Integrated Silicon Assembly

Thomas George Waring

Doctor of Philosophy

University of Edinburgh

1990

Abstract

The problem of silicon assembly contains several well-separated steps that can be identified. To solve the assembly problem, use can be made of the stepwise approach to assembly, which involves finding ways of performing the individual steps and then linking the steps together to realise a silicon assembler. However, experience has shown that this approach produces poor-quality results, not because of the failure of the individual steps, but rather because of the breakdown in communication between the steps. Silicon assembly is a programming problem whose results are significantly affected by how the problem is decomposed into sub-problems. Building on the experience gained from implementing a stepwise assembler a novel integrated approach to solving the assembly problem is presented, which overcomes the communication problems inherent in the stepwise approach. Experimental results obtained using an integrated assembler are comparable to or better than those produced by existing assemblers. The integrated approach is simple in concept and easy to implement, yet produces good results, and is a suitable platform for taking silicon assembly forward in the 1990's.

Acknowledgements

For the opportunity to study - an opportunity not available to everyone in my parent's generation - long may this opportunity continue.

The work described in this dissertation was undertaken as part of the Underlying Research Programme of the UKAEA.

To Alex Deas for his encouragement, inspiration and support.

To David Rees for his constructive advice and help.

Most importantly, to Sidney Michaelson, whom I had the privilege of having as my supervisor.

Declaration

I declare that this dissertation was composed by myself and that the work it describes is my own.

Table of Contents

List of Tables

List of Abbreviations

CVM	Constrained Via Minimization
DR cycle	Detailed Routing cycle
F&P	Floorplanning and Placement
GP cycle	Global and power routing, Placement adjustment cycle
UVM	Unconstrained Via Minimization
VLSI design	Very Large Scale Integrated circuit design

Chapter 1

Introduction

Do but take care to express yourself in a plain, easy Manner, in well-chosen, significant and decent Terms, and to give a harmonious and pleasing Turn to your Periods; study to explain your Thoughts, and set them in the truest Light, labouring as much as possible, not to leave them dark nor intricate, but clear and intelligible.

Cervantes, *Preface to Don Quixote*

VLSI (Very Large Scale Integrated circuit) design presents the opportunity to put enormously complex electronic systems on a chip. Mead states that he believes VLSI to be the most important opportunity since the Industrial Revolution [Mead81]. In the short run VLSI is simply a more cost effective way to implement hardware. In the long run there is the possibility of millions of autonomous communicating processing elements operating concurrently. The ultraconcurrent world of VLSI is effecting a revolution in the field of computer science. VLSI has the potential to change the modes of communication, commerce, education, entertainment, science and the underlying rate of cultural evolution.

The current position is that we want VLSI. The potential of VLSI gives rise to the motivation and impetus to research VLSI. The major bottleneck is that sometimes the 'what' of what we want to realise with VLSI is unobtainable because we do not yet have the 'how' of how to realise the VLSI.

1.1 Silicon Compilation

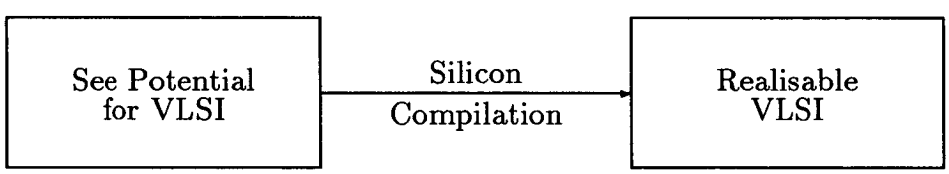


Figure 1–1: *The process of silicon compilation*

Figure 1–1 shows silicon compilation as being the 'how' of turning the 'what' of what we want to realise with VLSI into a mask level description suitable for fabrication. There are many approaches to realising the 'what' by the 'how' using silicon compilers. Exactly what is meant by the term silicon compilation is muddled. Johannsen in his description of the Bristle Blocks silicon compiler [Joha79] used the term to describe the concept of assembling parameterized pieces of lay-

out. Subsequently the term has gained in popularity and can be defined in a much broader sense as a translation process from a higher-level abstraction into layout. The ideal silicon compiler would be able to take a high-level design specification (for example, a behaviour defined by a program) and produce a system design that implements that program in hardware.

The reason for using a silicon compiler is that even a small VLSI design generates a large amount of mask level data. Consequently, to use the improving VLSI technology effectively, a means must be provided to allow the designer to work at a higher level of abstraction than the mask level, leaving the task of managing the huge volume of mask data associated with a VLSI design to an automatic system. The abstraction from mask level design is necessary both to reduce the design time and to prevent errors that can easily occur if manual mask level design is undertaken. The sheer volume of mask level data requires the automation of mask data generation from a more succinct representation at a higher level.

There are several other advantages in performing VLSI design at a higher level of abstraction than the mask level. Designing at the mask level makes it very difficult to alter a design to effect a small change in the specification or to correct a design error. Designing at a higher level permits more flexibility in altering designs. The designer is able to experiment and gain an intuitive feel about what is good and what is bad in VLSI design. Mask level design results in the VLSI geometry being specified as absolute positions. The sequence of actions through which the designer had to go to realise the geometry is lost. Such a sequence contains the precise information required to recreate the geometry - information which can be captured if VLSI design is performed at a higher level of abstraction than the mask level.

1.2 Silicon Assembly

Silicon assembly is the last stage of the silicon compilation process and involves the placement of cells that compose the VLSI design in a two dimensional finite space and the routing of the pins of the cells according to the specification. A solution to the assembly problem involves finding a placement and routing, which satisfies the set of *design constraints* (such as those based on the positions and sizes of the cells to be placed and routed), the set of *topological constraints* (such as those based on design rules and the number of layers that can be used in routing) and the set of *performance constraints* (such as those based on the timing to be implemented) [Sang87].

There are many approaches to VLSI design, and there are many types of silicon compiler. Often, the 'what' of what we want to realise with VLSI has properties that make one approach to VLSI design more suitable than another. The FIRST (Fast Implementation of Real-time Signal Transforms) system is an example of a silicon compiler specifically tailored to allow the rapid implementation of VLSI signal processors from a high level system description [Deny82].

A very general approach to VLSI design is to firstly design the cells and to then assemble them. This two-step approach to VLSI design allows the creation of custom cells which can take advantage of, and cope with, any special properties or requirements of the chip to be implemented. Consequently, this approach can handle design problems that the other approaches cannot. However, it also has the most complex assembly problem of all the approaches. The gate array and standard cell approaches allow the designer to assemble complete circuits from a library of cells, using a very constrained layout methodology [Moni85]. These approaches minimize design time, but at the expense of chip area and performance, over which the designer has very little control. The advent of the sea-of-gates

[Igus89] and the sea-of-cells [Kort89] approaches to design, which are variations on the gate array and standard cell approaches, reduce the area penalty involved in using such approaches. However, if the standard cell or gate array approach is used exclusively, then the whole chip has to be realised as one large cell. The two-step more general approach, enables individual cells to be realised using standard cell or gate array or other design techniques which can then be assembled to realise the chip. This dissertation is concerned with the assembly of such custom cells.

1.3 Goals of Silicon Assembly

In assembling a chip design, there are additional goals that the assembler must try to achieve:

- **Minimizing the design area** - the cost of fabricating a chip design is related to the design area. The smaller the design area, the lower the cost of each chip.

- **Minimizing the wire length and the number of wire-crossings** - generally, the shorter the routing wire length, the smaller the design area, and minimizing the number of wire crossings simplifies the wiring layout.

- **Assembling in a reasonable amount of time** - as will be seen in Chapter 2 it is not feasible to search exhaustively for an optimal solution. Instead, heuristics must be used to reduce the search space, to enable a solution to be found.

The problem is that the assembler goals conflict with one another and so a trade-off between them is necessary. It is generally true that the shorter the wire length and the fewer the wire crossings, the smaller the design. However, there are

cases where a longer wire length results in a smaller design area. An example of this is shown in Chapter 2. The other trade-off involves the time spent assembling the design. The longer the time, the more possibilities that can be considered, and hence, the greater the chance of obtaining a good solution.

1.4 Overview of the Dissertation

This dissertation is concerned with the problem of silicon assembly. The thesis is that an integrated approach to silicon assembly is a better way of performing assembly because it is based on a superior framework. The integrated approach represents an advance in the knowledge of 'how' to realise the 'what' of VLSI design.

1.4.1 Contents of the Chapters

In silicon assembly there are several well-separated steps that can be identified and these are described in Chapter 2. Some of these steps are themselves *NP*-Complete problems and not only is there the problem of finding ways of performing the individual steps, but also there is the problem of linking the steps together to realise a silicon assembler. It is made clear in Chapter 2, that to implement a silicon assembler is a large programming task and that silicon assembly is a hard problem to solve.

Chapter 3 describes the stepwise approach to assembly, which involves dividing the assembly problem into several sub-problems. The top-down approach, of breaking a problem into a number of sub-problems as a means of complexity control, is now standard programming practice in both University and Industry. The stepwise approach is all the more attractive because these sub-problems, due to their comparative simplicity, can be studied, and possibly solved, with far better

results than the assembly problem as a whole. Different groups can implement the sub-problems and if, in the future, a better solution for a sub-problem is found, then this can easily be slotted into the assembler. Unfortunately, experience has shown that realising a stepwise assembler, by implementing the constituent parts, results in poor quality layouts. An analysis of the bad performance of the assembler reveals that this is not due to failings in the constituent parts, but rather, the failure lies in poor communication between the parts. In breaking the problem down into a series of sub-problems the overall coherence of the assembler is lost and this is reflected in its poor performance. The implementation of a stepwise assembler reveals that the assembly results are significantly affected by how the problem is decomposed into sub-problems.

Until the analysis of stepwise assembly had been performed, it was surprising that poor results had been obtained. The analysis revealed a tension between the need to manage the complexity of the large assembly problem, by breaking it down into more manageable sub-problems, and the fact that the original stepwise implementation did not work. The key to improving the assembler implementation lies in tackling the problem, inherent in the stepwise approach, of poor communication between parts. Chapter 4 gives details of schemes for improving communication. However, despite these improvements, a fundamental communication problem remains between the global and detailed routing parts of the assembler. The problem is that the global routing part has to *estimate* the space requirements of the detailed routing part. It can only be an estimate, because the actual space requirement is not known until the detailed routing part is performed. It was this observation that led to the proposal and implementation of the Scotia integrated framework for assembly.

The Scotia implementation, which is described in Chapter 5, integrates the global and detailed routing parts. This advance overcomes the communication problem that previously existed and, as the benchmark results show, implement-

ing the Scotia integrated approach realises an assembler that produces compact layouts. Despite the advance in assembly brought about through Scotia, there are still improvements and enhancements that could be made, and these are discussed in Chapter 6.

Chapter 2

Components of Silicon Assembly

In silicon assembly there are several well-separated steps that can be identified:

- ▷ Floorplanning and placement.
- ▷ Channel definition.
- ▷ Global routing.
- ▷ Placement adjustment.
- ▷ Terminal assignment.
- ▷ Detailed routing.
- ▷ Compaction.

This chapter gives details of these steps some of which are known to be NP-Complete. Not only is there the problem of finding solutions to the individual steps, but also there is the problem of linking the steps together to form a silicon assembler. It becomes clear that silicon assembly is a hard problem.

2.1 Floorplanning and Placement

Floorplanning is the first design step which involves physical layout. The floorplanning problem takes as input:

- ▷ A set of cells with constraints on their areas, shapes and pin positions.
- ▷ Constraints on the area and aspect ratio of the chip.
- ▷ A net-list specifying the pins to be interconnected.
- ▷ Constraints on total power dissipation and timing-delays at the chip level.

The output of a floorplanner is an initial placement of the cells, which have had their areas, shapes and relative positions determined. There are many decisions to be taken in creating a floorplan, particularly if the floorplanner is able to determine the shape of a cell. If a floorplanner is able to decide cell shape, then further interaction with the cell design system occurs. Firstly, the cell design system passes the set of cells with their associated constraints to the floorplanner. The floorplanner decides the exact cell specification and passes this information back to the cell design system which then realises the cells according to the specification. Not all floorplanners have the flexibility of interacting with the cell design system. Consequently, it is required that the cell shapes, sizes and pin positions be known at the start of the assembly.

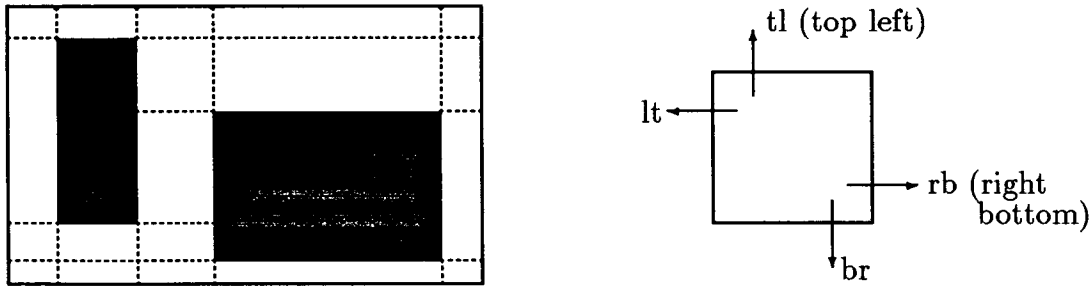
The placement problem has the goal of finding a layout of cells that, after having been routed, fits into an enclosing rectangle of minimum area with given height, width or aspect ratio [Dai89]. Placement is an *NP*-Complete problem [Sang87]. The goal of placement cannot be realised by considering the cells without the rest of the assembly process, because the global and detailed routing interact

with the cell placement. Floorplanning is an even harder problem than placement because it has more degrees of freedom due to cell shapes and areas not being fixed and the exact pin positions having still to be determined. Floorplanning and its interaction with cell design systems is currently an area of intense research.

2.2 Channel Definition

After floorplanning and placement the design consists of cells laid out in space. For assembly to be completed the design needs to be routed, both globally and in detail - steps which almost certainly will cause the placement to be adjusted. For the routings to be performed efficiently a representation of the cells and the spaces between the cells, together with a notion of adjacency of the spaces is required. Channel definition creates data structures that enable the subsequent steps in the assembler to be performed more efficiently.

2.2.1 HV-Slicing.



(a) hv-sliced design (b) tile with corner stitches

Figure 2–1: *HV-Slicing*

Figure 2–1(a) shows an hv-sliced design. The data structure is created by extending the edges of the cells until they hit either the boundary of the design

or another cell edge. This creates a design consisting of two types of tile: cell tiles and space tiles. For each tile, pointers are assigned to one of the pairs of diagonally opposite corners as shown in Figure 2–1(b). (Whichever pair of corners is chosen, the choice must be the same for all the tiles). These pointers point to their neighbouring tiles and are known as corner stitches [Oust84].

The advantage of using hv-slicing is that all space, both empty and occupied, is represented explicitly. The advantage of using corner stitches is that easy modification of the hv-slicing structure can be carried out, and efficient implementations of a variety of tile operations, such as searching, inserting and deleting, can be carried out.

2.2.2 Channel-Graph

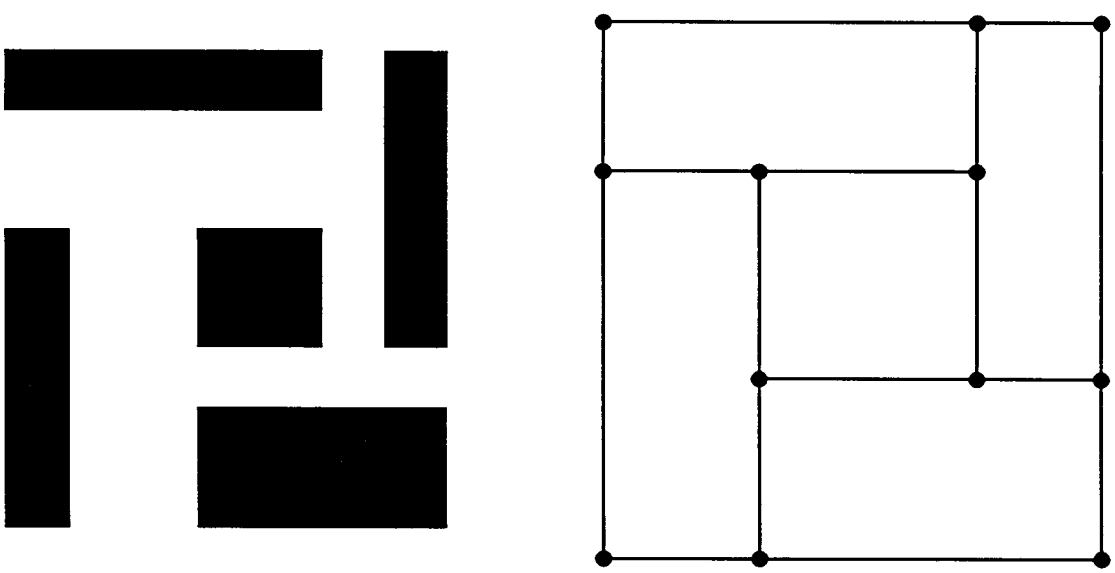


Figure 2–2: *Cell placement and corresponding channel-graph*

Figure 2–2 shows a cell placement and its corresponding channel-graph. An edge in the graph represents a channel (which is the space between 2 cells), while a vertex in the graph represents the intersection of 2, 3 or 4 channels. The advantage of using a channel-graph is that information, such as channel width or channel

occupation by routing, can be attached to the edges, and global wire routes can be found very efficiently.

2.3 Global Routing

2.3.1 Statement of the Problem

A global router takes as input a cell placement and a net-list specifying the required connections between pins on cell edges. An example specification is shown in Figure 2–3. Pins on the cell edges with the same letter belong to the same net and need to be connected together.

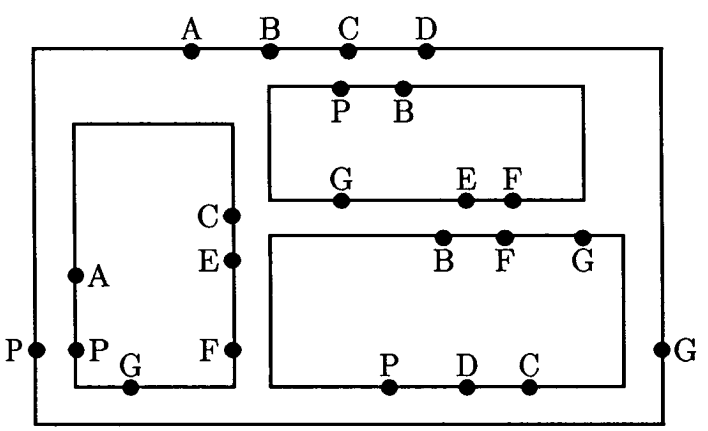


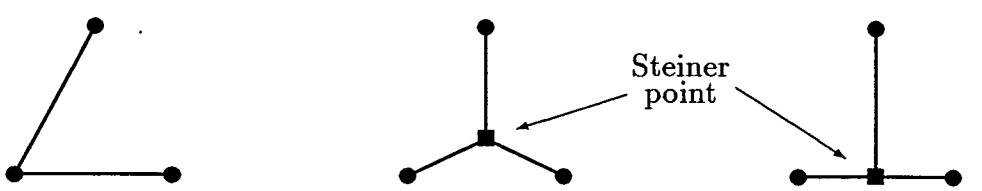
Figure 2–3: *Global routing specification*

2.3.2 Steiner Nets

One of the goals of global routing is to find the shortest total length of wiring that connects the pins of each net. This involves finding an optimum rectilinear Steiner tree for the pins. A rectilinear Steiner tree for the pins is defined to be a tree structure, composed solely of horizontal and vertical wire segments,

which interconnects all the pins [Gare77]. An optimum rectilinear Steiner tree for the pins is one in which the line segments used have the shortest possible total length. However, Garey and Johnson [Gare77] have shown that the problem of determining the minimum length given the pins is *NP*-Complete. Thus, the problem of finding optimum rectilinear Steiner trees for all the nets is probably computationally hopeless, and so the emphasis should be on finding heuristics and special case algorithms.

The shortest Steiner tree problem is similar to the shortest spanning tree problem, but it allows connecting points called Steiner points, in addition to the pins. The shortest Steiner problem involves the pins connected on the Euclidean plane being connected by lines such that the total length of the lines is a minimum. The rectilinear Steiner problem is the Steiner problem with the additional constraint that only horizontal and vertical lines may be used. Figure 2–4 shows the shortest spanning tree, a shortest Steiner tree and an optimum rectilinear Steiner tree for a 3-pin net.



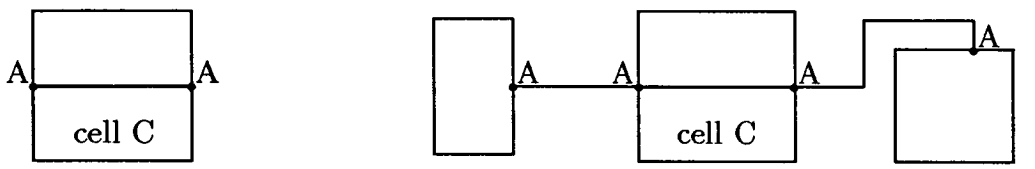
(a) shortest spanning tree (b) shortest Steiner tree in Euclidean space (c) optimum rectilinear Steiner tree

Figure 2–4: *Connection examples for a 3-pin net*

Global routers try to minimize the net length because the less wiring there is in a design, the smaller the overall design area will generally be. The Steiner trees shown in Figure 2–4 do not have their paths impeded by cells. A further complicating factor in chip design is that the global router must route nets around or over any intervening cells.

2.3.3 Implementation Issues

Pin Choice



(a) pin choice (b) use of internal cell connection

Figure 2–5: *Alternative pins and feedthroughs*

In routing a net there are additional variables that need to be considered. Often within a cell, instead of just one pin that has to be connected, there is a choice of pins, such that one of a set has to be connected. An example of a choice of pins occurs when a clock bus runs the length of a cell, as shown in Figure 2–5(a). In this example, it is sufficient to connect either of the pins of net A.

The situation is further complicated in that it is possible to feedthrough a cell as part of the net connection. In Figure 2–5(b) both pins of net A in cell C are connected to realise a shorter net. This illustrates a pre-existing feedthrough. An alternative form of feedthrough is where it is possible to route a wire through a cell. This feedthrough can then be assigned to any net to form part of its connection.

Feedthroughs can also be used in a hierarchical design environment. As an example consider the cells of Figure 2–5(b) as previously assembled parts of the hierarchy, with part C containing an unassigned feedthrough. Then, when assembling the parts a shorter connection length can be achieved by using the feedthrough. In the hierarchical design environment the parts can be large, and consequently significant savings in connection length can be made if feedthroughs are used.

Pin Ordering

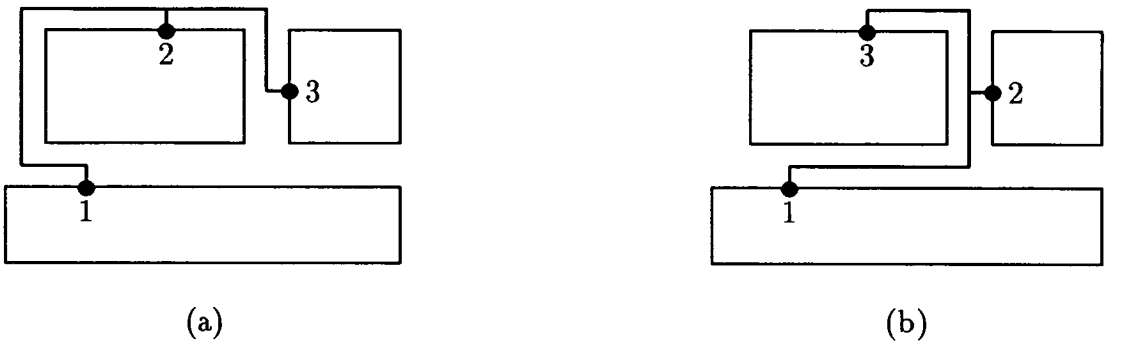


Figure 2–6: *Effect of pin ordering on global wire length*

A complete net is formed by joining the pins of a net together. If the pins are considered one at a time then the length of the global route is particularly sensitive to the order in which the pins are connected together. Figure 2–6(a) shows a pin ordering for a net that results in a longer path than if the pin ordering shown in Figure 2–6(b) is used.

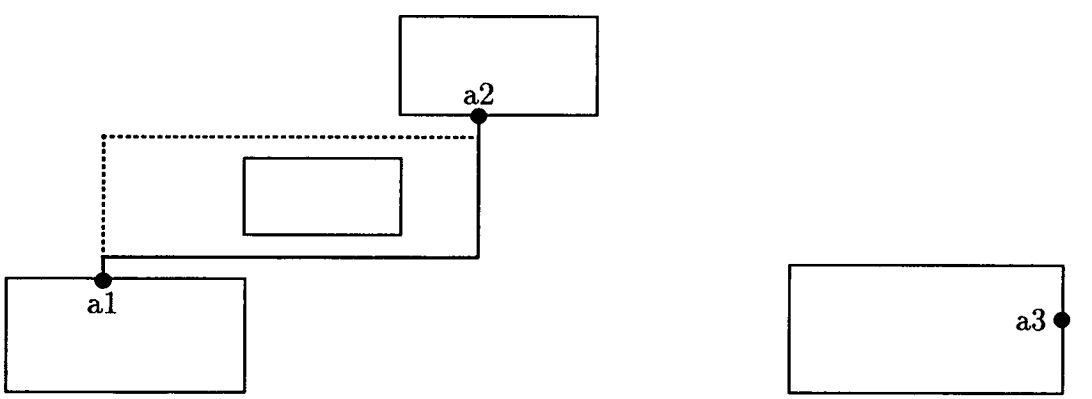


Figure 2–7: *Influence of as yet unconnected pins*

In connecting the pins together one at a time, knowledge of the positions of the as yet unconnected pins can reduce the length of the global route. Figure 2–7 shows two alternative paths for connecting pins a1 and a2. The path shown by

the solid line is preferable given the position of pin a3 that is subsequently going to be connected.

Power Routing

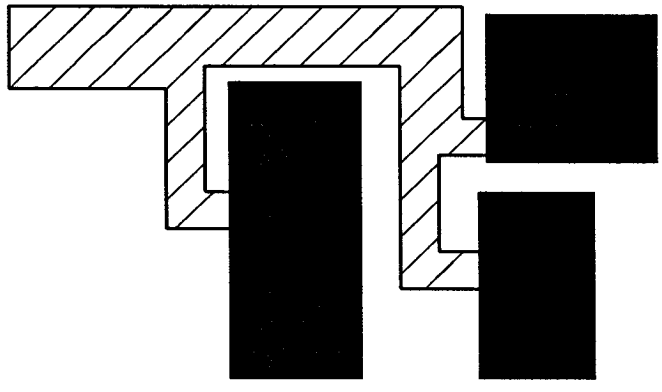


Figure 2–8: *Tapered power routing*

Each cell, at any level of the design hierarchy, needs to be connected to power and ground. The power and ground routings need to be all metal because of the high currents involved and the much higher resistivity of polysilicon and diffusion. The width of the metal wire is dependent on the current carried, resulting in the need for tapered-width routes as illustrated in Figure 2–8. Consequently, assemblers sometimes have a power router that is distinct from the global router. There is also an additional constraint that if there is only one metal layer available, then the power and ground routings must be non-overlapping (since if they overlapped part of the routing would have to be moved into polysilicon).

2.3.4 Summary

What is required is a global router that handles pin choice and pin ordering efficiently and effectively. Power nets have additional constraints that need to be satisfied. In a hierarchical design environment, it is not obvious when or where

to feedthrough parts of a design. Pin choice introduces another degree of freedom into an already difficult problem, and so it is necessary to find and use heuristics to reduce the time taken to find a solution.

2.4 Placement Adjustment

Having determined the global routing it is usually necessary to adjust the cell placement to obtain a better match between the wiring capacities of the spaces between the cells and the wiring routes found by the routers. This is illustrated in Figure 2–9(a). In adjusting the cell placement, the relative positions of the cells are altered and the channel structure is changed. If the adjusted cell placement is global routed again, then routes may be found that require further placement adjustment as illustrated in Figure 2–9(b).

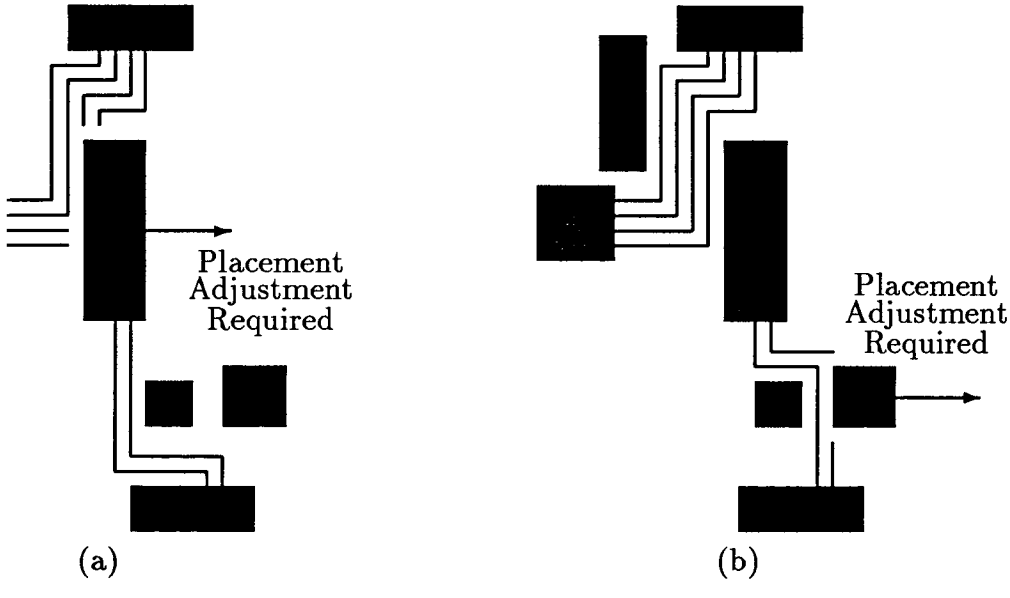
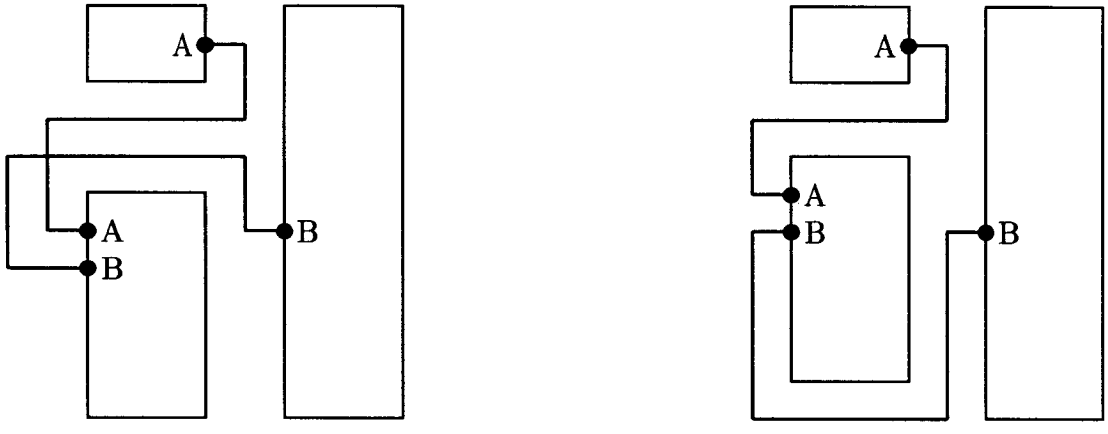


Figure 2–9: *Placement adjustment caused by global routing*

2.4.1 Interaction with Global Routing

One of the goals of Silicon Assembly is to minimize the design area of a chip. Unfortunately, choosing the shortest path for net B as shown in Figure 2–10(a) leads to a larger design area than if a longer path had been taken for the net as shown in Figure 2–10(b). This is an example of the trade-off between the assembler goals of minimizing the interconnection length and minimizing the chip area.



(a) shortest path (b) minimum design area

Figure 2–10: *Shortest path versus minimum design area*

The routing shown in Figure 2–10(b) results in a smaller design size due to the better match between the wiring capacities of the spaces between the cells and the number of global wires passing through the spaces. Figure 2–10(a) shows that the smallest layout area is not necessarily achieved by trying to find the shortest route for every net. This is because routing each net independently of all the other nets can result in the wiring capacities being exceeded in some parts of the design, whereas in other parts there is excess capacity. As is shown in Figure 2–10(b), it is sometimes better to use slightly longer routes for some nets to try to ensure that wiring capacity is not exceeded elsewhere in the design (though, it may be that design constraints make it more important to find the minimum length for net B, despite the increased chip area). Deciding when it is better to consider capacity

constraints in finding net routes is a hard problem. Capacity constraints should not be considered before there is a good match between the capacities and the wiring routes found. Otherwise, global routing driven by capacity constraints can lead to long nets which, use up more wiring capacity, resulting in the placement being adjusted in the wrong parts of the design.

There are two approaches that can be taken toward global routing driven by capacity constraints, but either way a net ordering problem is introduced. All the nets can be routed and then some of the nets can be ripped-up and re-routed in the areas where the capacity is exceeded. Alternatively, the nets can be routed such that the capacity is never exceeded in the design. With the former approach the time taken is longer, because some nets are routed more than once, and there is a net ordering problem in deciding which nets to rip-up and re-route. In the latter approach the nets become more constrained in their routings as the global route proceeds. Whichever approach is used, the global router will be unable to find a route for all the nets, if there is insufficient space in the design. When implementing a global router it is necessary to be able to detect that there is insufficient space to route all the nets. Otherwise the global router will not terminate.

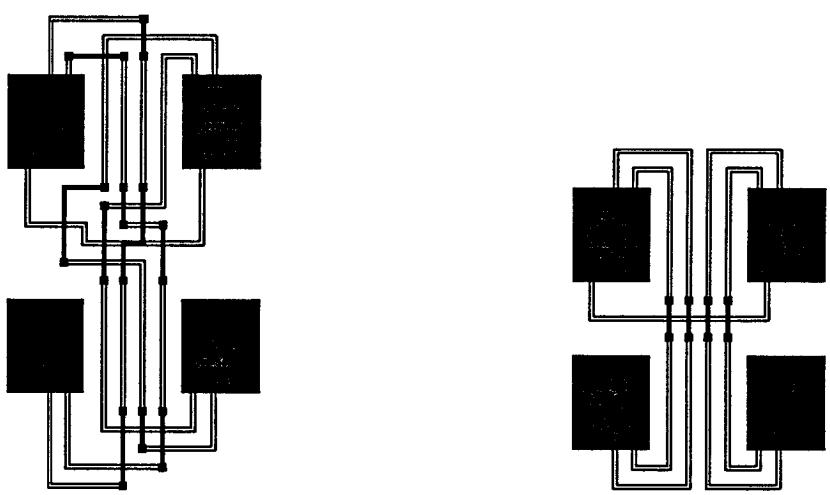
2.4.2 Summary

The interaction between global routing and placement adjustment imposes another constraint on the already difficult problem of global routing. There is a trade-off between the goals of minimizing the chip area and minimizing the length of the wiring. Minimizing the length of the wiring is generally a good idea, because it involves less channel occupation, but there comes a point where global routing driven by capacity constraints can lead to smaller chip design areas. Determining the point for changing between shortest path driven and capacity constraint driven global routing is a difficult task. A means of assessing the effect of the global

routing on the placement needs to be found, and heuristics for net ordering are required.

2.5 Terminal Assignment

A global router finds paths for wires between the cells. When the global routing has been completed paths have been found for all the wires. These paths can be specified as a sequence of channels that the wires pass through. However, for each channel there is no relative positioning of the wires, the only information there is about position is which channel edges a wire crosses to pass through a channel.



(a) Without terminal assignment (b) With terminal assignment

Figure 2–11: *The effect of terminal assignment*

After the global routing, detailed routing of the global wires in each channel must be performed. Along the common edges between adjacent channels, connecting pins need to be introduced to enable connections between the parts of the nets that span more than one channel. A detailed routing problem consists of a set of terminals which need to be connected together. These terminals are either pins on the cell edges whose positions are fixed, or connecting pins on channel

edges whose positions are to be determined. It is not necessary to have a separate terminal assignment step. Instead, many detailed routers are designed such that terminal assignment is performed automatically as part of the detailed routing. However, the locality of this "bottom-up" ordering of the terminals disregards the global topologies of the nets which span several channels. This may result in twisted wires, unnecessary vias and wasted area as illustrated in Figure 2–11(a).

The layout shown in Figure 2–11(a) could be improved. The goal of performing terminal assignment as a separate step is to determine an ordering for the terminals such that unnecessary twisting of the wires is prevented and the unavoidable wire crossovers occur only once. Performing terminal assignment this way has several advantages:

- ▷ The structured order of the terminals makes the channels easier to route. The routing area will decrease and, with that, possibly the overall chip area.

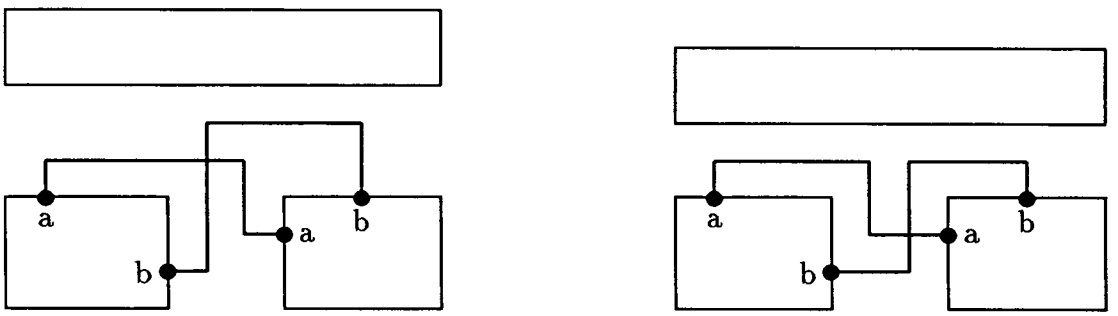
- ▷ Reducing the number of twisted wires leads to better electrical characteristics of the wiring layout. Fewer vias are required and the wires are generally shorter because they do not have to cross unnecessarily.

- ▷ Busses will be routed in a consistent order since they consist of a several wires which usually share the same topology.

Figure 2–11(b) illustrates the effect of terminal placement on the layout of the design. The design area is smaller, the wire length is shorter and there are fewer vias than in the layout of Figure 2–11(a).

The example shown in Figure 2–11 is taken from Groeneveld [Groe89]. In this paper an algorithm to prevent wires being twisted during detailed routing is presented. A consistent order for the terminals is found such that no unnecessary twisting of the wires is introduced. It is also recognised that there is usually a

choice as to where wires actually cross. An heuristic is described which has the aim of positioning any crossings such that the chip area is least affected.



(a) crossing assignment (b) better crossing assignment

Figure 2–12: *Effect of crossing assignment on routing area*

Figure 2–12(a) shows a crossing assignment that leads to a larger routing area than if the crossing assignment shown in Figure 2–12(b) were adopted.

2.5.1 Summary

There are clear advantages to be gained by performing terminal assignment as a separate step before detailed routing. The advantages derive from the ability of terminal assignment to take a global overview of the problem, something a detailed router cannot conveniently do.

2.6 Detailed Routing

2.6.1 Statement of the Problem

The specification consists of an area for routing - often called a channel - together with a set of terminals around the periphery of the routing area. Terminals of the same net need to be connected together such that the connecting wires satisfy the

design rules and there are no design rule infringements between adjacent wires. An example routing problem, taken from [Burs83], is shown in Figure 2–13.

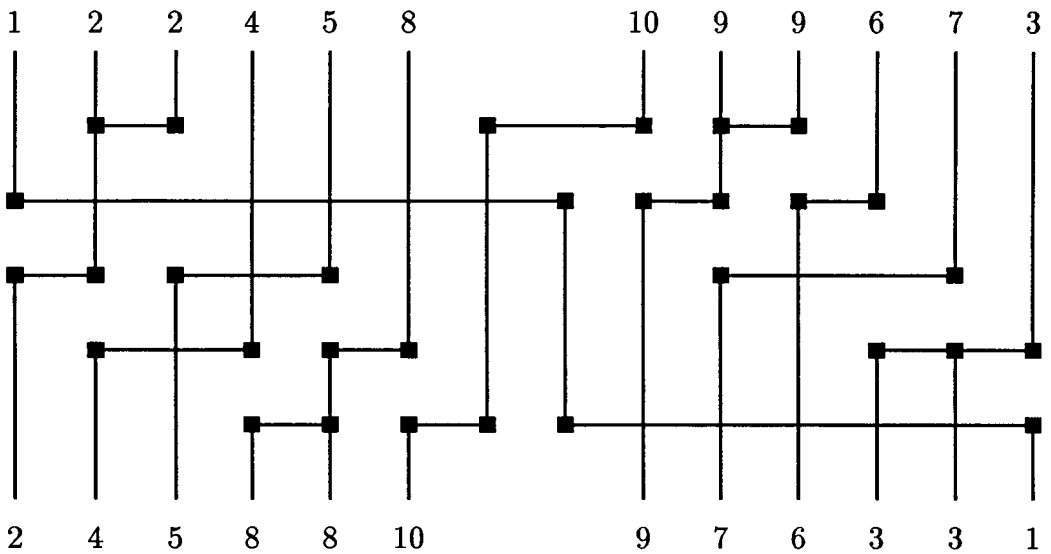
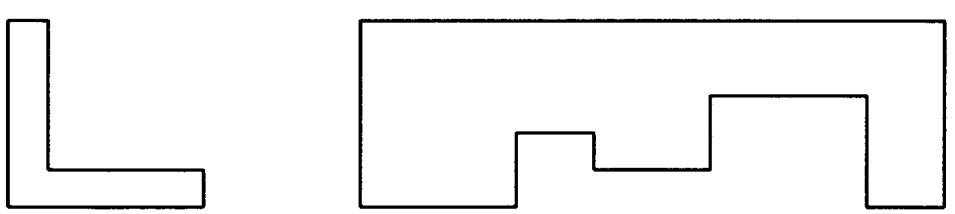


Figure 2–13: *Detailed routing specification and solution*

Routing Area

The general routing problem has terminals to be connected on any side of the routing area, which may not necessarily be rectangular. Depending on the silicon assembler, the area is usually rectilinear, of which the most common shapes are L-shaped as shown in Figure 2–14(a) and ragged edge as shown in Figure 2–14(b).

Within the routing area there can be obstacles, which have to be routed around or passed over or under by the detailed routing. These obstacles can be cells



(a) L-shaped (b) Ragged edge

Figure 2–14: *Common rectilinear routing areas*

(if the routing area is large) or previously wired nets. Previously wired nets in assemblers are often power and ground nets with their all-metal and tapered wiring requirements, or critical nets that have to be as short as possible and all-metal because the delay-times between cells are important for the correct functioning of the chip. There is now less need to route nets manually as performance and timing requirements are being automatically incorporated into silicon compilers and assemblers.

2.6.2 Routing Model

Routing Grid

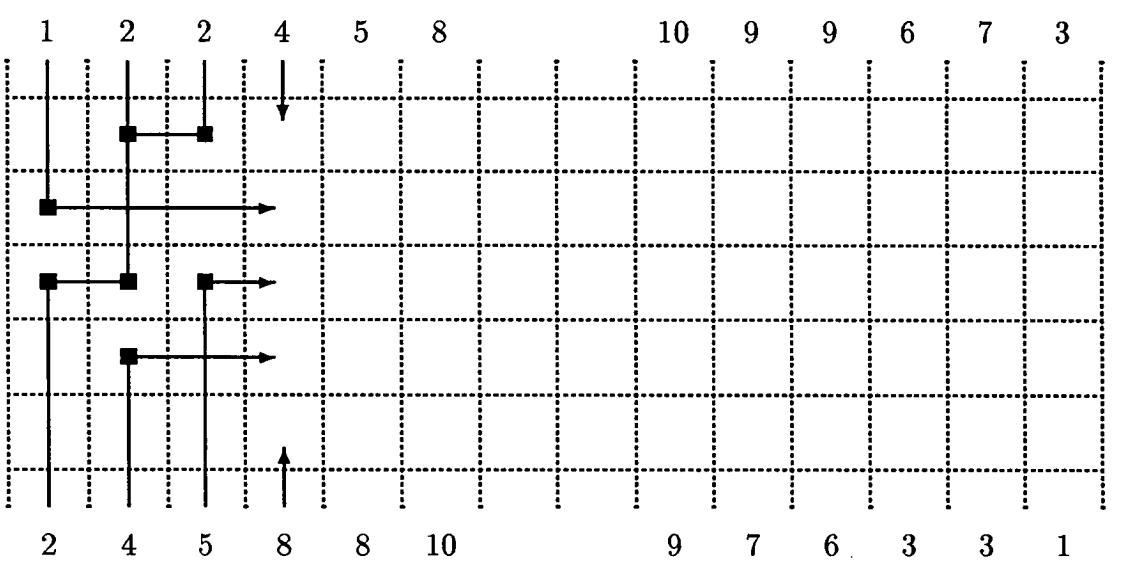


Figure 2–15: *Partially routed problem with grid*

The example shown in Figure 2–15 has the terminals regularly spaced and opposite each other. This allows the channel to be viewed as a series of vertical columns and horizontal tracks, which form a routing grid. The example has 14 columns and needs a minimum of 5 tracks for a detailed routing solution to be found. Using the column and track view of the problem, a detailed routing can

be found by routing each column in turn from left to right. For each column, the router has to consider the horizontal wires from previously routed columns, together with vertical wires emanating from terminals that may be present at the top or bottom of the column. This is illustrated schematically in Figure 2–15, which shows the track and column structure of the problem together with its routing completed as far as the third column.

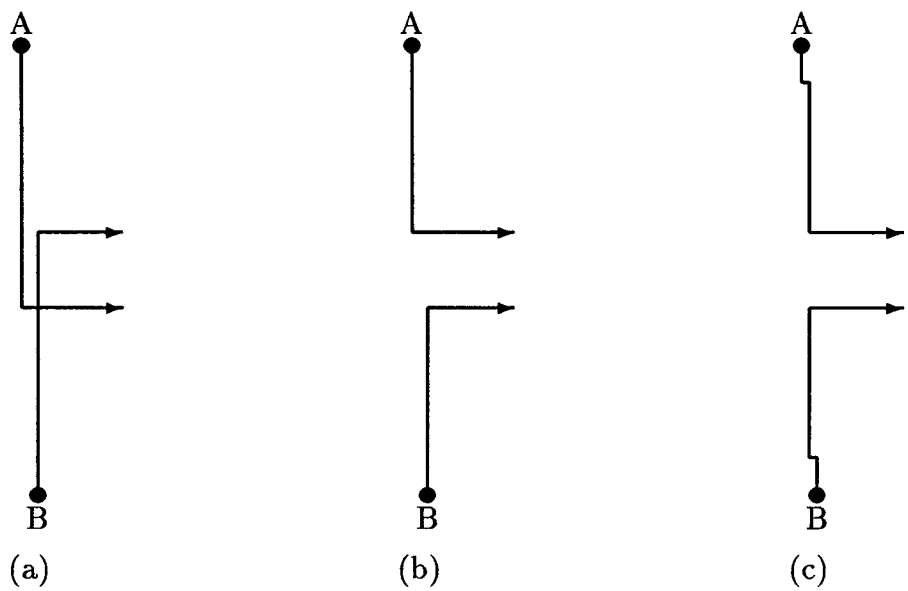
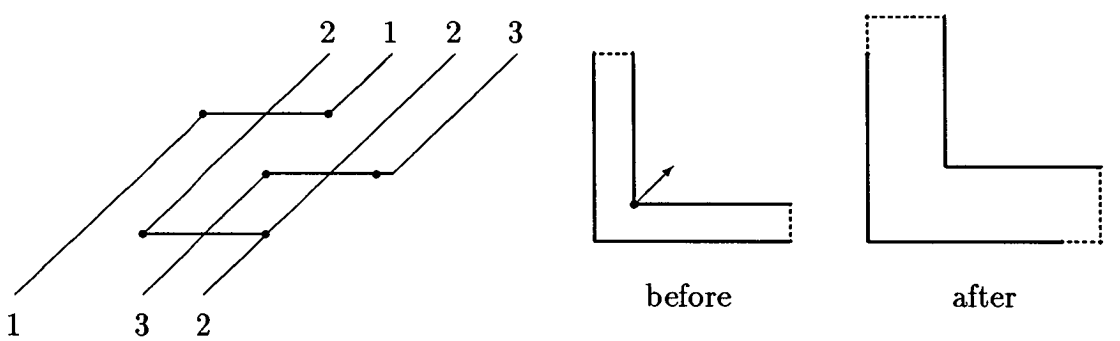


Figure 2–16: *Routing grid considerations*

In the general routing problem, however, the terminals do not have to be grid aligned. This makes the detailed routing algorithm more complicated because it is not then possible to consider a column at a time. This is illustrated in Figure 2–16(a). Terminals A and B lie on opposite faces of the channel and are not grid aligned. Consequently, the routing shown results in a design rule violation between the vertical wires emanating from A and B. To prevent this happening the detailed routing algorithm needs to be modified. When starting the route from net A, the router needs to look ahead and note that net B is present, and when routing net B the router needs to look back and note the routing of net A. This procedure enables

a routing that satisfies the design rules to be found, as illustrated in Figure 2–16(b). Alternatively, a routing grid can be imposed on the terminals by displacing the terminals by a small amount around the periphery of the channels so that they are aligned on the grid. The simpler column-based approach to detailed routing can then be applied. This is illustrated in Figure 2–16(c).

45° Routing



(a) example of 45° routing

(b) L-shaped channel adjustment when using 45° routing

Figure 2–17: *Example and use of* 45° *detailed routing*

Provided the design rules are not infringed, there is no reason why the detailed routing should be restricted to finding orthogonal routings. Instead 45° routing as illustrated in Figure 2–17(a) could be used. This has the advantage that in the L-shaped channel routing problem the two L-shaped faces can be separated by one 45° movement as illustrated in Figure 2–17(b). There is a problem if orthogonal routing is used to route L-shaped channels, because a horizontal adjustment of the L-faces affects the vertical routing and a vertical adjustment affects the horizontal routing.

Layering Constraints

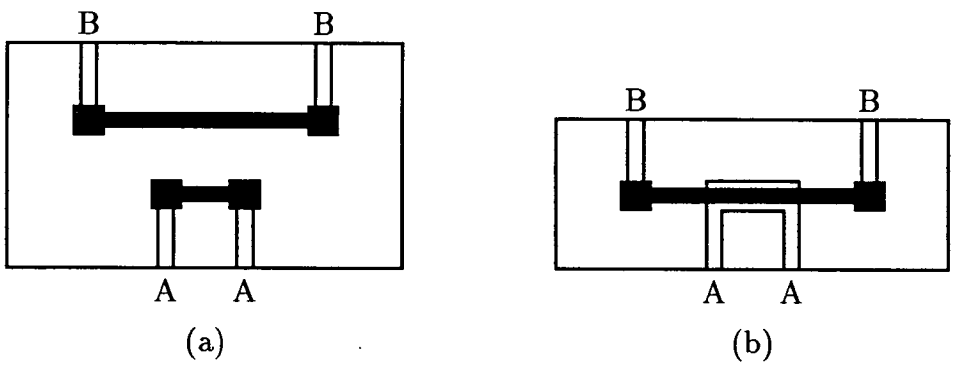


Figure 2–18: *Effect of layering constraints in detailed routing*

To make it easier to find solutions to routing problems, detailed routers usually have a routing layer restriction that all the horizontal wiring is on one layer and all the vertical on another, with connections between the layers being made by vias placed at the intersection points. When trying to find a detailed route that will fit in as small an area as possible, this layering constraint may cost a track as shown in Figure 2–18. Figure 2–18(a) shows the routing solution with the layering constraints applied. Figure 2–18(b) shows a routing solution with one less track, which can be found if the layering constraints are relaxed. However, in practice the opportunity to find routing solutions as shown in Figure 2–18(b) rarely occurs. Also, the layering constraint makes it much easier to find a solution in that to ensure no two nets are shorted it is only necessary to ensure separately that no vertical wiring overlaps and no horizontal wiring overlaps. Depending on the electrical characteristics of the fabricated wires, it may be that the layering constraint is imposed to prevent the layers overlapping to reduce signal cross-talk and wire capacitance.

The two-layer model just described can be generalized to n-layers. A typical n-layer scheme involves the horizontal wires being routed on layers $1, 3, 5, \ldots$ and the vertical wires being routed on layers $2, 4, 6, \ldots$ up to the number of layers n. The horizontal and vertical wires are assigned alternating layers to enable good

communication between layers, in that a wire on layer i can go down to layer $i-1$ or up to layer $i+1$, using a via to make the join between layers. It may be possible to use a via to communicate with other layers, but this is dependent on the rules imposed by the fabrication technology. The horizontal wires of different layers can be stacked on top of one another and the vertical wires of different layers can also be stacked on top of one another. Stacking wires this way saves a large amount of space, however the complexity of the routing problem is increased by the number of additional choices that have to be made.

The need to Dogleg

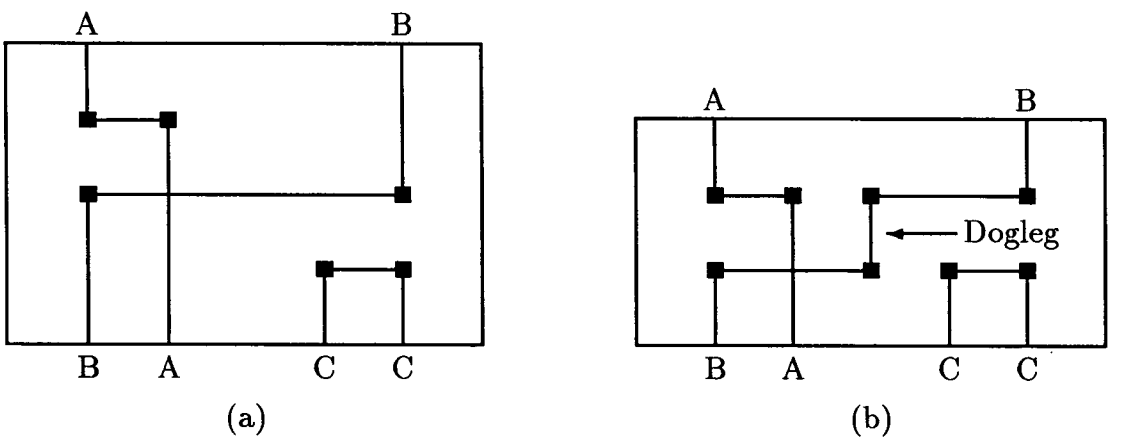
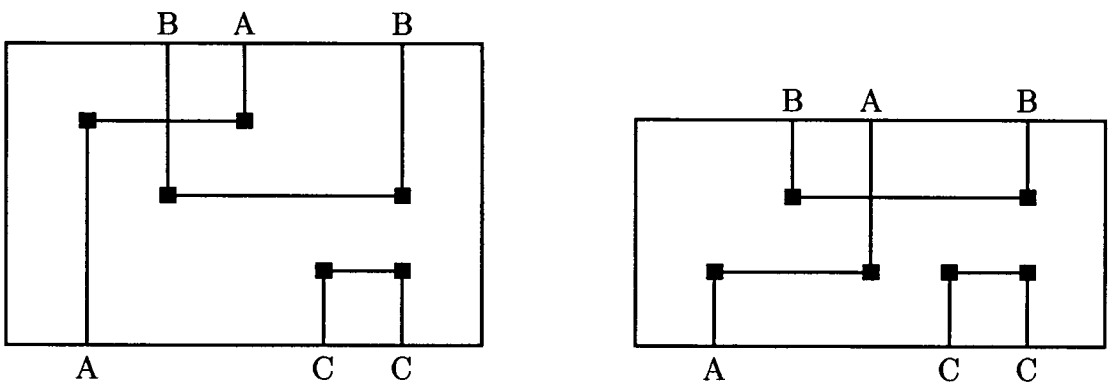


Figure 2–19: *Use of a dogleg*

The layering constraint resulting in no overlapping wires has implications during routing. If there are two vertical wires in a column of a channel belonging to different nets, then any horizontal wire attached to the lower vertical wire must be placed below any horizontal wire attached to the upper vertical wire (otherwise overlap would occur). This is seen in the first column of the routing example shown in Figure 2–19(a). In this example three tracks are required to complete the routing. However, if a dogleg is introduced then the routing can be completed in two tracks as shown in Figure 2–19(b). It is not possible to complete the

routing in two tracks without the dogleg, because of vertical constraints caused by terminals of different nets being opposite each other in the channel.

Net Interaction



(a) Column at a time (b) Column look ahead

Figure 2–20: *Advantage of column look ahead*

The example shown in Figure 2–20(a) requires three tracks to complete the routing, whereas, Figure 2–20(b) shows a routing solution requiring two tracks. If the detailed router finds a solution by considering a column at a time going from left to right then the solution shown in Figure 2–20(a) will be found. To find the solution shown in Figure 2–20(b), the router needs to look ahead and consider more than just a column at a time during the routing.

2.6.3 Goals of Detailed Routing

Not all the goals are always applicable. Depending on the routing model, the detailed routing problem may be a subset of the general problem. For example, there may only be terminals on two opposite sides of the channel or the channel may be rectangular, or both.

Smallest Space and Given Space

Detailed routers are often required to try to find a solution that occupies as small a space as possible. The example shown in Figure 2–20 shows how a track less can be used if the detailed router can find the better solution. Sometimes the goal of smallest space may not be applicable, instead the assembler may require a detailed routing to be found for a given space. (For example, when the cells have fixed positions and the channels are therefore of fixed width). However, to ensure that the router finds a route that will fit in the given space, some routers try to find the smallest space routing and then, if this space is smaller than the given space, expand the routing solution by lengthening wires, to fit the given space.

Using the routing model involving 2-layer wiring with horizontal wiring on one layer and vertical wiring on the other and only orthogonal routing (no 45° routing allowed), Szymanski [Szym85] has proved that it is an *NP*-Complete problem to determine whether an arbitrary channel can be routed using a specified number of tracks. Szymanski also gives as a corollary of his proof that the following problems are also *NP*-Complete:

- ▷ Determining the minimum number of tracks needed to route a channel.
- ▷ Determining whether a given routing for a channel is optimal.
- ▷ Producing a routing of minimum separation (or area, or perimeter, or total wire length) for a channel.

Although determining the minimum number of tracks needed to route a channel is *NP*-Complete, determining a mathematical lower bound - the channel density - on the number of tracks needed to route the channel is not [Deut76]. To calculate the channel density, it is necessary to find the extent of each net, which is the region between the leftmost and rightmost terminal positions of the net

in the channel. Any nets for which the leftmost and rightmost positions are the same (straight across connections) are ignored. For each terminal position in the channel, count the number of nets whose extents include that position. This is the local density. The channel density is the maximum local density and the span is defined as the number of terminal positions where this maximum occurs.

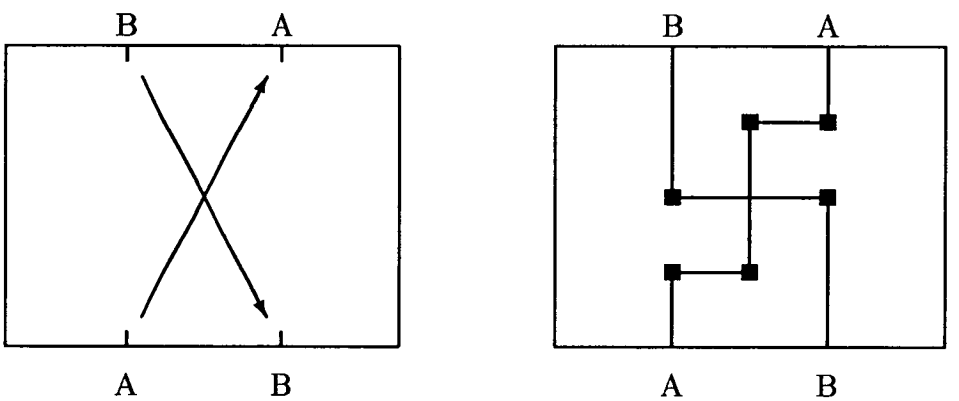


Figure 2–21: *A cyclic routing constraint and solution*

Both the channel density and span are properties of the terminal positions and the connectivity data and are independent of the routing, which is why determining the minimum number of tracks to route an arbitrary channel is not the same as determining the channel density. Figure 2–21 is an example of where the number of tracks needed to route a channel is greater than the channel density. The channel density is two, yet because of the terminal positions and connectivity three tracks are required to complete the routing. The terminal positions with net B above net A and then further down the channel net A above net B, form what is known as a cyclic constraint.

Given that the detailed routing problem is known to be *NP*-Complete, it is best to find heuristics and special case algorithms that consistently find detailed routings in a space equal to or close to the minimum amount given by the channel density.

Reasonable Time

The detailed routing problem is known to be *NP*-Complete [Szym85]. The channels to be routed can be large with over 100 nets in a channel. Consequently, the routing algorithms must control the computational complexity, otherwise routing situations could occur where the router may take too long to find a solution. This consideration is particularly applicable to those algorithms that employ backtracking as part of their search for a solution.

Minimize Crossings

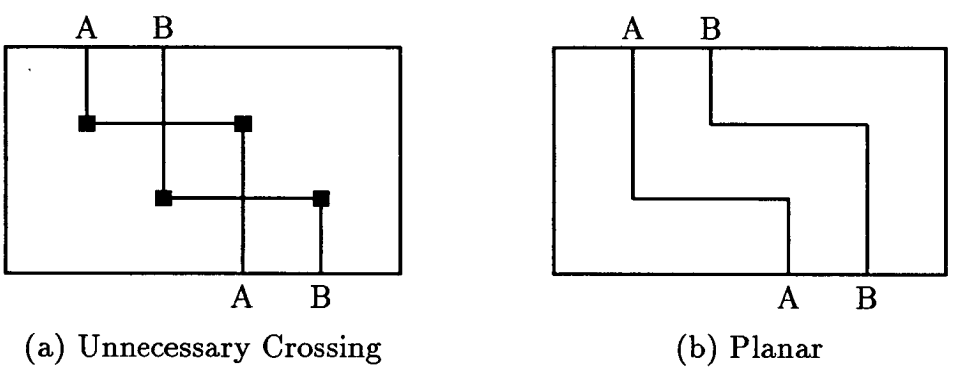
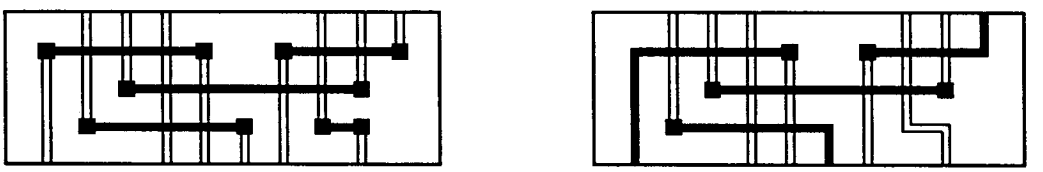


Figure 2–22: *Net crossing in detailed routing*

The goal of terminal assignment is to determine an ordering for the terminals such that unnecessary twisting of the wires is prevented and the unavoidable wire crossovers occur only once. Terminal assignment operates at the level of specifying the positions of the terminals on the channel faces, but not at the level specifying the positions of the wires in the channels. Figure 2–22(a) shows two wires crossing unnecessarily within a channel, and Figure 2–22(b) shows a simple routing for the two wires that is planar. The detailed router must take care to avoid introducing unnecessary wire crossings, otherwise the goals of terminal assignment will be undermined.

2.6.4 Via Minimization

In detailed routing where two conducting layers are available for interconnection, the routing model (to make it easier to find a routing) assigns all horizontal wire segments to one layer and all the vertical wire segments to the other layer. However, this method uses many unnecessary vias. Vias introduce some drawbacks. Besides increasing the manufacturing cost and complexity, vias degrade its performance and reliability [The89]. The via minimization problem is how to minimize the number of vias used in a layout.

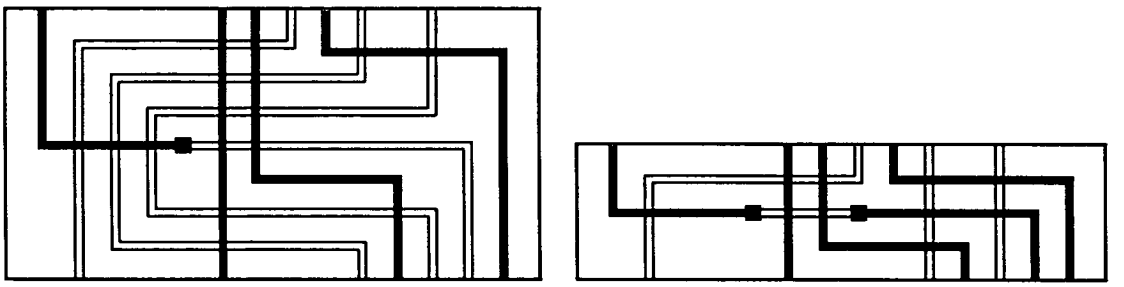


(a) Before via minimization (b) After via minimization

Figure 2–23: *Constrained via minimization*

An example of Constrained Via Minimization - CVM - is shown in Figure 2–23. Figure 2–23(a) shows the layout before via minimization and Figure 2–23(b) shows the layout after. The CVM approach takes a given layout and assigns layers to the wire segments such that the number of vias is minimized, subject to the constraint that the layout does not change. However, as this example shows, the layers of the wires emanating from the terminals may be changed.

An alternative approach known as Unconstrained Via Minimization - UVM - involves first finding a topological routing of the nets such that the number of required vias is a minimum, and then performing a geometrical mapping of the routing into layout. An example of a UVM layout is shown in Figure 2–24(a). However, as the example shows, absolutely minimizing the number of vias may require many nets to meander around a via, which results in a larger routing area. Compared to the seven tracks and one via needed in Figure 2–24(a), the



(a) Unconstrained via minimization with only one via

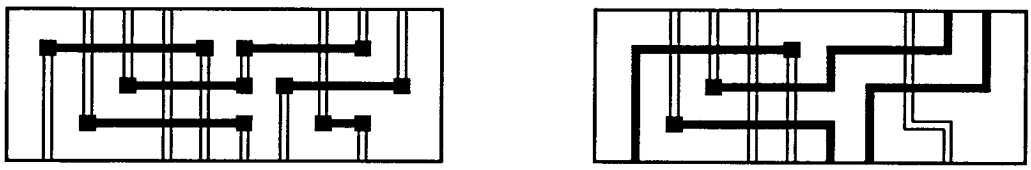
(b) A more compact layout with two vias

Figure 2–24: *Trade-off between layout space and number of vias*

routing can be achieved with only three tracks by using two vias as shown in Figure 2–24(b).

The UVM approach minimizes the number of vias, often at the expense of a larger layout area. The CVM approach minimizes the number of vias in the given layout. Neither approach is satisfactory. A new approach which combines the advantages of the UVM and CVM approaches is described in [The89]. That method is based on the observation that the same routing problem can have different layout solutions possibly requiring different minimum numbers of vias. An alternative layout for Figure 2–23(a) is shown in Figure 2–25(a). This layout when the number of vias is minimized requires three vias as shown in Figure 2–25(b), compared with the five vias required in Figure 2–23(b). This approach eliminates vias by systematic modification of the layout. The algorithm can produce better results than optimal CVM algorithms, and yet the routing area is not increased, because no new tracks are inserted during the modification.

A further complication in via minimization is maintaining layer communication at channel boundaries which are bounded by cells. Once a channel has been via minimized, the layer constraints of the routing model no longer hold. This can be seen in Figure 2–25(b) where some of the vertical wires emanating from the boundary are on one layer and the rest are on the other layer. Often in assembly, the terminals which are pins of cells are on a particular layer. This creates a



(a) Before via minimization (b) After via minimization

Figure 2–25: *Constrained via minimization of alternative layout*

further constraint on the via minimization problem, because the minimization process does not have the freedom to place the wires emanating from cell pins on any layer. Unless the wire emanating from the cell pin is placed on the same layer as the pin, a via must be placed to join the pin layer to the wire layer.

In assembly, if via minimization is performed on a channel-by-channel basis, there is a problem of communicating layer information across adjacent channel boundaries, with previous via minimizations in adjacent channels imposing layer constraints on a channel similar to those imposed by cells bounding channel edges. To avoid this problem, via minimization must be performed on a chip-wide basis, taking as input all the detailed routing together with the pin layer constraints of the cells. However, this is a much larger problem, the complexity of which may be too great for some via minimization procedures.

2.6.5 A Note on Terminology

There are three types of detailed router:

- ▷ Channel.
- ▷ Switchbox.
- ▷ Planar.

A channel router has terminals on just one face or two opposite faces of the routing area. This routing area is often called a channel. A switchbox router has

terminals on any face of the channel. A planar router, like the switchbox router, has terminals on any face of the channel, but its detailed routing operates with the additional constraint that the wires for each net must not cross each other. Consequently, there is no need for vias and the routing can be realised on a single layer. However, it is highly unlikely that a general routing problem would be planar, unless it was specially constructed with this end in mind.

The difference between channel and switchbox routers is that a switchbox router is more general than a channel router. The distinction between channel and switchbox routers is blurred in that some channel routers, allow terminals to be specified on the other faces of the channel, but unlike switchbox routers their exact position is fixed during the channel routing.

To further confuse the terminology, in silicon assembly the spaces between cells are broken down into channels which are routed in detail. This step is sometimes called channel routing with a channel router (which is really a switchbox router by the above definition) being used to route the channels. To avoid further confusion, this dissertation uses the term detailed router which finds a detailed routing for terminals positioned on the faces of the channel.

2.6.6 Summary

Detailed routing is an *NP*-Complete problem [Szym85]. Before a detailed router is implemented, it is necessary to be clear which routing model is being used. Often the routing model is a simplification of the general model. Simplifications include:

- ▷ Are the channels always a particular shape?
- ▷ Are the terminals only positioned on some of the channel faces?
- ▷ Do the terminals always lie on a grid?

- ▷ Is only orthogonal routing allowed?
- ▷ Are there no obstacles in the channel?
- ▷ Is a specific number of routing layers available?

The presence of simplifications of the general routing model can make the implementation of a detailed router much simpler and the router may be able to exploit the restrictions on the routing model by reducing the computational complexity of the search for a solution, whilst not excluding potential solutions.

2.7 Compaction

One of the goals of assembly is to minimize the overall design area of the chip. Often, in performing the other steps of assembly, more design area is used than is needed. A layout produced by an assembler usually results in a larger design area than if hand layout was used. Consequently, many assemblers use a compactor to try to reduce the overall design area of the chip by removing unnecessary spaces from the design.

The compactor takes as input an initial layout and without changing its topology, tries to find a layout with minimum design area consistent with the design rules [Mlyn86]. The constraint of not changing the topology is necessary in order not to render the previous steps of placement and routing obsolete. It is achieved by maintaining the adjacency of layout elements, for example transistors and cells, which are not allowed to jump across each other during compaction.

In assembly the layout compaction problem can be formulated as taking rectangular cells with pins on the boundaries which are connected by wires that have fixed width, but which can be stretched or shrunk and can have jogs inserted.

The goal is then to compact the layout subject to minimum distance constraints and topological constraints such that its bounding rectangle has minimum area. Schlag et al. [Schl83] have shown that under this formulation, the two-dimensional compaction approach of doing compaction simultaneously in both directions is *NP*-Complete. Consequently, to control the complexity, many compactors adopt the strategy of using two consecutive one-dimensional compactions of either a horizontal followed by a vertical compaction, or a vertical followed by a horizontal compaction.

2.8 Summary

Silicon assembly is a complex combinatorial optimization problem, and even simplified versions of the layout optimization problem are *NP*-Complete [Sang87]. The silicon assembly problem contains several steps - floorplanning and placement, global routing, detailed routing and compaction - that are *NP*-Complete. Clearly, the individual steps of assembly are hard problems, combining the steps together (as the analysis and discussion in the subsequent chapters will show) is a hard problem. Quite simply, silicon assembly is a hard problem.

Chapter 3

Stepwise Silicon Assembly

This chapter gives details of the stepwise approach to silicon assembly. As a prelude to explaining the ideas behind the stepwise approach, some earlier work discussing the problems of wire routing two-layer printed circuit boards is described. Viewed within an assembly context, the work can be seen as a simple, stepwise assembler. Details are given of the implementation of a stepwise assembler that was used to assemble an industrial chip design. The stepwise approach is analysed and the fundamental limitations of this approach are exposed. Other stepwise assemblers are described, and these too are shown to suffer from limitations inherent in the stepwise approach.

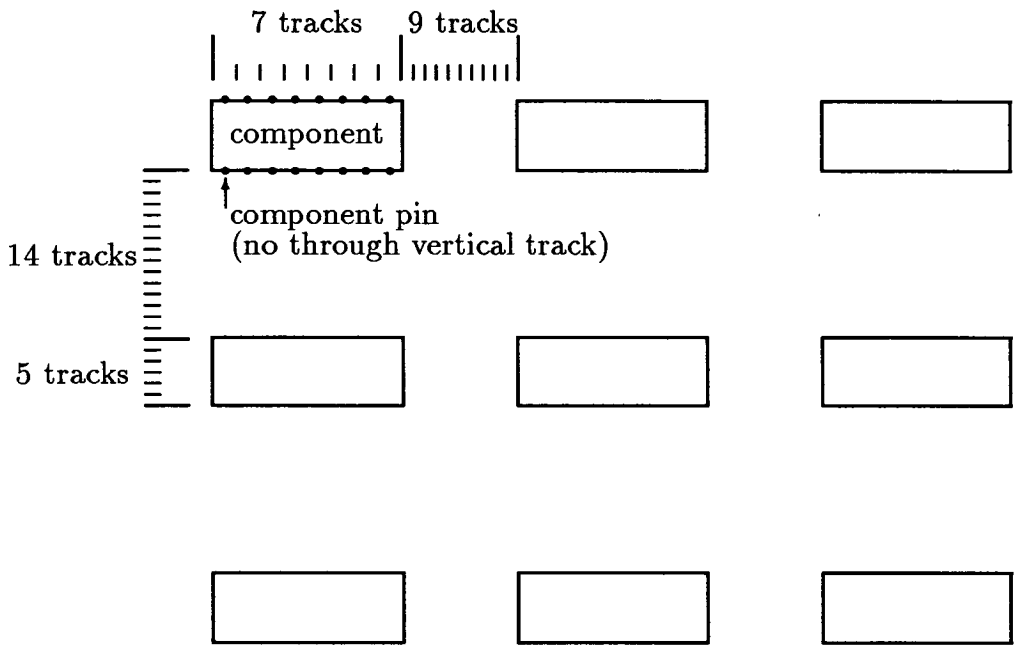


Figure 3–1: *Routing space available on a PCB board*

3.1 Early Work

In 1971 Hashimoto and Stevens published a paper entitled *"Wire Routing by optimizing Channel Assignment within Large Apertures"* [Hash71]. The paper introduced a new wire routing method for two-layer printed circuit boards. The problem was how to connect the components on the circuit boards. The solution involved viewing the problem as assigning the wire routing to horizontal and vertical channels which existed in the space between and under the components. The available space was broken down into channels, and each channel into tracks, and these tracks could have wire routing placed in them. A typical board layout is shown in Figure 3–1.

The algorithm comprised two stages; space assignment and channel assignment. The first - space assignment - involved breaking each connection into a

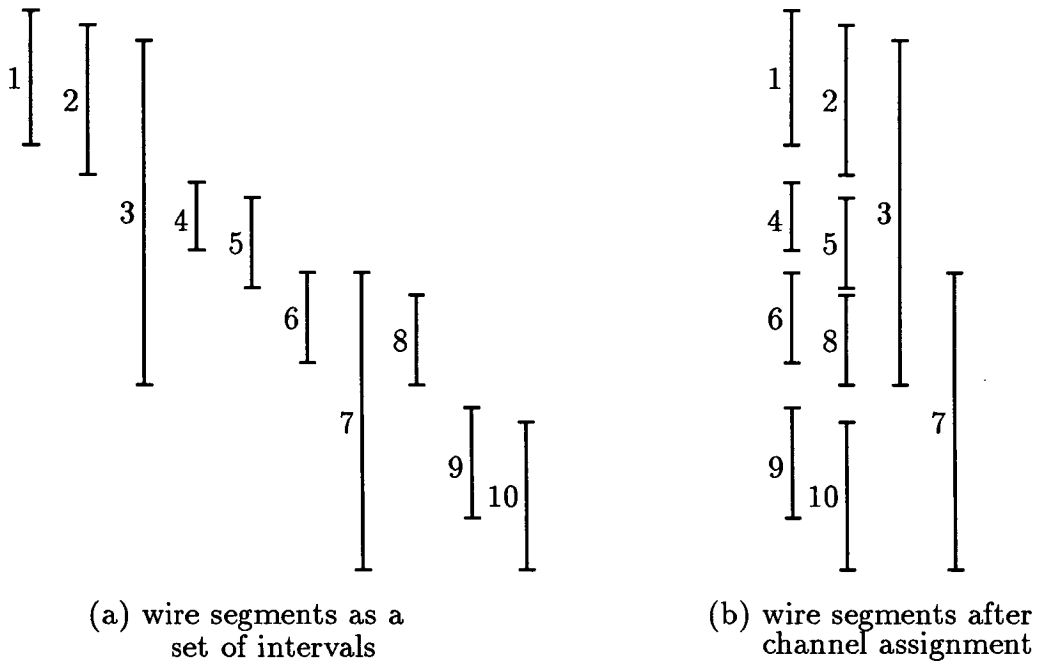


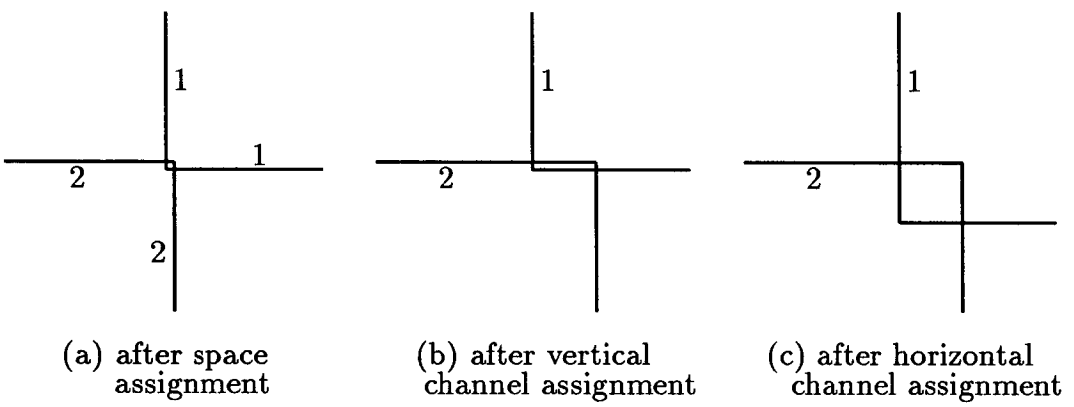
Figure 3–2: *Channel assignment of wire segments*

set of horizontal and vertical wire segments and assigning these segments to the channels on the board. In this initial assignment all the wire segments were assumed to occupy the centre of their assigned channels. This created the problem of overlapping wires that needed to be resolved.

The second stage of the algorithm - channel assignment - attempted to solve this problem. On completion of the space assignment, each channel contained a set of wire segments. These wire segments could be regarded as a set of intervals as shown in Figure 3–2(a). The object of channel assignment was to position all the wire segments in the least number of tracks without any of the wire segments overlapping. The first step in the procedure was to search the list of intervals for the element which had the greatest upper bound. This element was assigned to the first track and eliminated from the list. The list was then searched for the interval which had the greatest upper bound which was *less* than the lower bound of the previously chosen interval. This element was also assigned to the first track and then eliminated from the list. The search was repeated until no

element fulfilled the requirements, at which point the entire process was repeated for the next track. When all the intervals had been eliminated from the list, the channel assignment was complete, as shown in Figure 3–2(b). By performing channel assignment this way, it was proved that the algorithm used the minimum possible number of tracks to position the wire segments in a given space.

3.1.1 Analysis of Early Approach



(a) after space assignment (b) after vertical channel assignment (c) after horizontal channel assignment

Figure 3–3: *Example of channel assignment*

After the space assignment has been completed, many wires overlap in the centre of the channels. Channel assignment solves this problem. It is necessary to perform channel assignment for both vertical and horizontal directions. Figure 3–3(a) shows two nets after space assignment. Figure 3–3(b) shows the nets after vertical assignment, with an overlap in the horizontal channel. However, on completion of the horizontal channel assignment the overlap is removed as shown in Figure 3–3(c).

The channel assignment shown in Figure 3–3 succeeds because the end-points of the wire segments are not fixed and can be altered during the channel assignment. However, the wire segments emanating from a component pin have a fixed end-point at the component pin. This can lead to an overlap between wires emanating

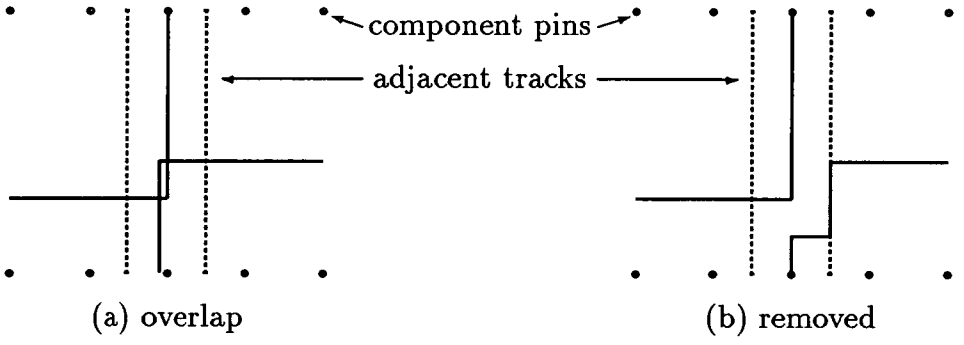


Figure 3–4: *Channel assignment problem*

from component pins that are opposite each other as shown in Figure 3–4(a). This overlap can be removed by local re-routing as shown in Figure 3–4(b).

In 1971, the problem addressed was that of routing components on PCB's. However, the solution can also be directly applied to assembling chip designs. The approach is a very simple, fast stepwise assembler. The stepwise approach is based on the observation that it is not possible to solve the assembly problem optimally for designs consisting of more than a few cells and a few nets [Sher89]. Accordingly, stepwise assembly involves dividing the problem into a number of sub-problems. These sub-problems, due to their comparative simplicity, can be studied, and possibly solved, with far better results than the assembly problem as a whole.

In assembly terms, the Hashimoto and Stevens approach [Hash71] involved taking an initial placement of cells (PCB components) laid out in regular rows and columns, defining channels (spaces between and under components), global routing (space assignment), and then detailed routing (channel assignment). The assembler is illustrated schematically in Figure 3–5.

The result of the assembly operation was either a completely routed PCB, or a partially routed PCB together with an indication of which channels had insufficient tracks to assign all the wire segments. To reduce the chance of failing to route the PCB completely, it was recognised that an even distribution of wire segments

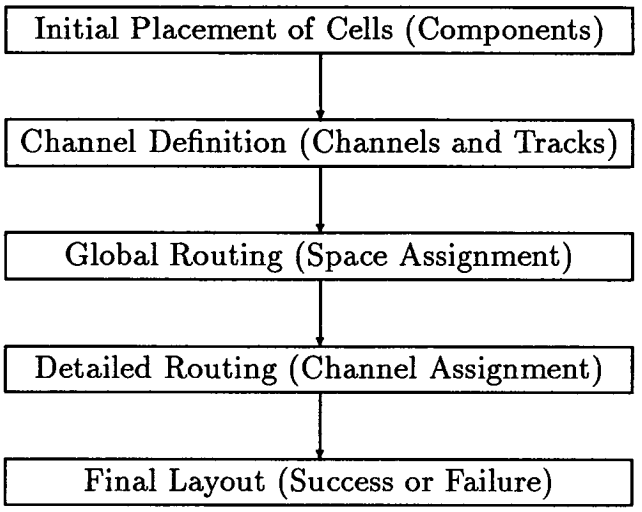


Figure 3–5: *Hashimoto and Stevens's stepwise assembler*

over the horizontal and vertical channels was required. In making a connection between components, there were usually many alternative routes that could be chosen. In an attempt to achieve an even distribution of wire segments over the channels, an arbitrary means of selecting an alternative route was used. In the test runs, the maximum number of components was 165. For each component there were 16 vertical tracks and 19 horizontal tracks available. Some boards with up to 103 components and 500 connections were wired completely in 30 seconds CPU time. For boards with up to 135 components, some were wired completely, others could be wired by reassigning a few wire segments to new channels and the remaining boards required extensive re-routing of the overlapping wire segments. It was considered that most boards with over 151 components could not be wired in two layers.

By present assembly standards, the constraints of a fixed, regular placement, arbitrary global wiring and manual feedback upon assembly failure, render the assembler useless. However, in its day, considering the limited computing resources available, this assembler was a milestone in assembler development.

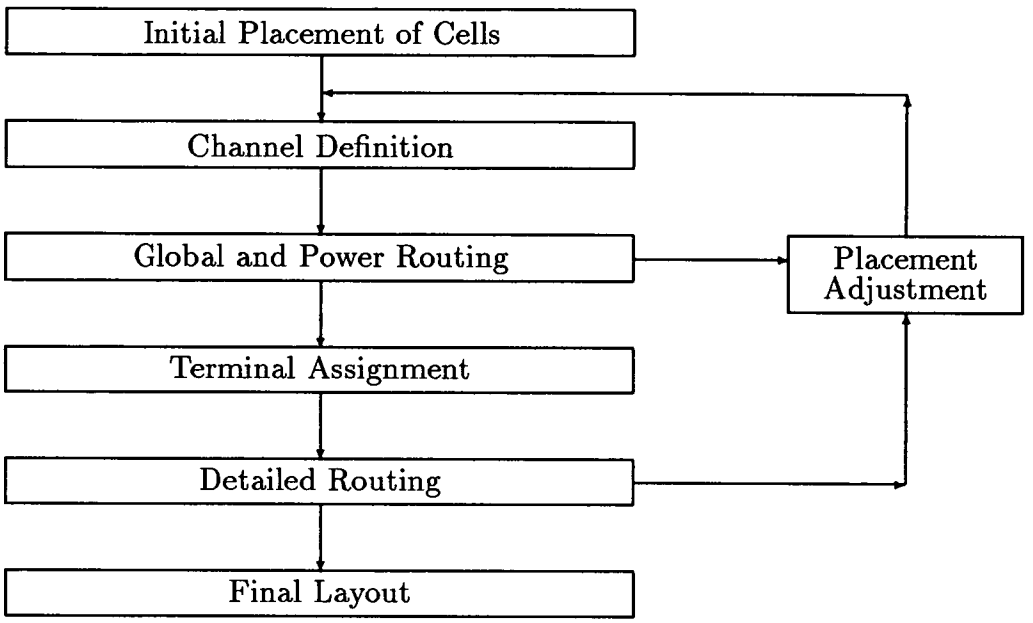


Figure 3–6: *Waring stepwise assembler*

3.2 A Stepwise Assembler

Figure 3–6 illustrates schematically a stepwise assembler that was implemented to gain experience of stepwise assembly. The assembly commenced by defining channels, which in the implementation involved hv-slicing the design and making each tile equivalent to a channel. Global and power routings were then found using the channel structure to hold the routing information. A separate power router was used to find planar routes for the power nets. The minimum channel widths were calculated and the placement was adjusted so that each channel had a width at least as great as the minimum calculated width. The placement was adjusted first horizontally, and then vertically in the subsequent cycle. The GP (global and power routing, placement adjustment) cycle continued until there was enough space for the global and power routes in all the channels. To ensure convergence of the cycle, the final iterations did not allow the placement adjustment to reduce any channel width, thus avoiding the possibility of cyclic oscillation.

Upon convergence of the GP cycle, the space between the cells was divided into channels for detailed routing (in the implementation this involved merging some of the tiles to form larger tiles). Terminals were then assigned along the channel edges with the aim of minimizing the number of wire crossings during detailed routing.

The DR (detailed routing) cycle involved finding a wiring layout that satisfied the design rules, in each of the channels. There was a loop back to the GP cycle from the DR cycle because sometimes there was insufficient space for the wiring layout found by the detailed router. If this happened, the GP cycle was called again to alter the cell placement to increase the space available in those channels which had had insufficient space.

3.2.1 Analysis of Stepwise Assembly

Figure 3–7 shows a chip that was assembled using the stepwise assembler just described. To use design data of a real problem, Harwell Laboratory [Harw90] provided a floorplan of a chip for an instrument, formerly implemented on two large boards of TTL (transistor-transistor logic). The design consisted of 339 nets and 152 cells, which the floorplanner had hierarchically partitioned into 9 sub-problems, 2 of which had a further sub-problem. The chip was assembled in about 90 minutes.

The chip in Figure 3–7 is much larger than it need be. It can be seen that the major reason for the large design size is the poor placement of cells at the top hierarchical level. The cell dimensions and connectivity did not yield a compact placement when floorplanned. Another reason for space wastage was the mechanism of ensuring convergence of the cycles in the stepwise assembler.

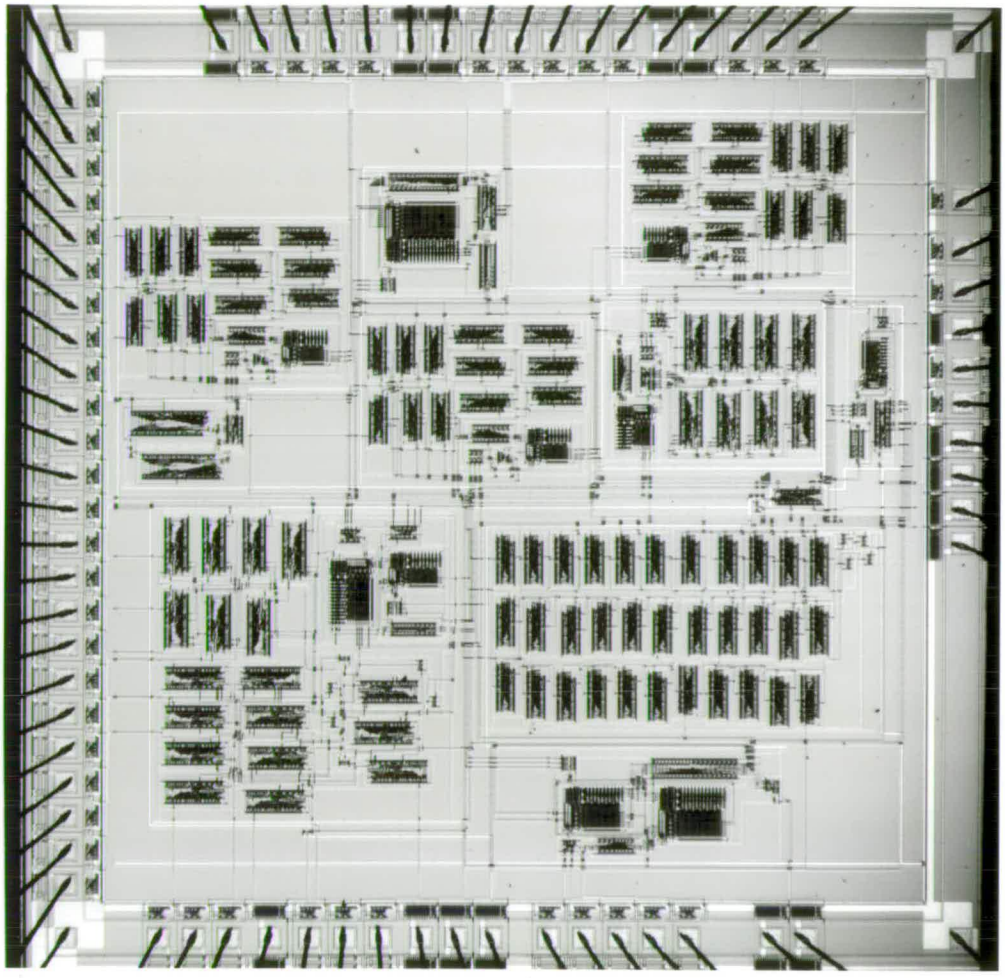


Figure 3–7: *Chip assembled using the stepwise assembler*

GP Cycle

In the GP cycle the assembler adjusts the placement to obtain a match between the channel widths required by the routings and the actual widths of the channel. Adjusting the placement causes problems, however, because usually the channel structure is altered, resulting in channels being created and destroyed. Consequently, the placement adjustment renders the match between the required and actual channel widths meaningless and it is necessary to start again and find new routings for the updated channel structure. Between successive iterations of the

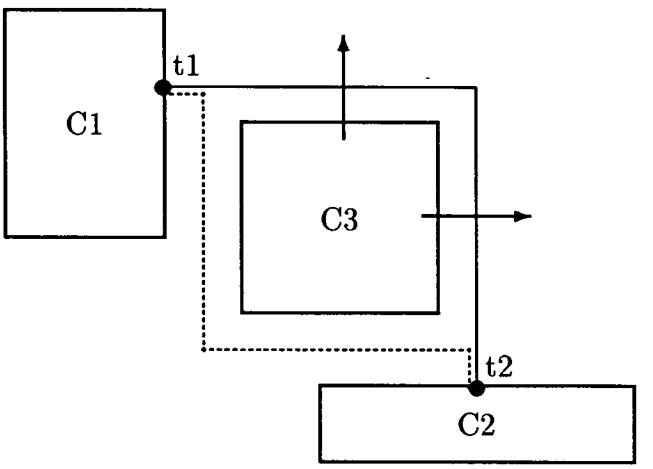


Figure 3–8: *Part of a cell placement where cyclic oscillation could occur*

GP cycle there is a lack of communication because the routings are specified in terms of the channel structure which is then changed by the placement adjustment. New routings have to be found after each placement adjustment and the knowledge of previous routings is lost with the changes in the channel structure. It is this lack of communication between cycles that gives rise to the possibility of cyclic oscillation.

Figure 3–8 illustrates a routing situation where there are two alternative shortest paths for connecting the component pins t1 and t2. The path shown by the solid line is preferable because it has fewer turns than the path shown by the dotted line. It is preferable to have fewer turns because there will be less vias when the design is routed in detail.

In Figure 3–8 cyclic oscillation can occur if during subsequent cycles, cell C3 moves right relative to C2 or up relative to C1. If this happens the path shown by the dotted line will be selected. Routing t1 and t2 around the other side of C3 may cause further placement adjustments. These further adjustments may cause C3 to revert to its original position with the consequence that the path shown by the solid line is selected once again (for C3 to be able to revert to its original

position, the space that was added between C1, C2 and C3 must be removed). At this point in the GP cycle it is not known whether cyclic oscillation is occurring. The fact that C3 has returned to its original position may be because of other placement adjustments rather than because of cyclic oscillation. Unless a copy is made of previous iterations of the GP cycle it is not possible to tell whether cyclic oscillation is occurring. (If copies are kept then if two copies are identical cyclic oscillation is occurring. However, keeping and comparing copies is expensive in both computer space and time.)

The possibility of cyclic oscillation can be prevented, and hence the GP cycle can be guaranteed to converge, by never decreasing the size of any channel during placement adjustment. Ideally, this channel size constraint should be added to the GP cycle after several iterations without this constraint. However, each iteration is computationally expensive and the GP cycle can be called again by the DR cycle. To control the time taken to assemble the design, it was necessary to apply this constraint on every iteration of the GP cycle. Imposing this size constraint usually results in excess space within the design.

DR Cycle

An example of where there can be insufficient space for a wiring layout found by the detailed router is shown in Figure 3–9. In this example, the number of wires crossing the space is two, yet three tracks are required by the router, due to the cyclic constraint. If there is insufficient space for the detailed routing, it is necessary to adjust the cell placement, which results in the GP cycle being called again. Further, if the channel structure is changed, the detailed routing problem that caused the need for the placement adjustment in the first place may no longer exist. Instead, a new detailed routing problem with insufficient space may have been created, requiring further placement adjustments.

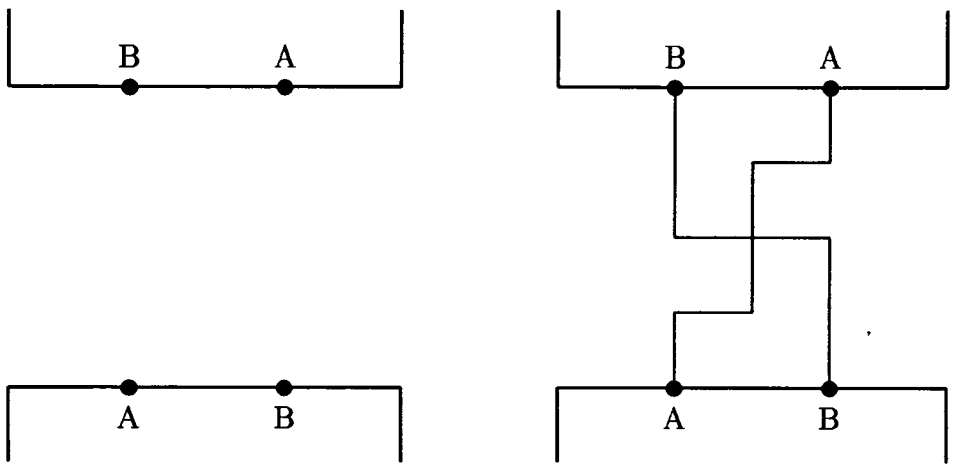


Figure 3–9: *A cyclic detailed routing constraint and solution*

Another problem with this assembler is that during the GP cycle the space required during the DR cycle is estimated. It can only be an estimate because it is not until the detailed routing is performed that the minimum space requirement is known. It requires only one more channel to need more space than is available for the GP cycle to be called again. The chance of a channel requiring more space is dependent on how good the detailed router is, what the channel densities are, how many cyclic constraints are present, and what the size of the channel is.

The probability of *at least* one channel requiring more space

$$= 1 - \text{(probability of zero channels requiring more space)}$$

$$= 1 - \text{(probability of one channel not requiring more space)}^{no\ of\ channels}$$

Figure 3–10 shows the probability of *at least* one channel requiring more space as a function of the number of channels and the probability of one channel requiring more space. Even when the probability of one channel requiring more space is very small, say 1 in 10,000, when considered for a large design of 500 channels the probability of at least one channel requiring more space is 4.9%. When the probability of one channel requiring more space is higher, say 1 in 1,000, the probability of at least one channel requiring more space goes up to 39.4%. If the

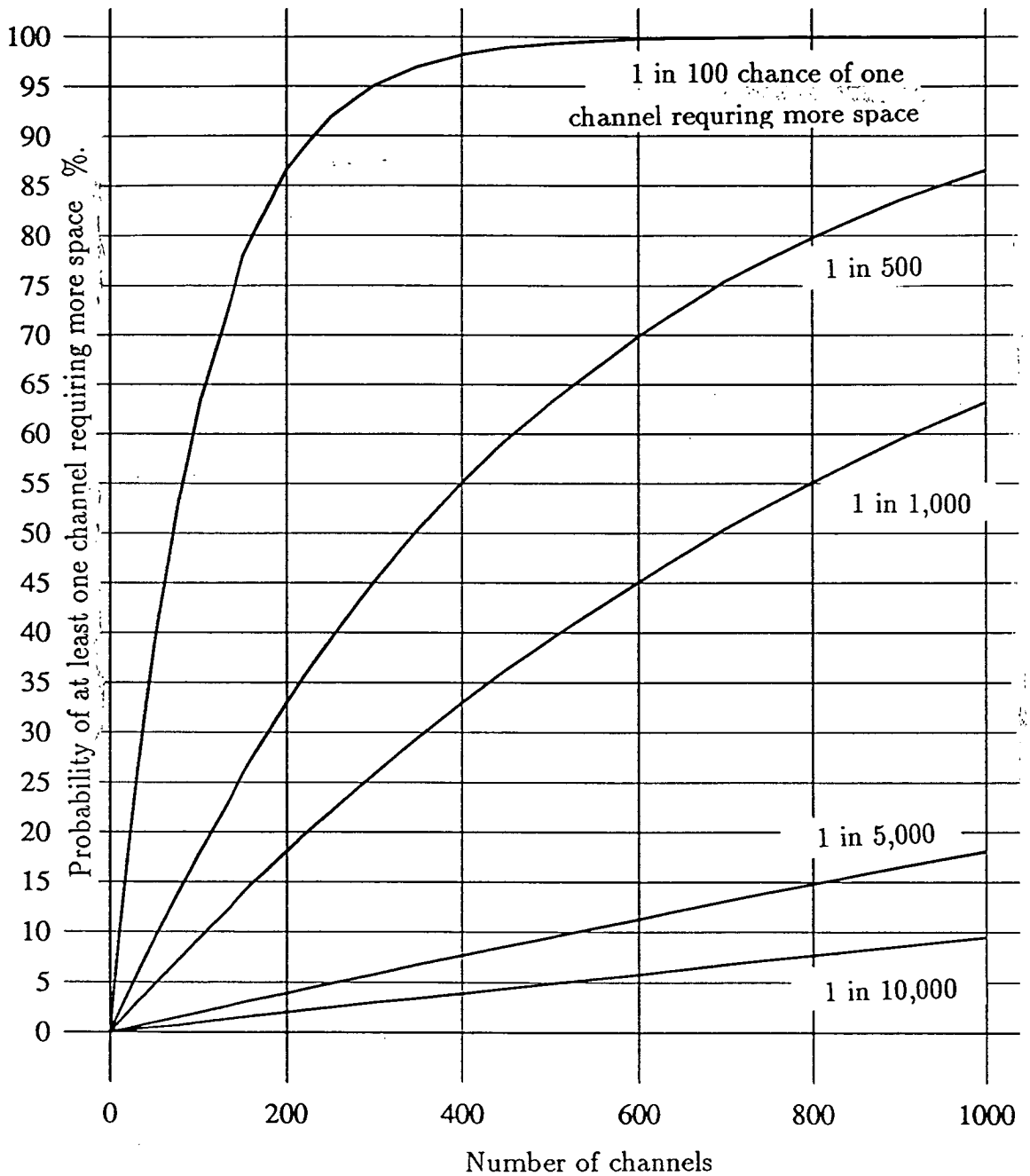


Figure 3–10: *Graph showing probabilities of failure during detailed routing*

probability of one channel requiring more space is even higher, say 1 in 100, the probability of at least one channel requiring more space is then 99.3%.

By using more conservative estimates of spacing requirements for the DR cycle during the GP cycle, the probability of a channel requiring more space is reduced. However, a large amount of space is wasted by using conservative estimates of spacing requirements, since for most of the channels this will be an over-estimate. Over-estimating occurs because the space required by the DR cycle has to be estimated during the GP cycle and it is not known until the DR cycle which channels require to have conservative space estimates. If many iterations of the DR cycle are to be avoided *all* the channels must have conservative estimates. In imposing this constraint, space will be wasted in most channels where a less conservative space estimate would have sufficed.

Summary

The stepwise assembler suffers from a lack of communication both within cycles and between cycles. Within the GP cycle, the global and power routings are specified in terms of the channel structure. Placement adjustments alter the channel structure, and so it is necessary to call the global and power routers again. All the information of previous global and power routings is lost when the channel structure is changed, and hence cyclic oscillation can occur. Whether cyclic oscillation is occurring or not, since each iteration of the GP cycle is computationally expensive, it is necessary to ensure that the GP cycle converges after a few iterations. Convergence is ensured by imposing the constraint of never decreasing the channel size. The effect of imposing this constraint is to trade-off the time taken for the cycle to converge against the redundant space added to the design. A further trade-off between the time taken and redundant space occurs in the DR cycle. If many iterations of the DR cycle are to be avoided then conservative estimates of the space required by the DR cycle must be made during the GP cycle. These two

time-space trade-offs are significant and a large amount of space can be wasted in assembling a design within a reasonable amount of time.

3.3 Other Stepwise Assemblers

3.3.1 PI

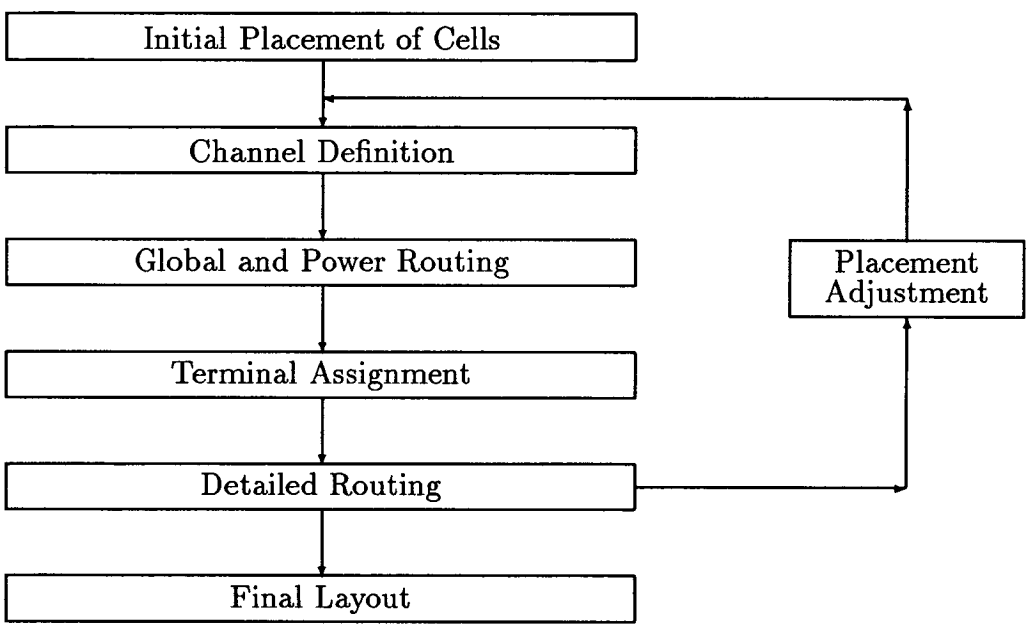


Figure 3–11: *PI stepwise assembler*

From 1981 to 1988 the PI system for custom VLSI placement and routing was being developed by Rivest and associates at MIT [Sher89]. Central to the development of the assembler was the methodology of the stepwise approach of breaking the assembly problem into sub-problems. Different groups could then implement the different sub-problems and if, in the future, a better algorithm for a sub-problem were to be found, this could easily be slotted in to the assembler.

Figure 3–11 illustrates schematically the structure of the PI assembler. The PI assembler is very similar in structure to the Waring stepwise assembler described

earlier in the chapter. The only significant difference is that there is no local GP cycle in the PI assembler. Instead, the placement part of the PI assembler computes an absolute position for each cell which remains fixed during the subsequent channel definition, global and power routing, terminal assignment and detailed routing. In computing the absolute placement PI attempts to leave enough room for the detailed routing by using routing area estimates. For detailed routing, instead of using only one channel routing algorithm, PI used a set of channel routing algorithms. Initially, a trivial routing algorithm was used, which ran each net independently and then checked for design rule errors. If that failed, more powerful routers were called. If they all failed, the placement was adjusted and the assembly process was started anew.

The results quoted for PI [Sher89] include a 20 cell, 9 net design assembled in less than 10 minutes and a 139 cell, 580 net design assembled in around 90 minutes. However, the 139 cell design was highly structured with the cells being abutted together to form strips of cells, and consequently there were fewer than 30 channels to be routed. In general, it was stated that the assembly time was usually dominated by the detailed routing.

Comment

The goal of the assembler was to provide a fully automatic layout system for custom VLSI taking as input a few dozen modules and a few hundred nets and producing an assembled design within two hours. For an assembler project being undertaken in the late 1980's the results are far from impressive. The fact that the assembly time is usually dominated by the detailed routing agrees with the experience of assembling the Harwell chip design [Harw90]. There have been and will be no significant results from PI, not because of the failure of the individual parts of the assembler; rather because of failure in tackling the cyclic problems inherent in the stepwise approach.

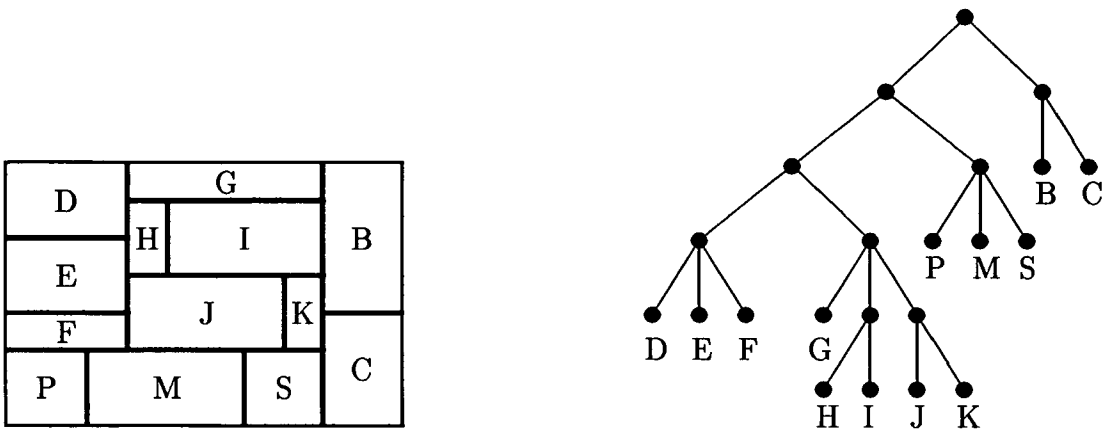


Figure 3–12: *Cell placement and corresponding slicing structure*

3.3.2 Lauther's Work

Lauther uses a min-cut floorplanner [Laut79] to find an initial cell placement. The min-cut floorplanner always produces placements that are in the form of a *slicing structure* [Szep80]. An example of a slicing structure is shown in Figure 3–12. If the placement is in the form of a slicing structure, then it is possible to partition the design using slicing operations. A *slicing operation* partitions a large rectangle into smaller rectangles using parallel lines. The same operation can be applied to the resulting rectangles - slices - but with lines perpendicular to the first set of dividing lines. Slicing can be repeated to any depth, alternating the orientation of the dividing lines. In the context of Silicon Assembly a slicing operation partitions a group of cells into smaller groups of cells. The advantage of the initial cell placement being in the form of a slicing structure is that each slice represents a channel and the order in which the slices are found is the reverse of the routing order. Using the routing order, each channel routing problem can be solved *independently.*

Figure 3–13 illustrates schematically the assembler of Lauther [Laut85]. After the initial cell placement has been obtained using the min-cut floorplanner, the placement is global and power routed with the routings being specified as sequences

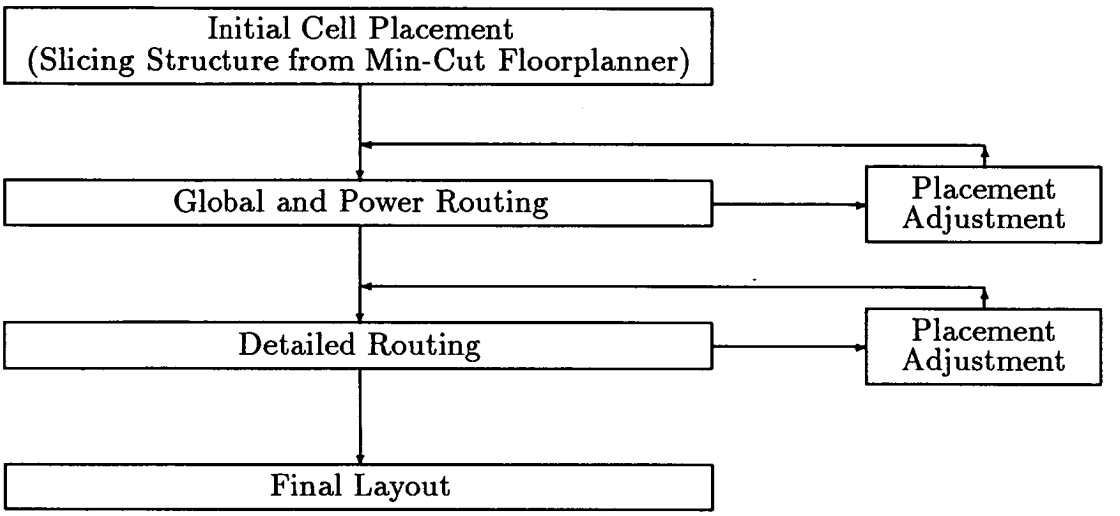


Figure 3–13: *Assembler of Lauther*

of channels. During global routing, the placement is adjusted, as each net is routed, so that there is enough space between the cells for the global nets. It is possible to adjust the placement and not suffer from the problem of cyclic constraints, because space is only ever being added to the design.

After the routings have been specified, the detailed routing is performed. During detailed routing, each time a channel is routed, the channel width is altered (the placement is adjusted) to accommodate the channel wiring produced by the detailed router, and the cells adjacent to the channel are merged with the channel to form a composite cell. To maximize the availability of routing space in the subsequent channels, the composite cells are formed with ragged borders, which enclose the channel wiring in as small an area as possible. An example composite cell with a ragged border is shown in Figure 3–14. It is necessary to adjust the placement before the next channel is routed because although a channel routing problem can be solved independently of the other routing problems, the solution of the routing problem forms part of the specification of subsequent routing problems in the slicing tree hierarchy.

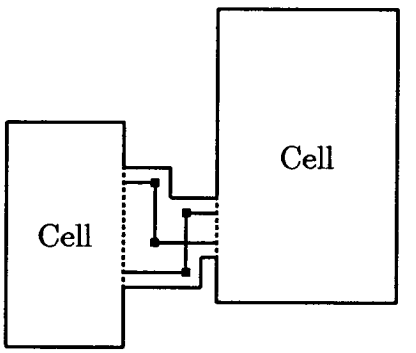


Figure 3–14: *A composite cell (with ragged border)*

Lauther quotes a result of an assembly time of 17 seconds on a 4 MIPS mainframe for a design containing 24 cells, 12 pads and 55 nets.

Comment

In assembly terms the effect of imposing a slicing structure on the cell placement is to localize the DR cycle from the GP cycle. Using the Waring stepwise assembler, a placement adjustment during the DR cycle caused the GP cycle to be called again, because the placement adjustment could change the channel structure. Lauther's assembler does not need to call the GP cycle again, because it ensures that the channel structure can be preserved whilst the placement is adjusted during detailed routing. It is possible to be certain to be able to preserve the channel structure because the placement is in the form of a slicing structure.

Lauther's assembler can assemble any cell placement that has a slicing structure. However, there are many cell placements that do not have a slicing structure. The simple cell placement shown in figure 3–15 has a non-slicing structure. An edge in the graph in figure 3–15 represents a channel (which is the space between 2 cells), while a vertex in the graph represents the intersection of 2, 3 or 4 channels.

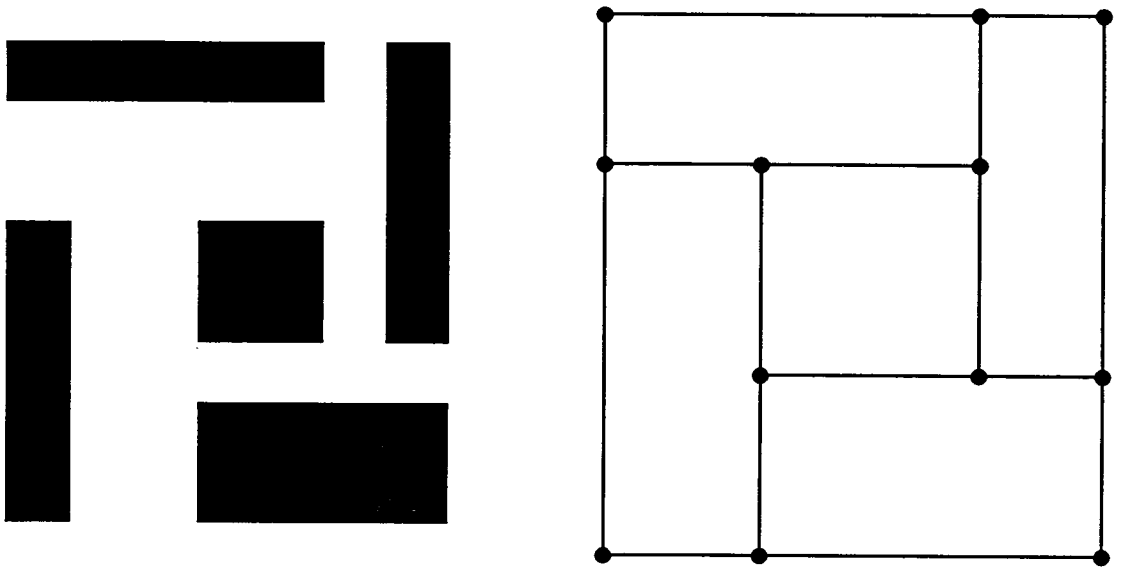


Figure 3–15: *Cell placement and corresponding channel-graph*

It is possible to adjust any cell placement so that it has a slicing structure, but in imposing this constraint a lot of space can be added to the design.

The major advantage of assembling slicing structure placements is that there are no convergence-cycle problems. Although there is a DR cycle involving routing a channel and then adjusting the placement, termination of the cycle always occurs when the last channel is routed and the placement is adjusted. The detailed routing of each channel is performed only once and so is much faster than in non-slicing structure stepwise assembly. However, the disadvantage is that detailed routing is achieved in one pass by adjusting the placement so that the channel slicing structure is maintained and space can be wasted in maintaining the structure particularly when routing the final few channels where the groups of cells are larger and the channel sides may be of unequal lengths.

Chapter 4

Towards Integrated Silicon Assembly

This chapter gives details of several significant advances in assembler frameworks. Two different approaches to tackling the problems of the GP (global routing and placement adjustment) cycle are described. Details are also given of an approach that attempts to localize the DR (detailed routing) cycle for any given placement. Two assemblers that incorporate these advances in assembler frameworks are described and analysed.

4.1 Tackling the GP Cycle Convergence Problem

4.1.1 Dynamic Layout Spacing

In stepwise assembly, the aim of the GP cycle is to obtain a match between the wiring capacities of the space between the cells and the wiring routes found by the routers. The major problem is the lack of communication between successive cycles.

Work at Berkeley [Dai87] has identified the GP cycle convergence problem as being caused by the separation of the topological (global routing) and the geometrical (placement adjustment) operations. Their *dynamic layout spacing* system was the first to unify the two operations by dynamically updating the global routing as the placement was adjusted. This meant that between cycles the global routing was retained. The advantage of unifying the operations was that there was far more communication between cycles and consequently convergence could be obtained much more easily.

The work at Berkeley unified the topological and geometrical operations by using a dynamic layout representation of the problem. The representation uses circle graphs, chords and the operations of circle partition and unification. A circle graph is formed by the edges of cells and tile edges, an example of which is shown by the dashed lines in Figure 4–1(a). The notion of circle graphs is introduced in the paper [Dai87], and examples of circle graphs are shown. However, a precise definition of a circle graph is not given, and it is unclear how the circle graphs are derived. Within the circle graph framework, chords, which cross the circles, are used to represent the global routing. Figure 4–1(b) shows how a global net is represented using chords within circles, together with terminals around the circle

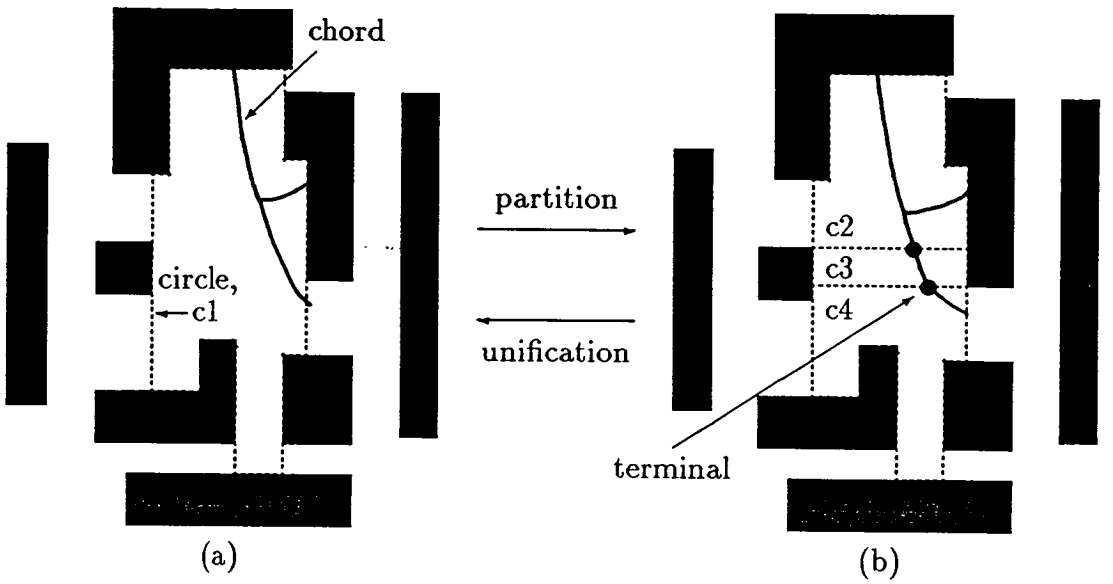


Figure 4–1: *Circle partition and unification*

perimeters. For each circle perimeter, the set of terminals forms a record of the net crossings.

In the GP cycle, when the channel structure is changed by placement adjustment, channels are created and destroyed. To reflect the changes in the channel structure, the representation uses the operation of circle partition when channels are created, and the operation of circle unification when channels are destroyed. The circle operations necessitate changes to the chords which represent the global routing. By finding equivalent chords before and after the circle operations, it is possible to update the global routing dynamically. An example of circle partition and unification is shown in Figure 4–1. If the circle shown in Figure 4–1(a) is partitioned, then chords have to be inserted as shown in Figure 4–1(b). On the other hand, if the circles shown in Figure 4–1(b) are unified, then chords have to be deleted as shown in Figure 4–1(a).

Using the dynamic layout representation, a dynamic layout spacing system can be created which enables the updating of the global routing as the placement is

adjusted. The spacing system can be used to adjust the placement to obtain a good match between the wiring capacities of the spaces between the cells and the number of global wires passing through the spaces. In obtaining this match, the stepwise approach suffered from a cycle convergence problem due to poor communication between the global routing and the placement adjustment. However, there is no cycle convergence problem if the dynamic layout spacing system is used.

Comment

The dynamic layout spacing system represents an advance in assembler frameworks, because the cycle convergence problem that occurs in stepwise assemblers no longer exists. To reflect changes in the channel structure and the global routing as the placement is adjusted, the spacing system performs the operations of circle partition and unification and modifies the chord structure accordingly. These circle operations can be complex, reflecting the fact that minor placement adjustments can cause significant changes in the channel structure. What the circle operations and chord modifications are doing is updating the global routing in response to changes in the placement. However, because the routing is specified with respect to the channel structure and the channel structure can be significantly changed when the placement is adjusted, the whole process of dynamic layout spacing is more complicated than it need be. This raises the question as to why is the global routing specified in terms of the channel structure? The only justification seems to be that, complicated though specifying the routing this way is, it works!

4.1.2 Simulated Annealing

An alternative approach to tackling the GP cycle convergence problem involves the use of simulated annealing. Simulated annealing [Vecc83] provides a general framework for dealing with large-scale optimization problems, and is intended for

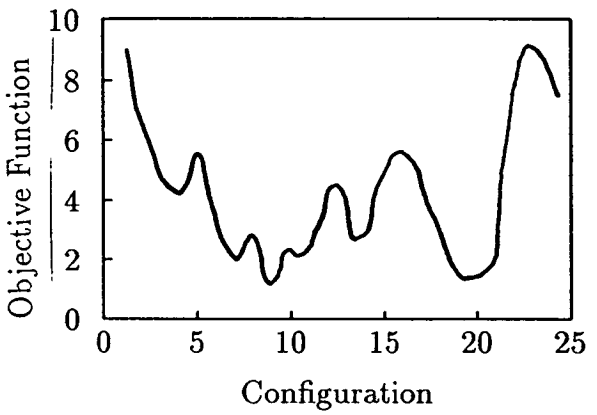


Figure 4–2: *Objective function for multivariate system with conflicting goals*

Fluid	*Optimization Problem*
internal energy	objective function
atomic positions	parameters
cool into stable, low energy state	find a near optimal configuration

Table 4–1: *Simulated annealing identifications*

problems with very many degrees of freedom and an objective function which combines conflicting goals. This objective function may have many goals as illustrated in Figure 4–2. The simulated annealing approach consists of a series of identifications between the many parameter problem and a hypothetical fluid consisting of many interacting atoms as listed in Table 4–1.

To bring a fluid into a low energy state (as, for example, in growing large single crystals), the most effective procedure is careful annealing. This involves melting the fluid and lowering the temperature slowly, spending a long time at temperatures near the freezing point to allow defects to anneal out of the growing crystals, and then cooling the crystals more rapidly to bring the atoms to rest. The same sequence can be followed in optimization by introducing a pseudo-

temperature, which is just a control parameter in the same units as the objective function. Several techniques exist for the computer simulation of the motion of the atoms of a fluid in equilibrium at a given temperature. The Metropolis Monte Carlo method [Metr53], [Kirk83] is particularly easy to extend to the optimization context. The method involves a probabilistic algorithm in which one atom is moved at each step. The new atomic configuration is accepted with probability one if its energy, E, is less than before, and with probability $exp(-\Delta E/T)$ for temperature, T, if the energy is greater than before.

It is the ability of being sometimes able to accept a new configuration with greater energy than before that is a fundamental feature of the algorithm that enables further exploration of the configuration space. In Figure 4–2, acceptance of a higher energy configuration is equivalent to a 'hill climbing' move which takes the objective function away from a local minimum and may result in a new minimum being found. As $T \rightarrow 0$, the probability of accepting an energy greater than before decreases, effectively 'freezing' the configuration, which simulates the end of the annealing process.

Sechen [Sech86], [Sech88] uses simulated annealing to find placements that require little adjustment during detailed routing. The annealing based system consists of two stages:

Stage1 involves deriving an initial placement. During this stage a dynamic interconnect-area estimator is used to determine a value for the interconnect area around cells to be used in the placement determination. The estimator is based on three factors:

- ▹ **The average net traffic.** This is an estimate of the average number of interconnections passing through a channel, which is used to derive the expected average channel width.

- ▷ **The position of the channel in the chip.** Sechen states that in cell layouts, the shortest possible route is often used for a net and so it is to be expected that the widths of channels would be greater nearer the centre of the layout.

- ▷ **The relative pin density of a cell edge.** This can be calculated by finding the pin density of a cell edge, which is given by the number of pins on the edge divided by the length of the edge, and dividing this total by the average pin density of the layout, which is given by the total number of pins divided by the total length of the edges.

Stage2 involves placement refinement. A low temperature annealing algorithm accomplishes this step. Instead of using a dynamic interconnect-area estimator, the design has channels defined, is global routed and the placement is refined based on the results of the global routing. It is stated that three iterations of this step are sufficient for the final chip area to converge.

Comment

Sechen's annealing based system achieves the goal of ensuring the convergence of the GP cycle. In Stage1 the global routing is modelled by using the dynamic interconnect-area estimator to assign area values to cells, which can be changed as the placement is adjusted. In Stage2 the global routing is modelled explicitly. From the paper [Sech88] it is not clear how convergence is ensured.

Simulated annealing is just one of many possible Monte Carlo methods that accept perturbations that result in an increase in objective function value. In their paper *"Experiments with Simulated Annealing"* Nahar et al. [Naha85] compared the performance of simulated annealing to that of other Monte Carlo methods for optimization. Their experiments showed that these methods often perform better

than simulated annealing. This is not surprising because although annealing has a sound theoretical basis in the physical domain (all the atoms are alike and the ground state is a regular crystal), no such basis for its application to arbitrary combinatorial problems exists (a typical optimization problem will contain many distinct, non-interchangeable elements, so a regular solution is unlikely).

4.2 Tackling the DR Cycle Convergence Problem

4.2.1 General Slicing Structures

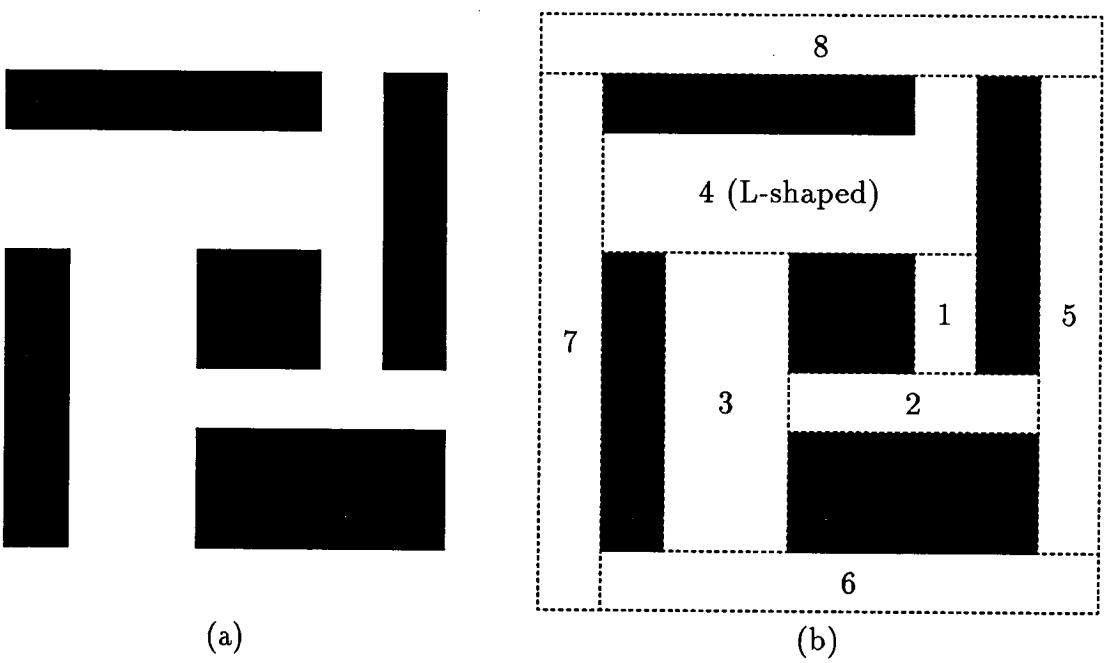


Figure 4–3: *Cell placement and corresponding slicing structure*

In Chapter 3, work was described in which imposing a slicing structure on a placement resulted in the DR cycle being localized. The DR cycle is localized from the GP cycle because the slicing structure of a design yields a channel ordering where each channel width can be adjusted independently of all previously routed channels. However, it was stated that there were many cell placements that did

not have a slicing structure and the cell placement shown in Figure 4–3(a) was given as an example. The problem was that adjusting the cell placement to be in the form of a slicing structure, involved space being added to the design.

Work at Berkeley [Dai85] has shown that by allowing slices with K-bends (K=0,1,2,3...) in them, it is possible to guarantee being able to find a slicing of any cell placement without having to adjust the placement to conform to the slicing constraints. A slicing structure can be found for a design containing cells of any Manhattan shape. It has been shown in [Dai85] that if only rectangular cells are present, an L-shaped, or 1-bend slice is the most complex slice that can result from the slicing algorithm. The cell placement of Figure 4–3(a) does not have a slicing structure if only rectangular, 0-bend, slices are allowed. However, as is shown in Figure 4–3(b) a slicing structure does exist if 0- and 1-bend slices are allowed. Also shown in Figure 4–3(b) is the order for routing the channels during detailed routing which is the reverse of the slicing order.

Comment

Using channels with K-bends removes the need to adjust the placement to find an order for routing the channels during detailed routing. However, for the case of K-bends with K >= 1, the DR cycle can no longer be guaranteed to converge. An example of an unroutable L-shaped channel routing problem, which is taken from a paper by Chen [Chen87], is shown in Figure 4–4(a). No matter how far up and to the right the upper channel boundary is moved, the channel cannot be routed. The independence property of being able to adjust a channel width independently of previously routed channels still holds. With only K=0 channels altering the channel width always ensured successful channel routings. However, with K >= 1, altering the channel width is not sufficient to ensure successful channel routings, because placement adjustments of previously routed channels may cause routing situations as in Figure 4–4(a) to occur. The corner pincer situation shown in

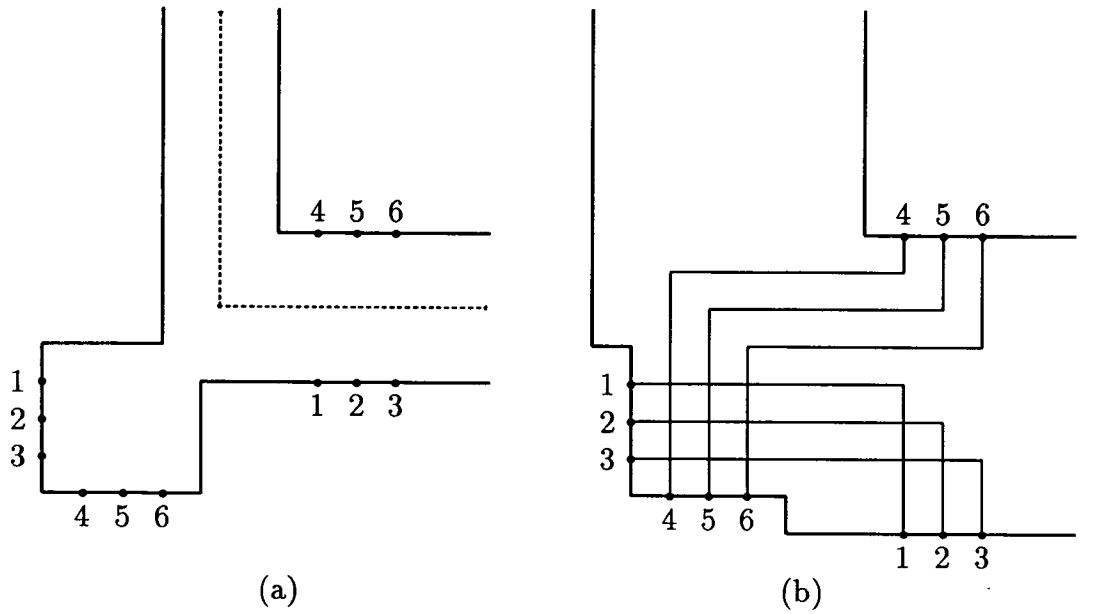


Figure 4–4: *L-shaped channel routing problem*

Figure 4–4(a) illustrates the point that there are other necessary conditions for ensuring the localized convergence of the DR cycle. Chen gives details of rules to ensure the routability of L-shaped channels. The rules place constraints on the channel edges which prevent corner pincers from occurring. If the rules are obeyed then the channel can be routed as shown in Figure 4–4(b).

4.3 Silicon Assembly at Berkeley

4.3.1 Mosaico

The Mosaico assembler [Burn87] at Berkeley is illustrated schematically in Figure 4–5. Every tool in Mosaico inputs and outputs the design in a common format using a data manager to store the design at each stage of the assembly

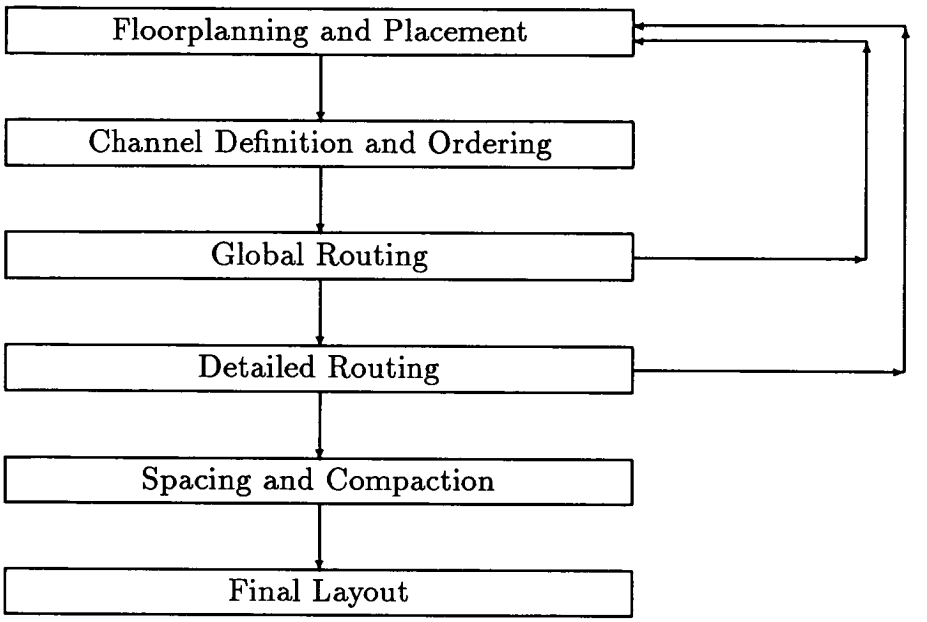


Figure 4–5: *Mosaico assembler at Berkeley*

process. Consequently, it is possible to swap tools in and out of the assembler via this interface.

The floorplanning and placement step is based on the simulated annealing algorithm of Sechen [Sech88] described earlier in the Chapter. A strength of Mosaico is that no constraints are imposed on the layout style used to implement the cells and hence a variety of layout styles can be supported.

The channel definition and ordering step slices the placement and finds the routing order for the channels during detailed routing. The channel definition is able to generate K-bend channels if required. Consequently, there is no need to adjust the placement during channel definition.

The global router is then called followed by the detailed router. The routing of the power and ground nets is performed using the same routers that are used for the other nets, but with additional processing to handle the tapered-width requirements.

The Mosaico assembly is completed by the spacing and compaction step. All steps in Mosaico are carried out at the symbolic-layout level. Consequently, symbolic layout spacing (or compaction) is used to ensure that designs satisfy the design rules. The use of symbolic layout also has the advantage of providing a mechanism for producing technology-independent designs.

The Mosaico assembler shown in Figure 4–5 has two feedback loops from the global and detailed routing to the floorplanning and placement step. The loops are provided to account for situations where there is insufficient space for the global or detailed routing. The number of times that the loops are executed is closely related to the accuracy of the routing area estimation performed at the placement stage. What is unclear is why, if the design is purely symbolic until the final spacing and compaction step, are any feedback loops necessary at all?

4.3.2 Bear

The Bear (Building-block Environment Allocation and Routing) system [Dai89] has a similar assembler structure to that of Mosaico except that dynamic layout spacing [Dai87] (as described earlier in the chapter) is used to integrate the global routing with the floorplanning and placement to produce placements that require very little modification during detailed routing. Following the dynamic layout spacing, channels are defined and ordered with K-bend channels being generated if required and the detailed router is then called to route each channel in turn.

4.3.3 Comment

Benchmark results (which are listed in the next Chapter) are given for Mosaico and Bear, with the results obtained by Bear being slightly better than those obtained by Mosaico. The results demonstrate the viability of implementing an assembler based on the framework of simulated annealing or dynamic layout spacing together

with K-bend channel definition. However the drawback of using Mosaico or Bear is that once the detailed routing has commenced, the channel structure is fixed because the channels and routing order are defined before commencing the routing. Detailed routing failure can occur if placement adjustments of previously routed channels make it impossible to route a channel. If a good match is obtained between the wiring capacities of the spaces between the cells and the number of global wires passing through the spaces, then the placement adjustments will usually be minor and so it is very unlikely that routing failure of a channel will occur. It was reported in Mosaico that routing failure had never occurred while assembling any chip design. However, if failure were to occur, then the assembler would have to go back to the floorplanning and placement stage and use a more conservative estimate for the routing area requirements.

Despite advances in tackling the GP cycle and DR cycle convergence problems, Mosaico and Bear are still fundamentally limited by the loop back between the DR cycle and the GP cycle. So long as the GP cycle makes a good estimate of the DR cycle requirements a loop back will not occur. However, if as much space as possible is to be squeezed out of the design, then the GP cycle must not make conservative estimates of DR spacing requirements or space will be wasted. The less conservative the GP cycle estimates, the more likely the possibility of looping back following DR failure.

It is the separation of the global and detailed routing that causes the loop to exist. Why not integrate the global and detailed routing?

4.4 Summary

Two questions involving assembler frameworks have been posed in this Chapter:

- ▷ Why is the global routing specified in terms of the channel structure?

- ▷ Why not integrate the global and detailed routing?

These questions are beacons of the communication problems still present in assembler frameworks. It is not necessary to specify the global routing in terms of the channel structure and it is possible to integrate the global and detailed routing. The next Chapter states the case for integrating the global and detailed routing, and for integrating the routings with the placement adjustment. Details of an integrated assembler framework that overcomes the communication problems present in the Berkeley assemblers are given. The benchmark results obtained by the integrated assembler are better than those obtained by Mosaico and Bear.

Chapter 5

An Integrated Silicon Assembler

This chapter presents the case for an integrated silicon assembler framework. The implementation of an integrated silicon assembler called Scotia is described. Assembly results are given for two well known benchmarks, which compare favourably with published results. A comparison is made between Scotia and a stepwise assembler by using the two assemblers to assemble the same chip design. Comparing the results, it is clear that the integrated approach of Scotia is superior to this particular stepwise approach to silicon assembly.

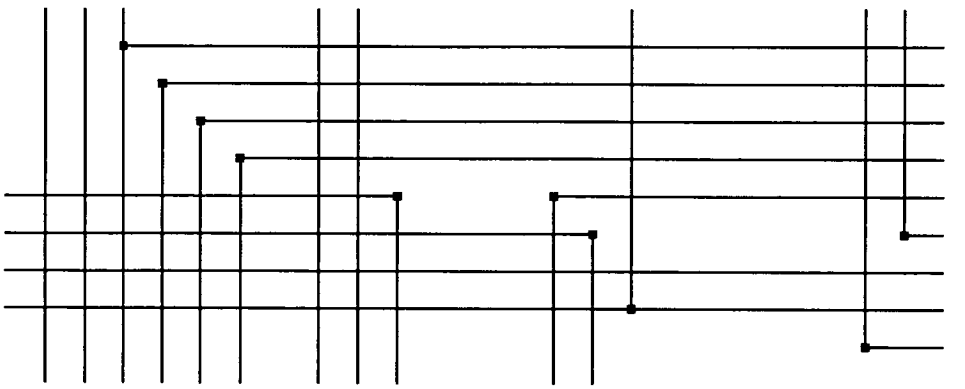


Figure 5–1: *Example of detailed routing in an assembler*

5.1 The Case for Integrated Silicon Assembly

5.1.1 Integrating the Global and Detailed Routing

There is a strong case for integrating the global routing and the detailed routing. During the global routing the placement is adjusted to accommodate the subsequent detailed routing. The goal of terminal assignment is to order the wires so that the number of crossings is minimized. Experience of detailed routing during assembly shows that the problem is simple with wires for most, if not all nets, forming straight or T-junctions or corner connections within a channel as shown in Figure 5–1.

During assembly the spacing of the global routing together with the terminal assignment is almost equivalent to the detailed routing for most of the nets. There would be very significant savings in computing time if the global wiring with terminal assignment were to be directly equivalent to the detailed routing because:

- ▷ There is a time-cost in calling the detailed router. The more powerful routers have to build data structures to allow the possible routings to be considered.

Often, as mentioned previously, the detailed routing problem is simple and so the full power of the router is seldom required. Nevertheless the data structures still have to be built to enable the router to function. Recognising this, PI [Sher89] uses a set of channel routing algorithms, starting with a trivial one that in essence runs wires for each net independently and then checks for design rule errors. If one fails, the next in line, being more powerful but slower, is called. The purpose of using a set of routers is to reduce the overall time taken to route a design in detail.

▷ The loop back between the DR cycle and the GP cycle would be eliminated. In effect, the two cycles would be merged into one. For a design with n channels that requires m iterations to assemble it, there is a time saving of up to $m * n$ detailed routings[1].

The problem is that if the global wiring is to realise the detailed wiring then the placement may have to be adjusted.

5.1.2 Integrating the Placement Adjustment

The dynamic layout spacing system [Dai87] had sometimes to perform complex circle operations due to placement adjustments causing changes in the channel structure and it was not clear why the global routing had to be specified in terms of the channel structure. An alternative approach, that enables the dynamic updating of the routings as the placement is adjusted, involves specifying the routing as a set of wire segments that lie in the fluid space between the cells.

---

[1]As the placement is adjusted during assembly the number of channels will vary slightly.

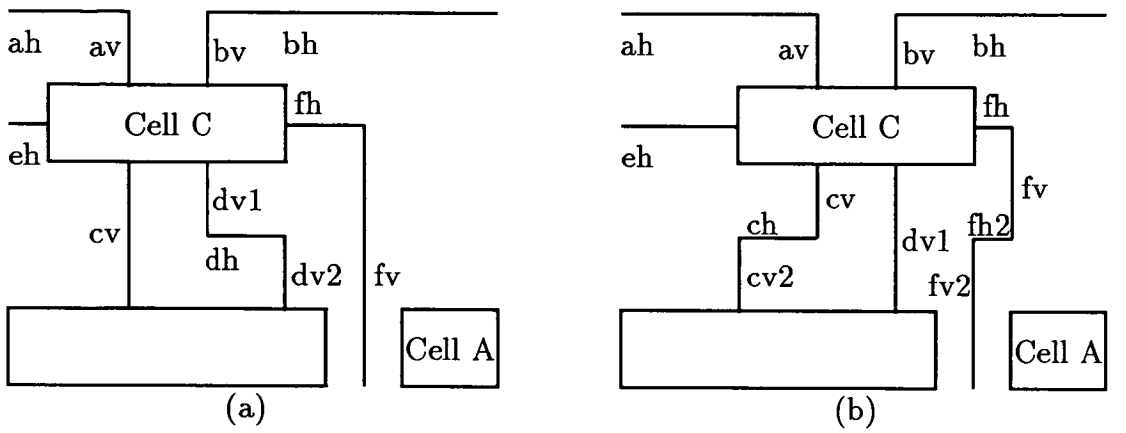


Figure 5–2: *Wire segment adjustments caused by a cell placement adjustment*

Figure 5–2 shows an example of the operations that are performed on the wire segments when the cells are moved during placement adjustment.

Figure 5–2(a) shows part of a cell placement with attached wire segments and figure 5–2(b) shows the cell placement and wire segments after cell C has been moved right. Wire segments av, bv, cv and dv1 are moved to the right with cell C as they are attached perpendicular to the the direction of cell movement. To maintain the connectivity of the wires, segment ah is expanded; bh is contracted; cv, because its bottom end is connected to another cell, has segments added leaving cv, ch and cv2; dh is contracted to zero length and so dv1 and dv2 are merged leaving dv1; eh is expanded; fh is contracted to within the design rule separation of fv and the cell edge, fh and fv are then moved right the rest of the way with cell C, and fv has segments fh2 and fv2 added because its bottom part cannot move right as it is blocked by cell A.

The wire segment framework goes further than the other integrated frameworks because, by assigning widths and separations to the segments, a layout of segments that satisfies the spacing constraints imposed by the widths plus separations can be mapped directly into detailed routing. This framework integrates the detailed routing with the placement adjustment and the global routing. Even

the tapered-width power routing can be represented by assigning different widths to the segments that represent the power busses within the design.

5.2 Overview of Scotia Integrated Assembler

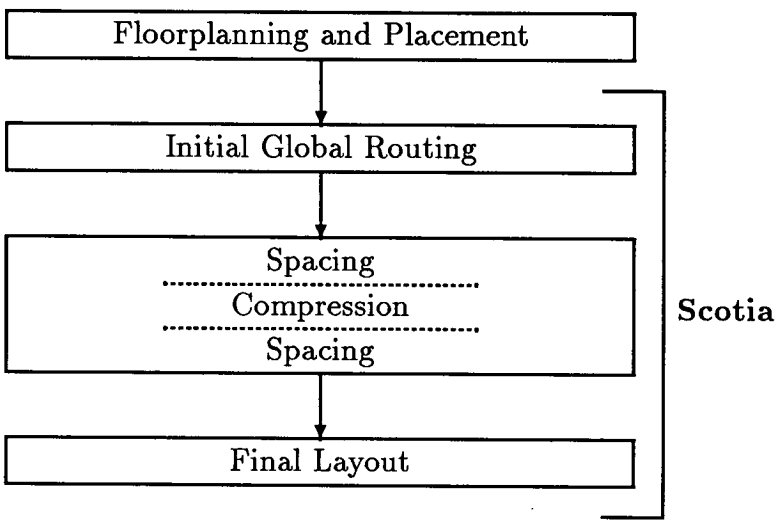


Figure 5–3: *Scotia integrated silicon assembler*

Figure 5–3 illustrates schematically the Scotia integrated assembler. As was discussed in chapter 2, floorplanning and placement is the first step in the assembly process. However, this step has not been implemented in Scotia. An initial placement together with the global routing specification forms the present starting point for the Scotia assembler, which is described in this chapter. The Scotia implementation is of a 2-routing-layer integrated assembler that produces as output a layout which satisfies the design rules. In the next chapter, the integration of floorplanning and placement into Scotia is discussed, together with a discussion of the feasibility of implementing an n-routing-layer integrated assembler.

5.3 Initial Global Routing

Starting with a given placement, Scotia finds global routes for all the nets. In chapter 2 the goals of global routing were detailed as:

- ▷ Finding the shortest connection between the pins of each net (Steiner trees).
- ▷ Making efficient use of feedthrough possibilities.
- ▷ Finding good pin orderings - if the pins are considered one at a time, the length of the global route is particularly sensitive to the order in which the pins are connected together.
- ▷ Coping with additional power routing requirements - the width of the metal wire is dependent on the current carried, hence the need for tapered-width wire routings.

The global router was implemented with the intention of achieving these goals. Scotia currently finds global routes which lie within the fluid space between cells. This restriction is due to Scotia being a 2-routing-layer implementation. The global router could be modified to include over-the-cell routing.

The global routing algorithm is based on the admissible A^* algorithm [Nils71], adapted for global routing by Clow [Clow84]. In the context of searching, an admissible algorithm is one which has the property of always finding a minimal cost path between two points when such a path exists.

5.3.1 Significance of the A^* algorithm

If there were no obstacles, then it would be relatively easy to find a Manhattan path to connect any two pins together. However, because the cells form obstacles

which have to be routed around, the connection problem becomes more difficult and a search algorithm is required to find the least-cost path to connect any two pins together.

During the search, a series of nodes that form the frontier of the search is generated, and there is a choice as to which of these nodes the search is continued from. The A* algorithm is a best-first type of algorithm (also known as branch-and-bound). A best-first algorithm relies on information generated by the route so far to predict which nodes are the most likely to be on a minimal cost path [Clow84]. The ideal algorithm would operate on *perfect* information, thereby always choosing the correct node to continue from, at each stage of the search. However, perfect information implies the path is already known, so there is little reason to search.

Very significant savings in computational time and memory space can be made using a best-first type of algorithm. By ordering the nodes on the frontier of the search, the A* algorithm is guaranteed to find a minimal cost path between two points when a path exists, without necessarily having to search exhaustively through all the possibilities. This is because the A* ordering of the nodes allows a safe cut-off (or pruning) of the search, soon after a path has been found between the two points. (A safe cut-off is one where it can be guaranteed that no better path will be found by searching through the remaining possibilities).

5.3.2 The Algorithm A*

Define an evaluation function, $\hat{f}$, so that its value, $\hat{f}(n)$, at any node n, is an *estimate* of the cost of a minimal cost path *constrained to go through* n. Let the function $k(n_i, n_j)$ give the *actual cost* of an unconstrained minimal cost path between two arbitrary nodes n_i and n_j. Let

$$h(n_i) = k(n_i, t)$$

where t is the goal node. An *optimal* path from n_i to the goal node is one that achieves $h(n_i)$. Define the cost of an *optimal* path from the start node s to some node n_i by

$$g(n_i) = k(s, n_i)$$

for all n_i accessible from s.

Now define a function f, such that, $f(n)$ is the actual cost of an optimal path constrained to go through node n. Then

$$f(n) = g(n) + h(n)$$

Let the evaluation function, $\hat{f}(n)$, be an estimate of f given as

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$$

where $\hat{g}$ is an estimate of g and $\hat{h}$ is an estimate of h. For $\hat{g}(n)$ the cost of the path which has been found by the search process in getting to node n is used. $\hat{h}(n)$ represents the best estimate of the cost of completing the connection between two points using Manhattan geometry while avoiding all obstacles between the two points. Nilsson proves that if $\hat{h}$ is a *lower bound* on h, then the A* algorithm is admissible. Therefore the obvious choice for $\hat{h}$ is the rectilinear (Manhattan) distance from n to the goal. This will always be a lower bound on the actual distance since the route may be forced to go out of its way to avoid obstacles, which only increases the length of the wire. In Manhattan geometry the shortest connection length is equal to the rectilinear distance between two points. Therefore, $\hat{h}$, the rectilinear distance distance to the goal, will always be a lower bound on h, the actual distance to the goal, and hence Algorithm A* will always find an optimal route.

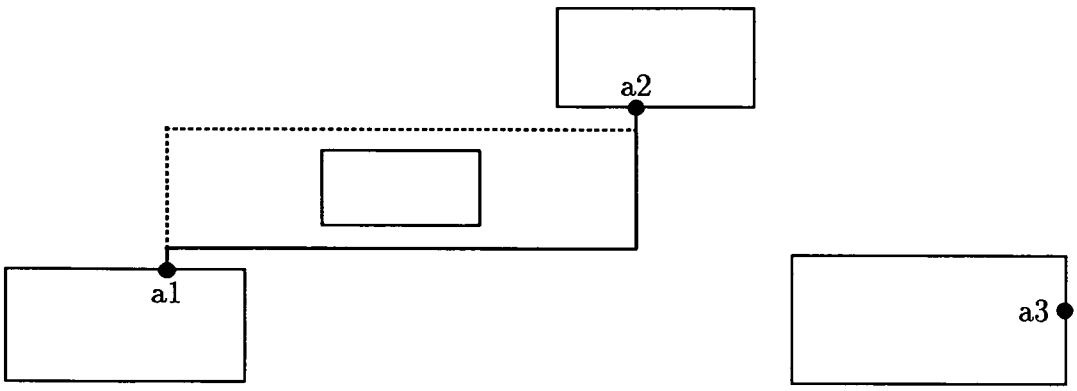


Figure 5–4: *Influence of unconnected pins*

5.3.3 Use of A* algorithm

Having explained how to make use of the A* algorithm to find a minimal cost path between two points efficiently, Clow then gave details of how to handle nets with more than two pins and how to make effective use of feedthrough possibilities.

An improvement to Clow's method of handling nets with more than two pins was described by Hsu [Hsu87]. The improvement was based on the observation that Clow's (and others) connection algorithms were *blind* in the sense that they did not consider the existence of *active* pins (pins in the net which were not yet connected). Hsu showed that by attaching a small magnetic force to each active pin better results were obtained than the results obtained by other search algorithms which do not consider the active terminals. In the A* algorithm the magnetic forces of the active pins generated small negative costs that reduced path costs. This is illustrated in Figure 5–4. For a global router where wire length is directly proportional to path cost, the two paths shown for connecting pins a1 and a2 have equal cost. However, if the influence of the unconnected pin, a3, is taken into account the cost of the path shown by the solid line becomes less than the cost of the path shown by the dotted line, and so the path shown by the solid line is chosen.

The Scotia global router is based on the router described by Clow, together

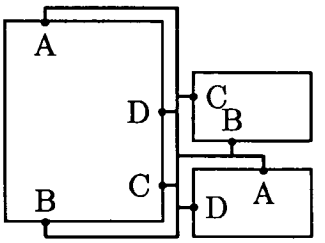
with the improvement described by Hsu. During global routing the power nets are routed in the same way as the other nets. After the global routing has been completed, the net connections are represented in Scotia as a set of horizontal and vertical wire segments. By numbering the nets and assigning the net number to the wire segments, the set of wire segments which represents the net connection can easily be identified. In addition to being assigned a net number, the wire segments are assigned widths determined by the design rules. The additional width requirements of the power nets are calculated and the wire segments of the power nets are updated to reflect these requirements. Finally, all the wire segments are assigned separations, again determined by the design rules, which represent the minimum space required between the different segments of the different nets.

5.4 Spacing & Compression

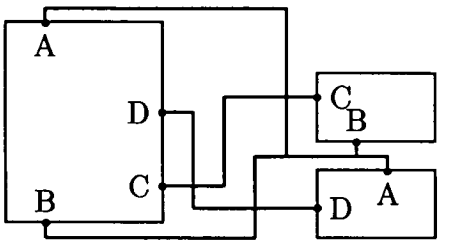
5.4.1 Spacing

The spacing operation takes as input a segment layout that may or may not satisfy the design rules and outputs a segment layout that satisfies the design rules. The design rules are embodied in the wire segments. In Scotia, a wire segment satisfies the design rules if there is sufficient space for the segment wire width and separation amongst the other wire segments and cells. If all the wire segments satisfy this constraint, the layout can be directly mapped into a wiring layout that satisfies the design rules using the direct correspondence between the wire segments with their associated width and the actual physical wiring.

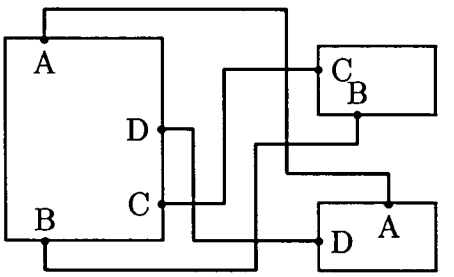
Spacing consists of sweeping through the design alternately horizontally and vertically, moving the wire segments and cells apart until termination occurs when the layout satisfies the design rules. Figure 5–5 shows the spacing of a very small layout with 3 cells and 4 nets. Figure 5–5(a) shows the placement and initial



(a) placement after initial global routing



(b) after horizontal spacing



(c) after vertical spacing

Figure 5–5: *Spacing Example*

global routing. At this stage in the layout process, the global wires are threaded through the design between the cells, and in many places the wires overlap.

Figure 5–5(b) shows the layout after it has been horizontally spaced. The wire segments and cells have been moved apart horizontally so that each wire segment is separated from its neighbour, which is either another wire segment or cell, by a space at least equal to the width of the segment plus its separation. When spacing global wires which overlap, there is a choice as to which wire is spaced first. This choice is decided using a terminal assignment routine which aims to minimize the number of unnecessary wire crossings.

Figure 5–5(c) shows the layout on completion of the vertical spacing. Termination of the spacing occurs when no further adjustment of the placement of the wire segments or the cells occurs. At this point each wire segment is separated from all the other wire segments of the other nets and from all the cells by a space at least equal to the assigned segment wire width plus separation.

In the simple example of Figure 5–5, termination occurred after one vertical sweep and one horizontal sweep, but this is not always the case. If all wire segments were joined at either end to other wire segments then termination would be guaranteed after one sweep in each direction. However, some wire segments have their ends joined to terminals of a cell and so there is not the same degree of flexibility in moving the segments. This is the same problem encountered by Hashimoto and Stevens during channel assignment [Hash71]. A problem occurs when two cells have wire segments emanating from terminals that are opposite each other as shown in Figure 5–6(a). During a spacing sweep this overlap is resolved by inserting a dog-leg as shown in Figure 5–6(b). However, this creates a violation of the design rules between the cell edge and the newly created dog-leg wire segment, which is then removed during the subsequent spacing sweep in the orthogonal direction. It is for this reason that more than two sweeps may be required to achieve a wiring layout that satisfies the design rules.

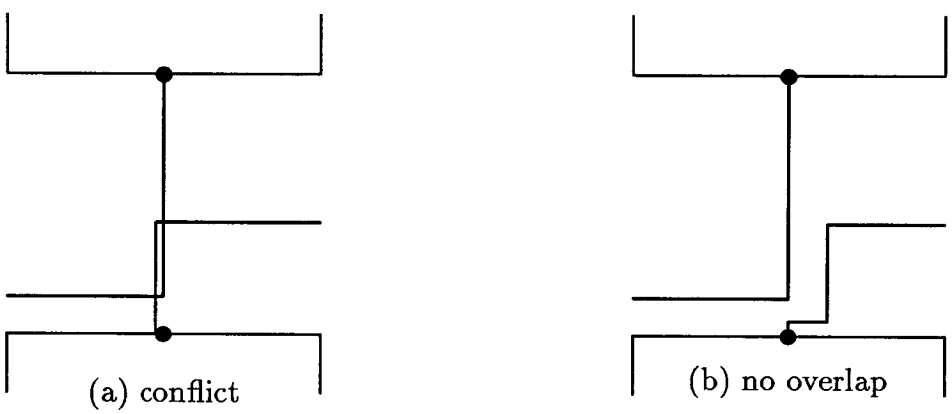


Figure 5–6: *Spacing Problem*

5.4.2 Compression

Spacing can introduce more space into the design than is necessary. The compression step recoups some of this space. Compression consists of sweeping through the design alternately horizontally and vertically, moving the cells and wire seg-

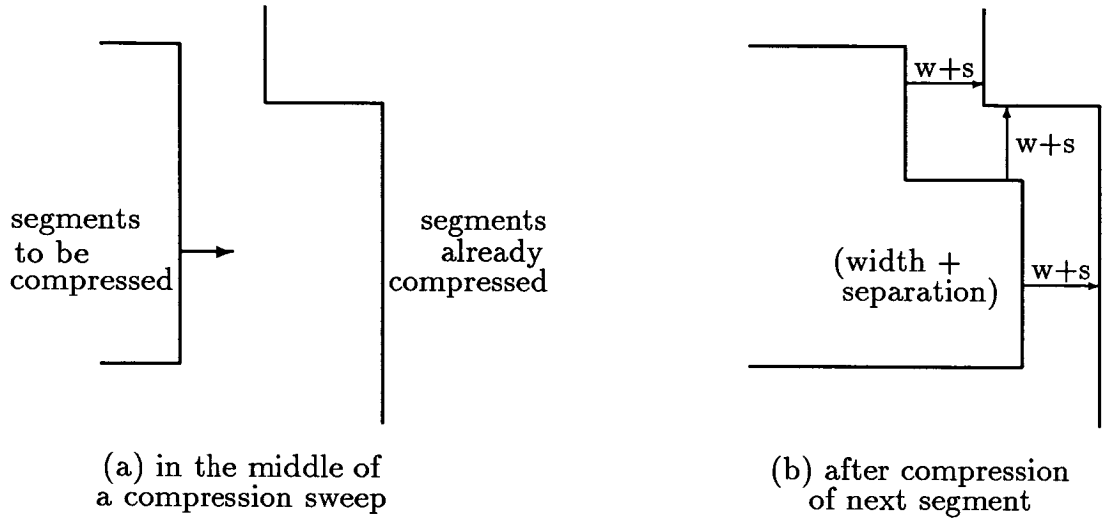


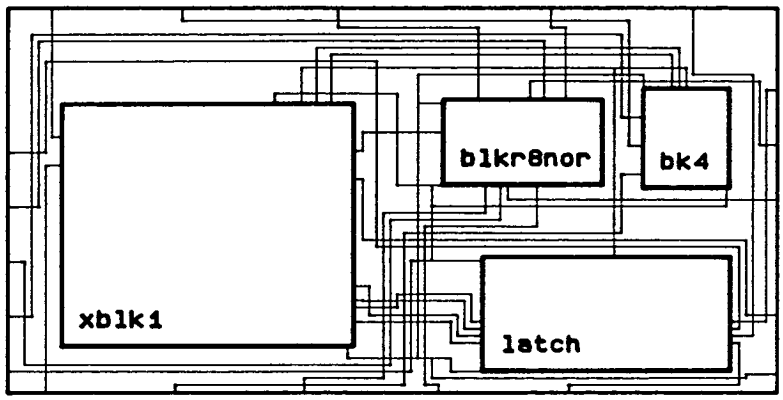
Figure 5–7: *Wire segment compression*

ments together within the design rule constraints. Jogs are inserted in the wire segments as necessary. The compression of a wire segment is shown in Figure 5–7. Figure 5–7(a) depicts the situation in the middle of a compression sweep. The segments on the right have been compressed, the segments on the left are to be compressed. Figure 5–7(b) shows the result of the compression of the next segment. It has been moved right to within a distance of the other wire segments determined by the assigned width and separation of the segment. In being compressed right a jog has been inserted in the segment, splitting the segment into three segments.

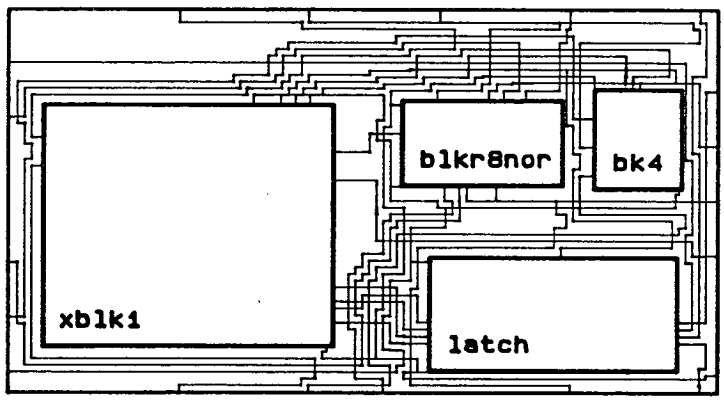
An example of a compression sweep is shown in Figure 5–8. Figure 5–8(a) shows a wiring layout after spacing and Figure 5–8(b) shows the same layout after it has been compressed from left to right.

Wire Straightening

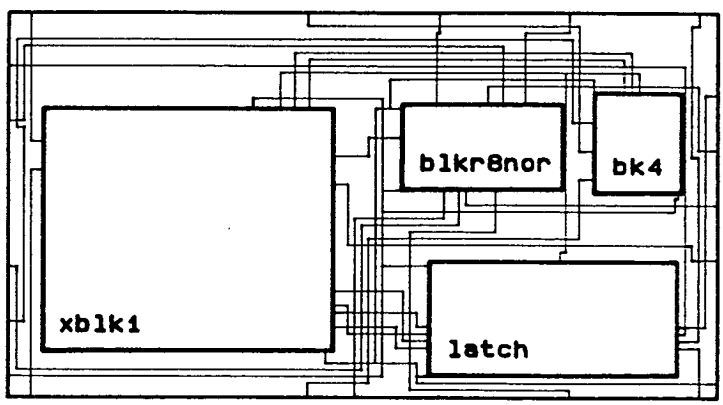
The last part of a compression sweep involves sweeping through the wiring layout trying to straighten wires. There are two types of wire straightening operation: pushes and jumps.



(a) Wire layout after spacing



(b) Wire layout after compression from left to right



(c) Wire layout after wire straightening

Figure 5–8: *Example of compression sweep*

Figure 5–9: *Examples of wire straightening push operations*

Examples of wire pushing are shown in Figure 5–9. The solid lines show the initial positions of the wire segments and the arrow indicates the direction of the push. A wire push involves moving the wire segment in the push direction whilst keeping within the design rule constraints. A wire push tries to move the wire segment far enough to straighten it. This operation is similar to that of wire compression shown in Figure 5–7, *except* that no jogs are inserted and the push distance is bounded by only going as far as to straighten it.

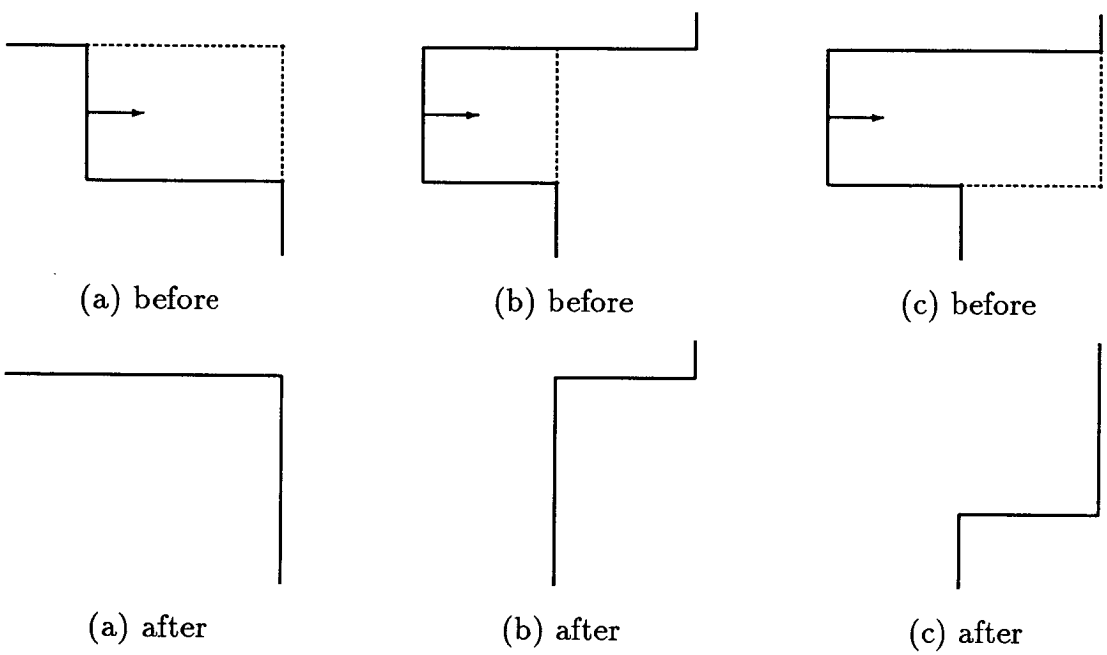


Figure 5–10: *Examples of wire straightening jump operations*

Examples of possible wire jumps are shown in Figure 5–10. The solid lines

show the initial positions of the wire segments and the dotted lines indicate the position of the moved wire segments after a successful jump. A jump is successful if the new positions of the wire segments satisfy the design rules.

A wire push does not move a wire segment past any intervening wire segments, whereas a jump may. Thus, a wire push preserves the relative ordering between wire segments, whereas a jump may alter the ordering. Experience has shown that the combination of these wire straightening operators, pushing and jumping, which remove jogs, together with compression sweeps, which insert jogs, mould the wire segments so that they fit more snugly into the fluid space between cells, allowing subsequent compression sweeps to reduce the overall design area even further. The compression example of Figure 5–8(b) after wire straightening is shown in Figure 5–8(c).

Compression Strategy

Compression Sweep	*Wire Straightening*	*Key*
LR	LR, RL, TB, LR, BT	L=left
BT	BT, TB, RL, BT, LR	R=right
RL	RL, LR, BT, RL, TB	T=top
TB	TB, BT, LR, TB, RL	B=bottom

Table 5–1: *Compression Strategy*

Experience has shown that good results are obtained if the compression strategy shown in Table 5–1 is used. The compression operation involves performing each of the compression sweeps followed by the associated wire straightening sweeps. After compression the spacing operation is called again to remove any remaining violations of the design rules and upon termination of this operation, the layout is complete.

For an individual layout a slightly smaller area may be obtained by changing the order of the compression sweeps. Also, a slightly smaller area is usually obtained by repeating the compression and spacing operation several times.

5.5 Physical Layout

Scotia is a 2-routing-layer integrated assembler. A 2-routing-layer scheme, involves horizontal wiring being realised on one layer and vertical wiring on the other. Often the routing layers are metal1 and metal2. Communication between the layers is made by vias. Sometimes the design rule widths and separations are different for metal1 and metal2. The integrated framework is able to take these differences into account by assigning different widths and separations to the horizontal and vertical wire segments, corresponding to the different layers of the routing scheme.

When the wire segments are mapped into physical wires, a via minimization step is performed. If the design rule widths and separations are different, then an additional constraint is placed on the minimization step, because although there may be enough space for a wire to be realised on one layer, there may not be enough space for the wire to be realised on the other layer.

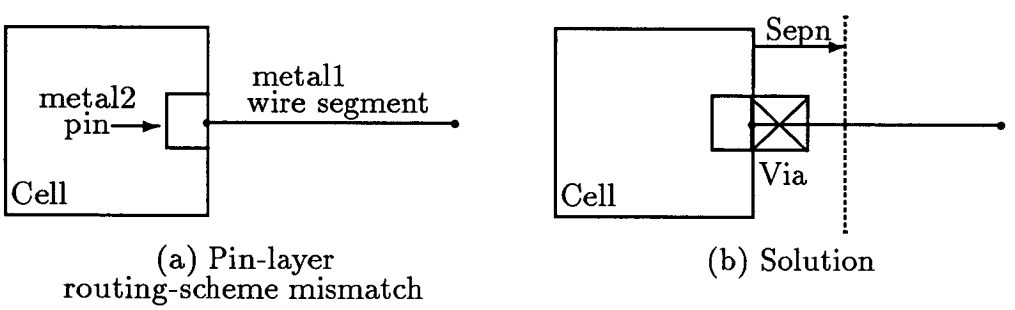


Figure 5–11: *Pin layer constraints*

Another problem caused by the routing scheme is that the layers of the pins of the cells and design edges may not be on the same layer as the adjoining wire

segments in the routing scheme. This situation is shown in Figure 5–11(a). The solution is to insert a via to join the pin layer to the wire segment layer as shown in Figure 5–11(b). The integrated framework is able to ensure that there is enough space to insert vias by assigning a separation to the cell or design edge, which prevents any wire segments parallel to the edge from coming too close and causing a violation of the design rules with the via. This is also shown in Figure 5–11(b).

5.5.1 Power Nets

It has already been mentioned that the tapered-width requirements for power nets can be accommodated in the integrated framework by assigning different widths to the power wire segments. An additional constraint is that the power wiring be all metal. This creates problems in a 2-routing-layer scheme where the layers are polysilicon and metal1. However, the integrated framework is flexible enough to accommodate these requirements.

Since there is only one metal layer available, the power nets need to be planar. (If they were not planar, the power nets would cross one another and where they crossed, one of the nets would have to be realised in polysilicon). To realise planar nets in Scotia, it is necessary to find global routes for the power nets that are planar. Except for the case of wire jumps, spacing and compression preserve the planarity of the wiring. Planarity during wire jumps can be preserved by checking, when a jump involving a power net is being considered, that the jump does not result in any other power net being crossed. There is also a problem in the routing layer scheme, which has metal1 for wires in one direction and polysilicon for wires in the other direction. With all metal power nets, it is required that the power nets be on metal1 where they would otherwise be on polysilicon. This in turn means that the other metal1 signal wires crossing the power net have to underpass the power net in polysilicon. This is illustrated in Figure 5–12, where the routing scheme is metal1 wires vertically and polysilicon wires horizontally.

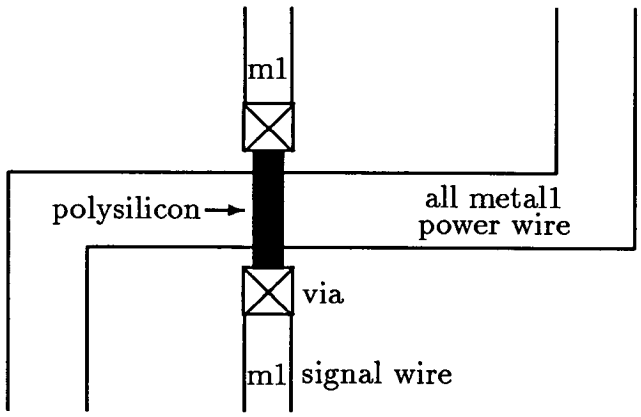


Figure 5–12: *Polysilicon underpassing by signal nets*

In the integrated framework the provision of space for vias to enable a polysilicon underpass can be realised by increasing the assigned separations of those power net segments that are not in accordance with the routing scheme.

5.6 Benchmark Results

The benchmark results quoted are for assembly examples taken from the MCNC 1988 International Workshop on Placement and Routing. The MCNC workshop provided a set of assembly benchmarks which were to be assembled by the participants and then the assembly results were to be compared at the workshop. The workshop recognised that often the assemblers used would be in a state of development. Consequently, several levels of participation were proposed. Participating at Level_1 involved placing the cells and routing all the nets so that the layout fitted into the least-area rectangle. For Level_1 the power and ground nets were to be treated as signal nets. Participating at Level_2*a* involved the same requirements as for Level_1, but with the additional requirement that the power and ground nets were to have proper wire widths so that electro-migration constraints were satisfied, and, if possible, voltage drop constraints were also to be satisfied.

Assembler	*Chip Area*	*Wire Length*	*Vias*
Scotia Level_1	25.48	569300	1331
Scotia Level_2*a*	26.13	585784	1334
Bear	28.47	633494	897
Mosaico	29.01	650009	1173
Vital[2]	31.17	865712	1029
Seattle Silicon[3]	28.63	762000	1235
Delft P&R[4]	26.57	615104	925

Table 5–2: *Results on PrimBBL1 with 10 Cells and 203 Nets*

Assembler	*Chip Area*	*Wire Length*	*Vias*
Scotia Level_1	2.58	141250	1041
Scotia Level_2*a*	2.61	144928	1087
Bear	2.83	131244	798
Mosaico	3.16	151824	813
Vital[2]	3.12	134599	763
Seattle Silicon[3]	2.94	125000	948
Delft P&R[4]	2.60	151656	967

Table 5–3: *Results on PrimBBL2 with 33 Cells and 123 Nets*

[2]treats power and ground nets as signal nets

[3]routes I/0 connections only to the chip boundary and not to the pins

[4]does the placement manually

Table 5–2 shows the assembly results obtained by the Scotia assembler for benchmark PrimBBL1 compared with other assemblers, and Table 5–3 compares the Scotia results for benchmark PrimBBL2. The Level_2*a* results quoted for Scotia also satisfy the voltage drop constraints. The assembly results of the other assemblers are quoted from [Dai89], but it is not stated which level of participation the results are for.

The purpose of quoting Level_1 results is to provide a basis for comparison with other assemblers. It would be expected that the Level_1 results would have a fractionally smaller chip area than the Level_2*a* results, because the power and ground nets have the same wire widths and separations as the signal nets.

The initial placements used by Scotia are derived from those used in [Dai89]. The placements differ only in the initial separations between the cells and are shown in Figures 5–13 and 5–14. The orientations of the cells are the same. For clarity, the layouts of Figures 5–15 and 5–16 show the wire segments immediately prior to being turned into geometry.

The Scotia results quoted in the Tables 5–2 and 5–3 have been obtained using a routing scheme of assigning metal1 horizontally and metal2 vertically, together with the compression strategy shown in Table 5–1. For the benchmarks the compression and spacing step has been applied to the layout no more than 12 times. Each of the results has been obtained in less than 2 hours total running time on a Sun3 workstation.

In the benchmark design rules, the metal1 and metal2 separations are the same, but the metal1 width is smaller than the metal2 width. The reason for assigning metal1 horizontally and metal2 vertically is that the best results for the benchmarks are obtained this way. Experience has shown that the first compression sweep usually removes more space than any subsequent sweep. By assigning the wires so that the wider metal2 wires are compressed in the first sweep, less space is removed than if metal1 wires had been compressed. However, in the subsequent

sweep metal1 wires are compressed and more space is removed than if the metal2 wires had been compressed. Overall, using this routing scheme, after all the compression sweeps have been performed, it was found that the design areas for the benchmarks were slightly smaller than if the metal wires had been assigned the alternative way.

An alternative routing scheme is to assign metal1 vertically, metal2 horizontally and to perform the first compression sweep from bottom to top rather than left to right. For other layouts it may well be that this produces slightly better results. However, for the case of assembling the two benchmarks slightly better results were obtained using the previous routing scheme.

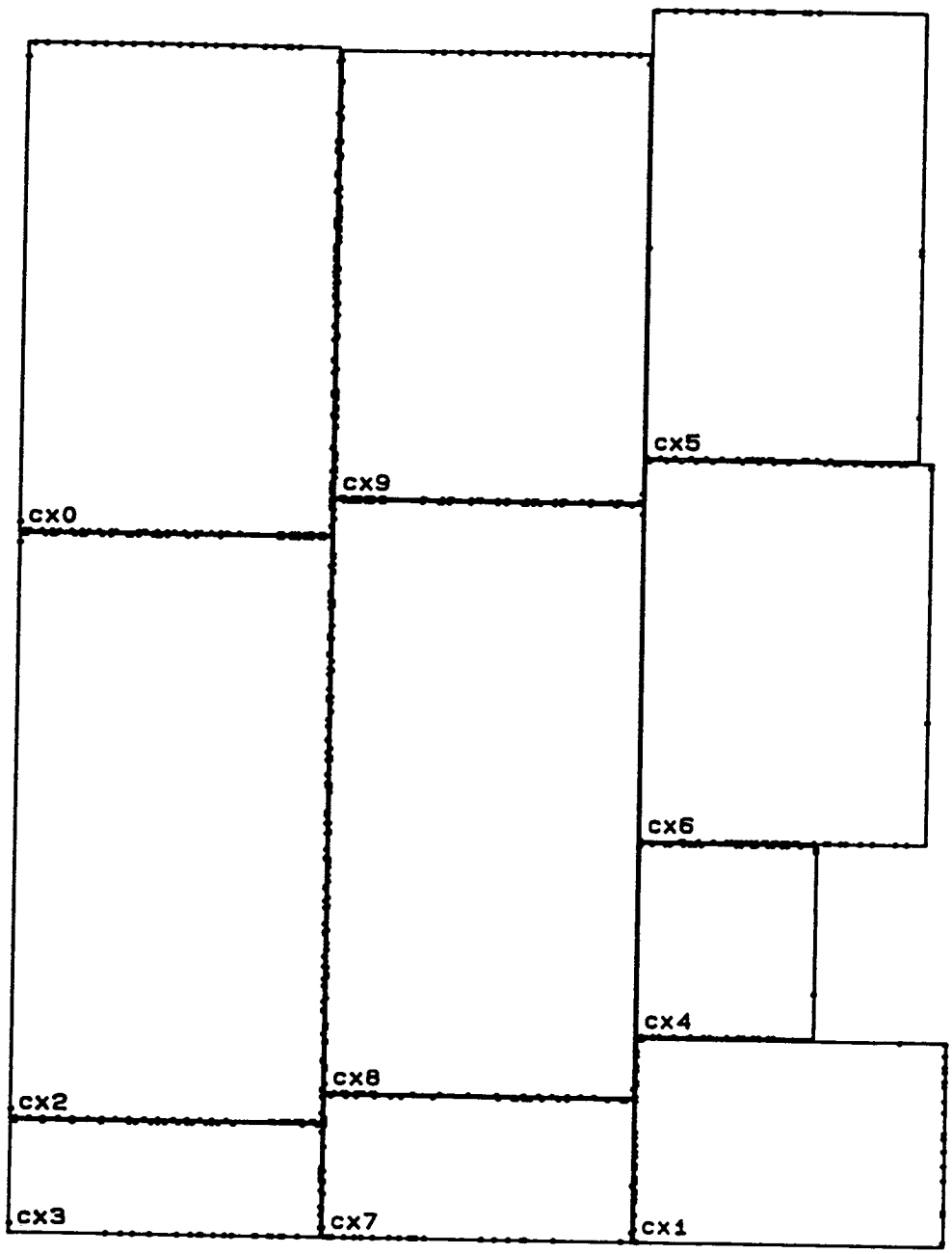


Figure 5–13: *Initial Placement of PrimBBL1*

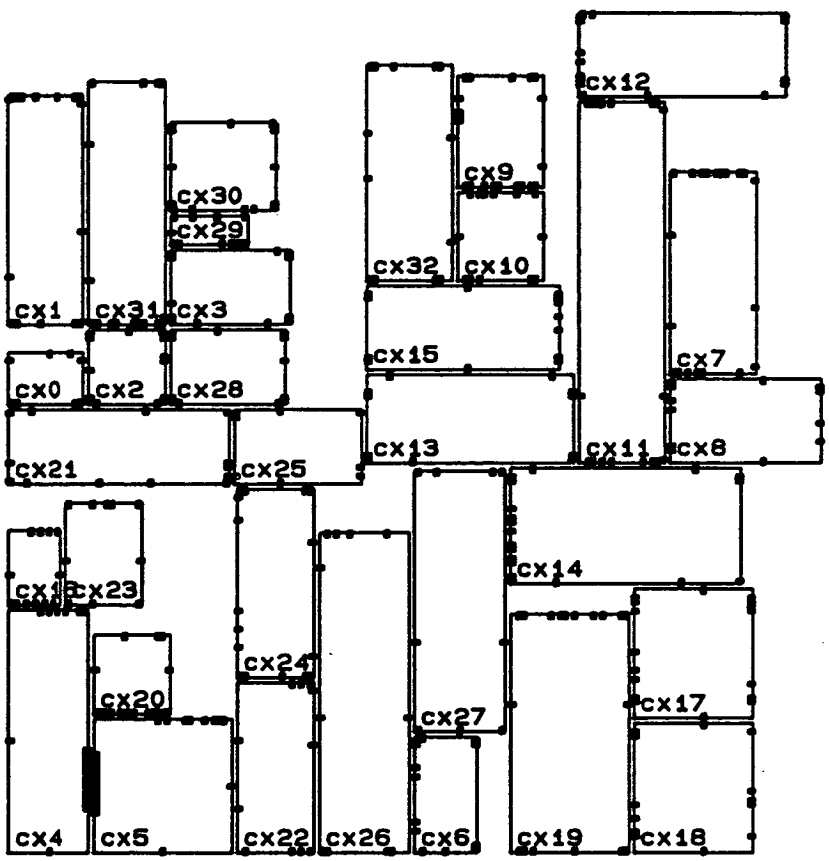


Figure 5–14: *Initial Placement of PrimBBL2*

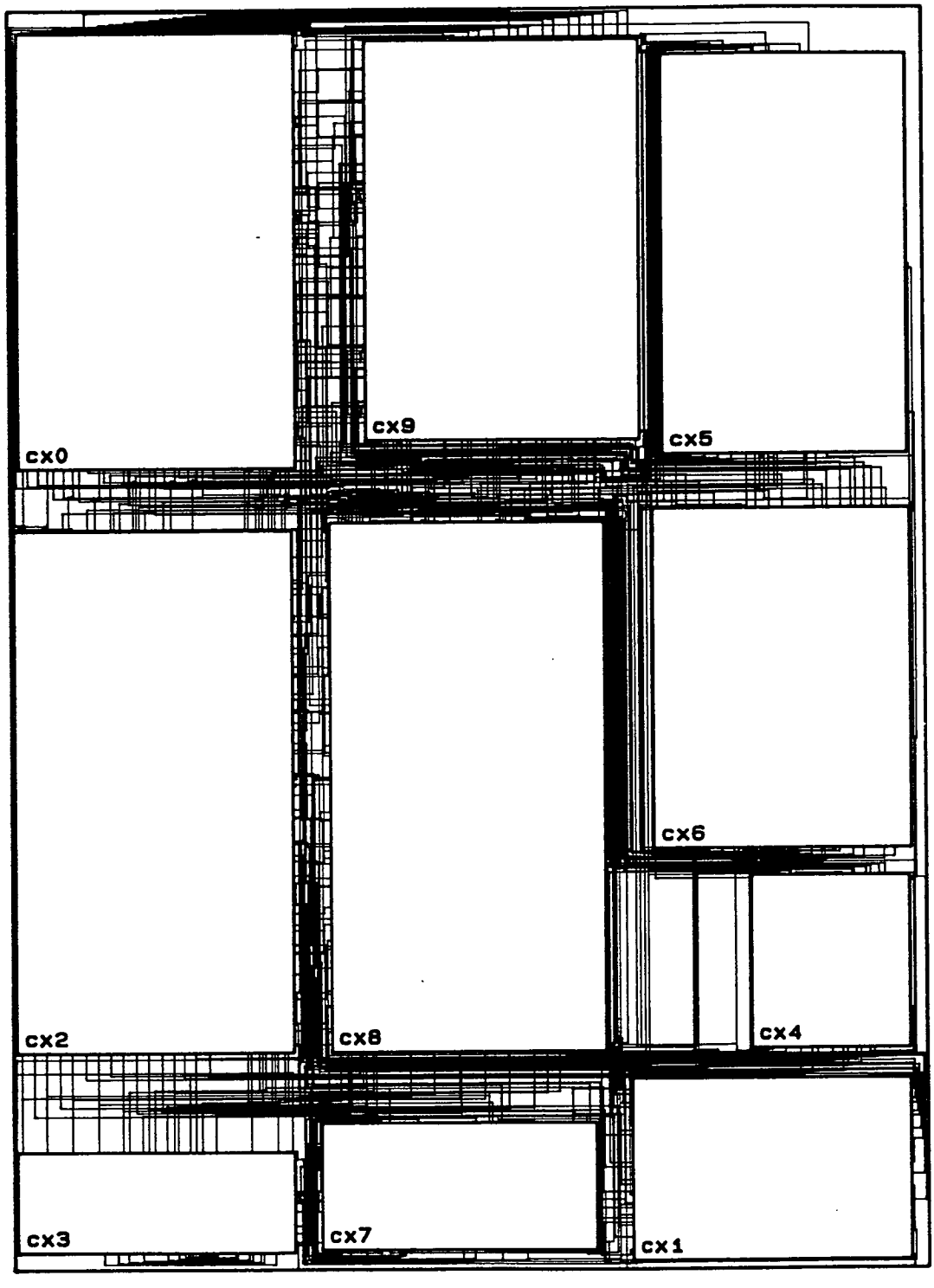


Figure 5–15: *Layout of PrimBBL1*

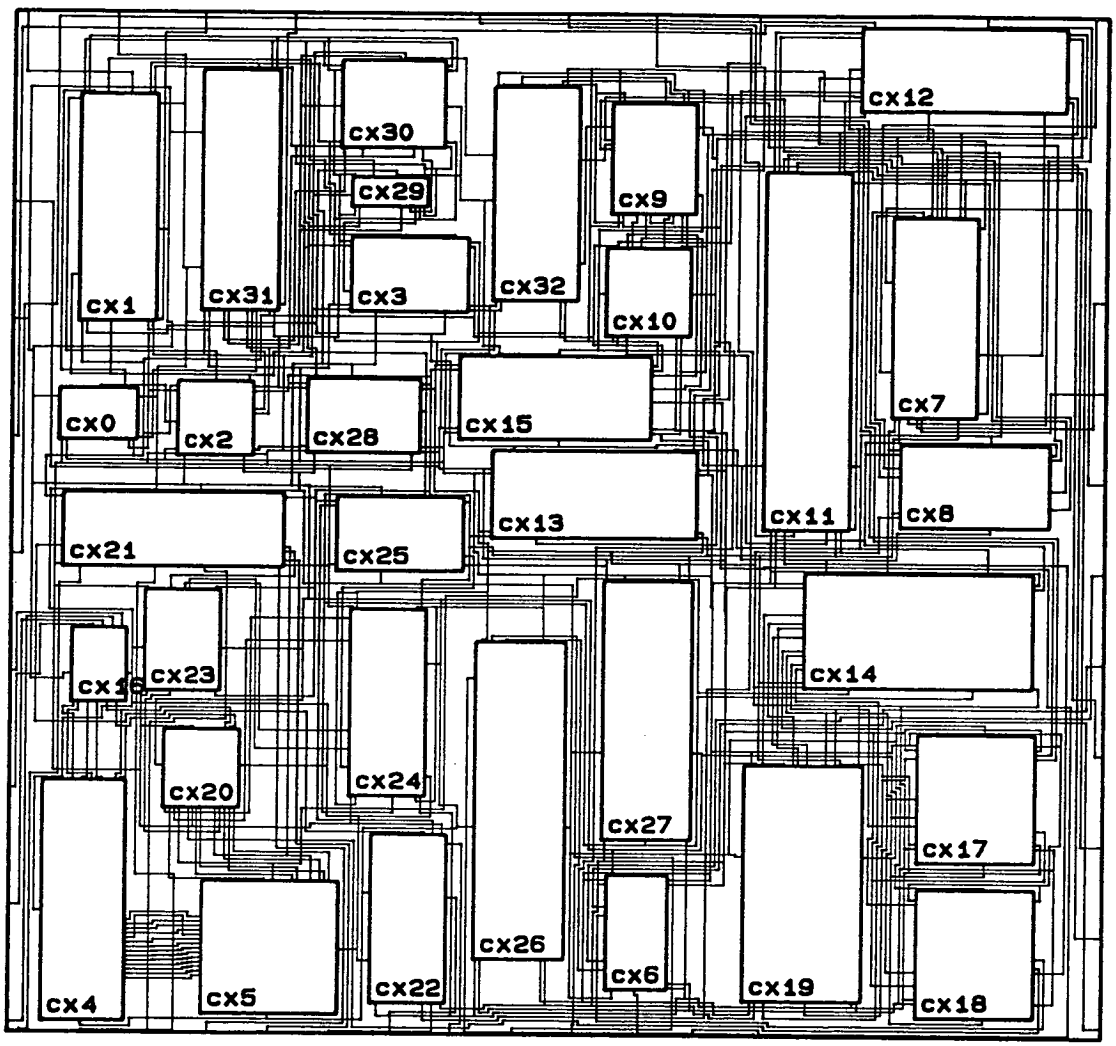


Figure 5–16: *Layout of PrimBBL2*

5.6.1 Delft Placement and Routing System

Looking at Tables 5–2 and 5–3, before the advent of Scotia, the best results for both benchmarks had been obtained by the Delft P&R system for custom VLSI layout design [Cai90a]. The Delft system is capable of finding a cell placement, routing the power and ground nets in a single metal layer and routing the signal nets in two to four layers.

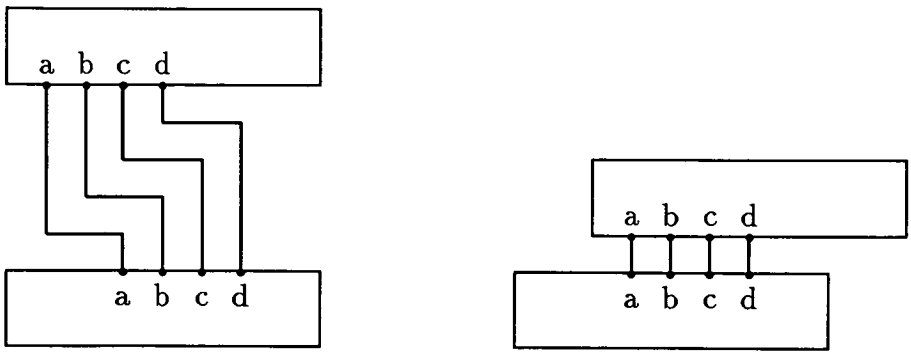


Figure 5–17: *Channel width variation with cell positioning*

The placement program is interactive and the system requires that the final placement be in the form of a slicing structure. Finding the global and power routes is automatic, as is the construction of the routing channels and the detailed routing. During detailed routing, each time a channel is routed, the channel width is altered to accommodate the detailed routing, and the cells adjacent to the channel are merged with the channel to form a composite cell with ragged edges, in the same way as in Lauther's assembler [Laut85]. The Delft system also provides the option for "experienced users" to execute the global router, and change the cell placement (to influence the channel construction) interactively via a graphical interface.

An enhancement of the slicing structure approach to assembly is based on the observation that the relative positions of the cells along the two sides of a channel in the channel length direction (the direction perpendicular to channel width adjustment) can have a great effect on the routing results. This is illustrated in Figure 5–17 which shows a large reduction in channel width by an alteration of the cell positions. The Delft system incorporates an algorithm which claims to calculate the *optimal* relative positions of the cells [Cai90b].

Assembly Approach	*Core Area of Chip*
Waring Stepwise	29.19
Scotia	16.63

Table 5–4: *Results on Harwell chip with 152 cells and 339 nets*

5.6.2 Harwell Chip

Scotia was used to assemble the Harwell chip design [Harw90] that had previously been assembled using the Waring stepwise assembler described in Chapter 3. As can be seen in Table 5–4, the chip area for the design is much smaller using Scotia. Figure 5–18 shows the layout of the cells and wire segments.

The comparison of assembly approaches is approximate. The design was assembled hierarchically using the stepwise assembler, whilst Scotia assembled the design as a single level of hierarchy. The stepwise assembly used three routing layers with the power nets being routed on metal2 and the signal nets being routed on polysilicon and metal1. Signal routing could be run underneath the power routing and wherever there was no power routing present the polysilicon was replaced with metal2. The Scotia assembly used only two routing layers (metal1 and metal2) and the power nets were routed using both layers. In the stepwise assembly the design was pad-bound along one of the sides, however, the chip areas quoted are for the core area without pads, so being pad-bound does not influence the result. In the absence of a given pad placement the Scotia assembly distributed the pads equally along the four sides of the design, so the pad layout is different from that in the stepwise assembly. Although, the comparison is approximate, it is still clear that the Scotia integrated approach to assembly is superior to the Waring stepwise approach.

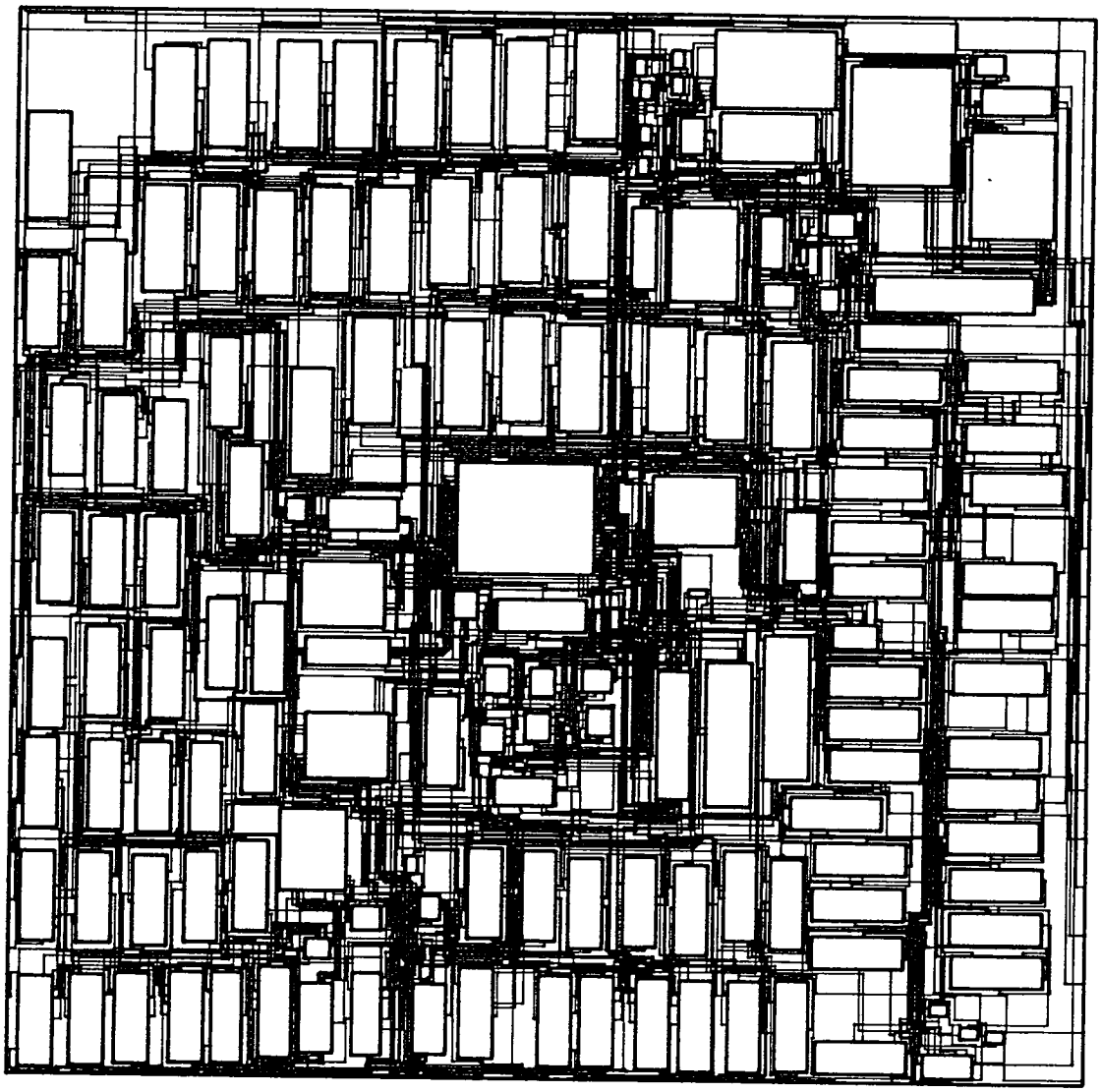


Figure 5–18: *Layout of Harwell chip using Scotia*

Chapter 6

Extensions of Scotia

The previous chapter described the implementation of an integrated assembler called Scotia. The benchmark results obtained by the implementation show that the integrated framework is currently one of the best ways currently known of assembling chip designs. There are potentially many ways of implementing an integrated assembler framework, Scotia is but one way. In this chapter, enhancements to Scotia, which have not been implemented, are described and assessed.

6.1 Manual Interaction

It is still true that studying the Scotia segment layout, a designer can see improvements that could be made to the layout. This fact is evidence that assembler implementations still have to embody algorithmically more human intuition. It is true that Scotia runs automatically with no manual interaction and produces good results, however, improvements in layout quality could still be gained if manual interaction were to be employed.

Scotia is good at coping with the complex detail of routing all the nets. Looking at the final layout, a designer can see improvements that can be made. A plausible scheme for manual interaction involves producing a layout, interacting manually, and then compressing and spacing the layout again to realise a layout that satisfies the design rules. Given the complexity of the layout, it would be very difficult and time consuming for the designer to manually interact and produce a layout that satisfies the design rules; it is far quicker to let Scotia take the design after manual interaction and produce a layout that satisfies the design rules. Two possible operators for manual interaction with the layout, net re-routing and cell placement adjustment are described below.

6.1.1 Net Re-Routing

One of the goals of assembly is to keep the wire length of the nets as short as possible. Sometimes though, cell placement adjustments during assembly result in wires that are longer than they need be. This is illustrated in Figure 6–1. Figure 6–1(a) shows the situation after the initial global routing. Figure 6–1(b) shows the routing of the net after assembly. Manual interaction could be used to rip-up the net and re-route it, as shown in Figure 6–1(c).

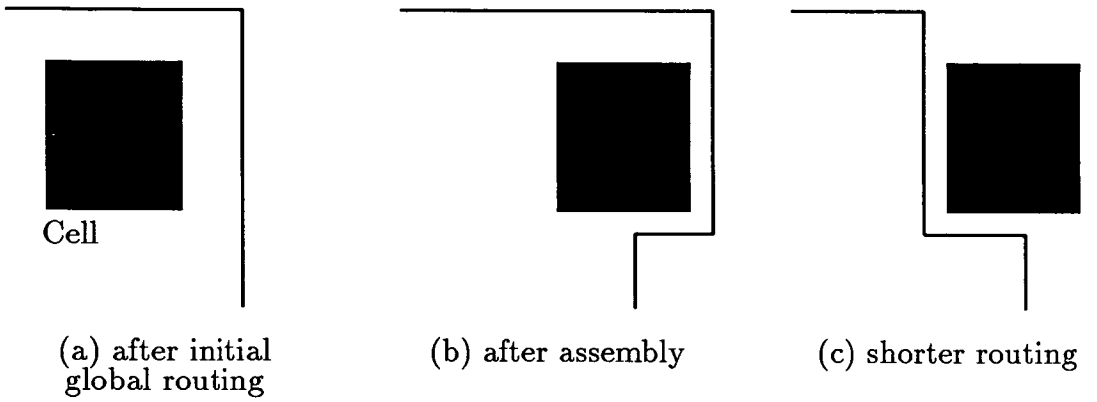


Figure 6–1: *Case for re-routing a net*

Instead of manually re-routing a net, Scotia's own global router could be called. An experiment was tried in which the final placements of the benchmark designs, which had been assembled by Scotia, were used as the starting point for reassembly by Scotia. The idea behind the experiment was that if the design was reassembled, then shorter routes for some nets would be found in the context of a placement that resulted in a good match between the wiring capacities of the spaces between the cells, and the number of wires passing through the spaces. However, the experiment was not a success, the reassembled designs were larger than the originally assembled designs. This was because after the design had been globally routed again, a placement adjustment of the cells was required, since in some places the number of wires passing through a space exceeded the wiring capacity. The Scotia reassembly made the necessary placement adjustments, but the resulting designs were larger than the originals. Initially, this result was surprising, but on investigation it was apparent that less moulding of the wiring into the spaces between the cells had taken place. Less moulding had occurred because the cells had been perturbed less than in the original assembly, since the cells of the reassembly had already been spaced. Scotia is more effective when the cells are initially closer together because there is more cell perturbation and so more moulding of the wires takes place. The initial benchmark placements shown in Figures 5–13 and 5–14 in the previous chapter have their cells close together.

Net re-routing using manual interaction warrants further research. As the simple example of Figure 6–1 shows, shorter wire lengths result from re-routing the nets, but, as the experiment showed, a larger design area currently results if all the nets are re-routed.

6.1.2 Cell Placement Adjustment

(a) initial placement (b) placement after assembly (c) alternative initial placement

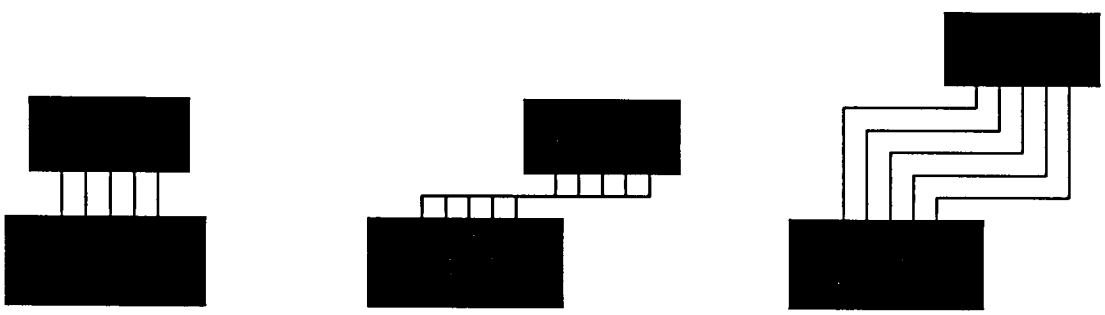
Figure 6–2: *Use of manual placement adjustment*

One of the goals of assembly is to assemble the design in as small an area as possible. Particularly in designs with many cells, manual interaction can be used to good effect, because although a placement may minimize the total net lengths, the layout area for the design may be larger than it need be. By adjusting the cell placement by moving the cells or rotating them or both (and probably increasing the total net length) a smaller layout area for the design may result. This is illustrated in Figure 6–2. Figure 6–2(a) shows the initial cell placement and Figure 6–2(b) shows the placement after assembly. Studying the post-assembly placement it can be seen that a smaller design area *may* occur, if the cell shown in outline is moved and rotated, so that the placement shown in Figure 6–2(c) is used as the starting placement. The starting placement will have a different wire distribution and so it is not certain whether a smaller design area will result, until the assembly is completed.

6.2 Improved Algorithm

As was stated previously, the segment layout produced by Scotia is not perfect and there are some obvious improvements that could be made. Two improvements cell locking and segment permutation are described, followed by a description of the possible use of two-dimensional sweeps in the spacing and compression operations.

6.2.1 Cell Locking



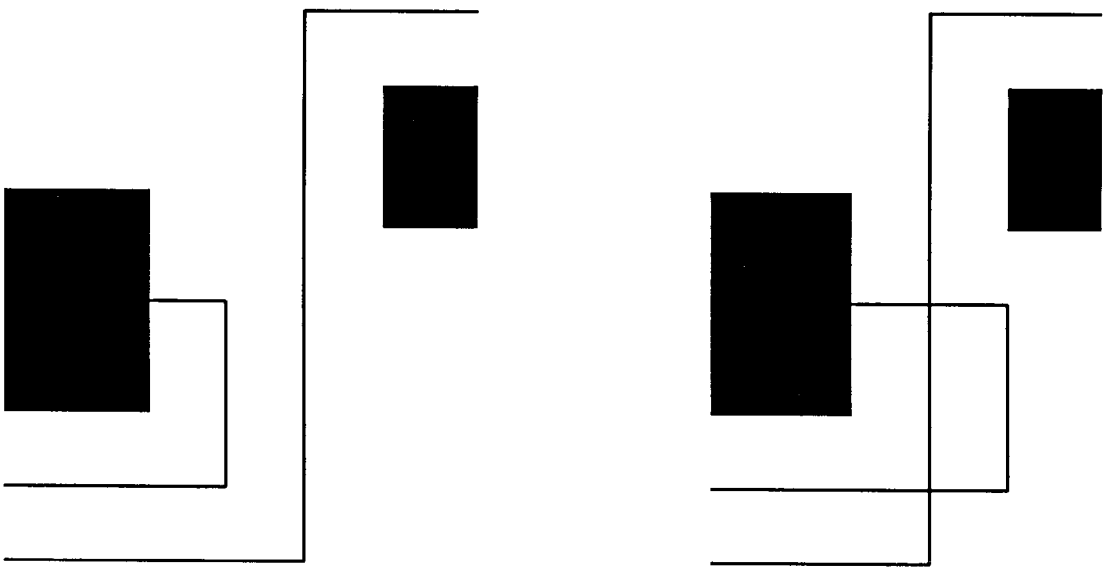
(a) good layout (b) after compression sweep from left to right (c) subsequent compression sweep from bottom to top

Figure 6–3: *The need for adjacent cell locking*

An improvement in layout quality could be made, if adjacent cells could sometimes be "locked" together. Consider the layout situation shown in Figure 6–3(a). There is a bundle of wires connecting the bottom and top cells together, a common layout situation. During the compression sweep from left to right, each cell tries to move as far right as possible and this can lead to the layout situation shown in Figure 6–3(b). The next wire straightening is ineffective, because the vertical segments of the former straight across connections have each been broken into two vertical segments joined by a horizontal segment, with the horizontal segments stacked on top of one another. A subsequent compression sweep from bottom to top will result in the layout situation shown in Figure 6–3(c). (The layout of benchmark

PrimBBL2 shown in Figure 5–16, has an instance of this problem in its bottom left corner). What is needed in Scotia is a means of detecting *when* adjacent cell locking is required, and a *means* of implementing the lock.

6.2.2 Segment Permutation



(a) minimal wire crossing (b) smaller design area

Figure 6–4: *Use of segment permutation*

The aim of terminal assignment is to minimize the number of wire crossings and a situation in which the number of wire crossings is minimized is shown in Figure 6–4(a). However, if the two vertical wire segments were to swap places, then a further compression of the design is possible as shown in Figure 6–4(b). Compared with Figure 6–4(a), there are more wire crossings and the wire lengths are slightly longer in Figure 6–4(b). What is needed in Scotia is a means of deciding when permuting the segments (which may increase the wire length and number of wire crossings) is desirable to try to reduce the overall design area. When to permute the segments is an example of the general problem of assessing when a local operation (segment permutation) has a beneficial global effect (reduction in the overall design area).

6.2.3 Spacing & Compression

At present, the spacing and compression operations use a series of one-dimensional sweeps, sweeping alternately in the horizontal and vertical directions. Instead of using two successive one-dimensional sweeps, there is the possibility of using one two-dimensional sweep. Use of a two-dimensional sweep could be faster than two one-dimensional sweeps, and could lead to smaller designs due to improved communication within the sweep.

Use of a two-dimensional sweep could also enable the compression and spacing operations to be merged into a single operation. At present a compression sweep can introduce violations of the design rules which are removed in the subsequent sweep in the orthogonal direction. If the two-dimensional sweep could always ensure that there were no outstanding violations at the end of the sweep, then there would be no need to call the spacing operation.

6.3 Incorporation of Floorplanning and Placement

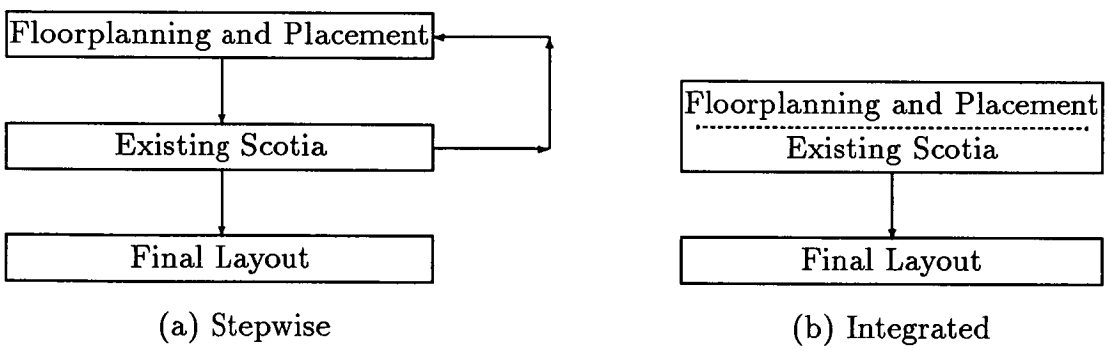


Figure 6–5: *Choices for incorporating floorplanning and placement*

Two possible ways of incorporating floorplanning and placement - F&P - involve either performing F&P followed by the existing Scotia in a stepwise manner as

shown in Figure 6–5(a), or integrating the F&P with the existing Scotia as shown in Figure 6–5(b).

Experience gained from implementing Scotia has shown that an integrated approach is better than a stepwise approach. If F&P is performed in a stepwise manner, the question of how to specify the interface between the two steps needs to be answered. Also, if it is intended to iterate between F&P and the existing Scotia, the question of what sort of feedback is provided needs to be addressed.

The alternative integrated approach to F&P poses the question of how to handle the vastly increased complexity generated by the additional F&P variables. The Berkeley BEAR system [Dai89] combines the F&P with the global routing to find placements that require very little adjustment during the detailed routing. The BEAR approach is very promising, because it could be extended to allow timing constraints to be imposed on the design.

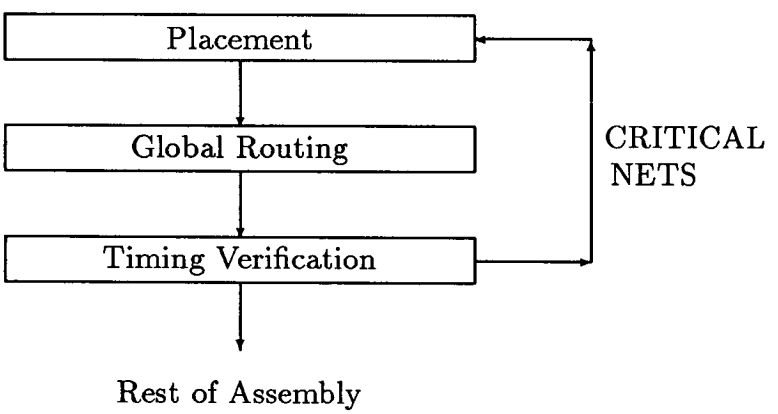


Figure 6–6: *Stepwise approach to handling timing constraints*

Work by Burstein and Youssef [Burs85] investigated how to include timing constraints during silicon assembly. The stepwise approach shown in Figure 6–6 involves finding a placement and global routing, checking that the timing constraints are satisfied for the critical nets, and then adjusting the placement to reduce the length of those critical nets which do not satisfy their timing con-

straints. Using the stepwise approach there is a convergence problem, because adjusting the placement to reduce the length of some critical nets, may increase the length of others causing them to no longer satisfy their timing constraints, resulting in further iterations of the timing-cycle.

To avoid timing-cycle convergence problems, Burstein and Youssef investigated unifying the timing constraints with the placement and global routing. Their preliminary results showed that in finding a placement that satisfied the timing constraints, there was little effect on the overall chip design size. This was because existing placement and wiring algorithms were non-optimal and there were several acceptable (good) solutions, so the reduction in lengths of the relatively small number of critical nets could be achieved without adversely affecting the total wire length and wiring congestion.

6.3.1 Scotia Floorplanning and Placement

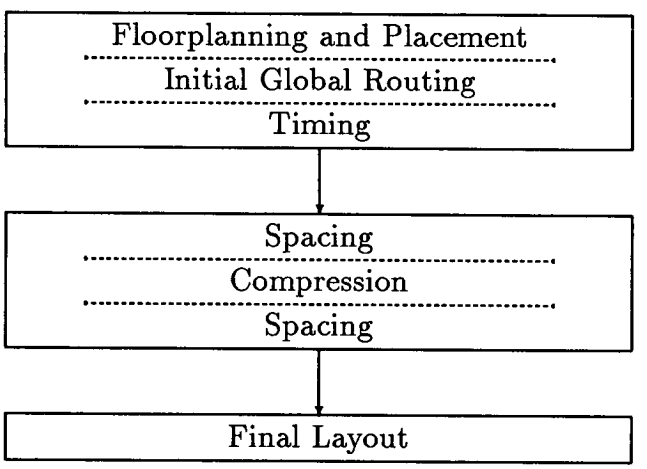


Figure 6–7: *Proposed integrated silicon assembler*

Considering the success of the BEAR work and the case for unifying the timing, a good way of extending Scotia to include a F&P step would be to integrate the step with the initial global routing as illustrated in Figure 6–7. Integrating the

F&P this way is a trade-off between the stepwise and totally integrated approaches previously described. As the BEAR work has shown it is possible to merge the F&P with the global routing and yet control the complexity. Including the global routing makes it easier to impose the constraints of timing and minimizing the design area. There is also good communication between the two steps because the spacing and compression step manipulates the cell placement and wiring - precisely what is output by the F&P and global routing step.

6.4 Extension to N-Layer Routing Scheme

6.4.1 Three Layers

The benchmarks of the previous chapter were assembled using two layers of metal. An assembly specification using three layers of metal was also provided in which metal3 could be routed anywhere, but subject to the restriction that vias to metal3 were not allowed over the area of any cell. In the specification however, it was not mentioned whether it was possible to go from metal1 to metal3 directly by using a metal2 to metal3 via stacked on top of a metal1 to metal2 via. (In some design rules stacked vias are not allowed).

The first impression of using three routing layers is that the amount of space occupied by the routing will be halved, because the metal3 can be routed on top of the metal1 and metal2 wiring and over the cells. However, the first impression is misleading, because although metal3 can be routed anywhere, it must be *planar*. It is only possible to route metal3 both horizontally and vertically, subject to the constraint of the horizontal and vertical wiring not crossing. It is due to this constraint that the reduction by half of the space occupied by the routing is unlikely to be achieved.

6.4.2 More Than Three Layers

There is potential for substantial reductions in the routing space if more than three routing layers are available. Metal3 and metal4 can be used in the same way as metal1 and metal2. What is needed, though, is a means of managing the extra computational complexity of routing with n-layers, because when using many routing layers it would take a very long time to consider all the possible wirings on all the different layers.

One approach, which is used in detailed routing, involves starting with a two-layer routing solution and systematically modifying the solution by moving some of the wiring onto the other layers. However, in detailed routing the wires were stacked on top of each other and the channel was compressed to recoup the space vacated by the stacked wires. This approach cannot be transferred directly into an assembler, because wires can be stacked not only on top of other wires but also over cells, and compression to recoup the space is not a simple problem.

An alternative approach is to partition the nets and the layers beforehand, so that groups of nets are to be routed on specified layers. This approach simplifies the 'which wire to which layer problem' and allows the assembly to make full use of the routing space available for each layer. However, if this approach is used then heuristics for the partitioning must be found.

Another problem which arises when using n-layer routing, is that of joining the cell pins to the wiring used to interconnect the cells. If the cell pin layers are fixed and stacked vias are not allowed, then a significant amount of the routing space can be used for joining the pin layer to the interconnect layer as illustrated in Figure 6–8.

The problem of joining the pin layer to the wiring could be avoided if the assembler could specify the layers of the pins of the cells. Using n-layer routing in a cell, the pins would no longer have to be positioned along the cell edges but

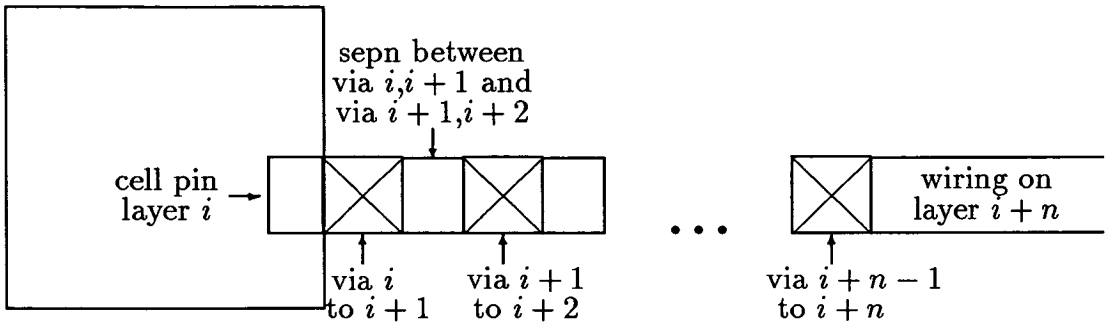


Figure 6–8: *Pin layer to wiring connection*

instead, since some routing can be made over a cell, a pin connection could be made inside a cell. Changing the constraints on cell design is not unreasonable, however, this further complicates the already complicated assembly problem.

6.4.3 Extension of Scotia

The previous sections have shown that developing an n-layer routing assembler is not simply a case of extending the 2-layer Scotia. Questions that need to be answered include:

- How many layers?
- How to manage the increased complexity resulting from using more layers?
- Are stacked vias allowed?
- Can the cell pin layers be specified by the assembler?
- What are the constraints on the cell design?

The answers to these questions have many implications for how assembly is performed. It could even lead to a new approach to assembly being developed.

Chapter 7

Conclusion

Here is my journey's end.

William Shakespeare, *Macbeth*

Silicon assembly contains several well-separated sub-problems and so lends itself naturally to the stepwise approach to tackling the problems posed by assembly. However, experience has shown that this approach is not successful, not because of the failure of the individual parts, but rather because of the breakdown in communication between the parts. In breaking the problem down into a series of sub-problems the overall coherence of the assembler is lost and this is reflected in its poor performance. Silicon assembly is a programming problem whose results are significantly affected by how the problem is decomposed into sub-problems. Consequently, a new integrated approach to solving the assembly problem has been described. The communication problems that plagued the stepwise implementation are no longer present. These communication problems have not been "worked around", they no longer exist. The integrated framework is built on the expertise gained from implementing a stepwise assembler. The stepwise assembler could have far better implementations of the individual steps, and yet would still produce worse results, in a longer time, than any integrated assembler. With a programming task of this size, having good implementations of the steps alone is not enough, the framework in which the steps operate is paramount. The proposal of the novel integrated framework for assembly was based on the observation that in the layout, the global routing and placement adjustment together with the terminal assignment is almost equivalent to the detailed routing for most of the nets. By totally abandoning the notion of channels, and instead, viewing the assembly problem in terms of cells and fluid space between the cells, a very effective approach to assembly is created. Since there are no separate channel definition and detailed routing steps, less code is required to implement an integrated assembler than to implement a stepwise assembler. The benchmark results obtained using an integrated assembler are comparable to, or slightly better than those produced by existing assemblers. Consequently, the integrated framework is a suitable platform for taking silicon assembly forward in the 1990's.

Bibliography

[Burn87] J. Burns, A. Casotto, M. Igusa, F. Marron, F. Romeo, A. Sangiovanni-Vincentelli, C. Sechen, H. Shin, G. Srinath, H. Yaghutiel. **Mosaico: an integrated macro-cell layout system.** *Proc. VLSI 87*, pp 165-179, Ed. C. H. Séquin, North-Holland 1988.

[Burs83] M. Burstein, R. Pelavin. **Hierarchical channel router.** *Proc. IEEE* 20th *Design Automation Conf.*, pp 591-597, ACM 1983.

[Burs85] M. Burstein, M. N. Youssef. **Timing influenced layout design.** *Proc. IEEE* 22nd *Design Automation Conf.*, pp 124-130, ACM 1985.

[Cai90a] H. Cai, P. Groeneveld. **Delft placement and routing system user's manual.** Delft University of Technology, Department of Electrical Engineering, Mekelweg 4 2628 CD Delft, The Netherlands.

[Cai90b] H. Cai, P. Th. Knijf, P. Groeneveld. **Routing optimization by adjusting relative positions of the building blocks.** Delft University of Technology, Department of Electrical Engineering, Mekelweg 4 2628 CD Delft, The Netherlands.

[Chen87] H. H. Chen. **Routing L-shaped channels in nonslicing structure placement.** *Proc. IEEE* 24th *Design Automation Conf.*, pp 152-158, ACM 1987.

[Clow84] G. W. Clow. **A global routing algorithm for general cells.** *Proc. IEEE* 21st *Design Automation Conf.*, pp 45-51, ACM 1984.

[Dai85] W. Dai, T. Asano, E. S. Kuh. **Routing region definition and ordering scheme for building-block layout.** *IEEE Trans. Computer-Aided Design* 4(3), pp 180-197, IEEE 1985.

[Dai87] W. Dai, M. Sato, E. S. Kuh. **A dynamic and efficient representation of building-block layout.** *Proc. IEEE* 24th *Design Automation Conf.*, pp 376-384, ACM 1987.

[Dai89] W. Dai, B. Eschermann, E. S. Kuh, M. Pedram. **Hierarchical placement and floorplanning in BEAR.** *IEEE Trans. Computer-Aided Design* 8(12), pp 1335-1349, IEEE 1989.

[Deny82] P. B. Denyer, D. Renshaw, N. Bergmann. **A silicon compiler for VLSI signal processors.** *ESSCIRC'82 Digest of technical papers*, pp 215-218, Brussels 1982.

[Deut76] D. N. Deutsch **A "dogleg" channel router.** *Proc. IEEE* 13th *Design Automation Conf.*, pp 425-431, ACM 1976.

[Gare77] M. R. Garey, D. S. Johnson. **The rectilinear Steiner tree problem is NP-Complete.** *SIAM J. Applied Mathematics* 32(4), pp 826-834, 1977.

[Groe89] P. Groeneveld. **On global wire ordering for macro-cell routing.** *Proc. IEEE* 26th *Design Automation Conf.*, pp 155-160, ACM 1989.

[Harw90] Harwell Laboratory. VLSI Design Section, B347.3, AEA Technology, Harwell, Didcot, Oxon OX11 ORA.

[Hash71] A. Hashimoto, J. Stevens. **Wire routing by optimizing channel assignment within large apertures.** *Proc. IEEE* 8th *Design Automation Workshop*, pp 155-169, ACM 1971.

[Hsu87] Y.C. Hsu, Y. Pan, W.J. Kubitz. **A path selection global router.** *Proc. IEEE* 24th *Design Automation Conf.*, pp 641-644, ACM 1987.

[Igus89] M. Igusa, M. Beardslee, A. Sangiovanni-Vincentelli. **ORCA A sea-of-gates place and route system.** *Proc. IEEE* 26th *Design Automation Conf.*, pp 122-127, ACM 1989.

[Joha79] D. Johannsen. **Bristle blocks: a silicon compiler.** *Proc. IEEE* 16th *Design Automation Conf.*, pp 310-313, ACM 1979

[Kirk83] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi. **Optimization by simulated annealing.** *Science* 220(4598), pp 671-680, 1983.

[Kort89] B. Korte, H.J. Prömel, A. Steger. **Combining partitioning and global routing in sea-of-cells design.** *Proc. IEEE ICCAD–89*, pp 98-101, IEEE 1989.

[Laut79] U. Lauther. **A min-cut placement algorithm for general cell assemblies based on a graph representation.** *Proc. IEEE* 16th *Design Automation Conf.*, pp 1-10, ACM 1979.

[Laut85] U. Lauther. **Channel routing in a general cell environment.** *Proc. VLSI 85*, pp 389-399, Ed. E. Hörbst, North-Holland 1986.

[Mead81] C.A. Mead. **VLSI and technological innovations.** *Proc. VLSI 81*, pp 3-11, Ed. J.P. Gray, Academic Press 1981.

[Metr53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller. **Equation of state calculations by fast computing machines.** *J. Chemical Physics* 21(6), pp 1088-1092, 1953.

[Mlyn86] D. A. Mlynski, C. Sung. **Layout compaction.** *Layout design and verification*, Chapter 6, Ed. T. Ohtsuki, Elsevier Science 1986.

[Moni85] L. Monier, J. Vuillemin. **Using silicon assemblers.** *Proc. VLSI 85*, pp 309-318, Ed. E. Hörbst, North-Holland 1986.

[Naha85] S. Nahar, S. Sahni, E. Shragowitz. **Experiments with simulated annealing.** *Proc. IEEE* 22nd *Design Automation Conf.*, pp 748-752, ACM 1988.

[Nils71] N. J. Nilsson. *Problem-solving methods in artificial intelligence*, Chapter 3, Mc Graw-Hill 1971.

[Oust84] J. K. Ousterhout. **Corner stitching: a data-structuring technique for VLSI layout tools.** *IEEE Trans. Computer-Aided Design* 3(1), pp 87-100, IEEE 1984.

[Sang87] A. Sangiovanni-Vincentelli. **Automatic layout of integrated circuits.** *Design systems for VLSI circuits*, pp 113-195, Ed. G. De Micheli, A. Sangiovanni-Vincentelli, P. Antognetti, Martinus Nijhoff 1987.

[Schl83] M. Schlag, Y. Liao, C. K. Wong. **An algorithm for optimal two-dimensional compaction of VLSI layouts.** *INTEGRATION, the VLSI Journal* 1, pp 179-209, Elsevier Science 1983.

[Sech86] C. Sechen, A. Sangiovanni-Vincentelli. **TimberWolf3.2: A new standard cell placement and global routing package.** *Proc. IEEE* 23rd *Design Automation Conf.*, pp 432-439, ACM 1986.

[Sech88] C. Sechen. **Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing.** *Proc. IEEE* 25th *Design Automation Conf.*, pp 73-80, ACM 1988.

[Sher89] A. T. Sherman. *VLSI placement and routing: the PI project.* Springer-Verlag 1989.

[Szep80] A. A. Szepieniec, R. H. J. M. Otten. **The genealogical approach to the layout problem.** *Proc. IEEE* 17th *Design Automation Conf.*, pp 535-542, ACM 1980.

[Szym85] T. G. Szymanski. **Dogleg channel routing is *NP*-Complete.** *IEEE Trans. Computer-Aided Design* 4(1), pp 31-41, IEEE 1985.

[The89] K. The, D. F. Wong, J. Cong. **Via minimization by layout modification.** *Proc. IEEE* 26th *Design Automation Conf.*, pp 799-802, ACM 1989.

[Vecc83] M. P. Vecchi, S. Kirkpatrick. **Global wiring by simulated annealing.** *IEEE Trans. Computer-Aided Design* 2(4), pp 215-222, IEEE 1983.