered, 56 exhibited wired-AND behavior. This is mainly attributable to the nmos transistors being stronger.

Circuit	Total Undetectables	Total Non-Feedback Shorts	Total Shorts
c2670	245	70724	86821
c3540	193	70688	97119
c5315	144	178822	212726
c6288	0	67948	134995
c7552	568	199038	243068
Total	1150	587220	774729

Table 5.1: Distribution of non-feedback shorts by circuit.

they may add circuit delay and increase ATPG time. In addition, nonpropagatable shorts can also cause current consumption to exceed rated values leading to reduced reliability and a violation of specifications since the shorted lines can be set to different values.

5.1.4 Locality of Undetectability

Table 5.1 shows the total number of undetectable non-feedback shorts for ten placements of each of the five largest circuits. Approximately 0.2% of the total non-feedback shorts are undetectable. This is promising since the router performs well when the percentage of targeted shorts is low.

In order to study non-excitable shorts and non-propagatable shorts separately, the undetectable shorts were subdivided into the two categories. This was done using the I_{DDQ} capabilities of Nemesis [FTL90], to see which shorts could not be excited. This introduces a slight inaccuracy in that a short that is both non-excitable and non-propagatable will be listed as only non-excitable.

When examining the two types of undetectable shorts, an important discovery was made. An examination of the undetectable shorts revealed that many of them can be determined to be undetectable by considering only local information, circuit information that is close to the short site. These undetectable shorts are *locally determinable* (LD) as undetectable.

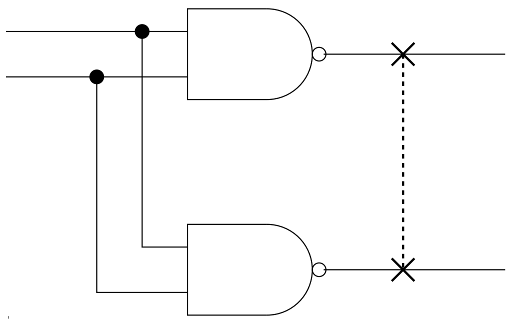

Figure 5.6: Example of a non-excitable LD short.

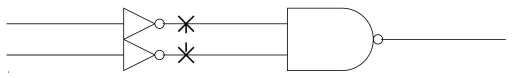

Figure 5.7: Example of a non-propagatable LD short.

Two types of LD shorts can be found by considering only the site of the short and logic that is no more than one level away. From this point on, LD will refer to the two types of shorts mentioned below and not the entire LD class[2].

The first type of LD short is a non-excitable short that can be identified by the shorted lines belonging to gates of the same type that share the same inputs, as in Figure 5.6. A short between these lines is clearly undetectable since the lines can never be set to different values. The second type of LD short is a non-propagatable short that can be identified by the shorted lines feeding only into a single gate that performs the same logic function as the short, as in Figure 5.7[3].

Table 5.2 shows the overall distribution of undetectable non-feedback shorts. From the table it can be seen that approximately 66% of the non-excitable shorts are LD shorts and that approximately 34% of the non-propagatable shorts are LD. The number of non-propagatable shorts is much higher than the number of non-excitable shorts. As a percentage of all undetectable shorts, LD shorts account for roughly 42%. This is encouraging

[2]In previous work [MF94b, MF94a] this sub-class was termed the LD_0 class.

[3]This example assumes that an inverter-inverter short acts as a wired-AND as it does in the standard cell library we used, MCNC SCMOS.

Circuit	Non-excitable		Non-propagatable	
	LD	Non-LD	LD	Non-LD
c2670	17	11	56	161
c3540	127	2	15	49
c5315	17	26	28	73
c6288	0	0	0	0
c7552	28	58	196	286
Total	189	97	295	569

Table 5.2: Distribution of undetectable, non-feedback shorts.

since the router should be able to reduce the probability of a large number of undetectable shorts occuring without incurring too large of a time or area penalty. As a note of interest, the LD class was able to predict some of the shorts for which the ATPG aborted.

5.1.5 Locality as an Objective

The idea of locality is interesting. A test pattern generator can be thought of as using all N levels of circuit information whereas a program for identifying LD shorts can be thought of as using only two levels of circuit information, one gate level forward and one gate level backward. Let us now consider how this relates to varying strength objectives. If identifying all the undetectable shorts in a circuit is necessary, test pattern generation can be used. However, this can be very expensive computationally. If identifying a subset of the undetectable shorts in a circuit is sufficient, a program for identifying LD shorts can be used since it is more computationally efficient. To allow greater trade-offs an undetectable short identifier can make the number of levels of circuit information it uses a variable. In this manner, the trade-off between identification and computation (run-time) could be tunable. Unfortunately, this endeavor is beyond the scope of this dissertation. However, techniques similar to those of Isern and Figueras [IF94] might be suitable for tunable multi-level redundancy identification.

Circuit	Decrease in Shorts	Decrease in CA
c2670	0.86%	10.37%
c3540	-0.66%	40.52%
c5315	0.24%	18.20%
c7552	1.56%	21.77%
Average	0.50%	22.72%
Overall	0.87%	24.8%

Table 5.3: Overall decrease in undesirable shorts and undesirable critical area totals with full look-ahead and weight penalty of 5000.

5.1.6 Results

In order to evaluate how well the router could reduce the likelihood of occurrence of undetectable shorts, an LD predictor was incorporated into it and the circuits were re-routed. The experiments were ran with a weight penalty of 5000, full look-ahead, and a targeting of both horizontal and vertical shorts. Table 5.3 shows how the router performed for 4 of the 5 different test circuits. The circuit c6288 is not included as it did not contain any undetectable shorts. The totals shown are the average reduction in undesirable shorts and the average reduction in critical area totals (horizontal, vertical, and cross-over combined.) Since the LD class predicts approximately 42% of the shorts, and the router is able to achieve a approximately a 57% overall reduction when targeting 1% of all possible shorts, one would expect close to a 24% overall reduction in critical area totals. From the overall total, this is indeed seen to be the case. The reason the average total is lower is that it represents average improvement and circuits with small improvements drag the average down. It is interesting to note that the overall reduction in shorts, .87%, is much smaller than the overall reduction in critical area totals, 24.86%. This means that while less than 1% of the shorts were eliminated that the probability of an undetectable short occurring was reduced by close to 25%. The reason more shorts were not eliminated is because vertical adjacencies can not be completely fixed and cross-over adjacencies, which are small in terms

Circuit	Decrease in CA
c2670	32.59%
c3540	54.35%
c5315	37.47%
c7552	54.75%
Average	44.79%
Overall	57.63%

Table 5.4: Overall reduction in LD critical area totals with full look-ahead and weight penalty of 5000.

Circuit	Decrease in Shorts	Decrease in CA
c2670	1.47%	6.09%
c3540	2.70%	41.86%
c5315	1.44%	18.71%
c7552	1.77%	26.29%
Average	1.84%	23.24%
Overall	1.83%	26.67%

Table 5.5: Overall decrease in undesirable shorts and undesirable critical area totals with look-ahead of 3 and weight penalty of 5000.

of CA, are not targeted.

Table 5.4 shows how the router performed considering only the LD shorts. Once again the overall totals show the expected performance. This is equal to the 57% percent improvement predicted in Figure 4.21 when 1% of all adjacencies are targeted.

Since the time penalty is at its worst for a full look-ahead when a small percentage of the total shorts are targeted, and since the number of undetectable shorts represent a small percentage, a full look-ahead might be too strong for the LD application. Since a look-ahead of 3 tracks performs very well for small target percentages, the LD application

Circuit	Decrease in CA
c2670	28.33%
c3540	55.43%
c5315	39.30%
c7552	55.51%
Average	44.64%
Overall	57.16%

Table 5.6: Overall reduction in LD critical area totals with look-ahead of 3 and weight penalty of 5000.

was re-run using a look-ahead of 3 tracks with the expectation that it might provide a similar improvement with a much smaller time penalty. Table 5.5 shows the results of using a look-ahead of 3. Surprisingly, it performed slightly better than when a full look-ahead is used. It appears that a full look-ahead perturbs the overall route more by encouraging early placement sooner than is necessary when the targeted number of shorts is small. Since the LD shorts do not account for all undetectable shorts, some of the new adjacencies are undesirable. Since fewer new adjacencies were created, a higher percentage of shorts were eliminated.

Table 5.6 shows how a look-ahead of 3 performs with respect to LD shorts. It performs almost as well as a look-ahead of N. If undetectable shorts accounted for a larger percentage of all shorts, this would probably not be the case.

When LD shorts are targeted and a look-ahead of N is used, the average increase in routing time over the unmodified version of the router was 353%. When LD shorts are targeted and a look-ahead of 3 is used, the average increase in routing time was only 48%. Neither look-ahead value changed the number of tracks required to route the circuit. In both cases the routing area required was the same as in the unmodified case. Since the improvement gained for the full look-ahead is slightly smaller, and the amount of time required is much higher than when a look-ahead of 3 is used, for undetectable shorts and

Circuit	Decrease in Shorts	Decrease in CA
c2670	4.16%	38.51%
c3540	2.70%	54.32%
c5315	4.88%	50.08%
c7552	3.78%	44.99%
Overall	3.88%	46.97%
Gross	3.91%	47.19%

Table 5.7: Overall decrease in undetectable shorts and undetectable critical area totals with full look-ahead, a weight penalty of 5000, and ATPG prediction.

for small percentages of undesirable adjacencies, a look-ahead of 3 is recommended. For the LD application, an average 24% decrease in the chance of an undetectable short occuring can be obtained with no increase in area and only a 48% increase in routing time. The total time required to route all 50 layouts was 324 CPU seconds on a DEC alpha. If the number of undetectable shorts as a percentage of all shorts increased sharply, the amount of look-ahead should probably also increase.

To see how the LD prediction method performed in comparison to ATPG, for each circuit, the set of possible shorts was extracted from the original layouts and ATPG was performed to generate a list of all undetectable shorts. MCR then used this list during routing to determine if a pair of lines should be allowed to be adjacent. Table 5.7 shows, as would be expected, that the decrease in undesirable critical area totals is much higher. The reason that the decrease in Table 5.7 is lower than 57% reductions that the separation techniques can obtain when targeting a 1% of adjacencies (Figure 4.21) is that in changing the routing, new adjacencies are created that did not exist in the original route. To overcome this, an iterative process could be used to update the list of undetectable shorts after each ATPG-MCR cycle. Unfortunately, ATPG is very expensive in terms of run-times. On a Sun4 computation server, it took 15,843 seconds to perform the ATPG and only 419 seconds to route all the circuits.

5.1.7 Conclusions

In this section, the idea of local determinability was presented. An LD predictor was able to efficiently identify large numbers of undetectable shorts, 42%, by examining only local circuit information. This predictor was able to identify some undetectable shorts that an ATPG system aborted on. It was shown that MCR can eliminate some undetectable shorts and greatly reduce the likelihood of occurrence of others. A reduction in adjacency of 24% was obtained with no increase in area and a 48% increase in routing time when the LD predictor was used. Since routing times are very low compared to placement times, this is a very small increase. It was shown that when separation is more important than running time, ATPG can be used to achieve a 47% reduction with little area penalty (.03%) and a very large time penalty.

5.2 Segmentation

This section shows how MCR can be used to increase the likelihood of a short being detected under the pseudo-exhaustive segmentation testing methodology.

5.2.1 Motivation

If all input combinations are applied to a circuit, all faults that are detectable with only a single static pattern are guaranteed to be detected. Unfortunately, since this requires 2^n patterns when there are n circuit inputs, this is too costly. One attempt to take advantage of exhaustive testing without experiencing as large a cost has been to divide a circuit into segments that each have a smaller number of inputs and to then exhaustively test each segment [MBN81, McC84, AM84, UM87, UM89, Ude92].

As was mentioned by Acken [Ack88] detectable shorts may go undetected if a short occurs between lines that do not share at least one segment in common. If a short between the two lines is detectable by applying only a single input pattern, it is not guaranteed that the pattern will be exercised since the lines are not in the same segment. However, if

Circuit	Original percent in different segments	Decrease in Undesirable Shorts	Decrease in CA
c2670	67.17%	.70%	13.38%
c3540	39.88%	.75%	22.14%
c5315	62.20%	.49%	13.87%
c6288	28.80%	1.5%	24.83%
c7552	67.53%	.59%	9.45%
Average	53.12%	.81%	16.73%
Overall	55.86%	.73%	13.82%

Table 5.8: Original percentage of shorts in different segments, percent reduction in shorts in different segments, and overall reduction in undesirable critical area totals with full look-ahead and a weight penalty of 5000.

they share at least one segment, the proper input combination will be exercised since all combinations of the inputs to the segment will be exercised. Therefore, the effectiveness of pseudo-exhaustive testing can be improved by not routing lines adjacent to each other unless they are both part of the same segment.

5.2.2 Experimental Procedure

Jon Udell's segmentation software was used [Ude92] to create reasonable segments for each of the ISCAS 85 benchmark circuits. MCR was then modified to try to separate lines if they have no segment in common. To do this, the generated segments were used to create a list of the segments each wire belonged to. This list was then checked during routing and if two lines were not in the same segment, MCR tried to separate them.

5.2.3 Results

In order to provide a proper comparison, the same routing parameters for improving static-voltage testing were used. The results of running the router are shown in Table 5.8.

In this case, the router was only able to achieve an average reduction of 17% and an overall reduction of 14%. The reason for this can be seen by looking at the second column of the table. The percentage of shorts between wires in different segments is close to 56%. There was an average 82% increase in routing times and an average 3% increase in area.

This means that while the likelihood of undetected shorts can be reduced, it comes at a higher cost and provides a smaller percent improvement than the improvements gained for static-voltage testing. However, considering the total improvement in terms of all shorts, it eliminates far more undesirable shorts and CA. With undetectable shorts in static-voltage testing, only .15% of all shorts were targeted and a 27% improvement experienced when an LD predictor was used and a 57% improvement when ATPG was used. This means that an overall improvement in testability, in terms of all shorts, of at best .085% is achieved. For segmentation, since 56% of shorts are between nodes in different segments, a 14% improvement for those 56% yields an overall improvement in testability of 7.8%, a larger overall improvement.

5.2.4 Conclusions

This section showed that the routing techniques can be used to increase the testability of circuits under the pseudo-exhaustive testing methodology. Because the percentage of shorts between wires in different segments is quite high, the router can only reduce the probability of a short occuring between wires in different segments by 14%. However, because so many shorts occur between wires in different segments, this translates to a significant improvement in overall testability. For further improvements, segmentation should be considered during circuit placement [Ack88].

6. Applications to Routing in General

This chapter discusses some of the important ideas that should be considered when creating a goal-oriented router. These ideas pertain to the strength with which a goal should be approached and which wires should belong to the target set.

6.1 Relative Importance of Goals

If there are multiple goals to achieve when routing a circuit, a decision on how they are best achieved needs to be made. The most crucial point to consider when making the decision is the relative importance of each goal. If the importance of a goal is very high, the effort applied to meeting the goal should be greater than if its importance is low. At the extremes, the importance of a goal may range from that of a weak objective, a goal that should be met if there are no adverse side effects, to that of a hard constraint, a goal that must be met at all costs.

As an example, consider the goal of reducing capacitive coupling. If the capacitive coupling in a circuit is too great, it may cause the circuit to fail. In this case, reducing the capacitive coupling should be treated as a constraint. If the capacitive coupling is not large enough to cause the circuit to fail, it is still desirable to reduce it in order to decrease delay and noise. In this case, reducing the capacitive coupling should be treated as an objective.

Always approaching goals strongly is inadvisable because of the conflicts that can arise with attaining other goals. For instance, goal-oriented routing methods may have large area or performance penalties. If these penalties outweigh the benefits provided, the methods won't be used. Constraints are usually undesirable since they must *always* be enforced irregardless of their drawbacks. Constraints do not allow tradeoffs to be made with other goals and can even prevent a solution by overconstraining a problem. Constraints are necessary though for goals that must be met. Objectives are desirable since they allow tradeoffs to be made with other goals and therefore have fewer drawbacks.

When determining how much effort should be used to meet a goal, the importance of achieving the goal with respect to other goals must be considered. Among the goals that a circuit function correctly, perform well, and cost little, the goal of functionality is vital. The others are meaningless if the circuit does not work. This means that functionality is a constraint and should always take precedence over performance and cost. Among themselves, performance will sometimes take precedence and cost will sometimes take precedence, in which case the more important should be given preference. At other times they will be ranked equally, in which case they should be treated equally. In all cases, each goal should take all other goals into account. The only difference should be in how strongly a given goal lets another affect it. It is important to not treat goals individually since an improvement in one often leads to a decline in another.

Finally, since the relative importance of differing goals may change from circuit to circuit, the strength with which a router attempts to achieve a particular goal should not be fixed. The end user should be able to control how strongly the router meets a given goal from the "only if no adverse effects" level to the "at all costs" level. In some sense, the router should be given a amplitude control for how strongly a given goal is attacked. An excellent way to achieve this is to use a weighting scheme that allows a goal's importance to be varied.

6.2 General Versus Selective

An additional point to consider is whether a goal should be approached in a general or selective manner. Consider capacitive coupling. One can reduce either the overall capacitive coupling (general) or only the capacitive coupling between target pairs of lines (selective). In cases where there is a subset that is more important, for instance line-pairs that will fail from capacitive coupling versus all line-pairs, it is best to first apply selective methods and then apply general methods, taking care to insure that the general methods do not undo the results of the selective methods.

Performing operations in this order provides several benefits. An overall improvement can be gained that will have given preference to the more important members of the set.

If the complete set were first targeted, fixing a relatively unimportant member of the set might hamper the fixing of an important member of the set. Additionally by using multiple passes, stronger methods can be used at first to increase the chance of success for important members with a smaller penalty than if the same methods were applied to all members.

6.3 Illustration of Ideas

To put the ideas of the above two sections into perspective, consider MCR. In order to make circuits more testable, decisions on how strongly to approach testability and the manner in which to approach it had to be made. Since, historically, testing has not been considered a crucial goal and enhancing it through routing can impact the goals of area and time, it was decided that testing should not be met too strongly. This was accomplished by using a weight that kept the area penalty low. For static-voltage testing, the time penalty was kept low by the use of an LD predictor. If the importance of testability, in relation to time and area, changes, the router can change accordingly. The weight penalty and amount of look-ahead can be varied to achieve the desired balance. Ideally, the levels of circuit information utilized by the undetectability predictor could also vary.

For shorts that are undetectable with static-voltage testing, a selective method was used to achieve the best results. Only shorts that are known to be undetectable are targeted. Since the target percentages are low, large improvements were made. For pseudo-exhaustive testing, more general methods were used since an undetectability predictor was unavailable. The methods were general since all shorts between lines in different segments were targeted while not all shorts between lines in different segments would necessarily go undetected. Since the target percentages were higher, smaller improvements were made.

6.4 Future Work

The most important area for future work are in devising automated methods for controlling the strength with which a goal is approached. Ideally, a user should be able to set a high level goal, such as no more than an X% increase in time and a Y% increase in area, and a router would be able to determine the proper values for the variable parameters, such as amount of look-ahead or weight penalty, in order to meet the high level goal.

The problem of efficiently mixing selective and general methods so that they can be performed concurrently should also be solved. In a sense, this can be thought of as sub-prioritizing within a given goal. For weight-based schemes, this might be accomplished by "weighting" the weights by scaling a weight based on the importance of the element the weight represents.

7. Conclusions

In this dissertation, I presented the first implementation of P-DFT techniques that improve the testability of circuit interconnect with the channel router, MCR. Other researchers in this area presented general suggestions without an implementation. The techniques presented in this dissertation achieve significant improvements in testability with insignificant area overhead and acceptable performance overhead.

The P-DFT techniques presented in this dissertation discourage placing wires next to each other if a short between them would cause a fault that is a potential testing problem. These techniques work over a wide range of targeted adjacencies and were applied to two testing problems with different distributions of undesirable wire pairs, one with 0.2% undesirable and the other with 56% undesirable. In both cases the testability of the suite of test circuits improved significantly.

Since testability is usually emphasized less than other design issues, it is best to treat it as a weak objective during the physical design of a circuit. P-DFT techniques incorporated into MCR treat testability as a weak objective. MCR can be tuned to increase testability by increasing the weight penalty for undesirable adjacencies and increasing the number of tracks of look-ahead. Increasing the weight penalty increases the likelihood of separation at the expense of area. For weak objectives, a weight penalty of 5000 is sufficient. Increasing the look-ahead increases the likelihood of separation at the expense of time unless the percentage of target adjacencies is very small or very large. For small and large percentages of target adjacencies, the number of tracks of look-ahead should be small.

P-DFT techniques should target as selective a group as possible in order maximize improvements and minimize area costs. MCR performed very well when targeting known undetectable shorts since the target group was small. When the target group was larger, potentially undetected shorts in different segments, MCR did not remove as large a percentage of undesirable adjacencies.

P-DFT techniques can make use of local information to efficiently target undetectable static-voltage shorts. By examining only local information, many shorts are identified as undetectable without incurring the expense of ATPG. Examining local information sometimes identifies undetectable shorts that ATPG misses. Nearly half of the undetectable shorts were identified by an LD predictor.

References

[Ack83] J.M. Acken. Testing for bridging faults (shorts) in CMOS circuits. *Proceedings of Design Automation Conference*, pages 717–718, 1983.

[Ack88] John M. Acken. *Deriving Accurate Fault Models.* PhD thesis, Stanford University, Department of Electrical Engineering, September 1988.

[AM84] E. C. Archambeau and E. J. McCluskey. Fault coverage of pseudo-exhaustive testing. In *Dig., Int. Conf. Fault-Tolerant Computing*, pages 141–145, 1984.

[Bak90] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Publishing Company, 1990.

[BBGL92] K. Balachandran, S. Bhaskaran, H. Ganesan, and C. Lursinsap. A yield enhancing router. In *IEEE International Symposium on Circuits and Systems*, volume 4, pages 1936–1939. IEEE, 1992.

[BF85] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1985.

[CK86] Howard H. Chen and Ernest S. Kuh. Glitter: A gridless viarable-width channel router. *IEEE Transactions on Computer-Aided Design*, 5(4):459–465, October 1986.

[COK93] Kamal Chaudhary, Akira Onozawa, and Ernest S. Kuh. A spacing algorithm for performance enhancement and cross-talk reduction. In *Digest of Technical Papers of the International Conference on Computer Aided Design*, pages 697–702. IEEE and ACM, 1993.

[CSV90] Umakanta Choudhury and Alberto Sangiovanni-Vincentelli. Constraint generation for routing analog circuits. In *Proceedings of Design Automation Conference*, volume 27, pages 561–566. ACM and IEEE, 1990.

[CSV93] Umakanta Choudhury and Alberto Sangiovanni-Vincentelli. Constraint-based channel routing for analog and mixed analog/digital circuits. *IEEE Transactions on Computer-Aided Design*, 12(4):497–510, April 1993.

[Fer87] F. Joel Ferguson. *Inductive Fault Analysis of VLSI Circuits.* PhD thesis, Carnegie Mellon University, Department of Electrical and Computer Engineering, October 1987.

[Fer93] F. Joel Ferguson. Physical design for testability for bridges in CMOS circuits. In *Proceedings of the 1993 VLSI Test Symposium*, pages 290–295. IEEE, 1993.

[FKM92] D. Feltham, J. Khare, and W. Maly. Design for testability view on placement and routing. In *EURO-DAC '92*, pages 382–387, 1992.

[FL91] F. Joel Ferguson and Tracy Larrabee. Test pattern generation for realistic bridge faults in CMOS ICs. In *Proceedings of International Test Conference*, pages 492–499. IEEE, 1991.

[FS88] F. Joel Ferguson and John P. Shen. A CMOS fault extractor for inductive fault analysis. *IEEE Transactions on Computer-Aided Design*, 7(11):1181–1194, November 1988.

[FTL90] F. Joel Ferguson, Martin Taylor, and Tracy Larrabee. Testing for parametric faults in static CMOS circuits. In *Proceedings of International Test Conference*, pages 436–443. IEEE, 1990.

[GL93] Tong Gao and C. L. Liu. Minimum crosstalk channel routing. In *Digest of Technical Papers of the International Conference on Computer Aided Design*, pages 692–696. IEEE and ACM, 1993.

[GL94] Tong Gao and C. L. Liu. Minimum switchbox channel routing. In *Digest of Technical Papers of the International Conference on Computer Aided Design*, pages 610–615. IEEE and ACM, 1994.

[Gn89] Ronald S. Gyurcsik and Jzan-CHing Jee n. A generalized approach to routing mixed analog and digital signal nets in a channel. *IEEE Journal of Solid-State Circuits*, 24(2):436–442, April 1989.

[GRKG90] J. Greene, V. Roychowdhury, S Kaptanoglu, and A. E. Gamal. Segmented channel routing. In *Proc. 27th ACM/IEEE Design Automation Conference*, pages 567–572, June 1990.

[HKtK90] Ikuo Harada, Hitoshi Kitazawa, and takao Kaneko. A routing system for mixed A/D standard cell LSI's. In *Digest of Technical Papers of the International Conference on Computer Aided Design*, pages 378–381. IEEE and ACM, 1990.

[HS71] A. Hashimoto and J. Stevens. Wire routing by optimizing channel assignment within large apertures. In *Proc. 8th IEEE Design Automation Workshop*, pages 155–169, 1971.

[IF94] E. Isern and J. Figueras. Analysis of I_{DDQ} detectable bridges in combinational CMOS circuits. In *Proceedings 12th IEEE VLSI Test Symposium*, pages 368–373. IEEE, 1994.

[JF93] Alvin Jee and F. Joel Ferguson. Carafe: An inductive fault analysis tool for CMOS VLSI circuits. In *Proceedings of the IEEE VLSI Test Symposium*, 1993.

[KBRM91] R. Kapur, K. Butler, D Ross, and M.R. Mercer. On bridging fault controllability and observability and their correlations to detectability. In *Proc. 2nd Annu. European. Test Conf.*, pages 333–330, 1991.

[KDG$^+$85] C. D. Kimble, A. E. Dunlp, G. F.. Gross, V. L. Hein, and Others. Autorouted analog VLSI. In *Proceedings of the IEEE 1985 Custom Integrated Circuits Conference*, pages 72–78. IEEE, 1985.

[Koe87] Siegmar Koeppe. Optimal layout to avoid CMOS stuck-open faults. In *Proceedings of Design Automation Conference*, pages 829–835. IEEE, 1987.

[KSV94] Desmond A. Kirkpatrick and Alberto L. Sangiovanni-Vincentelli. Techniques for crosstalk avoidance in the physical design of high-performance digital systems. In *Digest of Technical Papers of the International Conference on Computer Aided Design*, pages 616–619. ACM and IEEE, 1994.

[Kuo93] Sy-Yen Kuo. YOR: a yield-optimizing routing algorithm by minimizing critical areas and vias. *IEEE Transactions on Computer-Aided Design*, 26:1303–1311, September 1993.

[LA90] Marc E. Levitt and Jacob A. Abraham. Physical design of testable VLSI: Techniques and experiments. *IEEE Journal of Solid-State Circuits*, 25(2):474–481, April 1990.

[Lar92] Tracy Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, pages 4–15, January 1992.

[LP90] Andrea S. LaPaugh and Ron Y. Pinter. Channel routing for integrated circuits. In Joseph F. Traub, editor, *Annual Review of Computer Science*, volume 4, pages 307–363. Annual Reviews Inc., 1990.

[MBN81] E. J. McCluskey and S. Bozorgui-Nesbat. Design for autonomous test. *IEEE Transactions on Computers*, C-30:866–875, November 1981.

[McC84] E. J. McCluskey. Verification testing - a pseudo-exhaustive test technique. *IEEE Transactions on Computers*, C-33:541–546, June 1984.

[McG92] Richard McGowen. An improvement on yoeli's channel router. Master's thesis, University of California at Santa Cruz, Computer Engineering Department, December 1992.

[MF94a] Richard McGowen and F. Joel Ferguson. Elimination of undetectable shorts during channel routing. In *Proceedings 12th IEEE VLSI Test Symposium*, pages 402–407. IEEE, 1994.

[MF94b] Richard McGowen and F. Joel Ferguson. A study of undetectable non-feedback shorts for the purpose of Physical-DFT. In *Proceedings of the European Design and Test Conference*, pages 371–375. EDA Association, 1994.

[MFS84] W. Maly, F.J. Ferguson, and J. P. Shen. Systematic characterization of physical defects for fault analysis of MOS IC cells. In *Proceedings of International Test Conference*, pages 390–399. IEEE, 1984.

[MNRC92] Sujoy Mitra, Sudip K. Nag, Rob A. Rutenbar, and L. Richard Carley. System-level routing of mixed-signal ASICs in WREN. In *Digest of Technical Papers of the International Conference on Computer Aided Design*, pages 394–399. ACM and IEEE, 1992.

[MTCC87] W. Maly, M.E. Thomas, J.D. Chinn, and D.M. Campbell. Double-bridge test structure for the evaluation of type, size and density of spot defects. Technical Report CMUCAD-87-2, Carnegie Mellon University, SRC-CMU Center for Computer-Aided Design, Dept. of ECE, February 1987.

[PTLG89] A. Pitaksanonkul, S. Thanawastien, C. Lursinsap, and J. A. Gandhi. DTR: a defect-tolerant routing algorithm. In *Proceedings of Design Automation Conference*, pages 795–798. ACM and IEEE, 1989.

[RF82] Ronald L. Rivest and Charles M. Fiduccia. A 'greedy' channel router. In *Proc. 19th ACM/IEEE Design Automation Conference*, pages 418–424, 1982.

[RSVS85] James Reed, Alberto Sangiovanni-Vincentelli, and Mauro Santomauro. A new symbolic channel router: YACR2. *IEEE Trans. on Computer-Aided Design*, 4(3):208–219, July 1985.

[SCS$^+$92] M. Saraiva, P. Casimiro, M. Santos, J.T. Sousa, F. Gonçalves, I Teixeira, and J.P. Teixeira. Physical DFT for high coverage of realistic faults. In *Proceedings of International Test Conference*, pages 642–647. IEEE, 1992.

[SSV85] C. Sechen and A. Sangiovanni-Vincentelli. The timberwolf placement and routing package. *IEEE Journal of Solid-State Circuits*, April 1985.

[TBM94] A. Tyagi, M. Bayoumi, and P. Manthravadi. Yield enhancement in the routing phase of integrated circuit layout synthesis. In *Proceedings of 1994 International Conference on Wafer Scale Integration (ICWSI)*, pages 52–60. IEEE, 1994.

[TTA$^+$91] J.P. Teixeira, I.C. Teixeira, C.F.B. Almeida, F.M. Gonçalves, and J. Gonçalves. A methodology for testability enhancement at layout level. *Journal of Electronic Testing: Theory and Applications*, 1(4):287–300, January 1991.

[TWC89] K. S. The, D. F. Wong, and J. Cong. Via minimization by layout modification. In *Proc. 26th Design Automation Conference.*, pages 799–802. ACM and IEEE, 1989.

[Ude92] J. G. Udell, Jr. Efficient segmentation for pseudo-exhaustive BIST. In *CICC*, pages paper 13.6/1–5, May 1992.

[UM87] J. G. Udell, Jr. and E. J. McCluskey. Efficient circuit segmentation of pseudo-exhaustive test. In *PrICCAD*, pages 148–151, 1987.

[UM89] J. G. Udell, Jr. and E. J. McCluskey. Pseudo-exhaustive test and segmentation: Formal definitions and extended fault coverage results. In *Dig., FTCS 19*, pages 292–298, 1989.

[Wad78] R.L. Wadsack. Fault modeling and logic simulation of CMOS and MOS integrated circuits. *Bell System Technical Journal*, 57(5):1449–1474, May-June 1978.

[XHJ94] E Hua Xue, Ed P. Huijbregts, and Jochen A. G. Jess. Routing for manufacturability. In *Proc. 31st Design Automation Conference.*, pages 402–406. ACM and IEEE, 1994.

[YK82] T. Yoshimura and E. S. Kuh. Efficient algorithms for channel routing. *IEEE Trans. on Computer-Aided Design*, 1:25–35, 1982.

[Yoe91] Uzi Yoeli. A robust channel router. *IEEE Trans. on Computer-Aided Design*, 10(2):212–219, February 1991.