The General Interconnect Problem of Integrated Circuits

John Y. Ngai

Computer Science Department
California Institute of Technology

5143:TR:84

CALIFORNIA INSTITUTE OF TECHNOLOGY

Computer Science

5143:TR:84

THE GENERAL INTERCONNECT PROBLEM OF INTEGRATED CIRCUITS

by

John Y. Ngai

In Partial Fulfillment of the Requirements for the
Degree of Master of Science

May, 1984

The work described in this document was sponsored by the
Defense Advanced Research Projects Agency, ARPA order number 3771,
and monitored by the Office of Naval Research under
contract number N00014-79-C-0597

ACKNOWLEDGEMENTS

I would like to express my gratitude to all the folks at the Caltech Computer Science Department for their help and support during the course of this project. In particular, I owe special thanks to my advisor, Chuck Seitz, who guided me in the right direction, inspired me with new ideas, encouraged me to explore alternatives and allowed me to work on my own pace. Special thanks also go to Gary Clow for the many discussions of ideas and the references to literature he provided.

I would also like to thank Eric Holstege and Mike Newton for their help on using UNIX and programming in C. I also appreciate the friendship of Doug Whiting who had to listen to a lot of my griping along the way. Thanks also go to the students in the VLSI class at Caltech for their helpful suggestions.

I am also indebted to Caltech whose financial support allowed me to obtain such an excellent education in this country. I thank my parents for their love, encouragement and brought up that makes me whom I am. Finally, I thank Theresa for her love and understanding.

ABSTRACT

This thesis is concerned with the interconnection problem of custom integrated circuits. It may be broadly defined as the transformation of circuit description represented by the notion of modules together with the circuit connectivity requirements, into wiring patterns which implement the required connectivities. Conventional approaches to its solution are presented. Issues such as partition to placement and routing and various layout optimization tradeoffs are discussed. A detail hierarchical routing model with timing considerations that extends naturally to multiple conducting layer environment is presented. Several of the implications of this extension are also discussed.

The rest of this thesis deals with an experiment with the stepping approach to routing as an alternative to the conventional cellular approach emphasizing simplicity rather than optimization. Algorithms for routing signals and power developed for the stepping router are presented. An implementation of this approach by the author together with some test examples and their results are also described. This thesis concludes with a few suggestions for further research work in this area which the author considers very important from the experience gained during the work on this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The period since 1970 has been marked by rapid advances in integrated circuit fabrication technology. We can now make very large scale integrated (VLSI) circuits with 100,000 or more transistors reliably. The development of an integrated circuit of such complexity presents challenges to logic design, simulation, layout and extensive testing techniques to ensure an error-free product. Hierarchical methodologies [Mead & Conway 80] are generally employed. Entire circuits are partitioned into smaller and simpler pieces called modules or cells. Each module is composed of other modules or primitive cells and the compositions recur until the modules arrived at are simple enough to be understood and designed comfortably. Such an approach relies heavily on the ability to compose modules successfully and efficiently. Computer aided design techniques are routinely employed to reduce the development cost.

1.1. The Composition Problem

To compose the circuit modules is to connect electrically the appropriate signals of the circuit modules together to form bigger super-modules. The super-modules normally have their own sets of signals that will be used in future compositions. In accordance with the structured design style for VLSI systems presented in [Mead & Conway 80], a module is normally rectangular in shape and all signal connections to a circuit module occur at the perimeter of the module (see Figure 1.1). The connections can

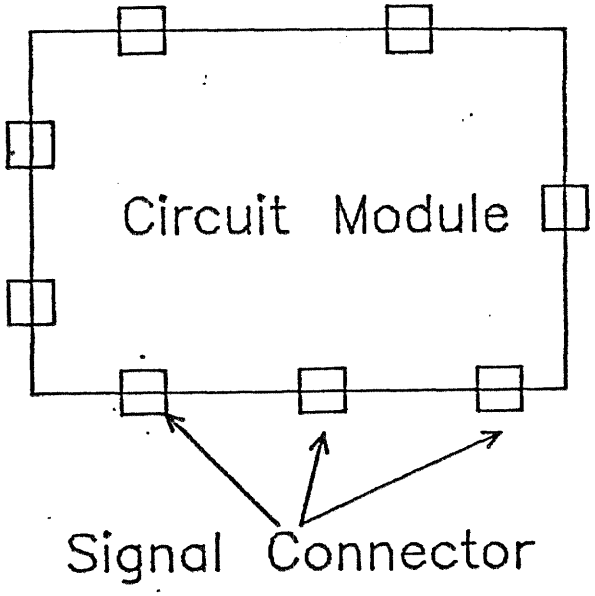

Figure 1.1 A circuit module

be achieved by direct abutment which requires exact matching of the signal port positions between neighboring modules (see Figure 1.2). Stretching of modules to enforce matching of ports is a common technique employed by such systems [Kingsley 82]. A different way of connecting signal ports is to

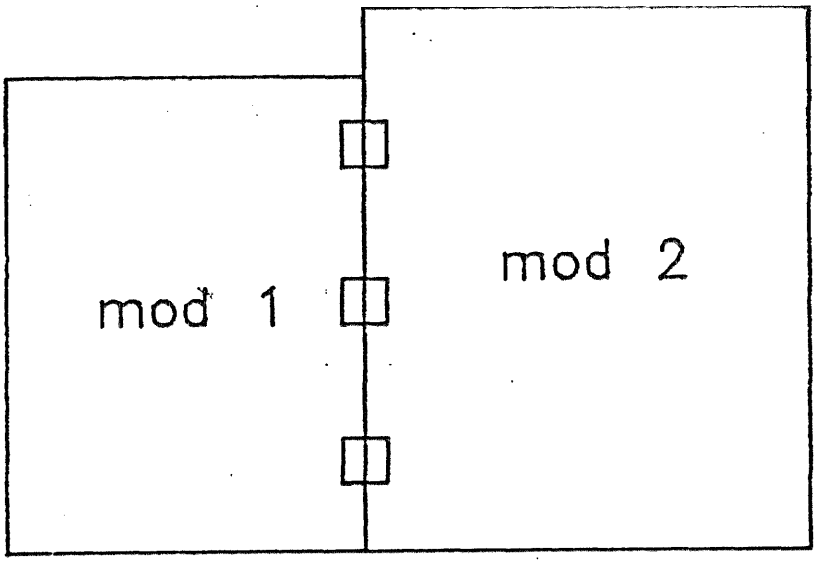

Figure 1.2 Composition by Abutment

route the signals through wires to the required port locations [Rivest 82a] [Hassett 82] (see Figure 1.3). The former approach works well at the lower and more regular levels of design. The latter is much more suited to the last several assembly stages of a large chip when a large number of signal, power and clock ports which are very much geometrically constrained, have to be connected together. This thesis is concerned with the latter approach.

1.2. Placement and Routing

To handle the overall complexity of the integrated circuit composition problem, the solution process is usually divided into two phases: the *placement phase* and the *routing phase*. In the placement phase, each of the composition modules is assigned a fixed geometric location on a routing plane. There are various restrictions to be satisfied, and a number of objectives to be optimized by the placement assignments. Since the yield of fabricating an integrated circuit decreases as the exponential of its area, the

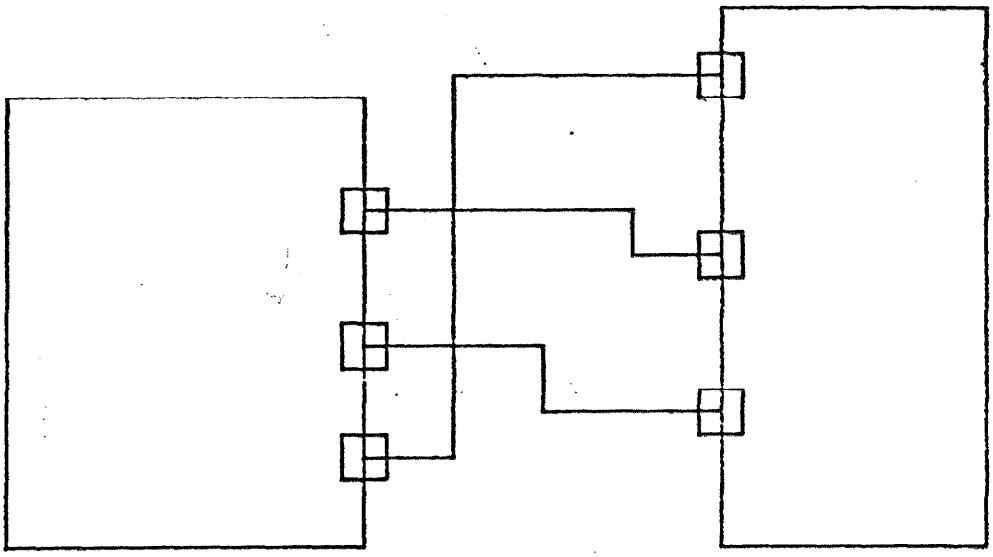

Figure 1.3 Composition by Routing

prime goal is to minimize total chip area, and yet allow enough space between modules to generate the required interconnect wires. Once the modules are placed, the actual wiring patterns implementing the net connections must be generated, which constitutes the routing phase. While the serial partition of the composition problem into placement and routing simplies the solution process, the optimization strategies involved in the two phases are often highly interdependent.

In custom VLSI design, structured design methods with advance floorplanning often result in much cleaner separation of the placement and routing phase. In these applications, the designer taking advantage of his knowledge of the circuits can come up with placements which are very close to optimal. In addition, enforcement of sub-optimal placements may be desirable in exchange for other design criteria. After all, the composition problem is only one of the many problems encountered in the overall design process.

Another complication that arises in the area of custom VLSI design is the issue of *standard cells* versus *general cells*. Standard cells are building blocks which have fixed height (see Figure 1.4). This approach limits the range of cell complexity considerably, since a height suitable for simple gates will force the layout of complex modules to be very wide. This limitation together with the limit on the dimensions of a chip impose a rigid placement topology which greatly simplifies the routing process. General cells are structures on the other extreme. They can have arbitrary dimensions and placements (see Figure 1.5). The placement and routing of general cells are necessarily much more complicated than those of standard cells. In exchange for the complications, the circuit designer has freedom in

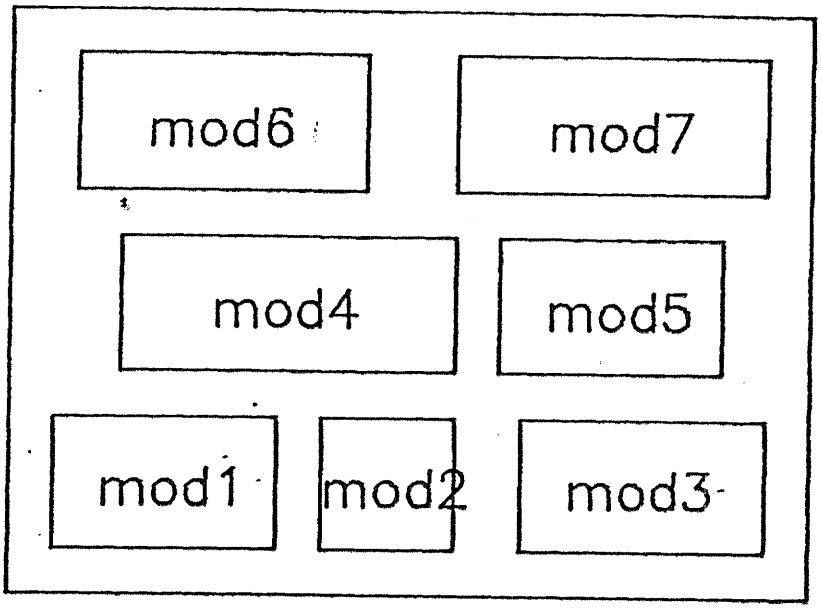

Figure 1.4 Standard Cells

deciding the sizes and aspect ratios of the cells laid out. A restrictive form of general cells called the *slicing floorplan* [Johannsen 81] has also been widely investigated [Liu & Atkins 82]. This thesis is concerned with routing of general cells in their extreme form.

1.3. Routing Approaches

Given an input specification of circuit module placements and the various power and signal port positions, the router has to come up with a final layout of power and signal wires which correctly connect the various signals together. This layout must satisfy all the requirements set forth by the routing model employed.

In general, to reduce complexity, the problem is further broken down into several simpler phases [Hightower 80] [Hassett 82] [Rivest 82a].

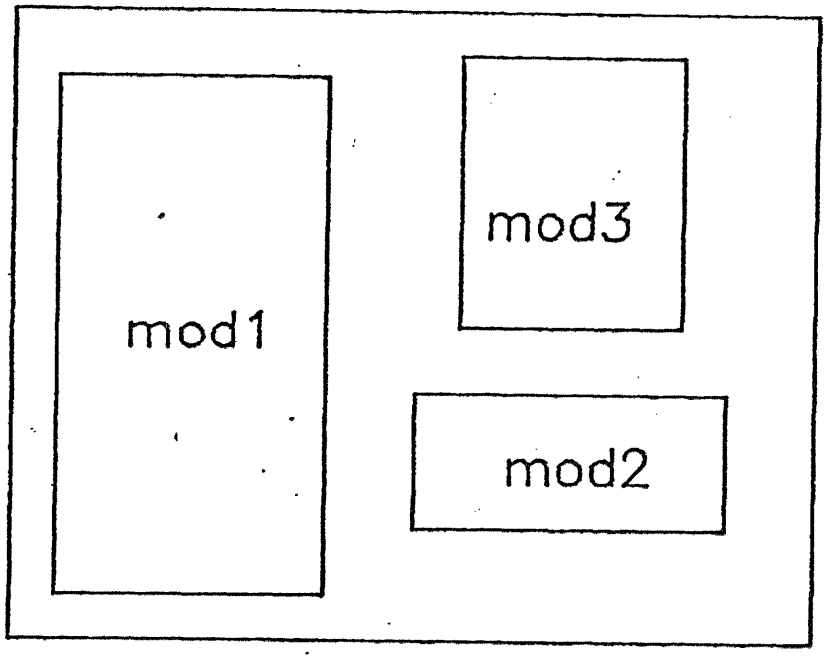

Figure 1.5 General Cells

Initially, large rectangular routing channels are defined with respect to the module placements[1]. Then, a rough *Global Routing Phase* operates on interchannel connections. Global signals are assigned to intermediate channels that form its wiring path. It finds rough topological connections for all signal nets with the primary objective in avoiding channel overflow. Other objectives such as minimizing the total expected wire length of the global nets; minimizing the critical path in the global layout and maximizing usage of the best conducting layer are also to be selectively optimized. In general, the quality of the topological layout is sensitive to the channels defined and the order of routing nets [Hightower 74].

After the *Global Routing Phase*, a *Channel Routing Phase* operates on detail wiring of signals within each routing channel. It finds actual geometric

[1] Router based on the cellular wiring model only.

connections for the signal nets within the routing channel subjected to objectives such as avoiding channel overflow; minimizing jogs and vias[2] and constraint loop handling. Channel routers [Deutsch 76] [Kuh 82] [Rivest 82b] [Chan 83] [Burstein 83] are usually employed. These channel routers are fast and capable of producing high quality layout. However they can only process channels with fixed pins on one set of parallel edges and floating pins on the other set. It is not always feasible to find a routing order for the channels to ensure completion. Employing such channel routers requires much global planning in channel definition and some post-channel-routing processing. Maze routers [Lee 61] [Moore 59] [Hightower 69] [Hsu 82] are also employed. These maze routers are much slower compared to the channel routers but are capable of connecting pins on all four sides of the routing channel. They are usually applied as a last resort when other faster methods failed. In some router [Rivest 82], there is an additional *Crossing Placement Phase* after the *Global Routing phase* which assigns exact crossing locations to global wires passing the intermediate channels. The intended benefits are better, simpler and independent channel routing.

Power and ground connections differ from normal signal connections due to their requirement to be routed exclusively on the best routing layer. They also have variable wire widths due to different power requirements and the need to avoid DC or transient voltage drops and metal migration. These requirements make such paths especially hard to route in a general cell composition environment. In standard cell layout, the regularity of structures enable a predefined global power bus topology to be imposed on the composition layout. No such simple scheme is available for general cell composition. They must be routed like other nets but subjected to their

[2] In Integrated Circuits, they are called *contact cuts*.

specific requirements. Special power routing algorithms were described in [Mlynski 81] [Syed 82] and [Moulton 83].

1.4. Optimization versus Generality

Many routers are oriented toward the objective of producing the best layout possible with little or no human intervention. This approach is incapable of taking advantage of the human insights available from the circuit designer. Extensive heuristic computations are employed to extract what might have been obtained easily from the designer. Due to the complexity and various contradicting objectives and constraints of the routing problem, the quality of the final layout is prone to subjective interpretations. Further, much fine tuning of the routing algorithms is needed before a satisfactory layout is obtained. While the above approach may be necessary in a production environment, it is too costly in an educational environment where the computing budget is very limited. A more intuitive and less "costly" router is obviously needed. With limited resources, heavy optimization is undesirable. The educational environment must support flexibility in experimenting new ideas, be able to test new ideas on small and yet realistic projects. The emphasis should be shifted from optimization to supporting generality.

[Oestreicher 72] described an alternative approach to route printed circuit boards. The final routing layout is produced one raster at a time and all its algorithms are oriented with the raster as a basic operation unit. Pin assignment, component placement and wire routing are all carried in parallel within each raster, and incrementally for the whole PC board. While the routing environment of an IC is different from that of the PC board, the

simplicity of the stepping approach enables a easy modification to fit the IC routing requirements and forms the basis of the experimental approach discussed in this thesis.

CHAPTER 2

THE ROUTING MODEL

A general model for describing the interconnect problem for integrated circuits is discussed in this chapter. Such a model is important in defining both the problem we are tackling and its acceptable solutions. It also provides insights into possible extensions.

2.1. Geometric Model

It is best to start describing the integrated circuit composition model from a description of the underlying electrical and geometric model used by the designer.

Integrated circuits are built from conducting materials laid down on a substrate wafer (usually of silicon). There exist many different collections of conducting materials and different fabrication technologies. In this thesis, we present a generic wiring model that captures the essential ingredients concerning routing of most fabrication technologies. The standard Mead-Conway geometric design rules [Mead & Conway 80] for laying out nMOS integrated circuits are used as illustrations.

A typical fabrication technology offers several layers on which to run wires and build circuit elements. The assumed geometric unit in the design of integrated circuits is a size parameter denoted by λ. All other spatial dimensions are expressed as multiples of λ. This relative sizing enables any layout to remain correct in face of the progressive shrinkage of feature sizes as a result of advances in fabrication technology. λ is approximately the

maximum amount of accidental displacement that is expected in depositing a feature on the wafer. It is typically 1 micron in current state of technology. λ restricts among other things, the width of a wire. A wire less than 2 λ wide cannot assure conduction because a mismatch of λ on each edge of the wire would reduce the wire width down to zero. Similar arguments leads to a whole set of spacing requirements known as the geometric design rules for the technology. Figure 2.1 shows the design rules for the single metal layer nMOS technology. Only those rules that affect routing are shown in the figure.

In order to be able to route arbitrary signal patterns, it is necessary and sufficient to use two non-interacting conducting layers [Busacker 65]. In general, a process has more than two layers to route on. A standard method is to select two non-interacting layers and agree to lay them on an orthogonal coordinate system (see Figure 2.2), although non-orthogonal geometry also exists. Wires that need to switch between the orthogonal axes do so through contact cuts that have their own design rules to follow. The rectilinear system, also known as the Manhattan system, is the most popular. Manhattan geometry is adopted in this thesis because of its simplicity. To further simplify the model, a *routing grid* is imposed onto the routing plane. Wires can only be laid down along the grid lines. Contact cuts can then be inserted on any grid points to allow wires to change their directions. This requirement imposes a minimum separation between adjacent grid points that is usually bigger than the minimum wire separation specified by the technology. For single metal layer nMOS process, this minimum spacing is 7 λ.

The grid model also assumes that minimum width wires are used. While this is adequate for most signal wires, power and ground signals frequently

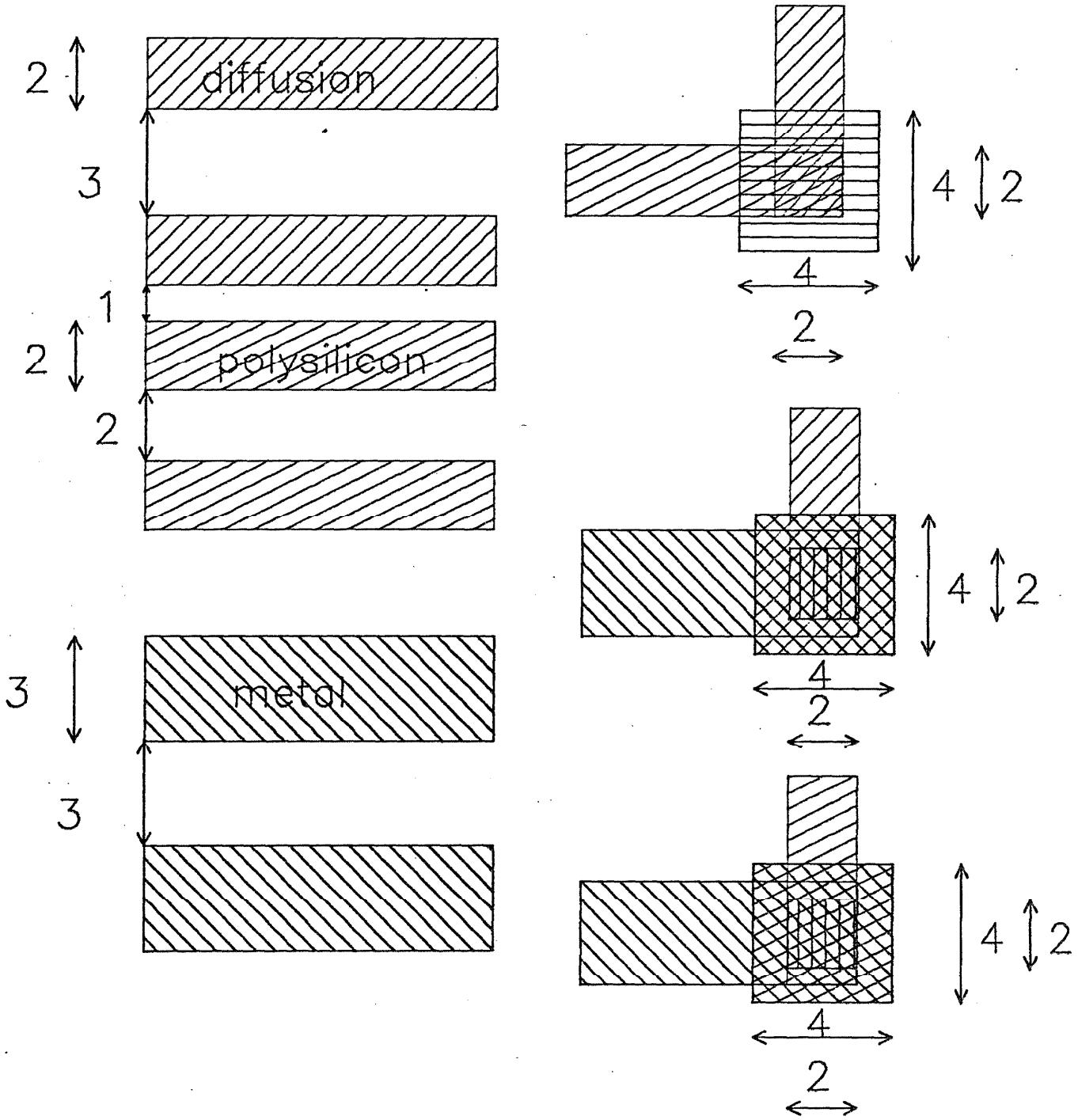

Figure 2.1 nMOS geometric design rules

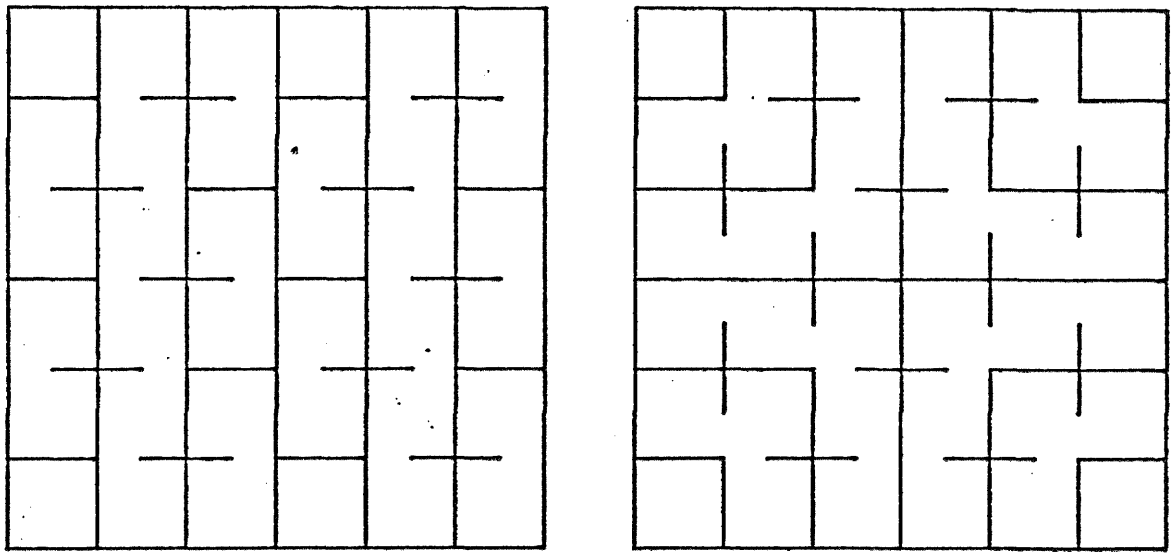

Figure 2.2 Orthogonal wiring coordinate systems

require much wider conducting wires, because the smaller resistance on wide conducting paths reduces the DC voltage drops across the wires. Further, they are less sensitive to metal migration. The extra capacitance along the wider power and ground wires also helps in reducing transient voltage fluctuations induced by switching of transistors. Hence power and ground wires can have arbitrary wire widths and do not have to lie on grid lines.

2.2. Timing Consideration

The geometric model described in the last section is not enough to ensure a successful design and layout of integrated circuits. To have more successful control of the timing behavior of circuits, an understanding of the electrical issues involved is required . Again, only those which are relevant to

the interconnect problem are described.

The primary purpose of routing signal wires is to provide a pathway on which signals can propagate from one spot to another in the circuit. The speed of signal propagation determines how fast a circuit can run and hence its overall throughput. Signal propagation speed is determined by the capacitances and resistances of the wiring layers and the driving capacities of the sources. Capacitance is a measure of how much charge must be supplied or removed in order for a unit square of a layer to swap its voltage. Resistance measures the difficulty in supply current for charging and discharging the layer. It is clear that the R's and C's depend on the specific process involved. Table 2.1 lists the typical resistance and capacitance figures for the nMOS process.

Figure 2.3 shows an inverter driving the gate of another inverter through a long wire not atypical in global wiring. A conservative estimate of the propagation delay T can be roughly expressed as [Mead & Conway 80][Mead & Rem 82] :

$$T \approx (R_t + R_w)(C_w + C_g) \qquad (2.1)$$

where R_t is the resistance of the charging/discharging transistor, R_w is the resistance of the wire, C_w is the capacitance of the wire and C_g is the capacitance of the gate of the transistor being driven. Assuming a minimum width wire is used, R_w and C_w are both proportional to the length of the wire

Table 2.1 Typical electrical parameters for nMOS

Substance	Resistance/square (Ω)	Capacitance/micron (pF)
Metal	0.03	3×10^{-5}
Poly	15-100	4×10^{-5}
Diffusion	10	10^{-4}
Channel	10 K	4×10^{-4}

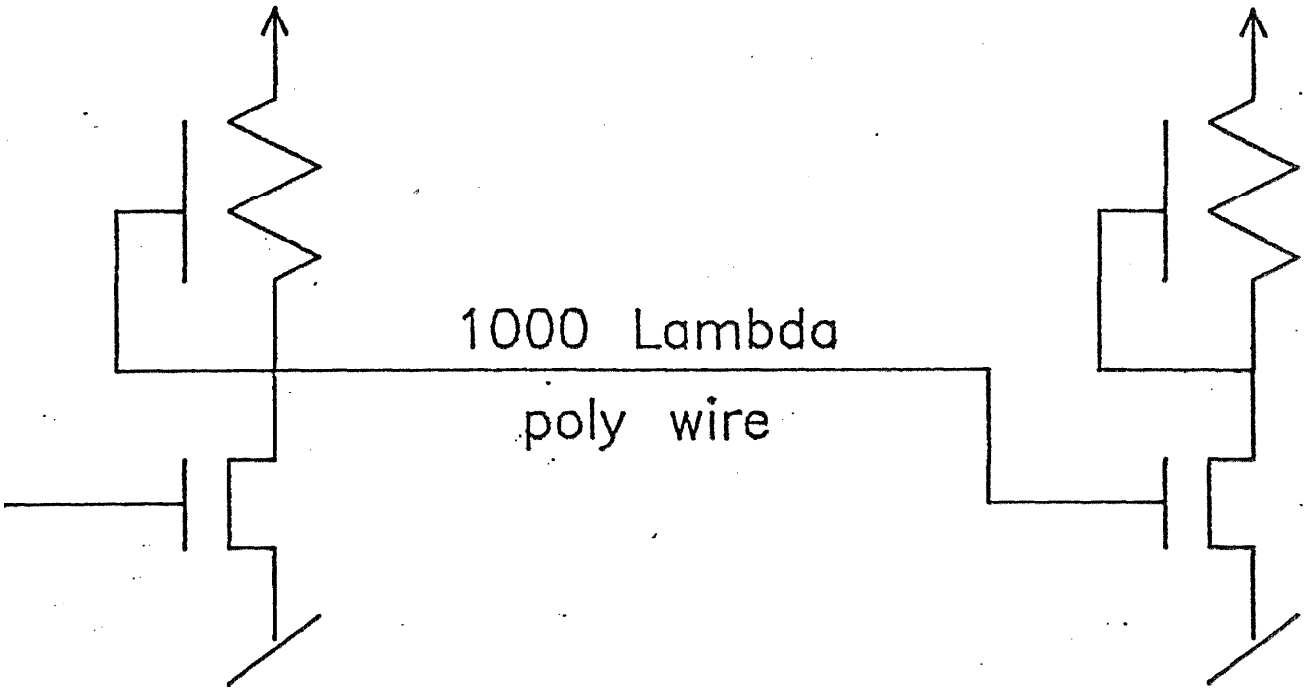

Figure 2.3 Driving signals over a global wire

l. Hence T has a delay term that is proportional to l^2. For global wires, C_w almost always dominates C_g. Whereas the terms R_t and R_w could be about the same magnitude for long global wires (see Table 2.1). Hence T can be rewritten as :

$$T \approx R_t C_w + R_w C_w \qquad (2.2)$$

Conceptually, we can interpret the terms (see Figure 2.4) using the Thevenin equivalent circuit. The first term is the delay contribution assuming a zero resistance wire is being used. The second term is the delay contribution assuming a perfect voltage source is used to drive the wire. An analysis in [Mead & Conway 80] Chapter 1 shows that a reduction from linear dependence to a logarithmic dependence of wire length of the first term in Eqn 2.2 can be achieved by using successively larger buffers to drive the capacitive load presented by the long global wire.

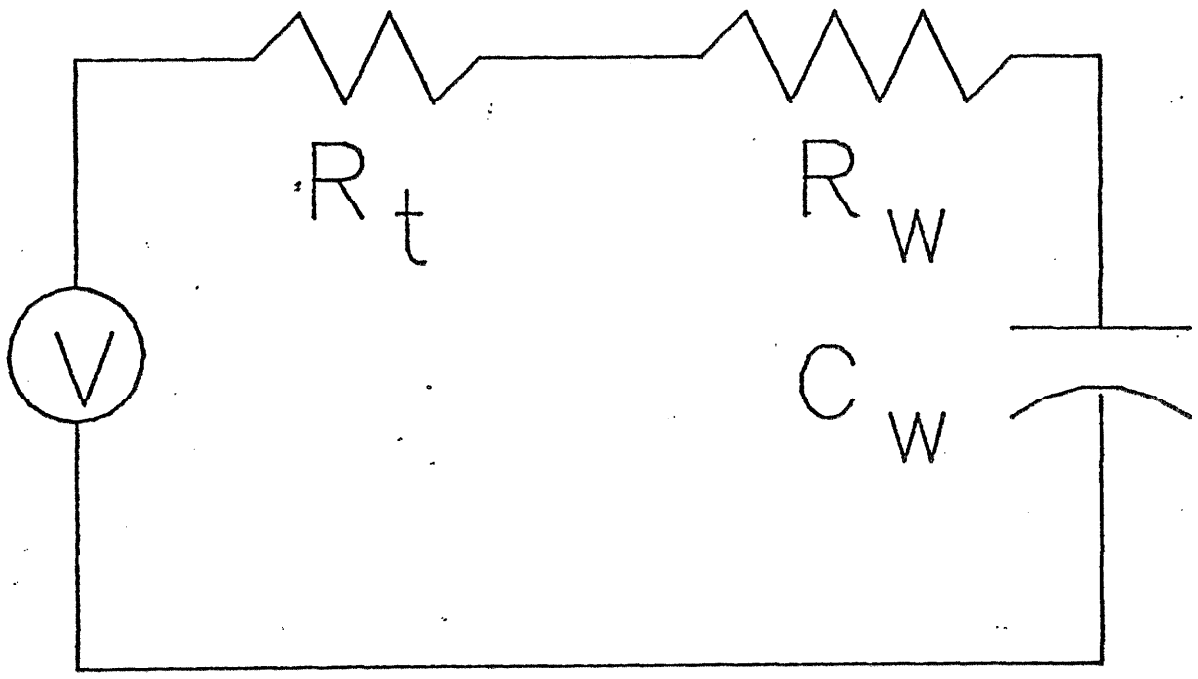

Figure 2.4 Thevenin's Equivalent of the wire delay circuit

When multiple conducting layers are available, it is straight forward to extend the above analysis [Mead & Rem 82]. In particular, we let w be the width of the wire used, σw be the thickness of the layer and the spacing to the layer beneath it, τ be the time taken to charge a minimum transistor gate. Then

$$R_w = \frac{\rho l}{\sigma w^2} \tag{2.3}$$

where ρ is the resistivity of the conducting layer and

$$C_w = \frac{\varepsilon l}{\sigma} \tag{2.4}$$

where ε is the permittivity of the insulation from substrate. It is clear that the time taken to charge a capacitance C_w is $\frac{C_w}{C_g}\tau$. Hence using successively bigger buffers of ratio α to drive the wire requires $\approx \log_\alpha\left(\frac{C_w}{C_g}\right)$

stages. Therefore the total charging time is $\approx \alpha\tau\log_a\left(\frac{C_w}{C_g}\right)$ and the total propagation delay is given by :

$$T \approx \alpha\tau\log_a\left(\frac{\varepsilon l}{\sigma C_g}\right) + \rho\varepsilon\frac{l^2}{\sigma^2 w^2} \qquad (2.5)$$

Eqn 2.5 shows clearly the logarithmic and quadratic terms of the propagation delay. It is clear that as ρ decreases and w increases, a larger propagation distance can be tolerated before the quadratic term becomes dominant in the total wire delay. This suggests a classification of signals according to the distances they have to travel, and subsequently assign the appropriate conducting layers for their distribution. A natural design methodology that facilitates the above classification is the hierarchical design model described in the next section.

2.3. Hierarchical Routing

The complexity of the general cell composition problem calls for a careful planning in order to successfully route instances from real applications. Hierarchical decomposition of the routing problem is introduced as a natural way to manage complexity [Johannsen 81][Rowson 80][Preas 79].

Generally, the hierarchical approach to the VLSI circuit design problem can be divided into a top down circuit floorplanning phase, a leaf cell layout phase and a bottom up circuit assembly phase. During the top down design phase, the overall floorplanning involves estimating the size and shape of each module at the current hierarchical level and make provisions for routing areas. Employing similar techniques, modules of the current hierarchical level are further decomposed to create new sub-hierarchies.

This process is carried up to the point where the modules arrived at can be comfortably laid out or can be fetched from an available leaf cell library. The bottom up assembly phase simply reverses the above process by composing modules of the same hierarchical level together.

2.3.1. Two Dimensional Hierarchy

The ability to do hierarchical composition imposes a discipline on both the composing and the composed blocks. It is best explained by means of a diagram (see Figure 2.5). In any hierarchical approach, there exists a set of conventions which ensure the black box abstraction of building blocks being valid at any level. In VLSI circuits composition, this is conveniently represented by the concept of connectors or *signal ports* along a module's

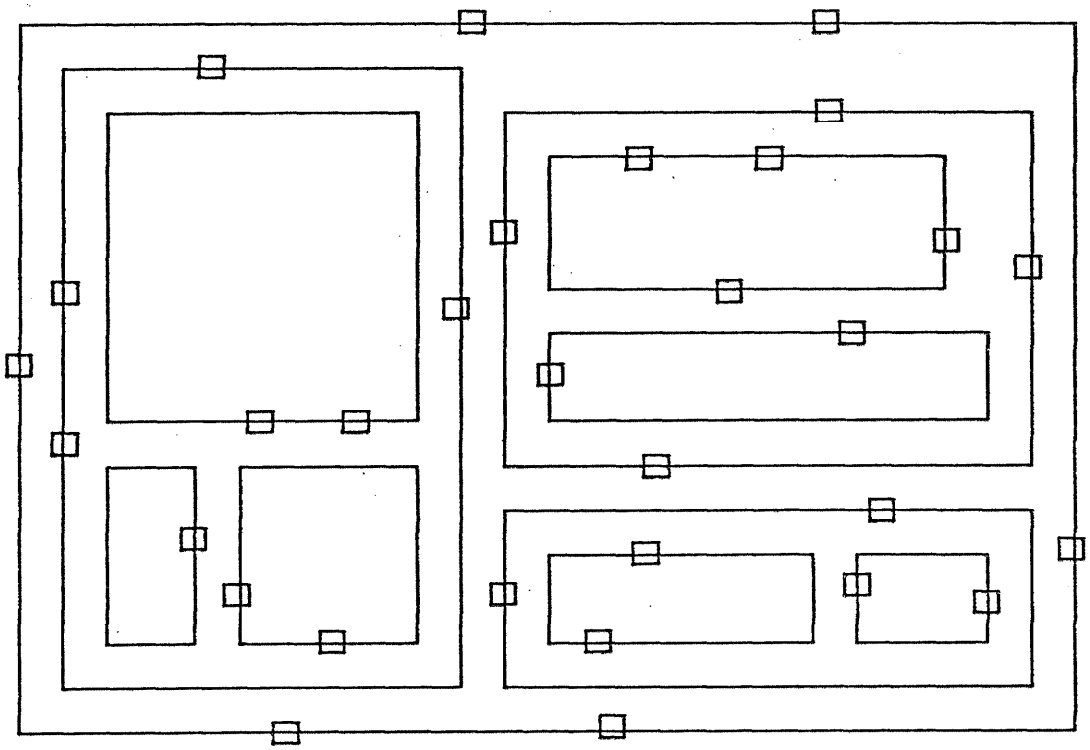

Figure 2.5 Hierarchical Composition of modules

perimeter. At each routing level, modules of similar complexity are grouped together to form a super-module. Every module has its own set of signal ports that define the connections doorways into the internal circuitry of the module. As long as the module is concerned, the size and shape of the module and its signal ports completely define the module at the current routing context. In order for the resulting super-module to be used as building block in its own routing context, it must have its own set of signal ports for connections to the outside. Hence any connection from its internal modules to outside the super-module must be channeled through these predefined signal ports along its perimeter (see Figure 2.5).

Another requirement which is very important in VLSI circuits layout and to a lesser degree in printed circuit board environment is the ability to use multiple copies of the same module in a routing context. For example, to build the register array of a microprocessor, the designer would lay out a typical cell that implements one bit of the register. Stacking thirty two of these bit cells together gives rise to a thirty two bit register which forms a unique module. In composing the data path of the microprocessor, the designer can put the same register module in places which need a register, e.g. as I/O data registers or as address registers. The various copies of the registers are different *instances* of the same register module. A hierarchical routing specification must allow efficient instantiations of modules in order to avoid redundant details.

2.3.2. Three Dimensional Hierarchy

The hierarchical model described in the previous section was formulated in an environment with limited conducting layers typical of the late 1970s

technology. Advances in fabrication technology since then promise a multiple layer environment in which many conducting layers are available for wire routing. As a general rule, as more and more layers are deposited on the wafer, alignment of features on the wafer becomes much harder because of the much more rugged wafer surface as a result of depositing the lower layers. The minimum feature size of the top layers must then be increased to allow for larger tolerance of feature mismatch. The analysis in section 2 shows that the larger feature size is not only necessary but also desirable. The thicker and wider conducting wires on the upper layers become excellent candidates for distributing global signals.

My advisor in this research suggested to me a natural extension of the hierarchical model by imposing a hierarchy on the usage of conducting layers. Recall that active logic are realized by using only the lower layers whereas the upper layers are provided for routing signals. A restriction on the top most conducting layer to be used is predetermined by the designer for every routing context (see Figure 2.6). As larger modules are being composed from the smaller ones, the routing *ceiling* is stepwisely relaxed to allow for usage of the upper layers. The hierarchical composition procedure forms a natural framework in sorting out the local signals from the more global ones.

The additional hierarchy imposed on layer usage allows a new alternative in bringing out signals from a module. By restricting usage of conducting layers, higher levels of composition may have an entire free routing layer over areas occupied by the lower level modules. *Internal ports* can now be introduced inside the lower level modules which bring signals out of the module through contacts to the upper free routing layers (see Figure

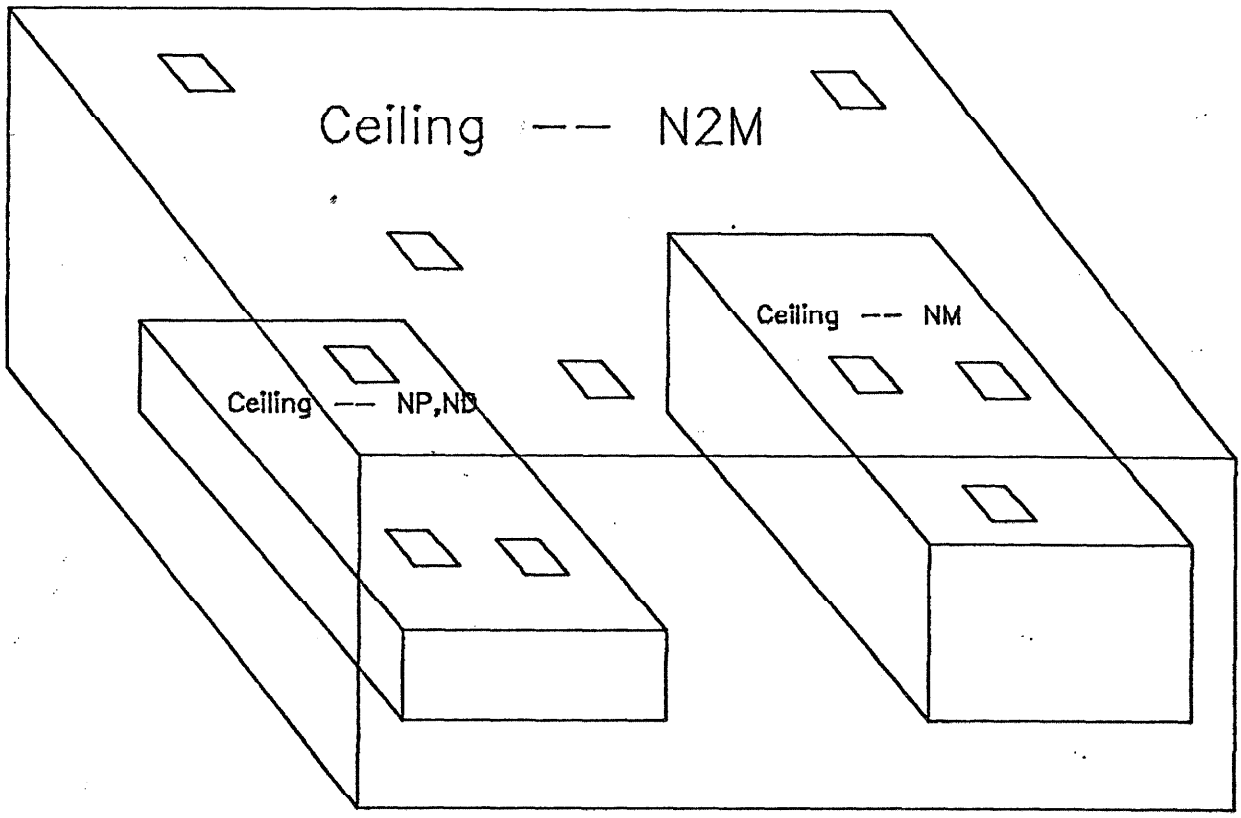

Figure 2.6 Hierarchy in Three Dimensions

2.6). The "black boxes" now have ports not only around their perimeters but also on their top faces. An internal port would simply be a receptacle formed by the lower part of a contact upon which the rest of the cut can be put in by the router.

This extension of the hierarchical routing model into the third dimension requires new algorithms to take advantage of the extra degree of freedom. Automatic placement and routing have to support deferred routing of signals until a more favorable layer is available, and much remains to be done towards such a goal. The next chapter will be concentrating on the second goal of this thesis: that of expermenting with a stepping approach in the more restricted form of routing.

CHAPTER 3

ROUTING ALGORITHMS

In this chapter, algorithms needed to achieve the routing requirements are described. These algorithms are designed to work with the model we discussed in the last chapter. The main motivations behind the algorithms are simplicity and efficiency. The goal is to provide fast turn around time wire routing.

3.1. The Stepping Approach

Wire routing of power and signal connections in general cell composition is inherently a two dimensional problem once the routing layers and principal directions are specified[1]. The conventional cellular approach[2] tries to solve it in a two level hierarchy : a top level that operates on channels, and a lower level that operates on pins within the channels. But within each level, it solves in a two dimensional fashion. This is especially true for the top level in which a channel graph is usually built and solved subjected to the various objectives. In this thesis, an alternate approach is described. In comparison, the stepping approach tries to solve the routing problem by reducing an inherently two dimensional problem down to one dimensional through incremental feedback and late binding of signal paths.

The stepping approach is also organized in two levels of hierarchy to reduce complexity. Both levels are processed in a stepwise fashion from one

[1] See section 2.1

[2] See section 1.3

edge towards the opposite edge, routing everything it sweeps across. The top level is responsible for propagating global wires from a local region to those beyond whereas the lower level tries aggressively to connect as much as it can within the local region. A major difference is that the two levels are processed alternatively rather than in a strictly sequential manner as in conventional approaches. Also, the global wiring paths are not completely determined at the top level. Instead, it determines the set of signals which need to be propagated and provides information for the lower level algorithms to select the actual paths. That decision is then fed back to guide further propagations. Local decision process is more apt to user interaction simply due to the locality of its consequences. The stepping approach provides a framework for easy tracking of failures made possible by its well defined processing order.

3.2. Channel Definition

The first step in the stepping approach is to divide the routing plane into a set of disjoint *channels*. These channels form the basic unit upon which the *Global Propagation Algorithm* steps across one at a time from the bottom edge of the routing plane towards its top edge. Each channel defines a free routing region within which the low level algorithms operate. The intention is to limit the amount of information and let the system make decisions and set up goals only on the basis of information available inside the current channel.

> A **channel** is a maximal horizontal rectangular strip bounded between adjacent modules (see Figure 3.1).

The left and right edges of a channel are always bordered by edges of placed modules. The top and bottom edges consists of combinations of module

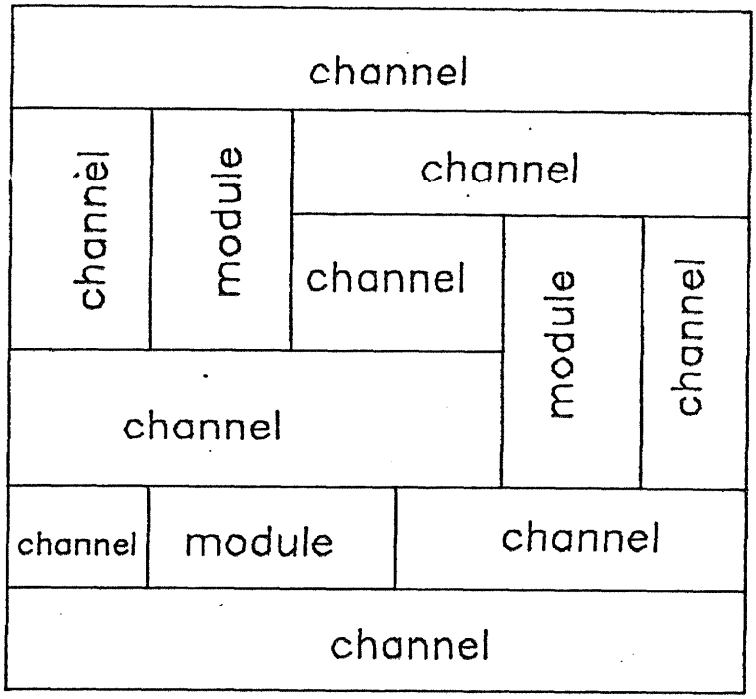

Figure 3.1 Slicing of the routing plane

edges and free *segments*.

> A **segment** is part of a channel's north (south) edge which is also a part of the south (north) edge of the channel above (below) it.

These segments form the doorways through which global signals can propagate into and out of the channels. One motivation behind selecting maximal horizontal channels is to avoid dealing with channel cross pin assignments in both dimensions. It is also the most natural in a stepping approach. A maximal horizontal strip is the largest monolithic rectangular free routing region that forms a stepping unit.

3.3. Metric Initialization

After the channels are defined for a certain given module placements, the stepping router proceeds to initialize the global metric between channels. The metric between channels is useful in estimating propagation

costs for the global signals. The metric calculation described below is designed to capture the unique behavior of the stepping approach. Given a fixed channel definition, the stepping router assigns a fixed processing order to each channel. This order predetermines and restricts at each stepping stage, the remaining set of free routing channels available for global connections.

The channels may be represented as the vertices of a directed acyclic graph. The existence of a directed arc from vertex A to another vertex B implies there is a segment adjacent to both channels and also that channel A is processed before channel B (see Figure 3.2a). We will assume that the channel graph for the placement has a unique source and sink. A topological sort can be applied to the channel graph to assign the processing ordering of the channels. Each channel is given a unique channel number which

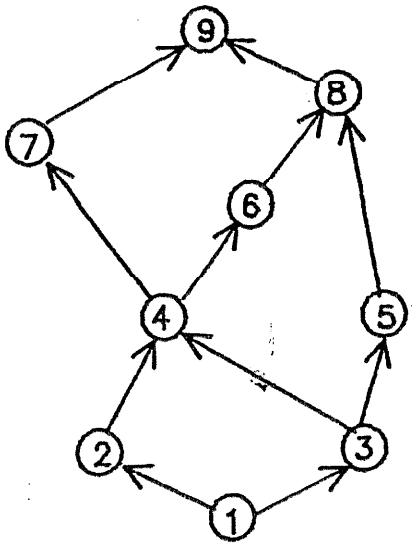

Figure 3.2a Channel Graph

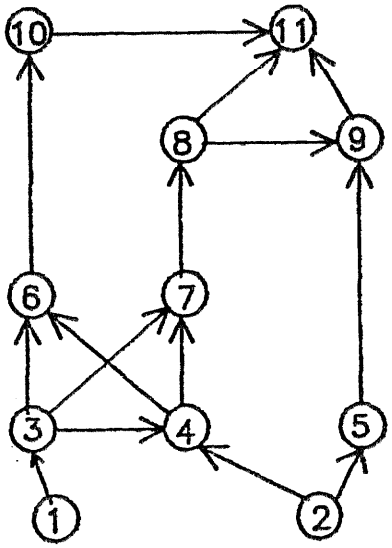

Figure 3.2b Segment graph

represents the order the channels are to be visited. The topological sort is organized so as to give a balanced visit to channels of same y-coordinate range in order to maintain a balanced sweep across the routing plane. It is obvious that the global propagation paths of signal wires must follow the channel graph flow pattern. The same topological sort also assigns a unique segment number to each segment adjacent to the channels it visits.

Given a fixed channel visit order, it is clear that any remaining unconnected ports of the global signals within the current channel must be located inside channels of higher channel number. Hence, to estimate the global propagation costs, it is only necessary to initialize those metric between the current channel and the ones beyond it. Associated with the channel graph is the segment graph (see Figure 3.2b). The segment graph of a fixed module placement is a graph whose vertices are the segments of the channels. A directed arc exists from segment A to segment B provided:

(1) Both segment A and segment B are adjacent to the same channel.

(2) Segment A is a south segment of the channel.

(3) Segment A has a segment number smaller than that of segment B.

The stepping router attempts to calculate the graph metric from a segment of low segment number to a segment of higher segment number. The location of a segment is represented by the coordinates of its center. Let,

$Tr(A)$ = transitive closure of segment A
$Tr(B)$ = transitive closure of segment B
$Tr(A,B)$ = $Tr(A) \cap Tr(B)$
$d(A,B)$ = distance from segment A to segment B

then :

$$d(A,B) = \min_{X \in Tr(A,B)} \left[d(A,X) + d(B,X) \right]$$

The metric calculation proceeds as follows: First, entries in the metric table are initialized to infinity. Then the distance between adjacent segments are initialized as the Manhattan distance between the segments' centers. An algorithm based on the Floyd-Warshall shortest path algorithm with decomposition [Hu 82] is then applied to obtain the shortest path between each pair of vertices.

From the criteria in inserting directed arcs to the segment graph, it is clear that the segment graph have only a unique sink (assuming unique sink in channel graph), but it could have several sources. The successors of a vertex must lie in the path from that vertex to any other vertex belonging to its transitive closure. Hence the successors of a vertex form the bottleneck needed for decomposition. The shortest path algorithm starts from the unique sink and works backward according to the segment numbers. Since all descendents of a vertex have segment numbers higher than their ancestor, therefore, the triangle operations are well defined for all pairs of vertices. Notice in the calculation of distances, it is always from a vertex of low segment number to one of higher segment number.

```
INPUT:      segment graph of a placement
OUTPUT:     metric between every pair of vertices

ALGORITHM:
for i:= sink downto 1 do
begin
        for j:= i+1 upto sink do
                begin
                        while (k:=descendent(i)) ≠ φ do
                                d(i,j) := min (d(i,j), d(i,k)+d(k,j))
                        d(j,i) := d(i,j)
                end
end
```

If we adopt the convention of referencing metric only from a low segment to a high segment, then a triangular array is enough. The time complexity of this algorithm has its outer two loops of $O(n^2)$ in the number of segments and hence in the number of channels (since the channel graph is planar). The innermost loop depends on the outdegrees of the vertices. For most placements, it is usually small compared to the total number of segments and equals one for a large fraction of vertices. Hence, the overall complexity is $\sim O(n^2)$ for almost all cases.

The same algorithm can also be applied to the channel graph to obtain metrics between pairs of channels. The purpose of these metric initializations is to precompute and save the data needed for global cost estimation, thus avoiding redundant calculations for every signal and channel during the *Global Propagation Phase*.

3.4. Global Propagation

The operation of the stepping router can be thought of as a sequence of cycles swapping between the *Global Propagation Phase* and the *Local Routing Phase* for each channel. The global phase is responsible in determining the set of signals which need to be propagated beyond the current channel and estimating the relative costs for these signals among the set of segments which they may be propagated.

> The **kernel** of a signal is the set of channels which contain unconnected port(s) of the signal.

Ports of a signal within the same channel are collectively represented by their centroid in the cost estimation of global propagation. The main objectives of the global propagation phase are:

(1) Determine the set of signals active inside the current channel which have their kernels containing channels beyond the current one. These signals must be propagated upwards in order to make connections to those remaining unconnected ports. A simple test on the cardinality of a signal's kernel readily determines if the signal needs propagation.

(2) For each global signal within the channel, each segment represents a possible propagation doorway. To determine the best choice, it is necessary to assign propagation cost to each of the segments for the signal. These costs are used by the local routing algorithms to determine the actual propagation crossing locations. The metric table provides the lookup values for the cost calculations.

The global propagation cost reflects the distance a signal has to travel among the set of segments it can choose. Below we give the exact steps of the propagation algorithm.

(1) Determine the set of signals not completed after the current channel. This amounts to no more than checking the cardinalities of the kernels of the signals active in the current channel and single out those which contain channels beyond the current channel into a candidate list. If list is empty or the channel has only one segment, return, else proceed to the next step.

(2) Come here when list is nonempty. Compute the effective cost of each signal for each north segment of the current channel by carrying out step (3), (4) and (5). These costs represent expected distances in routing through the segments.

(3) For each signal, select from its kernel the channel closest to the current one according to the channel metric table.

(4) For each north segment of the current channel, the graph distances from the center of the segment to the centroid of signal ports of the target channel are calculated. The effective cost is then calculated by adding their minimum to the grid distance from the centroid of signal ports of the current channel to the segment. This approximates the global distance the signal has to travel to the nearest channel inside its kernel through that segment.

(5) After the costs of each segment for the particular signal have been calculated, they are normalized by their minimum. These new costs will be used to compare with the *coefficient of tolerance*[3] of the signal in the Local Routing Phase.

It should be pointed out that global signals do not form multiple branches by propagating through several segments simultaneously. This restriction together with the stepping restriction occasionally may lead to extremely inferior path topology.

Assuming the indegrees and outdegrees of the channel graph vertices are small compared to the total number of channels, the time complexity of the global propagation algorithm of a channel is of $O(ns)$ where n is the number of channels and s the total number of signals. For those channels with outdegree being one, the algorithm simply returns after step (1), and hence has time complexity of $O(s)$.

3.5. Local Routing

The global routing phase described above determines the tentative set of signals needed to route pass the current channel. Within each channel, a

[3] See next section.

detail local routing phase then follows to complete the connections geometrically, and makes any adjustments to the global signals as a result. The local routing phase is the most complicated phase of the whole stepping approach.

3.5.1. Problem Definition

After the global propagation phase of each channel, the list of signals needed to be routed beyond the current channel has been determined. In general, a signal can be routed through one of the several segments. The global propagation phase simply determines the relative costs for each of such decisions. Hence, the local routing phase inside each channel has to accomplish the following:

(1) For the set of signals to be routed beyond the current channel, determine which segment they must be placed on.

(2) Generate detail wiring pattern connecting the signals and their pins in the channel. The result must be correct subjected to the geometric design rules.

Again, signals are to be routed on two layers, using one for the vertical and the other for the horizontal direction. A *switch box problem* (see Figure 3.3) is specified as :

(1) A rectangular routing channel, of length l and height h, where absolutely all routing has to lie within the channel area.

(2) Four lists of pins $T = (T_1,\ldots,T_t)$, $B = (B_1,\ldots,B_b)$, $L = (L_1,\ldots,L_l)$, and $R = (R_1,\ldots,R_l)$, correspond to pins on the four sides of the channel. Associated with each pin is its signal net number and position. Pins of the same signal net are to be connected together.

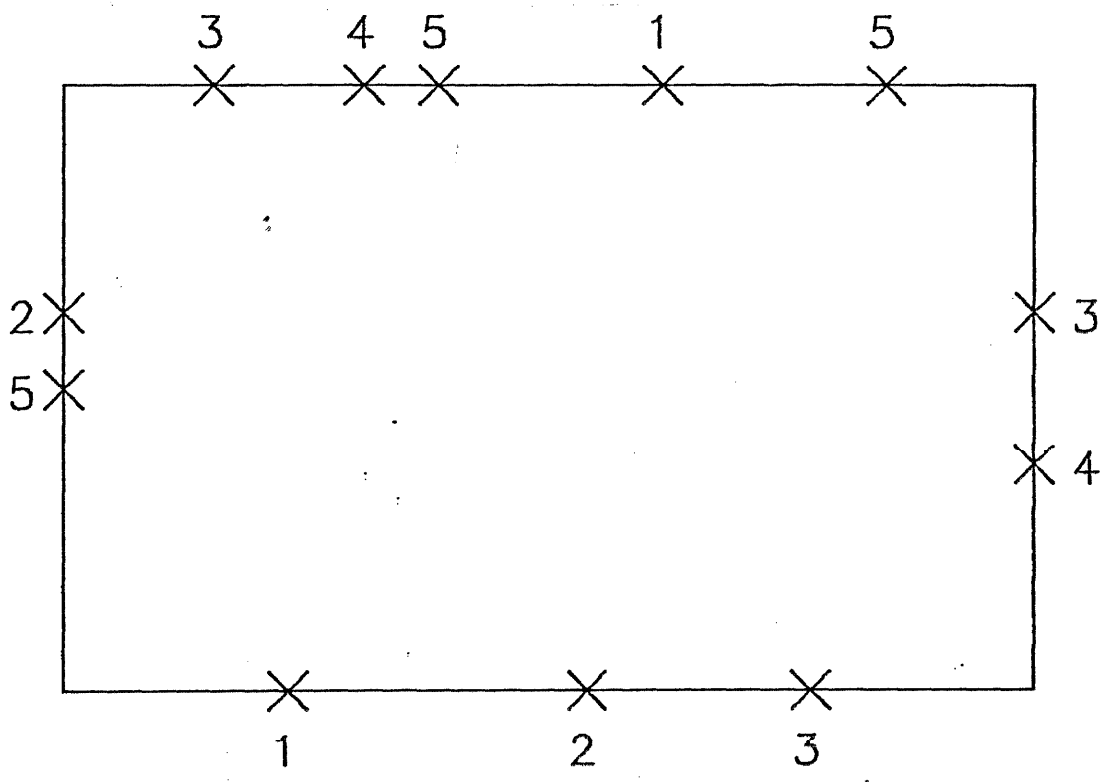

Figure 3.3 A switch box problem.

(3) All pins must lie along the intersections of the channel perimeter and the vertical or horizontal grid lines.

In practice, pins do not always line up with the routing grid. The local routing algorithm must then be responsible for jogging the mismatched pins onto the grid lines.

A solution to the switch box problem (see Figure 3.4) specifies for each signal net inside the channel a set of connected horizontal and vertical *wire sections* whose end points are either a pin of the net or a *contact cut*. Sections in the same direction run on the same layer, so they may not touch if they belong to different signal nets. Two sections for the same signal net in different directions are connected together by a *contact cut*. If the sections belong to different nets, they form a *crossover*.

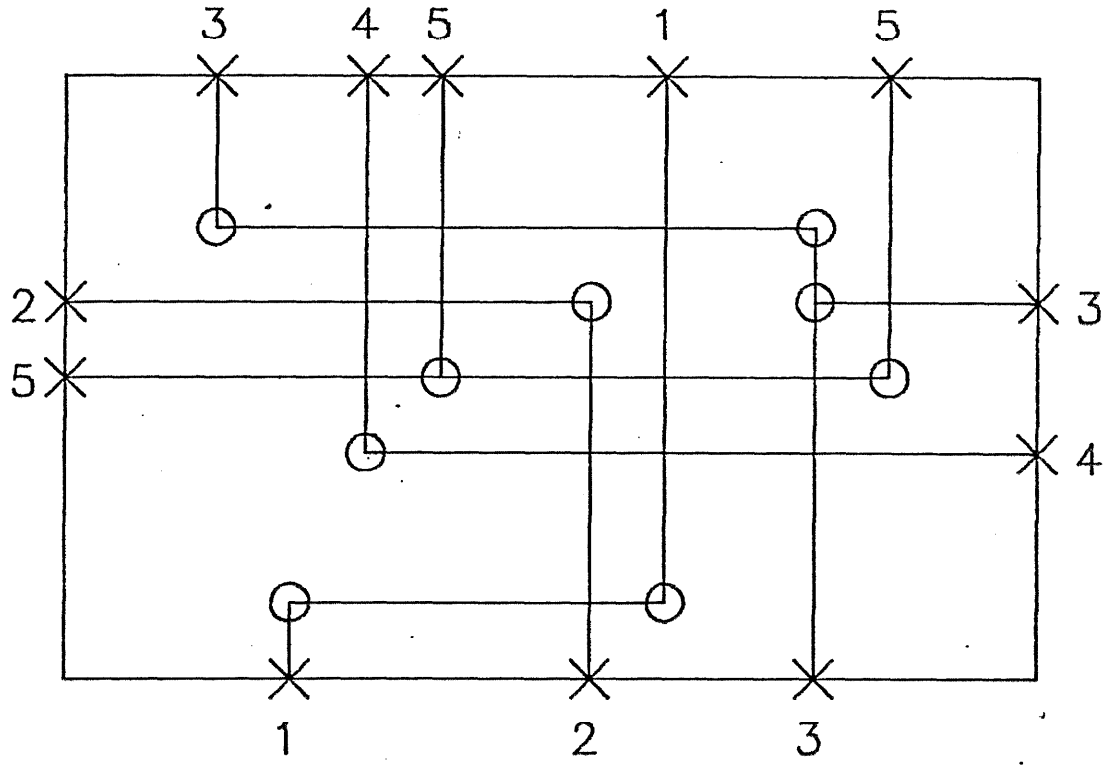

Figure 3.4 A Solution to the switch box problem.

The switch box problem is much harder then the simpler *channel routing problem*. Much work has been focused on good heuristics for solving channel routing problem, but little work has been done on the general switch box problem. [Hsu 82] and [Smith et al. 82] are two recent work on this problem. In this thesis, the Greedy Channel Router of Rivest and Fiduccia [Rivest 82b] is modified to handle the general switch box problem. The motivation is to take advantage of the speed and performance of the greedy router and the flexibility of its control structure to handle global signals propagation.

3.5.2. The Routing Algorithm

The switch box problem is very general in that it has to deal with connections from all four sides. This is inherently a two dimensional problem. To reduce the complexity, we adopt the same linear scanning

approach as in the global phase by compressing it into a one dimensional problem. Here are some definitions:

> A **raster** is a horizontal line along which, horizontal wires may be laid down. It is the basic operating unit in which most of the actions of the router are to take place. The router steps through the rasters from bottom to top. There is no backtracking.
>
> A **column** is a vertical line along which vertical wires may be laid down.
>
> A **branch** of a net inside the channel is a horizontal portion of wire connection needed to connect two adjacent net elements (a horizontal pin or a vertical wire). A branch is said to be **closed** if the connections has been made, else it is said to be **open**.
>
> A **subnet** of a net inside the channel is a set of adjacent open *branches*. A net has only one subnet (that of itself) before any wire connection is made. Connecting a branch in a subnet will either split the subnet into two new smaller subnets or shrink the old one if it is at the end of the original one (see Figure 3.5). Dogleg is allowed only to the wires originated from extreme pins of a subnet in order to avoid too many contact cuts.
>
> A horizontal pin is **free** if a vertical wire can be introduced into the channel with one end of it connected to the pin. All up-growing vertical signal wires are *free*.
>
> A vertical signal wire is **fixed** if it is growing towards a horizontal pin on top of the channel with the same signal and on the same column.

The greedy router scans across the channel from bottom to top in a raster by raster manner. It tries to generate as many connections as possible in a raster before it moves up to the next. The original greedy router follows a set of rules to create connections according to fixed priorities. In the conventional channel routing problem, pins with fixed locations come only from the vertical sides and the total number of columns available is the main constraint to be satisfied. Thus, the rules used in the original greedy router give exclusive priority to collapsing split nets into single wires in order to reduce the pressure on column demand. A new class of constraints emerged

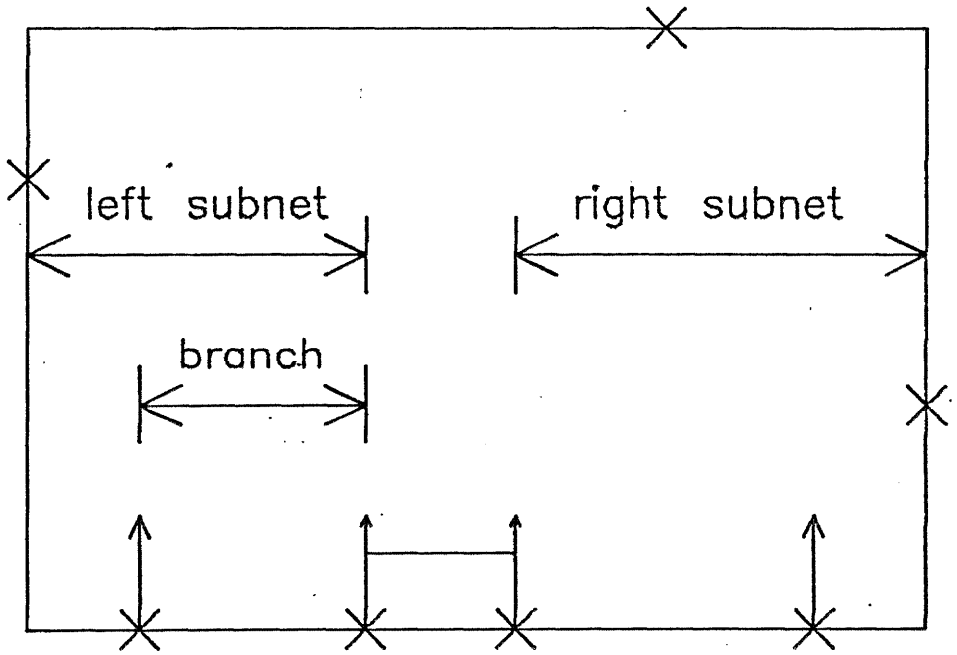

Figure 3.5 Examples of subnets and branches

in extending the greedy approach to handle the switch box problem. In addition to fixed pins on the vertical sides, there are now also fixed pins along the horizontal sides of the channel. A simple column collapsing scheme may end up in generating zig-zag wires across the channel (see Figure 3.6). Further, a signal may have multiple pins on the top side of the channel, in which case, the signal has to split to make connections to them. To resolve these complications, new rules must be added to determine collapsing of split nets and produce *fixed connections* to the top side of the channel. Once a signal wire is fixed, it will grow towards the top side and cannot be jogged away.

The rule based control structure of the greedy router can be easily extended to handle the floating global cross pins along segments. Following the same greedy approach, all connections to pins within the channel are given higher priority than connections to cross pins. The local routing

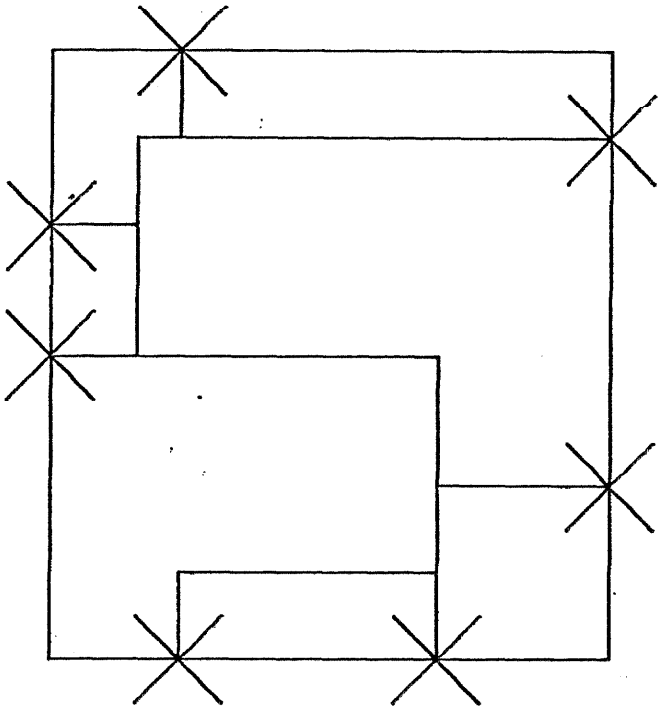

Figure 3.6. Zig-Zag wires generated by simple column collapsing

algorithm proceeds by repetitively laying down horizontal connections on the current raster which divides the raster to a left and a right free sub-raster. The same process is then applied to the sub-rasters. This allows for a recursive implementation. During every pass of the algorithm, local connections are always considered first, and the corresponding wires laid if one exists. When no local connection is available, cross pin connections are then considered. Such priority is chosen because of the flexible cross pin locations often have higher completion chances. Note the difference from the original greedy router which employs a complete combinatorial search for making as many split net connections as possible. Such an approach is infeasible in a switch box router since the number of split nets can be much larger than that of a conventional channel. Below we present the steps of the algorithm:

(1) **Create Pin List:** Scan the four list of pins, *T*, *B*, *L* and *R* to form a linear list of pins in the following order: left sided pins in decreasing y coordinates, top and bottom pins in increasing x coordinates, right sided pins in increasing y coordinates. If a top and a bottom pin share the same column, insert the bottom pin in front. Assign an order number to each pin in the list.

(2) **Straight Connections:** Scan along the pin list, if two pins share the same column and are of same signal, mark the lower pin as *fixed* and delete the upper pin.

(3) **Partition Columns by Segments:** Partition the columns into different sets according to which segment it is underneath. Recall that segments are the doorways for global propagations. The router tries to relocate wires carrying global signals into columns belonging to their preferred segments having the lowest propagation cost.

(4) **Initiate a Raster:** This marks the beginning of routing a raster. While the channel is not completely routed, a new raster is initiated. If no more raster is available, goto step (9) for channel clean-up.

(5) **Introduce Side Pins into Channel:** Look at the left pins and right pins, introduce them into the channel in a minimal fashion by routing them to the nearest free columns. This step introduces the edge pins into the one dimensional horizontal raster. Adjust the pin list to reflect this action. We now have the whole set of horizontal pins for the current raster. Next, recursively lay down horizontal connections on the current raster.

(6) **Recursively Connect Open Branches or Introduce Doglegs:** Given a section of the horizontal pin list corresponding to a sub-raster, locate

the branch or dogleg whose horizontal connection spans the most number of unconnected pins. This section will be routed thus removing the most congesting section preferentially. The raster is then divided into two remaining portions. Recur this step on the two portions until no more connections are possible.

(7) **Collapse Split Nets to Reduce Wiring Area:** After a horizontal connection is made between wires of a split net, the two wires are always collapsed unless both wires are fixed or both belong to different subnets (see Figure 3.7). When both belong to different subnets, the relative locations of their adjacent pins determine the choice. Referring to Figure 3.7, if $X \leq Y$ then the two wires will be collapsed else they are allowed to grow towards their neighbors.

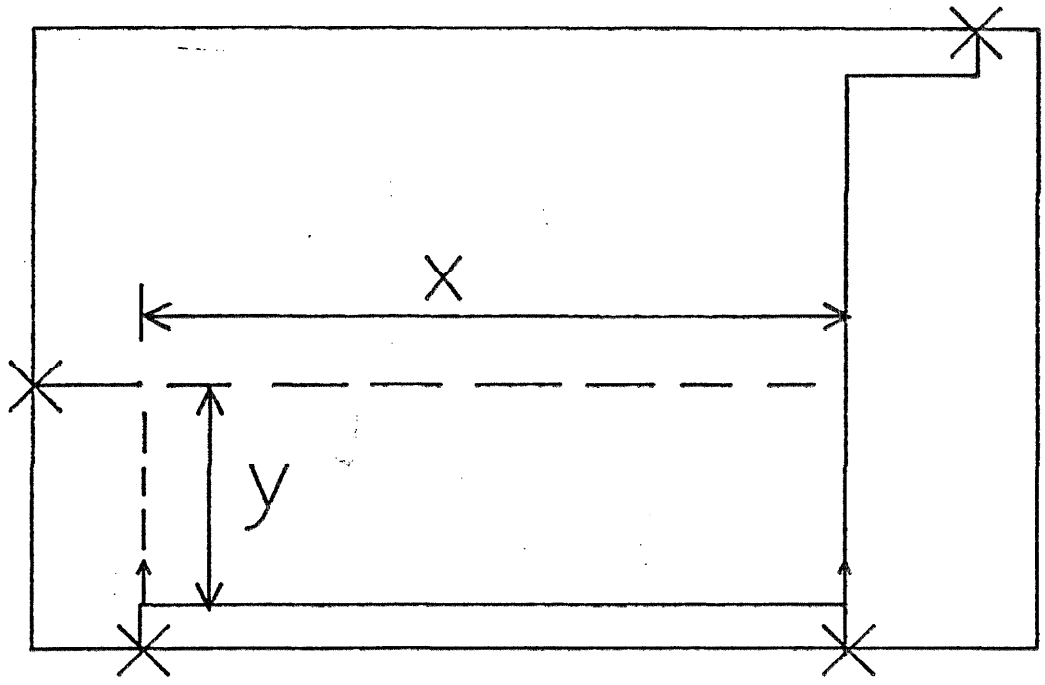

Figure 3.7 Collapsing split net

(8) **Cross Pin Connections:** When a sub-raster is void of local connections, cross pin connections are considered. For each segment directly above the sub-raster, scan the list of active global pins and select the one with least propagation cost and smaller than its coefficient of tolerance. If such a pin exists, make the horizontal connection and insert the cross pin on the top side of the channel by laying the shortest horizontal jog possible.

(9) **Channel Clean-up:** After the last raster inside a channel has been routed, a clean-up follows. The router checks the list of vertical wires against the list of top pins and cross pins. Any failure in making these connections is reported. Uncompleted connections within the set of vertical wires are allowed to propagate into the upper channels if the situation allows, otherwise failures are reported.

3.5.3. Layer Assignment

As pointed out in section 2.2, the conducting layers used in routing typically have different electrical properties. A layer with a smaller propagation time constant is usually a better choice than one with a higher time constant. The hierarchical model described in section 2.3 follows these asymmetries to impose a hierarchy on layer usage. However, within each routing context, two different layers are available. It is desirable to route as much wiring on the better conducting layer as possible provided the additional overhead is limited.

The local routing algorithm described above can be readily adapted to optimize usage of the better conducting layer. The local router always route the better conducting layer on the horizontal dimension and the inferior one

on the vertical dimension. For example, in nMOS technology, the router makes connections with horizontal metal wires and vertical polysilicon wires. The optimization involves replacing the poly wires by corresponding metal wires wherever possible.

During routing of each raster, horizontal wires are laid down when the router decides to make a horizontal connection. Vertical wires are treated differently. Making a vertical connection involves two steps: initiating the connection from a lower endpoint and terminating the connection at the upper endpoint. The actual wire is put in when the router decides to terminate the connection. Thus the wire always starts from a point below the current raster and ends at a point on the raster. The decisions to lay vertical wires always come after all the corresponding intersecting horizontal connections are made. The router simply keeps track of the intersecting horizontal connections made and put in metal wires if no such connection exists.

3.5.4. Discussion

The routing algorithm described tries to route the channel in a greedy fashion. It is greedy in its attempt to remove the most congesting section of horizontal wires within each raster section. It is best to view this as an approach to bin packing. At any moment during the algorithm execution, we have a range of horizontal wire sections which can be laid down. By removing the most congesting section first, we try to minimize congestion among the rest of the horizontal sections. By laying down horizontal wires, we are subdividing the rasters into smaller fragments. To enhance the chance of a successful fit between the raster fragments and the horizontal wire sections,

and thus maximizing the usage of the rasters, it is important to keep the remaining horizontal wire sections as short as possible. In practice, there is a strong empirical correlation between congestion and wire length. The direct impact, however, is on future rasters. By removing the most congesting section, the remaining sections are more likely to have their conflicts more localized. This is helpful in sorting out the congestions into disjoint blocks which can all be connected in upcoming rasters.

3.6. Power Routing

As mentioned in section 2.1, routing of power and ground busses has its own special requirements which single it out for separate processing from the other signals.

3.6.1. Planarity Restriction

The planarity requirement is intrinsically a global property of the entire layout whereas the stepping approach utilizes only local information. To approach a problem that requires global considerations from a local point of view, restrictions must be imposed which defines a subset sharing a common structure that can be exploited to reduce the problem to only local considerations. In this spirit, we choose the following:

(1) The VDD and GND signals do not both appear on the same edge of the same module.

(2) Opposite edges of the same module do not have pins of same polarity (see Figure 3.8).

The closure property for hierarchical design requires the external pins of the current module also to satisfy the above restrictions. Restrictive as it

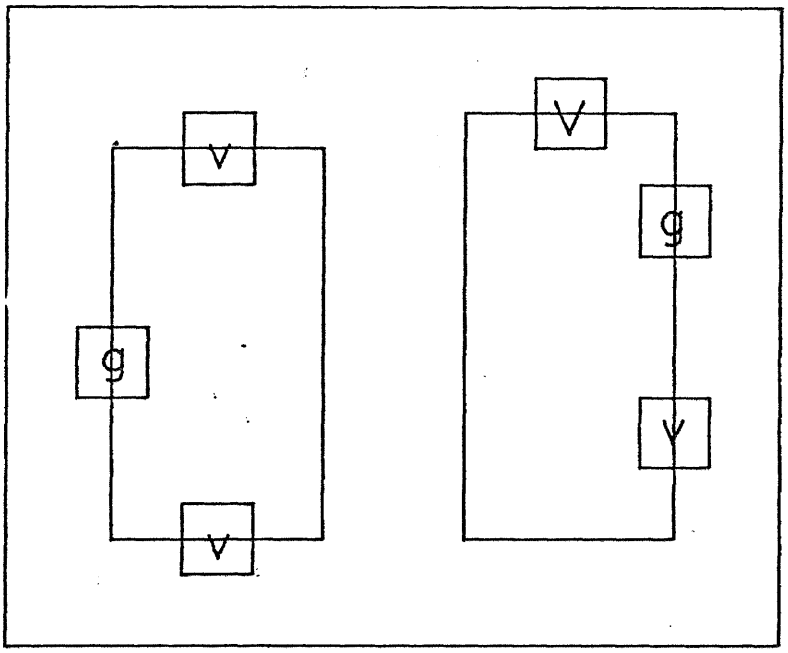

Figure 3.8 Violations of power routing requirement

might sound, it encompasses an overwhelming majority of module layouts. In fact, it actually helps in encouraging good and consistent design style.

To see that planarity is guaranteed, notice that the restrictions on VDD and GND pins along a module make it possible to represent the module by its homeomorphic reduction consisting of two vertices jointed by an edge (see Figure 3.9). The two vertices represent the VDD and GND pins respectively and the edge represents the area occupied by the module. The presence of an edge between the vertices provides an abstraction for the property that no wire can cross over a module's area in a planar embedding of the wiring connections. The super-module is also represented by its homeomorphic reduction. In order to distinguish between the inside and the outside of the super-module, two parallel edges are retained. Figure 3.10 shows the effect of placing a module inside the super-module and connecting the VDD and GND wires. It is simply a partition of a region into two sub-regions. The resulting

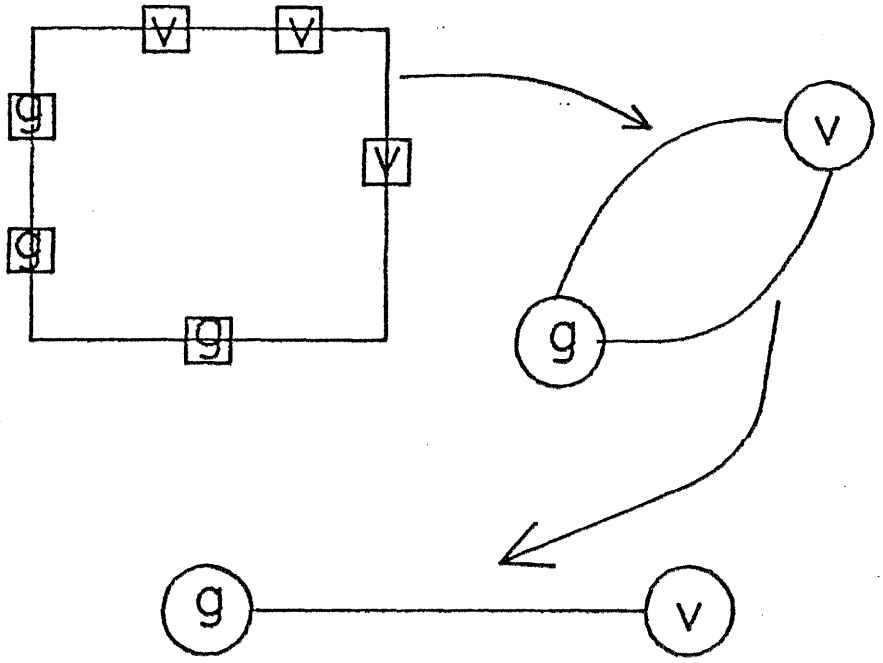

Figure 3.9 Homeomorphic reduction of a module

graph is then the planar embedding of wiring connections including the newly inserted module.

It must be pointed out that although planarity is guaranteed in the graph-theoretic sense, the actual layout of power and ground busses may not be feasible due to lack of wiring area.

3.6.2. The Algorithm

Given the restrictions described in the previous section, there exist a number of algorithms to systematically lay down the power and ground busses [Syed 82]. In this section, a new algorithm is presented which proceeds also in a stepwise fashion from one side of the layout to the other. This algorithm can be readily integrated with the previous routing algorithms described since they share the common stepping framework.

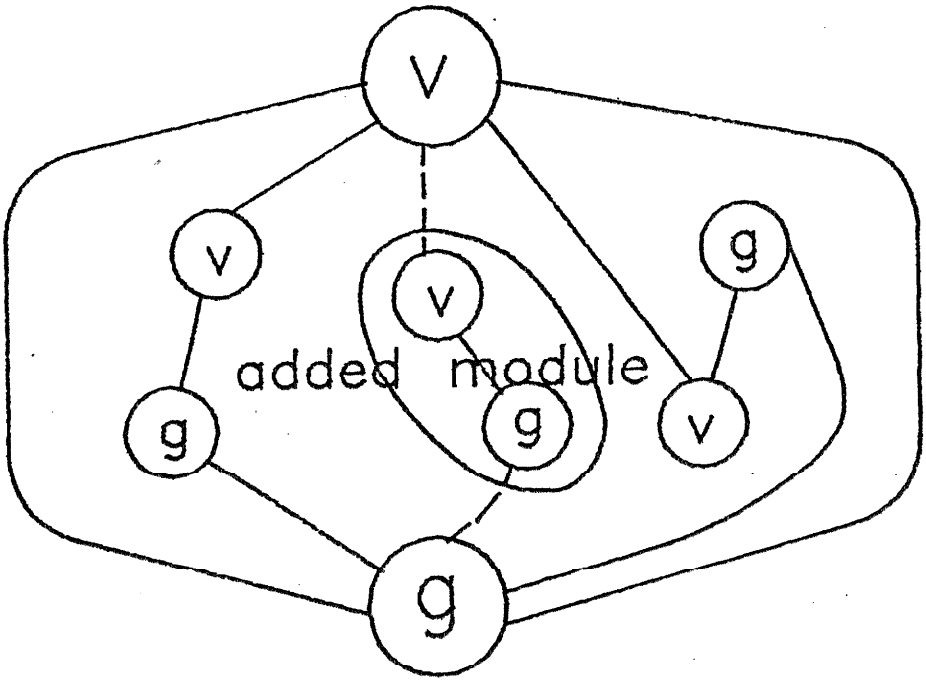

Figure 3.10 Planar embedding of Power and Ground busses

For ease of presentation, neglect the power and ground pins of the super-module for the moment. The power routing algorithm operates in four steps: two for the VDD net and two for the GND net. The power and ground busses are laid down along opposite edges of the channels. Since the VDD and GND pins may lie on any side of the modules, they could appear on an edge that has been assigned the opposite polarity. For these pins, it is necessary to route them to an edge of identical polarity. Thus the first step is to assign fixed polarities to the four edges of the channels such that the number of local relocation of power and ground pins needed is minimal.

Having fixed the polarities for the four sides of the channels, the next step is to relocate the VDD and GND pins which conflict with the polarity assignment. The power and ground nets are handled separately and their processing proceed in opposite directions. Suppose the south and east sides

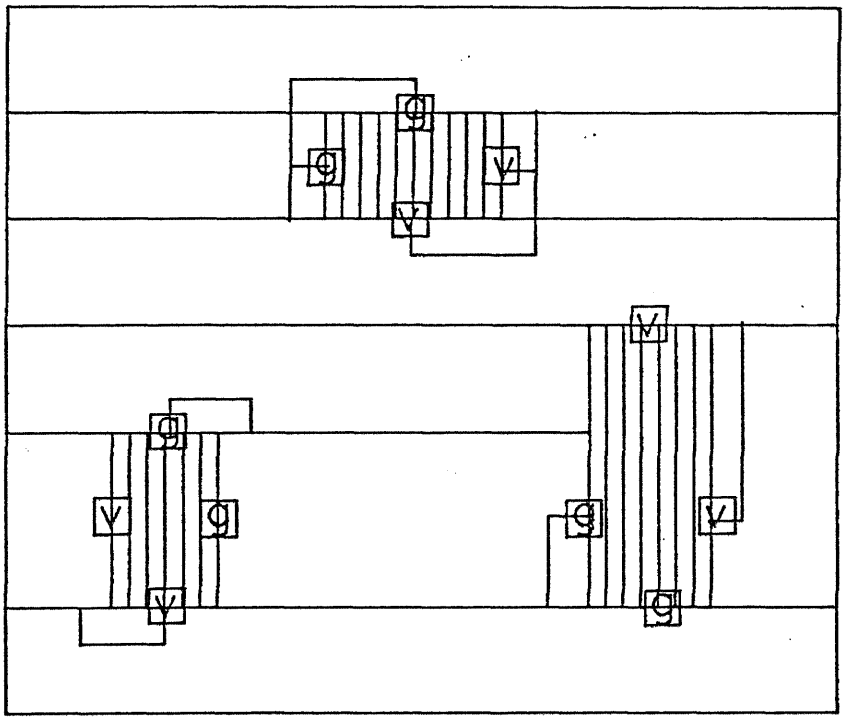

Figure 3.11 Relocation of VDD and GND pins

of the channels are assigned to route VDD busses, then the local relocation of VDD pins starts from the south side of the super-module and proceeds towards its north side (see Figure 3.11). As the algorithm steps across each channel:

(1) VDD pins on the west edge of the channel are relocated towards its north edge.

(2) VDD pins on the north edge of the channel are relocated to the west side of the module they belong if

(a) the west side of the module does not have any GND pins on it, and

(b) the east side of the module does not have any VDD pins on it.

Otherwise, they are relocated to the east side of the module.

Relocation of GND pins follow the same pattern but proceeds from the north side of the super-module towards its south side with pin and edge polarities

both reversed. The execution of this two pass algorithm ensures the matching of pin polarities with edge polarities.

With the exact matching of pin and edge polarities, another two pass across the routing plane can then join all the power and ground pins together into two planar nets. Again scanning upwards for the VDD net, all VDD pins along the south and east edges of the channel are joined together and routed towards its north edge (see Figure 3.12). The GND net is routed in exactly the same manner but in opposite direction. The result is a planar layout of the VDD and GND busses. External VDD and GND pins can now be simply added along the east/north edges and west/south edges respectively and be connected to the power and ground busses.

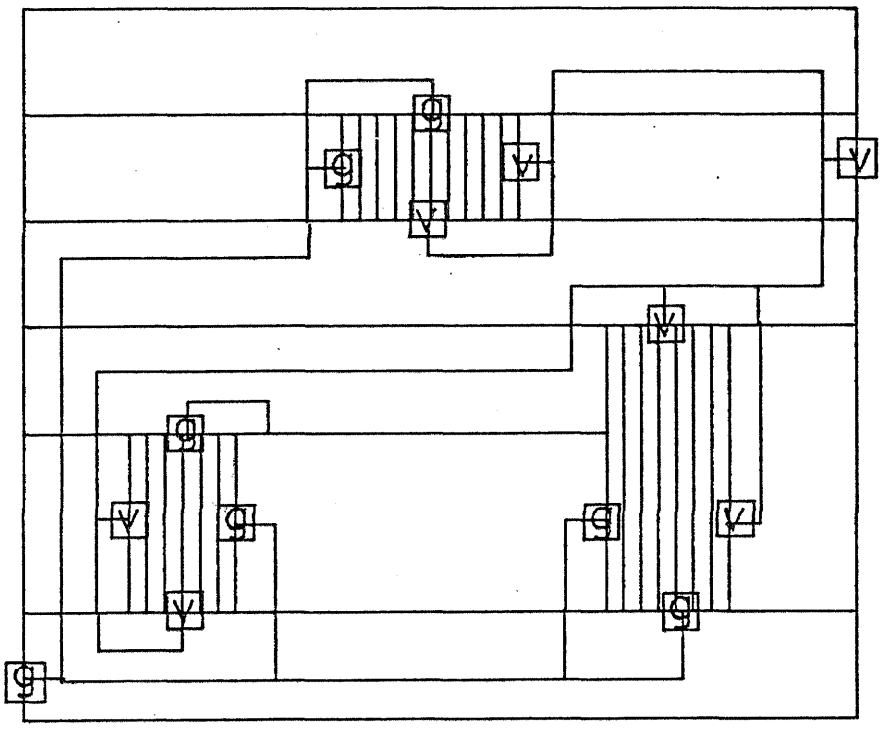

Figure 3.12 Global connection of VDD and GND nets

CHAPTER 4

Implementation and Results

In the last chapter, we described the various stepping algorithms used to arrive at the final routing layout. In this chapter, we describe briefly the implementation of SMART (Simple Minded Approach Routing Tool) by the author and some of the preliminary results obtained. A few suggestions for further research concludes the chapter.

4.1. SMART Implementation

SMART is a general interconnect tool for large-scale MOS custom integrated circuits. It implements the algorithms described in the last chapter according to the routing model presented in chapter 2. SMART currently supports single layer metal nMOS and CMOS/SOS processes and outputs its routing layouts in CIF.[1] It is written in C and runs under Berkeley UNIX Version 4.2.[2] The goal is to provide fast turn-around time wire routing. It currently runs at Caltech as a stand alone system. It is expected to be integrated with other existing layout software at Caltech in the near future.

4.1.1. System Organization

Figure 4.1 shows an overview of SMART's building blocks. The front-end user's interface consists of a LEX generated lexical scanner and a YACC generated parser together with their service routines. It handles all routing

[1]Caltech Intermediate Form

[2]UNIX is a trademark of Bell Laboratories

specifications and system commands issued by the user. SMART can be invoked with an optional list of input file arguments. The files are read in one by one in the same order as they appear in the argument list. SMART checks against all inputs for syntactic and semantic (against design rules and routing model) correctness when they are read in. It also keeps track of the modules defined in the input files, their hierarchical dependencies and creates the system's image of the modules read in. This image can be listed and is helpful in comparing what the user thinks and what SMART thinks regarding the modules.

The rest of SMART's building blocks come into play when the user issues the *route* command. Upon receiving the command, SMART's routing control block retrieves the internal images of the designated modules. It performs a depth-first search on the hierarchical composition dependency graph (which

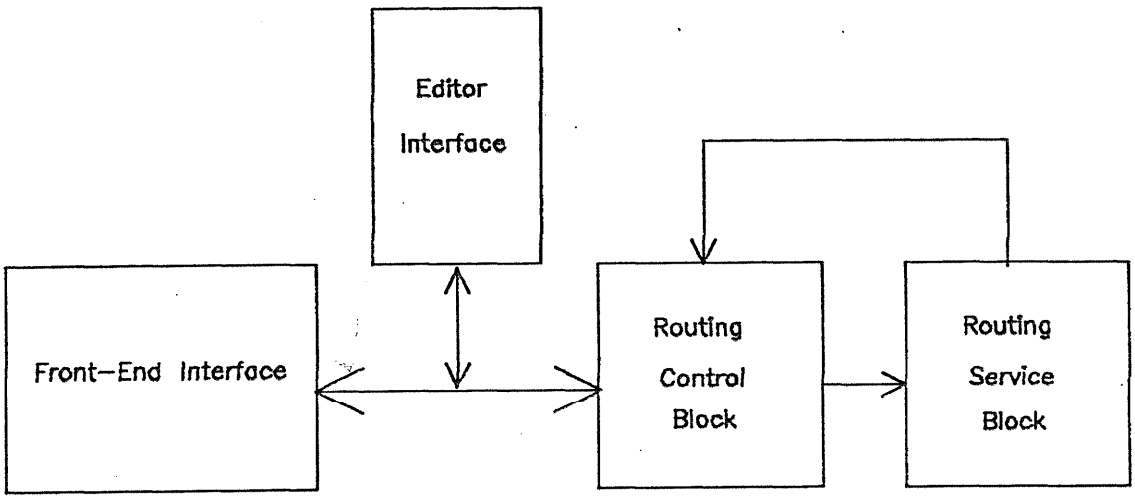

Figure 4.1 Overview of SMART's organization

must be acyclic) and starts routing the lower level contexts. The actual routing steps performed by the routing service block follows closely to those described in the last chapter. Any routing context which fails to achieve completion is reported. A phantom CIF symbol of the same dimensions is created to enable continuation of processing. All CIF symbols created by SMART are saved in temporary files. They may be deleted or saved permanently before exiting SMART depending on the user's command. SMART keeps track of any module routed by it to avoid redundant routing.

The editor interface is another building block put in to provide for user convenience. When a module is read in, SMART keeps track of its location in the source file. When the user decides to modify the source, probably due of syntax errors discovered, it is done by issuing an *edit* command. SMART retrieves the source code of the module and passes it to an editor. Upon exiting the editor, any modifications made are saved immediately and the source automatically read in for syntax and semantics check.

4.1.2. Input Language

SMART defines its own input language that allows the user to specify routing contexts hierarchically in terms of modules. Modules are the basic structural units which SMART recognizes. A module can be as simple as a leaf cell or as complicated as an entire integrated circuit chip. Through hierarchical nesting of modules, SMART allows the user to compose large complicated circuits in terms of simpler ones.

The BNF for module declaration is shown in Table 4.1 in which the following conventions are used: [ ] means 0 or 1; { } means 0 or more; ( ) is used for grouping; | for alternatives; - for range spanning and " " for reserved

words. White spaces (tab, newline, space) and "," are legal separators. Dimensions are specified in lambda units. Coordinates are measured with origin at the lower left corner of the module.

Briefly, a module declaration has four parts: attribute declarations; terminal declarations; sub-module instantiation declarations; and the netlist declarations. Attributes are information pertinent to the routing context as a whole such as the width of the power busses to be used. The remaining parts

```
MODULEDEC        ::=   MODULEATT "BEGINM" MODULEBODY "ENDM"
MODULEATT        ::=   MODULENAME [SIDE]
                       "BBOX" COORD ["SYM" INT] ["POWER" NUM]
MODULENAME       ::=   "MODULE" ID
MODULEBODY       ::=   [TERMINAL] [INSTANTIATION] [NETLIST]
TERMINALS        ::=   "TERM" { SIDELIST }
SIDELIST         ::=   SIDE ":" TERMLIST
SIDE             ::=   "EAST" | "WEST" | "NORTH" | "SOUTH"
TERMLIST         ::=   TERMLIST TERM
TERM             ::=   ID [LAYER] [WIDTH] POSITION
LAYER            ::=   "L" ID
WIDTH            ::=   "W" NUM
POSITION         ::=   "P" NUM
INSTANTIATION    ::=   "SUBMOD" { INSTANCE }
INSTANCE         ::=   ID ":" PLACEMENTLIST
PLACEMENTLIST    ::=   "{" PLACEMENTS "}" | PLACEMENT
PLACEMENTS       ::=   PLACEMENT { PLACEMENT }
PLACEMENT        ::=   ID COORD { TRANSFORMATION } [ EXPANSION ]
TRANSFORMATION   ::=   "MX" | "MY" | "ROT" INT
EXPANSION        ::=   "EXP" NUM
COORD            ::=   NUM NUM
NETLIST          ::=   "NET" { NET }
NET              ::=   ID ":" [NUM] PORTLIST
PORTLIST         ::=   "{" PORTS "}" | PORT
PORTS            ::=   PORT { PORT }
PORT             ::=   (ID|*) | (ID|*) "." (ID|*) | (ID|*) "." (ID|*) "." (ID|*)
ID               ::=   (a-zA-Z) { (a-zA-Z0-9) }
INT              ::=   (0-9) { (0-9) }
NUM              ::=   (0-9) { (0-9) } [.] { (0-9) }
```

Comments can be inserted by preceding the text of the comment with a semi-colon (";"). The remainder of the line which contains ";" will be ignored by the parser.

Table 4.1 The BNF for module declaration

form the main "guts" of a module. Terminals are connectors located on the four sides of a module. They are the principal structures through which connections can be made to the outside. Sub-module instantiation is the mechanism to specify hierarchical nesting of modules. Netlist information defines the desired wiring connectivities to be generated by SMART. Each module declaration specifies the routing to be performed inside that module. Latter modules can be declared to include former ones. All modules must be declared before their instantiations. An in depth description of the input language and user interface to SMART can be found in [Ngai 84b].

4.2. Examples and Results

In this section, we present a few preliminary test examples routed by SMART and examine their results. The first one is a contrived example that demonstrates some of the features and performances of SMART. The second one is the so-called *difficult example* of Deutsch [Deutsch 76]. The third one is a microprocessor chip implemented by the Silicon Structures Project at Caltech.

4.2.1. A Contrived Example

Figure 4.2 shows the layout of our first test example routed by SMART. The routing specification of this example can be found in Appendix A. It consists of three distinct modules placed together inside a single routing context. It has a total of twenty two signal nets together with power and ground. Power and ground nets are routed with 12λ wires. Notice how the VDD signal on the east side of mod1 is relocated to its north side. Local relocations of power and ground signals are done with wires conforming to

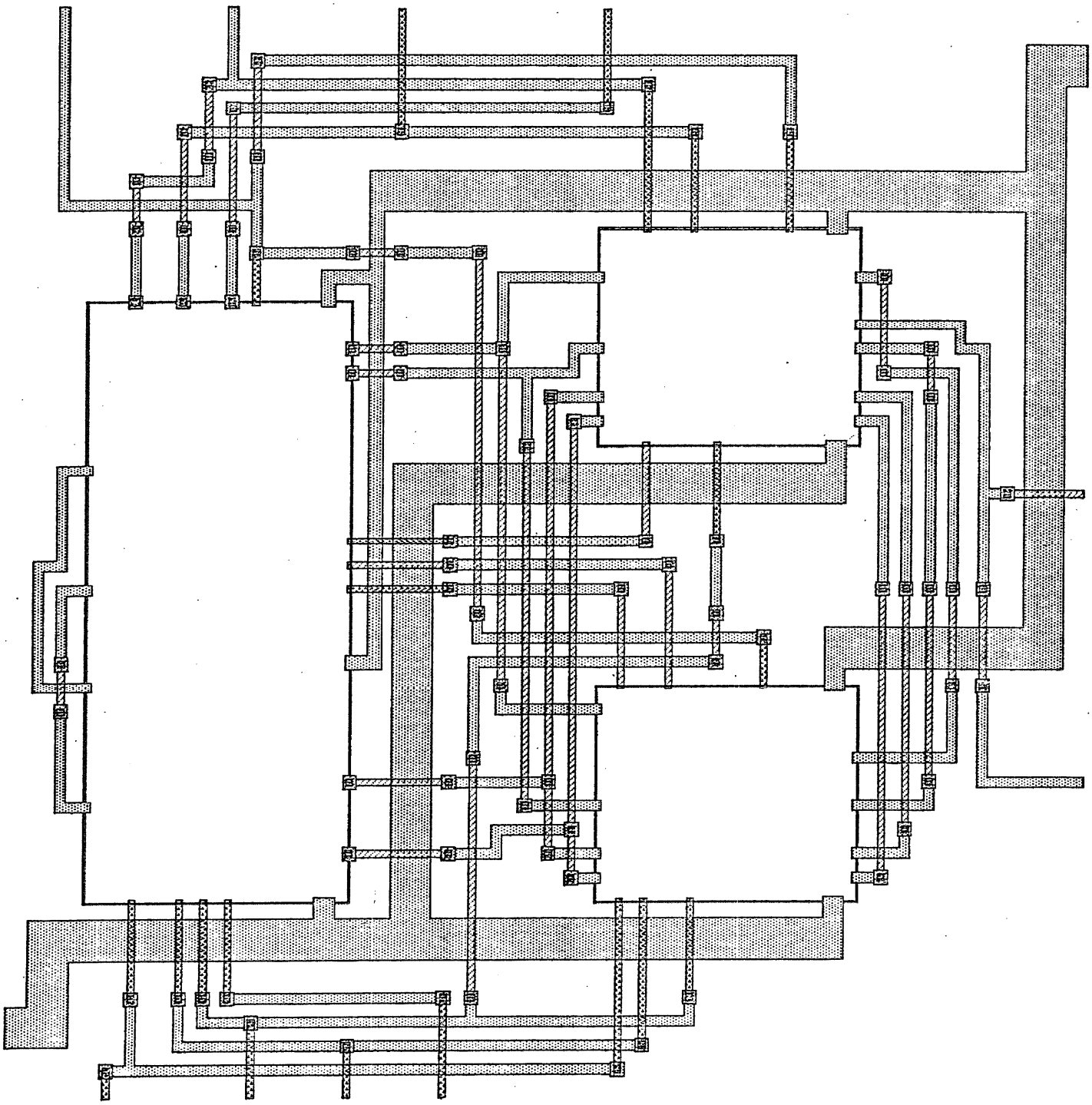

Figure 4.2 SMART's layout of the contrived example

the power wire width requirement of the relocated module, and in this example, it is 4λ. This example also illustrates the use of module bounding box expansion to avoid creating narrow channels and to guide the routing of power and ground. Recall that our power and ground routing algorithm routes the wires along opposite edges of the channels. A narrow channels created due to a certain placement may result in the power routing algorithm failing miserably. In our example, mod1 is instantiated with an expansion that aligns its top edge with that of mod3. The result is a continuous VDD bus running along the south edge of the top most channel.

The layer assignment strategy used by SMART is also demonstrate vividly in this example. SMART uses metal for all horizontal connections except those bridging over a vertical power wire. Metal usage optimization is performed on vertical connections. SMART uses a threshold value to determine if swapping of layer on a vertical wire is desirable. In our example the threshold is set to 21λ. Vertical connections shorter than the threshold value simply stays on polysilicon. Notice that diffusion wires are also used in vertical connections. This is not necessary but helps in saving contact cuts when signals are brought out by pins on diffusion.

Another property of the stepping routing algorithm also illustrated in this example is the tendency to introduce unnecessary corner turns. This effect is the result of the local information driven and greedy approach characters of the stepping algorithm. It also tends to introduce more crossovers than necessary and in this manner it tends to undermine the layer assignment strategy. Routing of this example took 3.6 CPU seconds on VAX-780[3].

[3]VAX is a trademark of Digital Equipment Corporation

4.2.2. Deutsch's Difficult Example

Deutsch's Difficult Example was originally posed by David N. Deutsch in 1976 as a test case for the LTX channel router described in his paper [Deutsch 76]. It was soon recognized as one of the best known benchmark for testing channel routers. It has a total of seventy two signal nets and a channel density of nineteen. Appendix B shows the specification of the example written in SMART's input language. [Burstein 83] is the only published result that succeeded in routing this example in nineteen tracks at the writing of this thesis.

This example is routed by SMART as a test case for its local routing algorithm. Recall that the local routing algorithm described in the last chapter is a switch box router which can handle fixed pins from all four sides of the channel. One desirable property a switch box router is expected to have is symmetry in handling pins regardness of the channel's orientation. The original greedy router [Rivest 82b] is capable of routing the example in 20 tracks. To test SMART's algorithm, the example was test routed in two orientations: one with pins along its horizontal sides and one with pins along its vertical sides.

Figures 4.3 shows the resulting layouts produced by SMART. In both cases, SMART complete the routing by using 21 tracks. These results have one track more than the original greedy router. This performance differential however, is the price paid in shifting from a specific channel routing algorithm to a general switch box algorithm. Further, the symmetry in the performances of the two cases is encouraging. It exhibits the balance in performance so desirable in general cell routing where modules may be rotated arbitrarily. Routing of the difficult example took 8.1 CPU seconds on

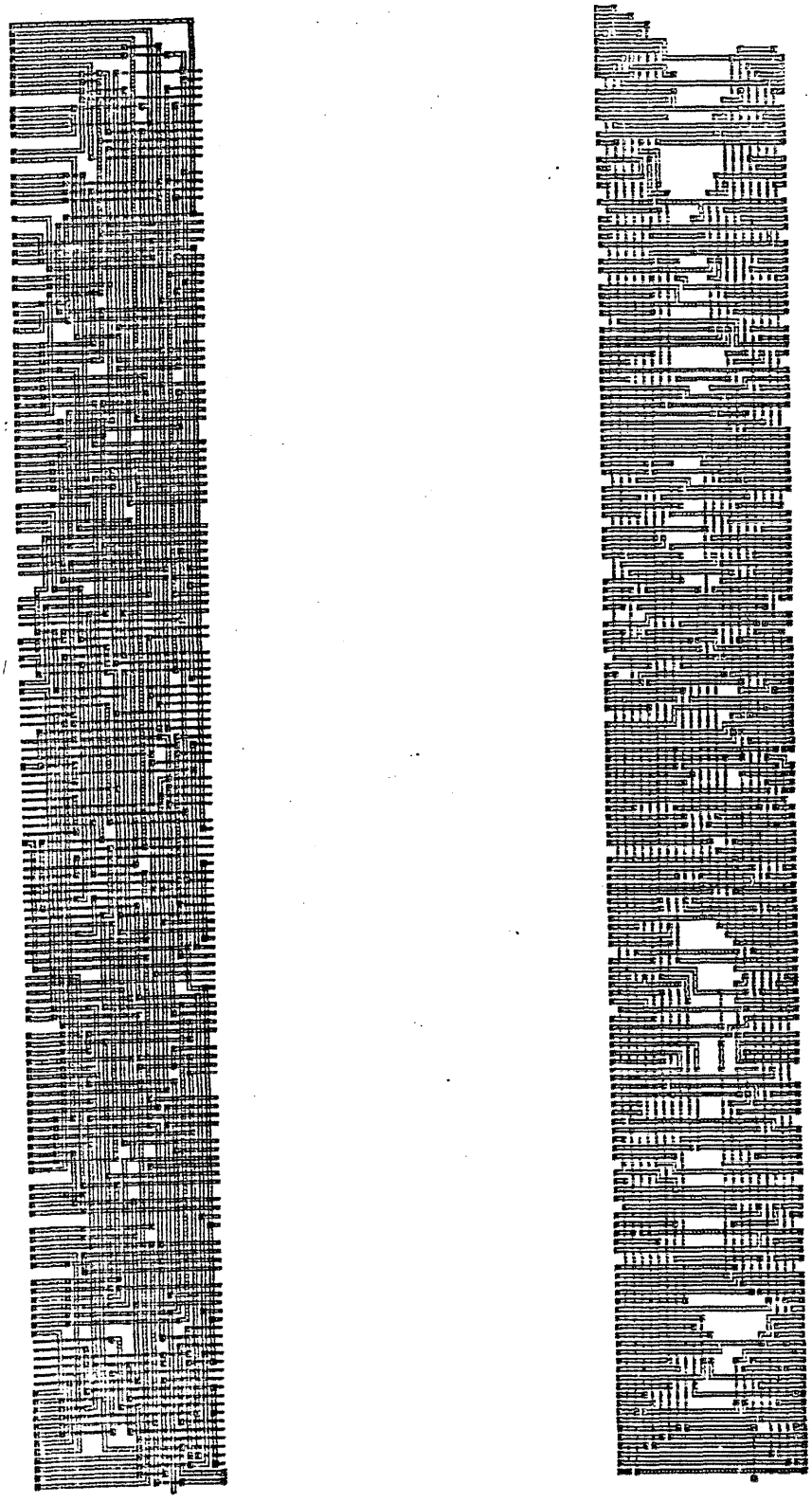

Figure 4.3 SMART's layout of the difficult example

VAX-780.

4.2.3. A Microprocessor Example

The microprocessor chip used for this example called *Smal16* was originally designed by Kenneth Slater from DEC[4] in the Silicon Structures Project at Caltech. The author obtained the routing specifications for the chip from Gary Clow, who was a former SSP representative and now a graduate student in the Caltech Computer Science Department. The example consists of four different modules: a data path; a FSM controller, a register decoder block and an ALU and PSW decoder block. It has a total of 189 signals and a size of 5800 $\lambda \times 4800\ \lambda$. Routing of the entire chip took 1 minute 47 seconds of CPU time on VAX-780. Because of the length of the input specification and the resolution required for plotting the layout, they are not included.

4.3. Conclusions

In this thesis, the general cell interconnection problem for composing integrated circuits is examined. Work has been focused in three related areas.

(1) The development of a generic routing model which is independent of the underlying fabrication technology. A detail look at the correlation between timing behavior and electrical properties of the connection wires leads to a natural extension of the hierarchical model to multiple routing layer environment.

---

[4]Digital Equipment Corporation

(2) Experimenting with an alternative approach to routing by shifting emphasis from optimization to simplicity. The stepping approach is chosen and the various routing algorithms which take advantage of the stepping framework are developed.

(3) Implementation of a prototype router based on the hierarchical routing model and the stepping algorithms developed. An input language that allows concise user specification of the hierarchical routing requirements and simple user interactions is developed.

From the preliminary test results obtained, the stepping approach seems quite promising in producing fast and average quality routing. However, much more extensive testing are needed before the final conclusions can be drawn. The major bottleneck in using the router remains in the specification of routing requirements, in particular, the pin coordinates. This bottleneck will be removed in the future when the router is integrated with the other existing layout tools at Caltech.

4.4. Suggestions for Future Work

Many problems remain unsolved in the area of integrated circuit composition. Some of the more prominent ones identified during the work on this thesis are described below. The ultimate goal in future research should be the development of design tools that enhence both the productivities of the designers and the qualities of the work done.

4.4.1. Routing Area Estimation

In the hierarchical approach to the VLSI circuit design problem (see section 2.3), top down floorplanning involves making provisions for routing

area for each routing context. Hence, the design system must provide accurate estimation of routing areas given the topological placements. Accurate estimates help to reduce the number of design iterations and hence an increase in productivity. While the most accurate method is to invoke actual routing, computationally, it is generally infeasible. Instead, abstract models should be employed to give effective estimations. However, the models employed are either too restrictive for general cell compositions [Ngai 84a], or too general to guide repetitive estimation refinements [El Gamal 81] [Heller et al. 78]. An effective estimation scheme for general cell composition is obviously needed.

4.4.2. Routing Model Extension

All the algorithms developed in this thesis use only two conducting layers and routing is restricted in area not occupied by the circuit modules. The three dimensional hierarchical routing model described in section 2.3 provides a natural extension to the multiple layer environment. The routing algorithms should be extended to take advantage of the extra degree of freedom. Single layer routing algorithms should be developed to handle over-the-cell routing where a free routing layer resides over areas occupied by the low level modules. Algorithms must also be developed to support deferred routing of global signals brought out through internal pins until more favorable layers are available.

Another restriction on the routing model is the limitation to generating Manhattan wires only. While this limitation provides for simplicity in most issues, it has its own complications such as the possibility of introducing constraint loops in channel routing. The general Boston Geometry wiring

model on the other hand, is very flexible but also very difficult to work with. A good compromise is the Forty-Five Degree Geometry which is Manhattan plus diagonal wires. The author does not know of any publication of work on this model. While diagonal wires have been routinely used in manual printed circuit board layouts, there is an obvious lack of systematic routing approach in Forty-Five Degree Geometry.

4.4.3. Direct User Interaction

The algorithms presented in this thesis and most of other conventional routers operate in *batch* mode. Once routing has been initiated, little or no interference is expected from the user. Because of the complexity of the routing problem, extensive fine tuning is needed before satisfactory layouts are obtained from such routers. Another problem inherent in such routers occurs when the router fails to complete its routing. While causes of failure in general are hard to localize, finding of efficient remedies are even more difficult. The result is usually a bigger routing area than necessary.

These problems are due to the lack of direct control of routing behavior by the user. Fine tuning of heuristic parameters used in the routing algorithms and the relaxation of routing area are only indirect measures available to the user. More direct control by the user is obviously needed. For example, the most critical signals may be hand-routed by the user in order to meet timing restrictions while the reemaining signals are routed by an automatic router. Another example would be the removal of the less critical signals from a congested channel by the user when a failure developed due to congestion.

These direct measures depend heavily on the ability of the router to interact with the user at any point when it is needed. During its processing, the user must be able to interrupt the routing and make modifications. The router should then be able to resume processing, and in the event of a failure, come back for further user interactions. This kind of close man-machine interactions can be very effective in raising designer's productivity and improving the quality of work done.

APPENDIX A

SMART Input of The Contrived Example

```
module mod1
      power  4
      bbox   77 175
beginm
term
north: f1 L ND P 14 n2 P 28 n3 P 42 n1 P 49 vdd L NM W 4 P 70
south: s1 L NP P 14 f2 P 28 s2 P 35 s3 P 42 gnd L NM W 6 P 70
east:  e1 L NM P 14 f3 P 35 vdd W 4 P 70 r1 L ND P 91
       r2 L NP P 98 r3 L ND P 105 e2 L NM P 154 e3 P 161
west:  w1 L NM P 28 f4 P 63 w2 P 91 w3 P 126
endm;

module mod2
      power  4
      bbox   77 63
beginm
term
north: r1 L NP P 7 r2 P 21 n1 L ND P 49 vdd L NM W 6 P 70
south: s1 L ND P 7 f2 P 14 s2 P 28 gnd L NM W 6 P 70
east:  e1 L NM P 7 f3 P 14 e2 P 28 e3 P 42
west:  w1 L NM P 7 f4 P 14 w2 P 28 w3 P 56
endm

module mod3
      power  4
      bbox   77 63
beginm
term
north: f1 L ND P 14 n2 P 28 n1 P 56 vdd L NM W 6 P 70
south: r3 L NP P 14 s2 L ND P 35 gnd L NM W 6 P 70
east:  e1 L NM P 7 f3 P 14 e2 P 28 t4 P 35 e3 P 49
west:  w1 L NM P 7 f4 P 14 w2 P 28 w3 P 49
endm

module complex
      north
      power  12
      bbox   313 315
beginm
term
north: n1 L NM P 14 f1 P 63 n2 L ND P 112 n3 P 172
south: s1 L ND P 28 s2 P 70 f2 P 98 s3 P 126
west:  gnd L NM W 12 P 20
east:  t2 L NM P 91 t3 L NP P 175 vdd L NM W 12 P 240;
```

```
submod
mod1:  I1 21 56 EXP 21              ; to avoid narrow channel
mod3:  I1 170 189
mod2:  I1 170 56
net
vdd:   *.*.vdd
gnd:   *.*.gnd
; all the rest signals have tolerance ratio 1.0!
sig1:  *.*.s1
sig2:  *.*.s2
sig3:  *.*.f2
sig4:  *.*.s3
sig5:  *.*.n1
sig6:  *.*.n2
sig7:  *.*.f1
sig8:  *.*.n3
sig9:  { mod1.e1, mod2.w1, mod3.w1 }
sig10: { mod1.e2, mod2.w2, mod3.w2 }
sig11: { mod1.e3, mod2.w3, mod3.w3 }
sig12: { mod1.f3, mod2.f4, mod3.f4 }
sig13: { mod2.e1, mod3.e1 }
sig14: { mod2.e2, mod3.e2 }
sig15: { mod2.e3, mod3.e3 }
sig16: { mod2.f3, mod3.f3 }
sig17: { mod1.w1, mod1.w2 }
sig18: { mod1.f4, mod1.w3 }
sig19: { *.*.t2, *.*.t3, *.*.t4 }
sig20: *.*.r1
sig21: *.*.r2
sig22: *.*.r3
endm
```

APPENDIX B

SMART Input of The Difficult Example

```
;
; SMART Input Specification for The Deutsch's Difficult Example
;
module DeutschDiff
        bbox  1225 154
beginm
term
south:
        s3 L NP W 2 P 7,
        s5 P 14,      s7 P 21,      s9 P 28,      s5 P 35,      s12 P 42,
        s14 P 49,     s15 P 56,     s7 P 63,      s12 P 70,     s14 P 77,
        s7 P 84,      s4 P 91,      s13 P 98,     s8 P 105,     s6 P 112,
        s15 P 119,    s18 P 126,    s14 P 133,    s8 P 140,     s6 P 147,
        s11 P 154,    s22 P 161,    s21 P 168,    s18 P 182,    s16 P 189,
        s18 P 196,    s16 P 203,    s8 P 217,     s6 P 224,     s26 P 231,
        s11 P 238,    s24 P 252,    s23 P 259,    s25 P 266,    s20 P 273,
        s1 P 280,     s29 P 287,    s22 P 301,    s3 P 308,     s22 P 315,
        s3 P 322,     s9 P 343,     s2 P 350,     s9 P 357,     s2 P 364,
        s32 P 378,    s23 P 385,    s33 P 392,    s19 P 399,    s6 P 406,
        s8 P 413,     s30 P 420,    s27 P 427,    s34 P 434,    s35 P 441,
        s36 P 448,    s37 P 455,    s39 P 462,    s31 P 469,    s39 P 476,
        s35 P 483,    s38 P 490,    s31 P 497,    s8 P 504,     s30 P 511,
        s37 P 518,    s41 P 525,    s19 P 532,    s6 P 539,     s44 P 546,
        s45 P 553,    s33 P 567,    s31 P 574,    s33 P 581,    s31 P 588,
        s27 P 602,    s35 P 609,    s36 P 616,    s48 P 623,    s49 P 630,
        s31 P 637,    s39 P 644,    s46 P 651,    s47 P 658,    s50 P 665,
        s52 P 672,    s20 P 679,    s53 P 686,    s24 P 693,    s47 P 707,
        s39 P 714,    s24 P 728,    s51 P 735,    s20 P 742,    s52 P 749,
        s20 P 756,    s52 P 763,    s23 P 770,    s8 P 777,     s30 P 784,
        s50 P 791,    s56 P 798,    s57 P 819,    s49 P 826,    s19 P 833,
        s6 P 840,     s6 P 847,     s19 P 854,    s49 P 861,    s59 P 868,
        s61 P 889,    s50 P 896,    s30 P 903,    s8 P 910,     s55 P 917,
        s24 P 931,    s64 P 938,    s20 P 945,    s52 P 952,    s67 P 966,
        s68 P 973,    s63 P 980,    s55 P 987,    s24 P 994,    s52 P 1001,
        s20 P 1008,   s69 P 1015,   s24 P 1022,   s46 P 1036,   s62 P 1043,
        s63 P 1050,   s68 P 1057,   s24 P 1071,   s65 P 1078,   s20 P 1085,
        s52 P 1092,   s70 P 1106,   s60 P 1113,   s62 P 1120,   s54 P 1127,
        s63 P 1134,   s24 P 1148,   s71 P 1155,   s20 P 1162,   s52 P 1169,
        s67 P 1176;

north:
        s2 L NP W 2 P 7,
        s4 P 14,      s6 P 21,      s8 P 28,      s10 P 35,     s11 P 42,
        s13 P 49,     s3 P 56,      s9 P 63,      s16 P 70,     s5 P 77,
```

```
        s17 P 84,    s11 P 91,    s5 P 98,     s14 P 105,   s14 P 112,
        s7 P 119,    s12 P 126,   s17 P 133,   s19 P 140,   s1 P 147,
        s20 P 154,   s21 P 161,   s23 P 168,   s24 P 175,   s16 P 189,
        s10 P 196,   s3 P 203,    s11 P 210,   s25 P 217,   s26 P 231,
        s11 P 238,   s26 P 245,   s11 P 252,   s27 P 266,   s28 P 273,
        s11 P 280,   s3 P 287,    s9 P 294,    s16 P 301,   s30 P 308,
        s27 P 315,   s5 P 322,    s31 P 329,   s1 P 336,    s5 P 343,
        s1 P 350,    s20 P 357,   s32 P 364,   s23 P 371,   s24 P 378,
        s9 P 392,    s1 P 399,    s20 P 406,   s29 P 413,   s23 P 420,
        s24 P 427,   s3 P 441,    s8 P 448,    s30 P 455,   s38 P 462,
        s28 P 469,   s19 P 476,   s6 P 483,    s40 P 490,   s27 P 497,
        s35 P 504,   s41 P 511,   s42 P 518,   s6 P 525,    s19 P 532,
        s34 P 539,   s43 P 546,   s30 P 553,   s8 P 560,    s31 P 567,
        s43 P 574,   s39 P 581,   s46 P 588,   s36 P 595,   s46 P 602,
        s47 P 609,   s48 P 616,   s31 P 623,   s24 P 637,   s23 P 644,
        s45 P 651,   s20 P 658,   s1 P 665,    s51 P 672,   s40 P 686,
        s39 P 693,   s40 P 700,   s39 P 707,   s8 P 721,    s30 P 728,
        s50 P 735,   s54 P 742,   s55 P 763,   s49 P 770,   s19 P 777,
        s6 P 784,    s47 P 798,   s42 P 805,   s47 P 812,   s42 P 819,
        s53 P 833,   s58 P 840,   s6 P 847,    s19 P 854,   s49 P 861,
        s50 P 868,   s30 P 875,   s8 P 882,    s60 P 889,   s62 P 896,
        s59 P 903,   s54 P 910,   s55 P 917,   s54 P 924,   s56 P 931,
        s63 P 938,   s55 P 945,   s65 P 952,   s66 P 966,   s68 P 973,
        s66 P 980,   s68 P 987,   s60 P 1001,  s68 P 1008,  s46 P 1022,
        s44 P 1029,  s46 P 1036,  s44 P 1043,  s69 P 1057,  s55 P 1071,
        s58 P 1078,  s55 P 1085,  s58 P 1092,  s64 P 1106,  s71 P 1113,
        s72 P 1127,  s63 P 1134,  s72 P 1141,  s63 P 1148,  s57 P 1162,
        s62 P 1169,  s54 P 1176,  s70 P 1183,  s67 P 1190,  s55 P 1197,
        s61 P 1204,  s63 P 1211,  s68 P 1218;

west:
        s1 L NM W 3 P 42;

net
        s1: *.*.s1   s2: *.*.s2   s3: *.*.s3   s4: *.*.s4   s5: *.*.s5
        s6: *.*.s6   s7: *.*.s7   s8: *.*.s8   s9: *.*.s9   s10: *.*.s10
        s11: *.*.s11 s12: *.*.s12 s13: *.*.s13 s14: *.*.s14 s15: *.*.s15
        s16: *.*.s16 s17: *.*.s17 s18: *.*.s18 s19: *.*.s19 s20: *.*.s20
        s21: *.*.s21 s22: *.*.s22 s23: *.*.s23 s24: *.*.s24 s25: *.*.s25
        s26: *.*.s26 s27: *.*.s27 s28: *.*.s28 s29: *.*.s29 s30: *.*.s30
        s31: *.*.s31 s32: *.*.s32 s33: *.*.s33 s34: *.*.s34 s35: *.*.s35
        s36: *.*.s36 s37: *.*.s37 s38: *.*.s38 s39: *.*.s39 s40: *.*.s40
        s41: *.*.s41 s42: *.*.s42 s43: *.*.s43 s44: *.*.s44 s45: *.*.s45
        s46: *.*.s46 s47: *.*.s47 s48: *.*.s48 s49: *.*.s49 s50: *.*.s50
        s51: *.*.s51 s52: *.*.s52 s53: *.*.s53 s54: *.*.s54 s55: *.*.s55
        s56: *.*.s56 s57: *.*.s57 s58: *.*.s58 s59: *.*.s59 s60: *.*.s60
        s61: *.*.s61 s62: *.*.s62 s63: *.*.s63 s64: *.*.s64 s65: *.*.s65
        s66: *.*.s66 s67: *.*.s67 s68: *.*.s68 s69: *.*.s69 s70: *.*.s70
        s71: *.*.s71 s72: *.*.s72
endm;
```

REFERENCE

[Burstein 83]
M. Burstein. and R. Pelavin, "Hierarchical Channel Router." *Proceedings of the 20th Design Automation Conference (1983).*

[Busacker 65]
Robert G. Busacker and Thomas L. Saaty, *Finite Graphs and Networks: An Introduction with Applications.* MacGraw-Hill, 1965.

[Chan 83]
W.S. Chan, "A New Channel Routing Algorithm." *Third Caltech VLSI conference (1983).*

[Deutsch 76]
D.N. Deutsch, "A *Dogleg* Channel Router." *Proceedings of the 13th Design Automation Conference (1976).*

[El Gamal 81]
Abbas A. El Gamal, "Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits." *IEEE Trans. on Circuits and Systems. Feb. 1981, pp. 127-138.*

[Hassett 82]
J.E. Hassett, "Automatic Layout in ASHLAR: An Approach to the Problems of *General Cell* Layout for VLSI." *Proceedings of the 19th Design Automation Conference (1982).*

[Heller et al. 78]
W.R. Heller et al., "Prediction of Wiring Space Requirements for LSI." *Journal of Design Automation and Fault-Tolerant Computing. (1978) (pp. 117-144)*

[Hightower 69]
D.W. Hightower, "A Solution to the Routing Problems on the Continuous Plane." *Proceedings of the 6th Design Automation Workshop (1969).*

[Hightower 74]
D.W. Hightower, "The Interconnection Problem: A Tutorial." *COMPUTER 7,4 (April 1974).*

[Hightower 80]
D.W. Hightower. and R.L. Boyd, "A Generalized Channel Router." *Proceedings of the 17th Design Automation Conference (1980).*

[Hsu 82]
C.P. Hsu, "A New Two-Dimensional Routing Algorithm." *Proceedings of the 19th Design Automation Conference (1982).*

[Hu 82]
T.C. Hu, *Combinatorial Algorithms.* Addison-Wesley 1982.

[Johannsen 81]
D. Johannsen, *Silicon Compilation.* Ph.D. Thesis, California Institute of Technology, 1981.

[Kingsley 82]
C. Kingsley, *Earl: An Integrated Circuit Design Language.* Caltech Technical Report 5021, 1982.

[Kuh 82]
T. Yoshimura and E.S. Kuh, "Efficient Algorithms for Channel Routing." *IEEE Trans. CAD of ICs and Systems CAD-1 (1) (1982).*

[Lee 61]
C.Y. Lee, "An algorithm for Path Connections and Its Applications." *IRE Trans. on Electronics Computers, Vol EC-10 No. 3 (Sept 1961).*

[Liu & Atkins 82]
W. Liu and D.E. Atkins, "On the Routability and Channel Routing Order of A General Cell Approach." *Proceedings of the IEEE International Conference on Circuits and Computers (1982).*

[Mead & Conway 80]
C.A. Mead and L.A. Conway, *Introduction to VLSI Systems,* Addison-Wesley, 1980

[Mead & Rem 82]
C.A. Mead and M. Rem, "Minimum Propagation Delays in VLSI." *IEEE Journal of Solid-State Circuits, Vol. SC-17, No.4, (August 1982).*

[Mlynski 81]
H.J. Rothermel and D.A. Mlynski, "Computation of Power Supply Nets in VLSI Layout." *Proceedings of the 18th Design Automation Conference (1981).*

[Moore 59]
E.F. Moore, "Shortest Path Through a Maze." *Annals of the Harvard Computation Laboratory, Vol 30, pt. II, Cambridge, Mass., Harvard University Press (1959).*

[Moulton 83]
Andrew S. Moulton, "Laying The Power and Ground Wires on A VLSI Chip." *Proceedings of the 20th Design Automation Conference (1983).*

[Ngai 84a]
John Y. Ngai, *Computational Models for Channel Routing Track Demand.* Caltech Technical Report 5107, 1984.

[Ngai 84b]
John Y. Ngai, *SMART User's Guide.* Caltech Technical Report 5118, 1984.

[Oestreicher 72]
D.R. Oestreicher, *Automatic Printed Circuit Board Design.* Ph.D. Thesis, University of Utah, 1972.

[Preas 79]
B.T. Preas, *Placement and Routing Algorithms for Hierarchical Integrated Circuit Layout.* Ph.D. Thesis, Stanford University, 1979.

[Rivest 82a]
R.L. Rivest, "The *PI* (Placement and Interconnect) System." *Proceedings of the 19th Design Automation Conference (1982).*

[Rivest 82b]
R.L. Rivest. and C.M. Fiduccia, "A *Greedy* Channel Router."

Proceedings of the 19th Design Automation Conference (1982).

[Rowson 80]
J.A. Rowson, *Understanding Hierarchical Design*, Ph.D. Thesis, California Institute of Technology, 1980.

[Smith et al. 82]
L.R. Smith, et al., "A New Area Router, The LRS Algorithm." *Proceedings of the IEEE International Conference on Circuits and Computers (1982).*

[Syed 82]
Z.A. Syed and A. El Gamal, "Single Layer Routing of Power and Ground Networks in Integrated Circuits." *Journal of Digital Systems, Vol 6 No. 1 (1982).*