Copyright © 1989, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

AUTOMATIC LAYOUT OF SILICON-ON-SILICON HYBRID PACKAGES

by

Massoud Pedram

Memorandum No. UCB/ERL M89/80

27 June 1989

AUTOMATIC LAYOUT OF SILICON-ON-SILICON HYBRID PACKAGES

by

Massoud Pedram

Memorandum No. UCB/ERL M89/80

27 June 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

AUTOMATIC LAYOUT OF SILICON-ON-SILICON HYBRID PACKAGES

by

Massoud Pedram

Memorandum No. UCB/ERL M89/80

27 June 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Contents

Table of Contents i

List of Figures iii

1 Introduction 2

2 IC Packaging Technology Overview 4
- 2.1 Conventional Packaging Technology 4
- 2.2 Multi-chip Hybrid Packaging Technology 5

3 General Cell Layout Overview 6
- 3.1 General Cell Placement 6
  - 3.1.1 Constructive Placement 7
  - 3.1.2 Iterative Placement 8
  - 3.1.3 Placement Summary 8
- 3.2 General Cell Global Routing 8
  - 3.2.1 Sequential Global Routing 9
  - 3.2.2 Iterative Global Routing 9
  - 3.2.3 Linear Programming Global Routing 9
  - 3.2.4 Hierarchical Global Routing 10
  - 3.2.5 Global Routing Summary 10
- 3.3 Channel Routing 10
  - 3.3.1 Constraint-Based Channel Routing 11
  - 3.3.2 Greedy Channel Routing 12
  - 3.3.3 Hierarchical Channel Routing 13
  - 3.3.4 Channel Routing Summary 13

4 The Design Automation Environment 14
- 4.1 Data Representation 14
- 4.2 Layout Synthesis 15

5 The Silicon-On-Silicon Hybrid Technology 17
- 5.1 Fabrication Issues 17
- 5.2 Performance Issues 18
  - 5.2.1 Packaging Efficiency 18
  - 5.2.2 Interconnect Density 19
  - 5.2.3 Package Delay 19
  - 5.2.4 Package Power Dissipation 20
  - 5.2.5 Package Power Density 20

6 Hybrid Layout 21
6.1 Die Mount Pattern Generation 22
6.2 Hybrid Placement and Routing 22
6.2.1 The Topological Model 23
6.2.2 Placement 24
6.2.3 Global Routing 25
6.2.4 Detailed Routing 26
6.3 Hybrid Pad Frames 27

7 Results 29
7.1 Summary 30

Bibliography 32

List of Figures

1 This hybrid layout was automatically generated from the schematics shown in Figures 2, 3, and 4. This circuit is the IO subsystem of a high performance workstation and contains 8 VLSI circuits. 35
2 This top level schematic defines the hybrid pad ring and the pad ring routing requirements. The layout (shown in Figure 1) corresponding to the blue and yellow portions of Figure 5 is generated from this schematic. 36
3 This second level schematic defines the main placement and routing problem (called the inner cell) for the hybrid. 37
4 This example shows the icon (interface summary), the schematic and the layout for one of the integrated circuit die mounts. The die mount layout was generated from the schematic and the layout of the integrated circuit. 38
5 This shows the areas of the layout of Figure 1 that were generated by the three layout generation procedures described in the text. The blue and yellow portions represent the top level layout. The green die mounts were generated by the lowest level layout procedure. The white area represents the routing of the inner cell. 39
6 This shows the detailed construction of the die mount patterns. A typical value of w is 0.1 mm. Taps to the Vdd and Ground planes are automatically inserted as required. Signal pins are touch to the periphery of the pattern for routing. 40
7 This Channel Intersection Graph corresponds to the layout of Figure 1. The red lines represent the routing channels. 41
8 This horizontal channel position graph corresponds to the layout of Figure 1 and is used to compute horizontal dimensions of the layout. 42
9 The global route improvement algorithm reroutes nets to reduce layout area. Topological model procedure names are shown in parenthesis. 43

Abstract

In high speed computer systems, performance and density limits are being set more by interconnect and packaging constraints than by transistor switching speeds. The most severe limitation comes from the single-chip packages that carry the VLSI circuits. Multi-chip, silicon-on-silicon hybrid packages can significantly improve performance by eliminating this level of packaging. A system has been developed to generate hybrid layouts automatically given a schematic description of the required interconnection and layouts of the VLSI circuits to be included. This report describes the hybrid technology, the design automation system foundation, and the hybrid layout system. The layout system is implemented as three layout generation procedures: die mount pattern generation, a general cell-like placement and routing procedure, and a pad frame generator. State-of-the-art algorithms are described. These algorithms are combined to form an automatic layout system for the hybrid technology. System operation is illustrated through a comprehensive example: a high performance workstation IO subsystem containing 8 VLSI circuits implemented on a 2.6" by 3.2" hybrid package. This layout method, in combination with the improved technology, produces layouts that are 5 to 8 times denser than the same circuit implemented with single-chip packages on a printed circuit board. Simulations show that clock speeds can be increased by a factor of two.

Chapter 1

Introduction

In high performance computer systems, the contribution of packaging to machine performance is becoming increasingly important. System performance and density limits are now being set by interconnect and packaging constraints. Signal delays due to transistor switching speeds have decreased dramatically while delays due to packaging have decreased only slightly. Demand for growing IO pin counts and greater packing densities along with increased performance requirements are making traditional packaging methods (single-chip packages on glass-epoxy printed circuit boards) inadequate. The most severe limitation comes from single-chip packages. Thus, multi-chip packages can significantly improve system performance by eliminating this level of packaging. Multi-chip packages can both increase system density and reduce signal delays due to interconnect and packaging.

At Xerox PARC, researchers have developed a high density, multilayer interconnect technology using thin film metallization and polyimide dielectric on a wafer-size silicon substrate. Silicon integrated circuit (IC) fabrication equipment is used to lay down metal and dielectric layers on a silicon wafer. Eight to twenty (unpackaged) VLSI circuits can be mounted and interconnected on a rectangular area inscribed on a silicon wafer. A single cover provides environmental protection for all of the chips. The combination of multiple interconnection layers, small features, ease of testing, reliability, and high performance makes this silicon-on-silicon hybrid technology very attractive.

Automatic layout synthesis at this packaging level is fully as important as the automatic layout synthesis of VLSI circuits. Layout synthesis for these hybrid packages has the characteristics of both printed circuit boards (PCBs) and silicon ICs. However, existing layout systems for neither printed circuit boards nor integrated circuits were able to lay out these circuits. PCB layout systems were unable to handle the high pin densities and the complexity of the circuits while IC layout systems were unable to generate layouts in the exact geometric shapes required. For these reasons, and because of the similarity of this layout problem has to general cell layout, we decided to enhance our existing

general cell layout system to produce silicon-on-silicon hybrid layouts.

The combination of this new fabrication technology, a relatively rich design automation environment and an existing general cell layout system allowed us to provide a new capability for silicon-on-silicon hybrid layout. Chapter 2 provides an overview of conventional and multi-chip hybrid packaging technologies. Chapter 3 gives an overview of placement and routing techniques for general cells. The existing design automation system provides a foundation on which to build automatic layout tools and greatly influences the design of the tools which the foundation supports. This foundation is described in Chapter 4.

Chapter 5 discusses the aspects of hybrid technology that are important for automatic layout. Since the silicon-on-silicon packaging technology is relatively new, it requires more discussion than would normally appear. Chapter 6 describes the automatic layout system for silicon-on-silicon hybrid circuits. The system is composed of three layout procedures. The lowest level layout procedure produces die mount patterns (the leaf cells for this system) for the ICs. Given the physical design of an IC (especially the bonding pad locations) and the binding of the hybrid interconnections to the bonding pads, patterns for die attachment and wire bonds are automatically created. The intermediate level layout procedure places and interconnects the die mount patterns. General cell layout techniques are used; this task is greatly simplified by exploiting topological domain algorithms. The highest level layout procedure connects the main layout (defined by the intermediate level layout procedure and called the *inner* object) to the hybrid bonding pads. A minimal amount of routing overhead is required to translate from the topological domain layout to the geometric domain of the pads. The IO subsystem from a high performance workstation (composed of eight VLSI circuits) is used as an example through out the report. Results from the layout of this circuit on a hybrid are presented in Chapter 7.

Chapter 2

IC Packaging Technology Overview

With the introduction of large-scale and very-large-scale integrated-circuit technology, the art of squeezing in the most chips per square inch - commonly called high-density packaging - is becoming more important. Familiar high-density schemes, such as packing dual in-line packages (DIPs) onto a single two-sided printed-circuit board, are inadequate: the DIPs required by the larger and more complex ICs take up too much space and their internal line resistance and capacitance limit circuit performance. To meet demands for higher density while maintaining performance and reliability, manufacturers of electronic equipment have tried a variety of alternatives: ceramic chip-carriers on multilayer PCBs or multilayer ceramic substrates, film chip-carriers on multilayer ceramic substrates, and bare chips wire bonded to multi-layer hybrids. In the following sections, we will review conventional and multi-chip hybrid packaging technologies.

2.1 Conventional Packaging Technology

Electronic systems typically consist of printed circuit boards (PCBs) connected by cables or backplane boards. ICs are connected to a PCB through packages. Packages provide the connection between ICs and the PCBs as well as environmental protection and a thermal path from the chips to the environment. PCBs have one or more layers of metal interconnections on a substrate [Ein77]. Except at *via* locations, wiring on one layer is insulated from wiring on other layers. This packaging scheme causes several problems. Over the past 30 years, printed circuit technology has developed less rapidly than IC technology and, at present, PCB feature sizes are about 50 times larger than those on ICs [Bru88]. Consequently, a package substantially larger than the IC, is needed to interface the IC to the board. Such a large package introduces parasitics and degrades system performance.

A more recent packaging technique mounts chip-carriers on ceramic mother-boards. The

ceramic chip carrier is a small, leadless, square package with a gold-plated cavity. Lead metalization is connected internally to gold solder pads on the bottom face of the unit. This method allows pretesting and burning in of the chips in the carriers and protects the chips from damage while handling. These advantages contribute to a higher yield than the bare-chip hybrid method described in next section. However, the chip-carrier/mother-board technique does not have a high packaging density. Chips may be mass-bonded to tape or film carriers to increase the packaging density [Lym80a].

2.2 Multi-chip Hybrid Packaging Technology

Multi-chip hybrid packaging has been used for some time in high performance systems. With this technology, several ICs are mounted on a substrate, typically ceramic. Multi-chip packages can be further divided into three categories depending on substrate type and fabrication technology used: *thin film*, *thick film*, and *silicon-on-silicon*. Thick-film hybrids are built by screening and firing alternate conductive, resistive and dielectric pastes onto a ceramic substrate. Thin film hybrids have resistors and conductors deposited in a vacuum onto a ceramic substrate [Lym80b]. Silicon-on-silicon hybrids are built by mounting silicon integrated circuits directly on a wafer sized silicon substrate. Interconnection wiring is deposited on the substrate using conventional silicon metallization processing. This technique allows fine (very narrow) lines and high packing densities compared to other hybrid or board-level packing technologies. It saves space by using the smallest package available - the IC chip itself. [Joh86] provides a review of the process.

The basic advantages of multi-chip hybrid technology are as follows: (1) Small size and weight; (2) Wide operating frequency range because of short lines and low capacitances; (3) Increased reliability because of a reduction in interconnections; (4) Functional trimmability; (5) Efficient thermal dissipation; (5) Economical small production runs; (6) Ease of achieving close component tolerances. Multi-chip hybrids are, however, more expensive to make and more difficult to test. Due to their packaging densities, multi-chip hybrids generate more heat than single-chip packages and require more elaborate heat removal techniques. Hybrid packaging imposes special demands on layout systems. The pin count and number of nets typically needed in a hybrid package are much higher than those for a PCB. The layout system should honor tight dimensional constraints and exact timing requirements. PCB or general cell design aids are, therefore, not adequate for hybrid layouts. Currently, there are no commercial design aids available for hybrid packages. A more detailed description and performance analysis of silicon-on-silicon hybrid packages is presented in Chapter 5.

Chapter 3

General Cell Layout Overview

Circuit layout is a crucial part of VLSI design. It deals with the physical design of chips to satisfy the cell and circuit specifications subject to constraints imposed by a given fabrication technology. The task of generating layout for a VLSI design is very complex. Therefore, the layout problem is often divided into subproblems of logic partitioning, chip floorplanning, cell placement, global and detailed routing. Logic partitioning and chip floorplanning fall outside the scope of this report. We, therefore, focus on the placement and routing stages of the layout process. A popular integrated circuit design style is *general cell* style. In the following, we review placement and routing techniques for this style.

3.1 General Cell Placement

Classical placement algorithms model cells as "point objects." This implies that the placement quality is a function only of the interconnections and is not affected by the shapes of cells. More recent algorithms, however, take cell shapes into account and therefore provide more accurate placement models. Standard cell layout systems arranges the cells in rows; in effect, the cells have only one important dimension. The cells are designed to have a uniform height; widths vary with the complexity of the cell. In this case, an accurate placement objective function must include the width of the rows and the heights of the channels. Simple wire lengths are no longer sufficient.

The goal of general cell placement is to find position and orientation for all cells of a circuit (subject to given constraints) such that (post routing) area of a rectangle enclosing all cells is minimized. A concurrent goal is to minimize the total wire length. General cell layout systems must take into consideration both dimensions of the cells. In this case, layout area is an accurate objective function, but it is very difficult to compute because of the complex interactions among the cells and

the routing areas.

Finding the optimum solution to the general cell placement problem is NP-complete. Therefore, various heuristic techniques have been proposed to provide a "good" solution. These techniques are divided into two categories: *constructive* placement and *iterative* placement. (See [Pre86].)

3.1.1 Constructive Placement

The constructive placement algorithms share the characteristic that their input is a partial placement and their output is a complete placement. These algorithms can be divided into the following classes: *clustering-based*, *partitioning-based*, and *global* placement techniques.

Clustering-based placement techniques are "bottom-up" methods that consider the most detailed level of abstraction. These techniques are further divided into *cluster growth* and *connectivity clustering*. Cluster growth algorithms operate by selecting unplaced cells and adding them to a partial placement [Han72, Wim88]. A major advantage of these algorithms is the considerably lower computational effort required to execute them compared to that required by other placement techniques. These algorithms, however, usually produce poor results because placement decisions must be made with local information which is incomplete.

Connectivity clustering algorithms initially derive a physical hierarchy in the form of a tree. This tree is generated by modeling cells as point objects and recursively grouping highly connected cells (or cell clusters) together. Clusters at the bottom level of the tree are then placed in such a way that the area of the bounding box enclosing cells of the cluster is minimized. This process is repeated by placing clusters of clusters and so on until root is placed and the process is terminated [Ott83]. [1] In the absence of user constraints (goals) such as chip aspect ratio and external IO location constraints, these algorithms may produce good layouts. However, a major shortcoming of these algorithms is that placement decisions for a node of the tree are entirely based on information in the subtree rooted by that node. This incomplete local information is not adequate to produce routable placements.

Partitioning-based placement is a "top-down" method that divides cells into two or more partitions (blocks) . The process of dividing blocks into smaller blocks continues until the number of cells per block is small [Bre77, Lau79, Ued85, LaP86]. Partitioning-based approaches consider higher level of abstraction before they consider more detailed levels and tend to avoid heavy wiring congestion typically found in the center of the layout surface. These algorithms are $goal-oriented$ and produce very routable placements. However, they have difficulty when cells are constrained to fixed positions and when partitions of grossly unequal areas are produced. Placement decisions made at higher levels

[1] Otten describes a floorplanning scheme based on combining shape functions for the cells. This approach is easily applicable to the placement problem if shapes of all cells are fixed.

of the hierarchy may suffer from lack of detailed information.

Global placement techniques consider all of the interconnections in parallel and consider all of the cells simultaneously to find a global solution. Quadratic assignment [Han72, Hal70], eigenvector decomposition [Fra86], resistive network technique [Che84], force directed placement [Qui75] and analytical methods [Sha85] are some of the global techniques. More data is needed to assess the relative merits of global placement techniques versus other placement techniques.

3.1.2 Iterative Placement

The goal of iterative placement is to transform a complete placement into an improved, complete placement. This process is repeated until some stopping criterion is met. Within one iteration cells are selected and moved to alternate locations. If the resulting configuration is better than the previous one, the new configuration is retained. Otherwise, the previous configuration is restored [Pre79a]. Simulated annealing technique [Kir83, Sec88] accepts even some of the trials that increase the score in order to avoid being trapped at a local minimum. Simulated annealing placement usually produces very good results at the expense of very long run times.

3.1.3 Placement Summary

Each of the constructive placement algorithms described has some desirable and some undesirable characteristics. Consequently, combinations of approaches have been proposed. Wipfler et al., [Wip82] reports a combined force-directed and min-cut approach. Akers [Ake81] reports the use of linear assignment combined with cluster growth. Eschermann et al., [Esc88] uses a combination of top-down partitioning and bottom-up geometric shape propagation techniques. Constructive placement algorithms are usually followed by iterative improvement algorithms.

3.2 General Cell Global Routing

The goal of a global router is to decompose a large routing problem into small, manageable problems in such a way that detailed routing can be completed. This decomposition is done by assigning each net a set of routing regions (channels, switchboxes or river-route areas). The objective is to plan the routing globally to reduce chip area, shorten connection length, improve routability, and balance congestion across the layout surface. [2]

[2] In the following, we assume that two layers are available for interconnection and that over-the-cell routing is not allowed.

Crucial for global routing is an appropriate model for the "routing space". In general cell layout style the routing regions are usually modeled by a *channel intersection graph* in which each node represents a channel intersection and each edge represents a channel or channel subsection [Pre79b]. Associated with edges is the geometrical length of the channel and its capacity. The global routing problem is then to generate for each net a tree on the graph in a way that the total length of wiring and the overflow of the channel capacity is minimized.

3.2.1 Sequential Global Routing

In general, it is not possible to route all nets with shortest paths because various nets compete with each other for paths to be used. One approach to solve this problem is to route all nets independently and then reroute some of the nets to eliminate overflows [Pri57, Tin83]. Another approach often used for this problem is to define a cost function on channel edges comprising geometrical length and actual wiring density within the channel and then to generate for each net a tree on the graph with the least total edge weight which connects pins of the net [Kan76, Fuj86]. A drawback of this sequential approach is that nets are not truly considered concurrently; nets processed early don't "see" the final channel densities. Clearly, the routing order becomes a crucial factor.

3.2.2 Iterative Global Routing

Iterative rerouting alleviates the net ordering problem. The basic idea is to start with a feasible global routing solution and improve it by repeatedly selecting connections, removing them, and reconnecting them by some different path. Simulated annealing can be applied as well [Vec83, Sec88]. By considering L-shaped and Z-shaped paths for each two-point connection or by finding M shortest routes for each net, the annealing technique can be used to distribute the connections uniformly.

3.2.3 Linear Programming Global Routing

Karp et.al., presented an integer linear programming formulation that considers interacting nets concurrently [Kar83]. Their algorithm operates on a channel graph. It is required that no more than ω_i of the nets path through an edge e_i. Associated with each pair of pins is a set of paths, one of which must be used to join the two pins. The global routing problem on this channel graph is then specified as an optimization problem which has the form of an integer linear program. It is shown that linear programming techniques can be used to solve the optimization problem since the rounded solution is the desired integer-valued solution to the integer problem. The simplifying assumptions of

the model are that the net list consists of two-pin nets only, that the shape of routes is limited to few possible patterns, and that the channel graph is unchanged during the global routing.

3.2.4 Hierarchical Global Routing

Another approach which avoids net ordering and sequential routing is hierarchical decomposition [Mar86]. The algorithm is based on a top-down hierarchical scheme. [3] In each step of the hierarchy, the current routing problem is partitioned (cut) into two subproblems. Linear assignment technique is used to assign nets to the boundaries which are included into the most recent cutline. The algorithm has the advantage that it considers all nets concurrently and can be applied to arbitrary (including non-slicing) circuit topologies and arbitrary shape cells. The disadvantage is that decisions made at each level of hierarchy are made in a greedy manner, i.e. local decisions which are "optimal" at a particular level of hierarchy can be detrimental at the subsequent levels of the hierarchy.

3.2.5 Global Routing Summary

In the case of layout styles with regular structures such as gate array style, it is possible to translate the global routing problem into a mathematical formulation rather precisely. [4] Therefore, there exist many good algorithms and good results. Global routing for general cells is more involved due to the following reasons: (1) Cell placement does not have a regular geometry, (2) There are various ways of defining the routing regions. (3) Circuit topology may change during the global routing. These features have made global routing for general cells less precise.

3.3 Channel Routing

Most modern, automatic systems for VLSI circuit layout are based on channel routing. Channel routing algorithms are used as detailed routers within standard cell, gate array, and general cell automatic layout systems. The wide range of uses and the importance of detailed routing in automatic layout systems have inspired many good channel routing algorithms. Taken as a whole, these algorithms provide a wide range of capabilities. However, many existing algorithms were (implicitly) developed with standard cell and gate array requirements in mind. No existing algorithm is sufficiently general to be used as the single detailed router in modern, VLSI design automation systems. Most of the existing algorithms suffer from one or more of the following drawbacks: (1) All wire segments

[3] See [Bur83, Mar84] for a similar technique applied to gate array layouts.

[4] Theoretically, all combinatorial problems can be formulated as integer programming problems. However, problem formulation may require an exponential number of equations.

must have the same width and that width must be uniform along the length. The terminals to be connected must have the same widths and be placed on a uniform grid. (2) The channel boundaries must be uniform. Irregular or rectilinear boundaries are not permitted. (3) Completion of all of the interconnections cannot be guaranteed in all cases. This condition usually occurs when cyclic vertical constraints are encountered. When cyclic constraints can be processed, they are addressed implicitly or iteratively; this approach is too inefficient when many complex, cyclic vertical constraints exist.

This simplified model is not appropriate in modern, general cell design environments. In very large circuits, complete automation is mandatory. Neither the designer nor the other parts of the design automation system is able to alter the arrangement of terminals on the sides of the channels to accommodate the router. Typically, power and clock busses are wider than signal wires and must be routed in the channels. In larger circuits, power busses must be tapered (width varies along the length.) Channels with extreme variations along the boundary are the norm. Terminals can have different widths and spacings because the cells along the channel boundary may be constructed using radically different design styles.

Channel routing algorithms are very specialized; they can operate only on channels and consider only restricted wiring patterns as potential solutions. However, these restrictions allow the algorithms to consider all the interconnections simultaneously. Background on channel routing is available in [Bur86, Lor88]. Channel routing algorithms can be divided into three categories: *constraint–based*, *greedy*, and *hierarchical*. This section describes these categories in sufficient detail to understand their strengths and limitations for general cell, VLSI design.

3.3.1 Constraint-Based Channel Routing

Constraint-based channel routing algorithms concentrate on assigning trunk segments to tracks. The minimal requirements to prevent overlap of branch layer routing belonging to different nets are captured in a vertical constraint graph. In a constraint graph the nodes represent trunk segments and the directed arcs represent constraints caused by branch layer wires. Thus, the constraint graph is a partial order of trunk segments. A complete order can be derived either by evaluating alternatives using geometric domain checks or by topological operations.

A complete ordering of the trunk position can be determined by evaluating the alternative assignments of trunks (within the limits allowed by the vertical constraint graph) to tracks. The evaluations are performed using geometric design rule checks. Deutsch [Deu76] describes a (terminal) dogleg channel routing algorithm where trunk segments between two adjacent terminals on a net are restricted to one track. Doglegs are allowed only at terminal positions. More recently, a different approach to handling constraints was reported [Ree85]. This router assigns trunks to tracks but allows

some vertical constraint violations to occur. A pattern router is then used to fix up the branch layer routing.

A complete ordering among the trunk segments can be derived from the partial order of the constraint graph by topological operations. [Yos82] defines an interval graph and merging operations on the constraint graph and the internal graph to determine which trunks are to be put on the same tracks. [Che86] defines an augmented constraint graph that contains both vertical (directed) constraints (to capture the branch layer constraints) and horizontal (undirected) constraints (to capture the trunk layer constraints). Operations are defined to assign directions to the horizontal constraints. Once directions are defined for the horizontal constraints, compaction techniques are used to determine the positions of the trunk segments.

As a group, these algorithms can route wires and terminals with different widths and spacings. These algorithms are easily adaptable to rectilinear channels, and can be extended to handle different width trunk segments. Unfortunately, the ability to guarantee that all interconnections can be routed remains problematic. Most of the algorithms discussed assume a vertical constraint graph that is free of cycles [Deu76, Yos82, Che86]. Authors assume that the terminals can be arranged by the designer or an automatic placement program [Per76] to provide a cycle-free problem instance. Several authors [Deu76, Sat80, Hig80] propose using non-terminal doglegs to break vertical constraint cycles but provide no algorithm for determining the positions. [Kel86] provides an algorithm but for a simplified channel model. Completion guarantees are not addressed. YACR2 [Ree85] can route channels with cyclic vertical constraints but does so inefficiently. It does not look for cycles but discovers them one at a time and must route the entire channel between iterations. The complicated heuristics are difficult to apply in a general routing model.

3.3.2 Greedy Channel Routing

The greedy channel routing algorithm determines positions for wiring segments in a left-to-right, column-by-column scan [Riv82]. The wiring within each column is completely specified before the next column is processed. In each column, the router applies a set of simple, "greedy" heuristics to maximize the utility of the wiring produced. Instead of performing doglegs at terminal locations, the greedy router allows any horizontal connection to change tracks, which causes it to insert a large number of vias. The algorithm specifies five greedy rules to assign wiring segments to positions.

The greedy algorithm typically produces very good resulting routes, and always finds a solution. Because it contains fewer assumptions about the topology of the connections, it is more flexible in the placement of doglegs than the left-edge approaches. It does not impose the restriction of one trunk section per net or one trunk section per terminal pair. Unfortunately, the rules require all

of the terminals to be on a grid. If terminals were allowed to be off-grid, there would be an interaction between columns, and the column-by-column approach would be infeasible. Wide power terminals could interact with many signal terminals.

3.3.3 Hierarchical Channel Routing

The third category of channel routing uses successive refinement in a hierarchical manner [Bur83]. First, the channel is split along the trunk direction into two routing regions and wiring segments are assigned to one of the two regions. This gives rise to a Two-By-N routing problem (with N columns or terminal positions). Both integer and dynamic programming solutions are described. Each routing region is then split into two more regions and the process continues until positions have been assigned for all wire segments. Additional tracks are added only when it is impossible to route a connection without violating the current channels' capacities.

The decomposition of the channel allows the generation of very robust Steiner points during the routing and tends to produce results very close to channel density. This algorithm produces very good routing solutions for gate array and standard cell problems. However, the approach is not applicable for general cell layout systems because uniformity of the routing surface is a critical assumption. These integer and dynamic programming solutions require that terminals be on a grid and that wires have the same width.

3.3.4 Channel Routing Summary

Constraint-based algorithms come the closest to satisfying all of the requirements for general cell, VLSI design. These algorithms have been (or can be) extended to meet the variable geometry requirements. Furthermore, they can complete all interconnections when enhanced with an algorithm to resolve cyclic vertical constraints. The resulting acyclic graph can then be input to any constraint-based channel routing algorithm.

Chapter 4

The Design Automation Environment

The Xerox PARC design automation system provides both the user interface and the programming foundation for the hybrid layout system described in this report. The characteristics of this design system had a large impact on the design and implementation of the hybrid layout system and thus, merits a separate discussion. Emphasis is placed on the data representation and layout synthesis portions of the design system.

4.1 Data Representation

The integrated design automation system at PARC uses an object-oriented paradigm [Ser86]. A small number of data types is used to represent an *annotated* net list. These types have property lists to allow arbitrary annotation. Instead of a fixed set of device types, a class mechanism facilitates new abstractions. Procedural translation from one abstraction to another allows new abstractions to be added easily [Bar88a].

The design automation system is based on the use of a single common hierarchical data structure to represent the complete design. This common data structure which is derived from extracted parametrized schematics and code is then used by all tools including those for simulation and layout synthesis. Although most aspects of a design are naturally and concisely described using graphic parametrized gates, registers, icons and so on, some features are much more readily understood when written in a high level language. The system enables mixed mode representation by allowing graphic icons to return the result of a procedure call and by providing a procedure which returns the result of extracting a graphic schematic. Both graphic extractions and procedure calls receive arguments or parameters from the context of the extraction or call.

The basic data structure used by the design automation system is a directed acyclic graph

with cells represented by nodes and instances by arcs. Each *cell type* is a member of a *class*. Each instance, cell type and class can have arbitrary properties associated with it to guide the behavior of simulation, layout or analysis tools. An important aspect of the system is the ease with which new cell type classes may be added without requiring updates to the existing tools. This is done by requiring programs to understand a small basic subset of cell type classes but requiring new classes to provide a *recast* procedure which will convert an instance of that class into a more primitive cell type.

The user's primary interface with the design system is via an interactive graphics editor through which the user describes his or her circuit as an annotated schematic [Bar88b]. A schematic describes the hierarchical structure of a cell (expressed in terms of subcells). The user is free to intermix graphical and textual (procedural) specifications. These intermixed specifications can be viewed as net list generators [Bar88a]. When a net list generator is evaluated (code is executed and/or a schematic is extracted) it returns the net list of a primitive device or merges results from subsidiary generators into a net list and returns it. Thus, evaluation of a single net list generator may cause many levels of schematic extraction and code execution.

4.2 Layout Synthesis

Layout generation is strictly a batch process. The layout of a cell is generated recursively from the layout of its subcells. These concepts are illustrated by an example: the layout shown in Figure 1 was generated entirely from the schematics shown in Figures 2, 3, and 4.

When a *layout generator* [Bar88c] is invoked on (the schematic of) a cell it returns the *object* (geometric layout) of that cell. Invoking the layout procedure at the top of the design hierarchy starts a chain of layout calls since generating layout at one level usually requires recursively obtaining the layout of the cell's children. For efficiency, a cache is maintained so successive invocations of a layout generator with the same cell will return the same layout. Each cell in a schematic for which layout is to be generated is annotated with a layout key. Cells without a layout key are *flattened.* (The cell's net list is merged with that of its parent.) Layout keys have *AttributeProcs*, *LayoutProcs* and *DecorateProcs* associated with them. The *AttributeProc* defines the user interface of the layout generator by gathering parameters from the graphical information in the schematic. For example, an *AttributeProc* could use schematic decorations and properties to determine relative placement of subcells or the side on which a port becomes public.

The layout generator next invokes the *LayoutProc* to construct the geometric object based on the net list and parameters obtained by the *AttributeProc*. Generally, a *LayoutProc* lays out the subcells by invoking their layout generation procedures. The subcell layouts can then be used as part

of the layout of the subject cell.

As the final step, a layout generation procedure invokes the *DecorateProc* to cache the object as a property on the cell and decorate the interface ports with graphical layout properties. These layout properties describe the position, size, and layer of public pins at the bounding box of the cell. This step also checks the consistency of the geometric layout and the cell definition. For example, every public port on the cell must have a corresponding geometric pin. At each level of the hierarchy, it is only necessary that a layout procedure understand the decorations of the cell's children.

The current set of layout generation procedures varies widely in complexity. The most primitive procedures just return a copy of some predefined mask object (for example, a transistor, gate or flip-flop). Higher level layout procedures transform an object's orientation or abut objects together into rows, columns, or arrays. The next level of complexity for layout procedures comes with the addition of routing. Cells can be stacked with interspersed routing channels or arrayed together with interspersed switchboxes. A standard cell place and route procedure [Con88] is available, as well as a data path generator that allows multiple interleaving of individual paths. There is also a procedure to route an object within a pad ring [Pre89] and one to do a general cell place and route.

The hybrid layout system described in the Chapter 6 is implemented as three layout generation procedures. The flexibility of this layout method made it quite easy to extend the notion of layout from an IC technology (2 metal layer CMOS) to a hybrid technology (4 layers of metal) while maintaining a common source schematic for all levels of simulation.

Chapter 5

The Silicon-On-Silicon Hybrid Technology

Silicon-on-silicon hybrid technology [Joh86, Bru89] can be viewed as a step toward wafer scale integration. In the next section, we will discuss the fabrication characteristics of the silicon-on-silicon packages.

5.1 Fabrication Issues

This technology uses a wafer-size interconnect pattern to provide connections among ICs. Standard metallization methods are used to deposit the interconnects over a silicon wafer. Separately processed and tested integrated circuits are glued onto the silicon substrate at the appropriate places. Wire bonds connect the integrated circuit bonding pads and the appropriate pads on the hybrid. A single cover protects the entire hybrid package. Bonding pads around the periphery connect to a PCB through TAB (tape automated bonding) or a specially constructed socket.

Fabrication of silicon-on-silicon hybrids involves four levels of metallization separated by a polyimide dielectric. The lowest metal level forms a ground plane and covers the entire surface. V_{dd} is supplied through the second metal layer. This V_{dd} layer covers the entire surface except under the die mount patterns. The top two metal layers are used for signal interconnection. The signal traces have 25 mm widths and spacings. Via-to-line spacing is also 25 mm. A single 2.6" by 3.2" package can be fabricated from a 5" wafer. Such a package can hold 8 to 20 VLSI circuits and allows 420 IO pins around the periphery of the hybrid with 635 mm (25 mils) pad-to-pad spacing. Figure 1 shows an example layout on a hybrid.

Hybrid packaging can greatly increase performance of computer systems, but imposes special

demands on layout systems. The pin count and number of nets typically needed in a hybrid package are much higher than those for a PCB. The layout system should honor tight dimensional and timing constraints. Next section describes the advantages of the hybrid packaging over traditional packaging techniques.

5.2 Performance Issues

Common figures of merit for different packaging approaches are packaging efficiency, interconnection density, package delay, power dissipation, power density, and power-delay product. In the following sections, we compare the silicon-on-silicon hybrid technology and other packaging using these figures of merit.

5.2.1 Packaging Efficiency

One goal of improved packaging is to fit more performance into the same volume [Bog87]. Silicon-on-silicon hybrids achieve this improvement through higher density (more integrated circuits for a given area), increased clock frequency (through lower interconnect capacitance and higher phase velocity media), and increased parallelism (more circuits per unit time are accessible [Bal85]). Traditional computer system packaging uses single-chip packages mounted on glass-epoxy PCBs. The PCBs are interconnected using backplanes (which often use PCB technology). The most severe limitation on system performance results from the use of single-chip packages.

As the number of IO pins per chip increases, the bond length (and hence the inductance) increases. Furthermore, single-chip packages cover correspondingly more area compared to the chip area. The geometric effects can be measured by *packaging efficiency*, ϵ:

$$\epsilon = \frac{Area_{chip}}{Area_{package}}.$$

Consider a chip carrier package with an IO pin pitch of $P_{package}$. Let the chip within the package have a bonding pad pitch of P_{chip}. In a pad limited package

$$\epsilon = (\frac{P_{chip}}{P_{package}})^2.$$

If the IC bonding pad pitch is 180 mm and the package IO pitch is 25 mils (635 mm), then ϵ = 7.8%. [1]

[1] Package dimensions are typically given in English units while IC dimensions are typically given in metric units. Silicon-on-silicon hybrids must be discussed in both worlds. We chose to follow convention at the expense of forcing the reader to deal with two systems of units. We provide metric translations of English units where appropriate.

In theory, pin grid arrays (PGAs) offer a high packaging efficiency for large IO pin counts

$$\epsilon = MIN[1, n(\frac{P_{chip}}{4P_{package}})^2]$$

(where n is the number of IO pins), but this efficiency is not achieved for today's chips. For example, a 400 pin PGA with $P_{package} = 0.1$ inches (2.54 mm) and $P_{chip} = 180$ mm has a packaging efficiency of 12%.

In contrast, the multichip hybrid packaging efficiency for ICs mounted on *die mount patterns* approaches 100% because $P_{chip} = P_{package}$. (The pad pitch of the die mount patterns are the same as the IC bonding pad pitch.) This high efficiency supports the claim that a level of packaging is skipped by silicon-on-silicon hybrid packaging.

5.2.2 Interconnect Density

Another measure of the quality of a packaging technology is *interconnect density*, δ

$$\delta = \frac{M}{P_{trace}}$$

where there are M layers of interconnection with pitch P_{trace}. When P_{trace} is in inches, d gives the maximum inches of trace per square inch of surface area. A high density PCB may have 10 signal layers with a trace pitch of 25 mils (635 mm). This gives $\delta = 400$ inches of trace per square inch. The silicon-on-silicon hybrid technology with 2 signal layers and a pitch of 50 mm can have 1016 inches of trace per square inch of area.

5.2.3 Package Delay

Package delay is given by

$$T_{package} = T_{buffer} + 3RC$$

where T_{buffer} is the output buffer delay, R is the on-resistance of the output driver, and C is the load capacitance. In silicon-on-silicon hybrid technology, smaller line capacitances and shorter IC-to-IC line lengths result in smaller C. In addition, because of thin-film lithography, the R can be higher, and therefore, the silicon area of the output drivers can be made smaller, allowing a larger number of IO buffers on an IC and reducing T_{buffer}. Smaller buffer sizes further reduce C because it decreases the capacitance of the driver buffers. Consequently, the hybrid package delay is small compared to other packaging approaches.

5.2.4 Package Power Dissipation

The *package power dissipation* is the sum of power dissipations of the ICs. The IC power dissipation is given by

$$P_{IC} = P_{internal} + nfCV^2$$

where n is the number of output buffers, f is the average operating frequency, C is the average load capacitance, and V is the output voltage swing. Design tradeoffs allow the package power dissipation to be either lower or higher for hybrids (due to higher number of output pins per IC, lower capacitance and higher operating frequency). In addition, the voltage swing at the output buffers may be reduced compared to other packaging approaches. All other factors being equal, P_{IC} will be smaller for ICs on a hybrid package due to smaller output capacitances.

5.2.5 Package Power Density

Package power density (power dissipation per unit package area) is larger for silicon-on-silicon hybrids. The IC long term reliability depends on its operating temperature. A target for commercial operation is 60-80 degrees Celsius. Hence efficient heat removal techniques must be used if this technology is going to succeed. The power-delay product for silicon-on-silicon circuits is much smaller than that of conventional packages.

Chapter 6

Hybrid Layout

This chapter describes an automatic layout system for silicon-on-silicon hybrid packages. This system exploits the hybrid technology described in Chapter 5 and is built on the design automation system discussed in Chapter 4. The author has made contributions to the hybrid global and detailed routing and pad frame generation procedures. A bottom-up description of the layout process follows. Layout is generated from a source schematic composed of at least three levels of hierarchy. A layout generation procedure is provided for the three levels. Figure 5 illustrates the areas of the layout of Figure 1 that were generated by the three layout generators.

The lowest level schematic cells represent the VLSI circuits (the leaf cells) and are used to generate the die mount patterns. These patterns provide places to attach the ICs as well as bonding pads on the hybrid that will connect to the bonding pads on the IC. Section 6.1 describes this layout operation.

Above the leaf cell generators is the layout procedure that generates the main placement and routing from one or more hierarchical levels in the schematic; intermediate schematic cells are flattened until only one hierarchical level exists between the top level and the leaf cells. This layout generator, described in Section 6.2, produces an inner object and is an adaptation of general cell placement and routing layout procedures.

The highest level layout procedure is described in Section 6.3 and generates the bonding pads around the periphery of the hybrid and routes from these pads to the pins on the inner object. Separate layout procedures for the top level and second level schematics provide a clean separation of topological domain code from geometrical domain code. Only a minimal area penalty is required for this separation.

6.1 Die Mount Pattern Generation

The *DieMount* layout procedure constructs a die mount pattern. In effect this pattern translates from the IC technology to the hybrid technology. This layout procedure is invoked once for each IC in the design. An example is shown in Figure 4. The icon summarizes the interface while the schematic defines the binding of the hybrid nets to the IC pads. The schematic and the IC layout are inputs to this layout procedure.

A die mount pattern is the hybrid layout that corresponds to a particular IC. This rectangular region is large enough to hold the IC die and has around its perimeter one or two rows of die mount pads. As shown in Figure 6, bonding wires are used to connect the pads on the IC (***die pads***) to the pads on the mounting pattern (***die mount pads***). Although design rules allow two bonding pad rows, the pad row nearest the edge of the die was constrained to contain only V_{dd} and *Ground* connections in this example.

The 4 hybrid metal layers are numbered 0 through 3. Layer 0 (the bottom layer) is always *Ground* and covers the entire surface. Layer 1 is usually V_{dd} except when used to connect signals to *Ground* as shown in Figure 6. Layers 2 and 3 are the signal layers. For the convenience of the router, all signal connections are provided at the periphery of the die mount in both signal layers.

Die mount layouts in the hybrid technology can be acquired in two ways. The first is simply to specify a previously constructed die mount from a library. The second is to apply the die mount generator to an IC for which the die mount pattern is to be generated. The die mount generator extracts first the IC layout to find the position and size of all pads. This information, combined with the signal names, is then used to determine the number of rows required as well as the individual pad types and locations. Figure 6 shows the various combinations of pad types and relative sizes. A typical value of w is 0.1 mm. The die mount pads are elongated to provide additional space for new bonds in case the ICs need to be replaced.

6.2 Hybrid Placement and Routing

The layout procedure that generates the main portion of the layout, the *inner* object, is adapted from existing general cell layout procedures [Pre85]. Automatic, graphical, and interactive placement techniques are described in Section 6.2.2. Global and detailed routing algorithms are described in Section 6.2.3 and 6.2.4 respectively. The placement and routing algorithms operate in the topological domain; the layout is converted to geometric form as the final step. The topological model that serves as a foundation to the algorithms is described in Section 6.2.1.

6.2.1 The Topological Model

The unifying concept throughout the hybrid placement and routing layout generator is the topological model [Cho85, Pre85]. In this abstraction only the relative positions of the IC primitives and the routing areas are important. The model automatically compensates for changes in the routing areas as they expand or contract to conform to the changing design. It is necessary to convert to the geometric domain only to compare alternative layouts during optimization or as the final step before generating the masks.

The topological model reduces the complexity of hybrid placement and routing by abstracting out the geometrical information (geometry). Computation of geometry is deferred until needed. The advantages of this approach are the following: removing and adding both components and wiring are simplified; topological operations are well defined; mapping the layout problem into the geometric domain is well defined and fast; routing areas can always be made just large enough to accommodate the required wiring since size determination is made after routing; and the complexity of dealing with masses of geometric data is obviated.

The routing areas are organized into channels, as shown in Figure 7, which are maintained throughout the layout process. Channels divide the space into topological holes, each of which contains an IC. Channel widths may vary depending on the number of wiring tracks required. The channel lengths are determined by the positions of the channels they intersect at their extreme points. Exactly one channel is associated with each side of each IC.

Three graphs are used to model the hybrid surface: the horizontal channel position graph, the vertical channel position graph, and the channel intersection graph. The channel position graphs are used to map between the topological and geometric domains while the channel intersection graph is used in topology modification and global routing. The horizontal channel position graph is a directed acyclic graph where arcs represent the widths (horizontal dimensions) of vertical channels or the horizontal dimensions of the IC die mount patterns. Figure 8 shows the horizontal channel position graph corresponding to the layout of Figure 1. The vertical channel position graph is analogous. The channel intersection graph is an undirected graph where the nodes represent the channel intersections and the arcs represent segments of channels between two intersections. Figure 7 shows the arcs of the channel intersection graph.

Three classes of operation are necessary for the topological model: operations which map geometrical domain problems into the topological domain; operations which compute geometrical aspects given a topology; and operations which modify or manipulate a topology.

The *DefineChannels* operation maps a geometrical domain layout into the topological do-

main. The input is a set of rectangles representing the ICs on a plane. First an outline is constructed around each rectangle; the outline always contains the rectangle and is made as large as possible without intersecting or containing any other outlines. Next a primary direction is chosen and all outlines perpendicular to this direction are collapsed along the primary direction into channels. Finally, the remaining outlines are collapsed along the secondary direction. The ICs and channels may need to be repositioned (using the *Geometerize* operation described below).

Operations are provided to compute aspects of geometry: *Size* finds the length of the longest path through a channel position graph (corresponds to height or width of hybrid). *ShortestPath* finds a shortest path between two nodes in the graph. *CriticalArcs* finds arcs that lie on any longest path through a channel position graph. *Geometerize* determines positions for the channels and ICs. All of these operations are linear in the number of arcs and nodes [Deo74].

Operations which modify or manipulate the topology are also provided by the topological model and are used primarily by automatic placement algorithms [Cho85, Pre85]. Examples of such operations are *Grow* which adds an IC, *Shrink* which removes an IC from the surface, and *TtoL* which modifies the channel intersection topology.

6.2.2 Placement

Placement procedures determine the relative positions of the ICs and the public pins of the inner object as well as organize the routing areas into channels. Hybrid placement is both simpler and more complex than general cell placement within ICs. Hybrid placement is simpler because fewer components are involved (8-20 ICs per hybrid compared to 10-100 general cells per IC), and the sizes and shapes of ICs on a hybrid are less variable than are general cells within an IC. Hybrid placement is more complex because many interrelated factors determine layout quality. Wide busses are very prevalent; propagation delays and even power distribution are much more important.

Two placement options are available to the user. An automatic placement algorithm built on the topological model is available. This algorithm finds a constructive initial placement, then improves the placement by iteration. This algorithm is described in [Pre85].

The second option uses graphical hints from the schematic to determine a floorplan. This option is popular because it provides a straightforward way for the user to specify a (topological) placement for a small number of ICs. In hybrid layouts there is often only one correct placement and that placement is obvious to the user. IC *pinouts* as well as public pads on the hybrid are often defined to accommodate a particular bus structure that further constrains placement.

The procedure to support this second placement option is minimal given the topological model and the design system foundation. The *DefineChannels* operation from the topological model

is invoked on the geometric domain layout defined by the IC icons in the source schematic. After the routing channels have been determined, actual layouts for the IC die mounts are substituted for the iconic representations. Routing channel topology may be modified interactively using the topology manipulation primitives of the topological model.

The placement for the layout shown in Figure 1 was obtained by the second placement option (without interactive modification). Note the correspondence of the positions of the IC icons in Figure 4 to the die mount patterns in Figure 1.

6.2.3 Global Routing

Global routing assigns segments of each net to a set of routing channels so as to reduce hybrid area and shorten connection lengths. Global routing for silicon-on-silicon hybrid packages differs in several important ways from general cell global routing. V_{dd} and *Ground* signals do not complicate global routing because these signals are distributed on separate layers, and taps to the power supply layers are easy to make. However, some problems are more complicated: overall dimensions must be tightly controlled (to fit within the package) and propagation delays must be carefully controlled.

Our global router operates on a channel intersection graph (CIG). The problem of determining the global routing for a single interconnection net is mapped into finding a minimum Steiner tree through the channel intersection graph. The interaction of this net with other nets and with the hybrid surface is captured through the use of appropriate arc weights. The algorithm to route a two-pin net is equivalent to finding a shortest path between two nodes in CIG. For a multi-pin net the global routing of the net is more complex. The complexity of the problem grows rapidly as the number of pins to be connected is increased.

The initial global route algorithm works as follows. Nets are sequentially selected for routing, and pins of the each net are mapped as new nodes onto the CIG. Arc weights are assigned according to the geometric lengths of the arcs. One pin is selected as the source and symmetric expansion is performed until any one of the other pins of the net is encountered. This subtree is then used as the source of the next expansion, and the process continues until all the pins in the net are visited. This approach guarantees that the path found between any two pins is of minimum length. However, it does not find a minimum Steiner tree. In case of electrically equivalent pins, we insert all the pins into CIG, but as soon as one is included in the Steiner tree for the net, the rest are no longer considered as targets for expansion. In this way, the global router can make use of equivalent pins to minimize the routing length of a net.

The initial global router routes each net ignoring all other nets in the design. This usually leads to a layout that is larger than necessary. An improvement phase follows which minimizes hybrid

area by selecting and rerouting critical nets. The idea is to start with the initial global route and improve it by repeatedly selecting critical nets, removing them, and reconnecting them by alternate paths. Critical nets are those nets which, if removed, make the layout smaller. These nets pass through positions of maximum density along critical channels. A critical channel is a channel will allow the size of the layout to be reduced if its width is decreased. The arcs corresponding to critical channels lie on a longest path through a channel position graph. The global route improvement algorithm is outlined in Figure 9.

In order to determine the channel widths, we compute the maximum track density of net segments passing through channel. We can consider the maximum track density to be an approximation of the actual width required for detailed routing. Arc weight for noncritical CIG arcs are the geometric lengths of the associated channel segments. For each critical CIG arc, the weight is the geometric length plus the half perimeter of the hybrid area. The idea is that, for most of the nets, the maximum Steiner distance between two pins is less than the half perimeter of hybrid. CIG arc weights are incrementally updated from iteration to iteration. The improvement phase is, however, sequential; a heuristic is employed to obtain the effect of a look ahead. Although rerouting of some critical net may not reduce the hybrid area, it may increase the probability of a future reduction. After all there may be more than one critical path in the position graphs. Also, if we succeed in reducing the range of maximum track density along a channel, we may make one of the channel segments non-critical, or if we reduce this range to zero, we can save one track, thereby reducing hybrid area.

We are investigating improvements to the global routing algorithm which use a dynamic path finding algorithm (the weights of the CIG arcs corresponding to channels are modified while finding the shortest path). Better results are also expected if a hierarchical global routing approach is used.

6.2.4 Detailed Routing

The detailed router used in the hybrid layout system is based on the *dogleg* router [Deu76] but several extensions (both conceptual and in implementation) enhance the generality. Because so many channel routing algorithms have been published, only the important extensions are discussed here. [1] The implementation extensions imply a more sophisticated model but are built on the theory presented in [Deu76]. The major conceptual advancement guarantees routing completion even in the presence of *constraint loops.* (The original dogleg router did not deal with constraint loops because they were removed by altering the placement of the standard cells. Such luxury is not available in the hybrid environment.)

[1] An introduction to channel routing may be found in [Lor88].

Implementation enhancements allow extra track ordering sequences (specifically alternating starting sides from one track to the next in addition to the starting corner specified in the original paper) and variable width branch and trunk wires. These variable width wires require that more sophisticated geometrical checks be made (both in determining constraints and in determining if a trunk wire will fit on a track). Actual design rules (metal spacings, metal widths, via-cut surround by metal, via-cut-to-via-cut) must be checked instead of the simplified rules checks that are possible when wire width for layers are uniform. Variable width track wires are dynamically assigned to staticly defined tracks. Full design rule checks must be made with other wires and with the channel edges.

The conceptual enhancement of this channel routing algorithm is the manner in which constraint loops are resolved. In summary the algorithm resolves constraint loops by adding a *non-pin dogleg* to enough wire segments to break all of the constraint loops. Sometimes a position to insert a non-pin dogleg cannot be found within the channel. In these cases it is necessary to defer the dogleg to one of the intersecting channels.

The starting point is a constraint graph where the nodes represent wire segments between two pins and the arcs represent the vertical constraints. Pins along the channel sides are investigated for vertical constraints, and arcs are added to the constraint graph. If a cycle will be formed by adding an arc, then that cycle is broken by adding a non-pin dogleg to one of the wire segments participating in the constraint loop. Each such wire segment is investigated by considering trial positions for the non-pin doglegs at pin locations and between each pair of pins. The trial wire segment and trial position for the non-pin dogleg that adds the least to channel density and wire length and do not propagate any constraints is chosen. This method can guarantee completion of all of the routing if the non-pin doglegs can be deferred (pushed off the ends of the channel and hence routed in the intersecting channel). The channels are routed in such an order to guarantee that this deferring can be done [Pre79b]. While this algorithm has poor worst case time performance, its average performance is quite good. Dogleg positions are usually found near their desired locations.

6.3 Hybrid Pad Frames

In the top-level schematic the pad frame layout generator expects to find five schematic cells: one inner cell to be laid out as described in Section 6.2 and four cells corresponding to the four pad rows. Schematic descriptions of the pad rows are usually composed of *abuts* of hybrid bonding pad cells as shown in Figure 2. However, the full power of the design system is available for more complex pad row definitions. In practice, a schematic template is available to simplify pad ring definition.

The layout generation procedure uses a method similar to [McG87] to route from the pad

frame to the inner cell. However, global routing at this level is not required since the layout generator described in Section 6.2 guarantees that each hybrid bonding pad has a pin on the inner cell just across the adjacent routing channel. A trapezoidal routing channel is constructed on each side of the inner cell by using the detailed router described in Section 6.2.4. Triangular routing obstructions are introduced at the ends of the channels to form the trapezoidal channels. The hybrid bonding pads and the trapezoidal routing channels regions are constructed by this layout generator.

The position for the inner cell is determined by estimating the width of the four channels (as approximated by track density). Any space in excess of the required channel widths is divided equally among the channels.

Chapter 7

Results

We have used the automatic layout system for the hybrid packages to layout an IO subsystem of a high performance workstation (Dragon). The function of this IO subsystem is to provide the interface among two unidirectional busses and a synchronous, high bandwidth bus designed to address the requirements of data consistency in a shared memory multiprocessor system. This IO subsystem includes four bus interface chips, a cache memory, an address mapping cache, a display controller, and an IO bus translation chip. The relative placement of the ICs on the package is functionally constrained. For instance, the four bus interface chips connect to the hybrid IO pads and hence, are constrained to be close to the boundary of package. The pin positions and orientations of ICs have been optimized so as to minimize the interconnection lengths. Two 68 bit busses connect to the bus interface chips to the IO pads and several hundred nets connect pins on the ICs.

The package has a fixed area of 2.6" by 3.2" inches, and a pad-to-pad spacing of 635 mm (25 mils) is used. The hybrid technology has two signal layers with a trace pacing of 50 mm. The spacing of the die mount pads is 180 mm. The IC circuits come in different sizes but are wire bonded to one of three standard-size die mounts.

Each IC was generated by invoking a layout generator on its source schematic. The hybrid layout procedure used cached layout results for these individual ICs. The user specified the placement graphically through positions of the IC icons on the schematic. The hybrid area was reduced by 5% as a result of global route improvement. This improvement is good considering the optimized placement and pin locations of the ICs on the hybrid package. The active cell area on the hybrid package accounts for 25% of the total area which indicates a packaging efficiency of greater than 25%. (This is conservative since it does not include the contribution of wiring on the hybrid.) If we were to implement this circuit using single chip packages, a PCB trace spacing of 25 mils (635 mm) with 10 signal layers, a total area of 4.7" by 9.8" inches would be required to implement the inner object.

This area is 8.2 times the size of the inner object and 5.5 times the size of the entire hybrid package.

7.1 Summary

In high speed computer systems, performance and density limits are being set more by interconnect and packaging constraints than by transistor switching speeds. Multi-chip silicon-on-silicon hybrid packages can significantly improve performance by eliminating single chip packaging.

This report describes the hybrid technology, the design automation system foundation, and a hybrid layout system capable of automatically laying out circuits on hybrid packages. The layout system is implemented as three layout generation procedures: die mount pattern generation, a general cell-like placement and routing procedure, and a pad frame generator.

System operation is illustrated through a comprehensive example: a high performance workstation IO subsystem containing 8 VLSI circuits implemented on a 2.6" by 3.2" hybrid package. This method produces layouts that are 5 to 8 times denser than the same circuit implemented with single-chip packages on a printed circuit board. Simulations show that clock speeds can be increased by a factor of two.

Acknowledgements

I am indebted to my research advisor Professor Ernest S. Kuh for his continuous encouragement and guidance. I gratefully acknowledge the many helpful suggestions and valuable comments by Dr. Bryan Preas at Xerox, Palo Alto Research Center. I have learned a great deal from my cooperation with Dr. Preas who has supervised my research at PARC. I wish to express appreciation to my colleagues and friends at UC Berkeley and at PARC. Finally, I am most indebted to my wife, Afsane, for her love, patience, and faith in this undertaking.

Bibliography

[Ake81] Akers, S.B., "On the use of the linear assignment algorithm in module placement," in *Proc. of the 18th Design Automation Conference*, pp. 137-144, 1981.

[Bal85] Balderes, D., and M. L. White, "Package effects on CPU performance of a large commercial processor," *Proc. of Electronics Components Conference*, 1985.

[Bar88a] Barth, R., and B. Serlet, "A structural representation for VLSI design," *Proc. of the 25th Design Automation Conference*, June 1988.

[Bar88b] Barth, R., B. Serlet, and P. Sindhu, "Parameterized schematics," *Proc. of the 25th Design Automation Conference*, June 1988.

[Bar88c] Barth, R., L. Monier, and B. Serlet, "Patchwork: layout from schematic annotations," *Proc. of the 25th Design Automation Conference*, June 1988.

[Bog87] Bogatin, E., "Beyond printed wiring board densities: a new commercial multichip packaging technology" Raychem Corporation Internal Report, May 1987.

[Bre77] Breuer, M.A., "A class of min-cut placement algorithms," in *Proc. of the 14th Design Automation Conference*, pp. 284-290, 1977.

[Bru89] Bruce, R., W.P. Meuli and J. Ho, "Multi chip modules," in *Proc. of the 26th Design Automation Conference*, June 1989.

[Bur83] Burstein, M., and R. Pelavin, "Hierarchical wire routing," *IEEE Transactions on Computer Aided Design*, vol. CAD-2, no. 4, pp. 223-234, October 1983.

[Bur86] Burstein, M., "Channel routing," in *Layout Design and Verification*, T. Ohtsuki, editor, Elsevier Science Publishers B. V. (North Holland), Chapter 4, pp 133-167, 1986.

[Che86] Chen, H. H., and E. S. Kuh, "Glitter: a gridless variable-width channel router," *IEEE Transactions on Computer Design*, vol. CAD-5, no. 4, pp. 459-465, October 1986.

[Cho85] Chow, C. S., Phoenix: Interactive Hierarchical Topological Floorplanning Placer, Masters Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1985.

[Con88] Cong, J., and Preas, B. "A new algorithm for standard cell global routing" in *IEEE Intl. Conf. on Computer-Aided Design*, pp. 176-180, November 1988.

[Deo74] Deo, N., Graph Theory with Applications to Engineering and Computer Science, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1974.

[Deu76] Deutsch, D. N., "A 'dogleg' channel router," *Proc. 13th Design Automation Conference*, pp. 425-433, June, 1976.

[Ein77] Einarson, N.S., *Printed Circuit Technology*, Printed Circuit Technology, Burlington, Massachusetts, 1977.

[Esc88] B. Eschermann, W. Dai, E. Kuh, M. Pedram "Hierarchical placement for macrocells: A 'meet-in-the-middle' approach", *Proc. ICCAD*, 1988 pp. 460-463.

[Fra86] Frankle, J. and R.M. Karp, "Circuit placements and cost bounds by eigenvector decomposition," in *Digest of the International Conf. on Computer Aided Design*, pp. 414-417, November 1986.

[Fuj86] Fuji, T. et al., "A new routing method based on tree-shaped regions for VLSI layout design," in *IEEE Proc. ISCAS* 1986, pp. 319-320.

[Hal70] Hall, K.M., "An r-dimensional quadratic placement algorithm," *Management Science*, vol. 17, pp. 219-229, November 1970.

[Han72] Hanan, M. and J.M. Kurtzberg, "Placement techniques," in M.A. Breuer, ed., *Design Automation of Digital Systems, vol. 1: Theory and Techniques*, Englewood Cliffs, New Jersy: Prentice-Hall Inc, Chapter 5, pp.213-382, 1972.

[Hig80] Hightower, D. W., and R. L. Boyd, "A generalized channel router," in Proc. 17th Design Automation Conference, pp. 12-21, June 1980.

[Joh86] Johnson, R. W., J. L. Davidson, R. C. Jaeger, and D. V. Kerns, "Silicon hybrid wafer-scale package technology," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 5, pp. 845-851, October 1986.

[Kan76] Kani, k., H. Kawanishi and A. Kishimoto, "Robin: building block LSI routing program," in *IEEE Proc. ISCAS* 1976.

[Kar83] Karp, R.M., R.L. Leighton, et al., "Global wire routing in two dimensional arrays," in *Ann. Symp. on Foundations of Computer Science*, 24(1983) 453-459.

[Kel86] Keller, F. V., and D. A. Mlynksi, "A graph-theoretical channel-router for constraint-loop-problems," in *Proc. 1986 IEEE International Symposium on Circuits and Systems*, pp. 311-314, May 1986.

[Kir83] Kirkpatrick, S., C.D. Gellat and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, May 13, 1983.

[LaP86] La Potin, D.P. and S.W. Director, "Mason: a global floorplanning approach for VLSI design," *IEEE Trans. on Computer-Aided Design*, vol. CAD-5, no. 4, pp. 477-489, October 1986.

[Lau79] Lauther, U. "A min-cut placement algorithm for general cell assemblies based on a graph representation," in *Proc. 16th Design Automation Conference*, pp. 1-10, 1979.

[Lor88] Lorenzetti, M. J., and D. S. Baeder, "Routing," in *Physical Design Automation of VLSI Systems*, Preas, B. T., and M. J. Lorenzetti, editors, Menlo Park, California, Benjamin-Cummings Publishing Company, Inc., 1988.

[Lym80a] Lyman, J., "Packaging technology responds to the demand for higher densities," in *Microelectronics Interconnection and Packaging*, edited by J. Lyman, McGraw-Hill Publications, New York, pp. 198-206, 1980.

[Lym80b] Lyman, J., "Advances in materials, components, processes ensure hybrid prosperity in the LSI age," in *Microelectronics Interconnection and Packaging*, edited by J. Lyman, McGraw-Hill Publications, New York, pp. 62-74, 1980.

[Mar84] Marek-Sadowska, M., "Global router for gate arrays," in *Proc. IEEE International Conf. on Computer Design*, (1984 332-337.

[Mar86] Marek-Sadowska, M., "Route planner for custom chip design," in *Proc. IEEE International Conf. on Computer Design*, pp.246-249, 1986.

[McG87] McGehee, R. K., "A practical moat router," *Proc. of 24th Design Automation Conference*, pp. 216-221, June 28-July 1, 1987.

[tt83] Otten, R.H.J.M., "Efficient floorplan Optimization," in *Proc. of the Intl. Conf. on Computer Design*, pp. 499-502, 1983.

[Per76] Persky, G., "PRO-an automatic string placement program for polycell layout," in *Proc. 13th Design Automation Conference*, pp. 417-424, June 1976.

[Pre79a] Preas, B. T., and W. M. VanCleemput, "Placement algorithms for arbitrarily shaped blocks," in Proc. of the 16th Design Automation Conference, pp. 474-480, 1979.

[Pre79b] Preas, B. T., and W. M. VanCleemput, "Routing algorithms for hierarchical IC layout," *Proc. of the Intl. Symposium on Circuits and Systems*, pp. 482-485, July, 1979.

[Pre85] Preas, B. T., and C. S. Chow, "Placement and routing algorithms for topological integrated circuit layout," *Proc. of the Intl. Symposium on Circuits and Systems*, pp. 17-20, 1985.

[Pre86] Preas, B.T. and P.G. Krager, "Automatic placement - a review of current techniques," *Proc. 23rd Design Automation Conference*, pp. 622-629, 1986.

[Pre89] Preas, B.T. and L. Monier, "Pad frame layout and routing for VLSI circuits," *Proc. Intl. Symp. on Circuits and Systems*, ISCAS-89.

[Pri57] Prim, R.C., "Shortest connecting networks and some generalizations," in *Bell System Technical Journal*," 36(1957) 1389-1401.

[Qui75] Quinn, N.R., "The placement problem as viewed from the physics of classical mechanics," in*Proc. of the 12th Design Automation Conference*, pp. 173-178, 1975.

[Ree85] Reed, J., A. Sangiovanni-Vincentelli, and M. Santomauro, "A new symbolic channel router: YACR2," *IEEE Transactions on Computer Design*, vol. CAD-4, no. 3, pp. 208-219, July 1985.

[Riv82] Rivest, R. L. and C. M. Fiduccia, "A 'greedy' channel router," in *Proc. 19th Design Automation Conference*, pp. 418-424, June 1982.

[Sat80] Sato, K., H. Shimoyama, T. Nagai, M. Ozaki, and T. Yahara, "A 'grid-free' channel router," in *Proc. 17th Design Automation Conference*, pp. 22-30, June 1980.

[Sha85] Sha, L. and R.W. Dutton,"An analytic algorithm for placement of arbitrarily sized rectangular blocks," in Proc. 22nd Design Automation Conference, pp. 602-608, 1985.

[Sec88] Sechen, C., "Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing," in *Proc. 25th Design Automation Conference*, pp. 73-80, 1988.

[Ser86] Serlet, B., "Object-oriented programming in cedar." *Actees des Journees Languages Orientes Object*, Bigre Globule, pp. 64-68, January 1986.

[Tin83] Ting, B.S. and B.N. Tien "Routing techniques for Gate Array," in *IEEE Transactions on Computer Aided Design*, vol. CAD-2 no.4 October 1983.

[Ued85] Ueda, K., H. Kitazawa and I. Harada, "CHAMP: chip floor plan for hierarchical VLSI layout design", *IEEE Transactions on Computer Aided Design*, vol. CAD-4, no. 1, pp. 12-22, 1985.

[Vec83] Vecchi, M. and S. Kirkpatrick, "Global wiring by simulated annealing," in *IEEE Trans. on Computer Aided Design*, vol. CAD-2, no.4, October 1983, pp. 215-222.

[Wim88] Wimer, S. and I. Koren, "Analysis of strategies for constructive general block placement," in *Transactions on Computer Aided Design*, vol. 7, no. 3, March 1988.

[Wip82] Wipfler, G.J., M. Weisel and D.A. Mlynski, "A combined force and cut algorithm for hierarchical VLSI layout," in *Proc. 19th Design Automation Conference*, pp. 671-677, 1982.

[Yos82] Yoshimura T. and E. Kuh, "Efficient algorithms for channel routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-1, no. 1, pp. 25-35, January 1982.

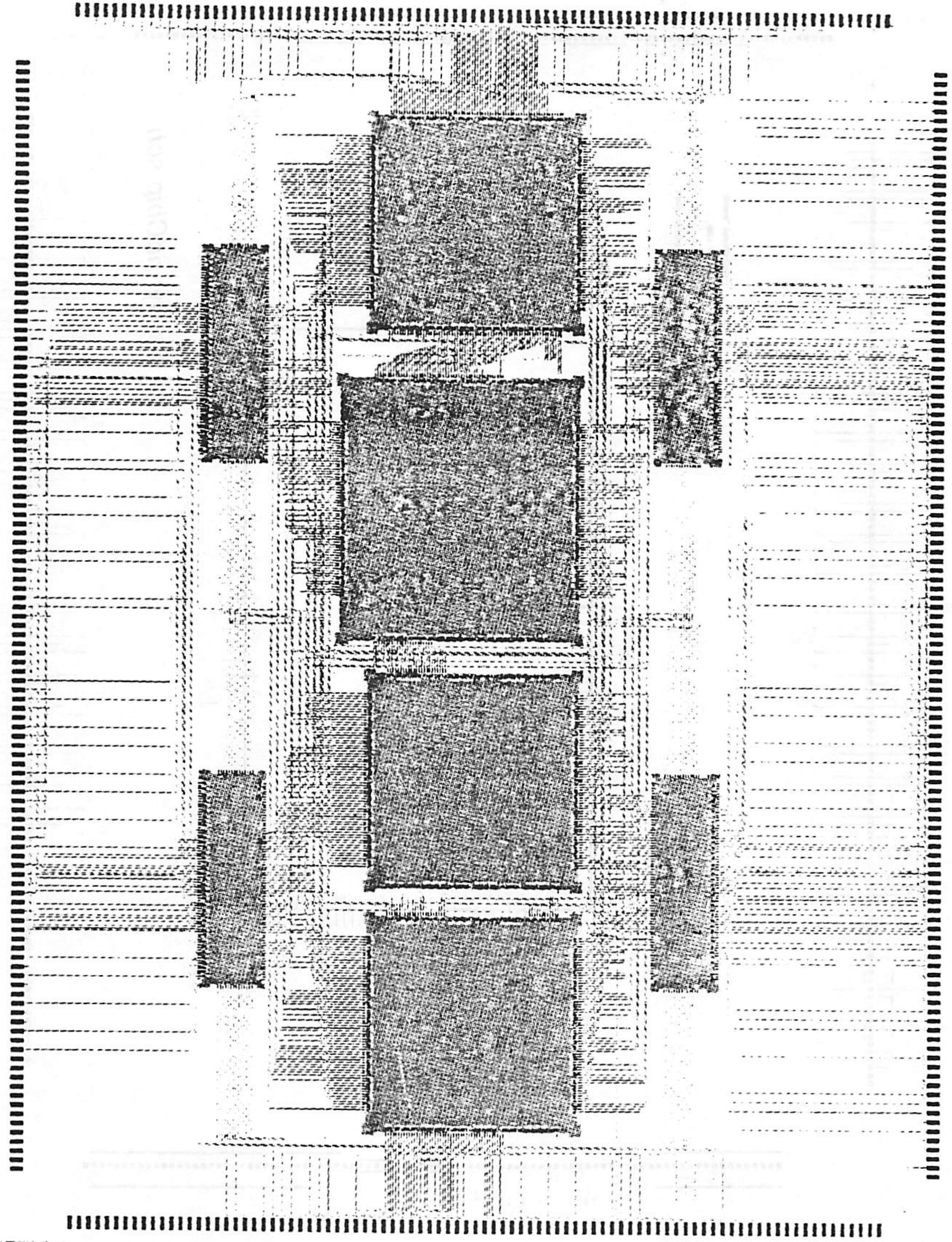


Figure 1. This hybrid layout was automatically generated from the schematics shown in Figures 2. 3. and 4. This circuit is the IO subsystem of a high performance workstation and contains 8 VLSI circuits.

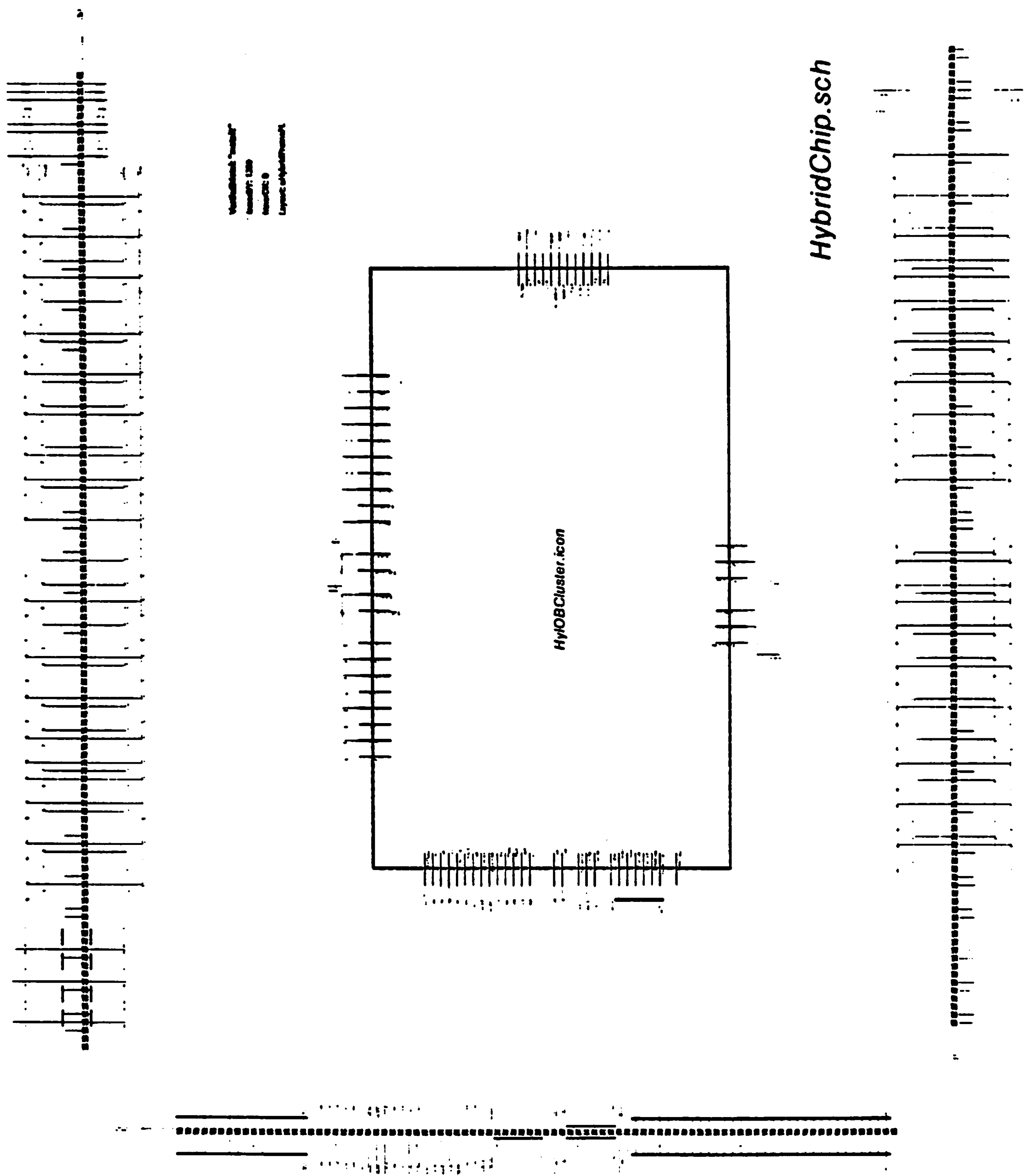


Figure 2. This top level schematic defines the hybrid pad ring and the pad ring routing requirements. The layout (shown in Figure 1) corresponding to the blue and yellow portions of Figure 5 is generated from this schematic.

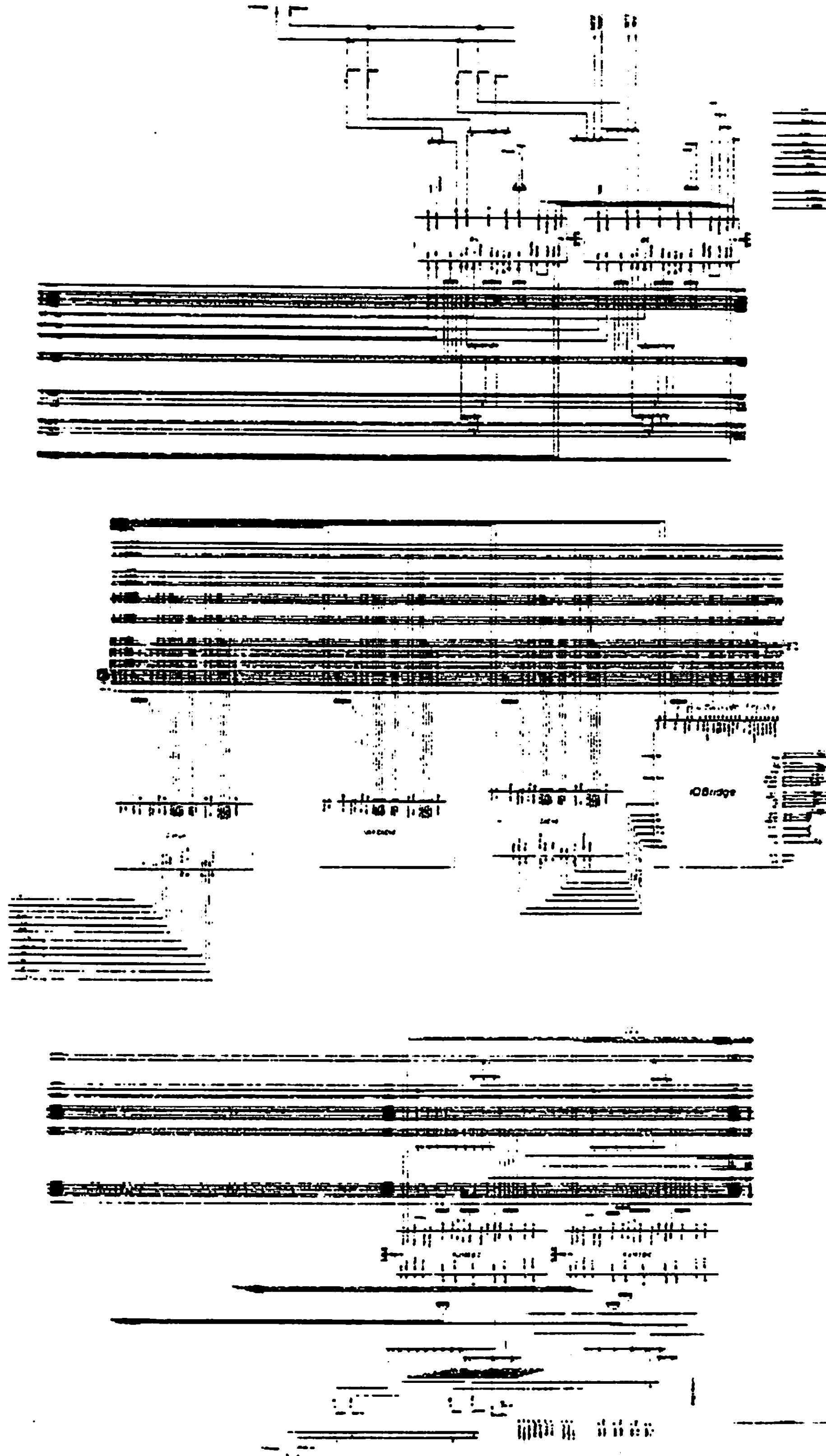
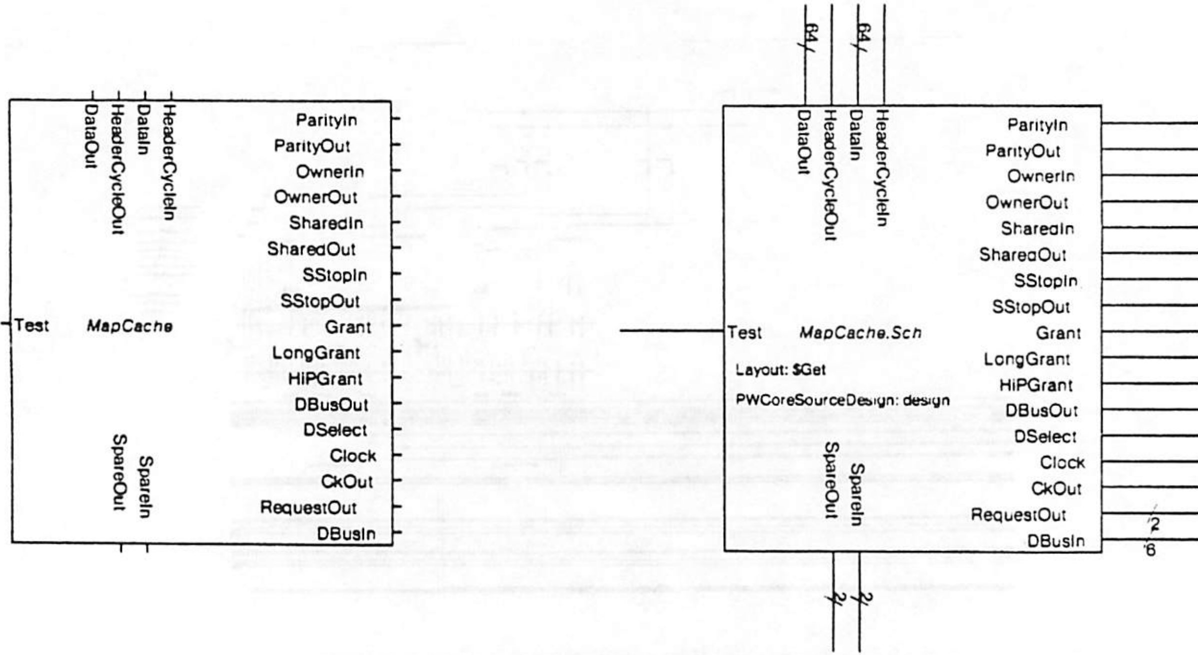


Figure 3. This second level schematic defines the main placement and routing problem (called the *inner* cell) for the hybrid.



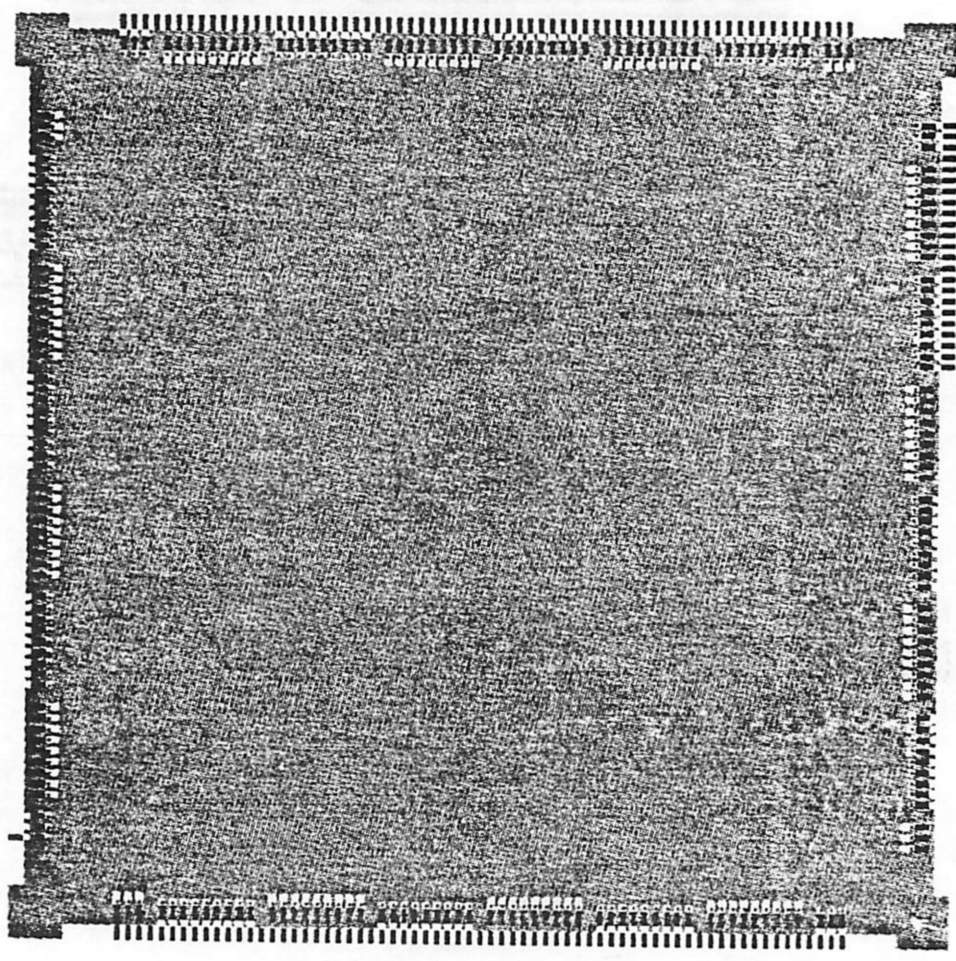


Figure 4. This example shows the icon (interface summary), the schematic and the layout for one of the integrated circuit die mounts. The die mount layout was generated from the schematic and the layout of the integrated circuit.

Figure 5. This shows the areas of the layout of Figure 1 that were generated by the three layout generation procedures described in the text. The blue and yellow portions represent the top level layout. The green die mounts were generated by the lowest level layout procedure. The white area represents the routing of the inner cell.

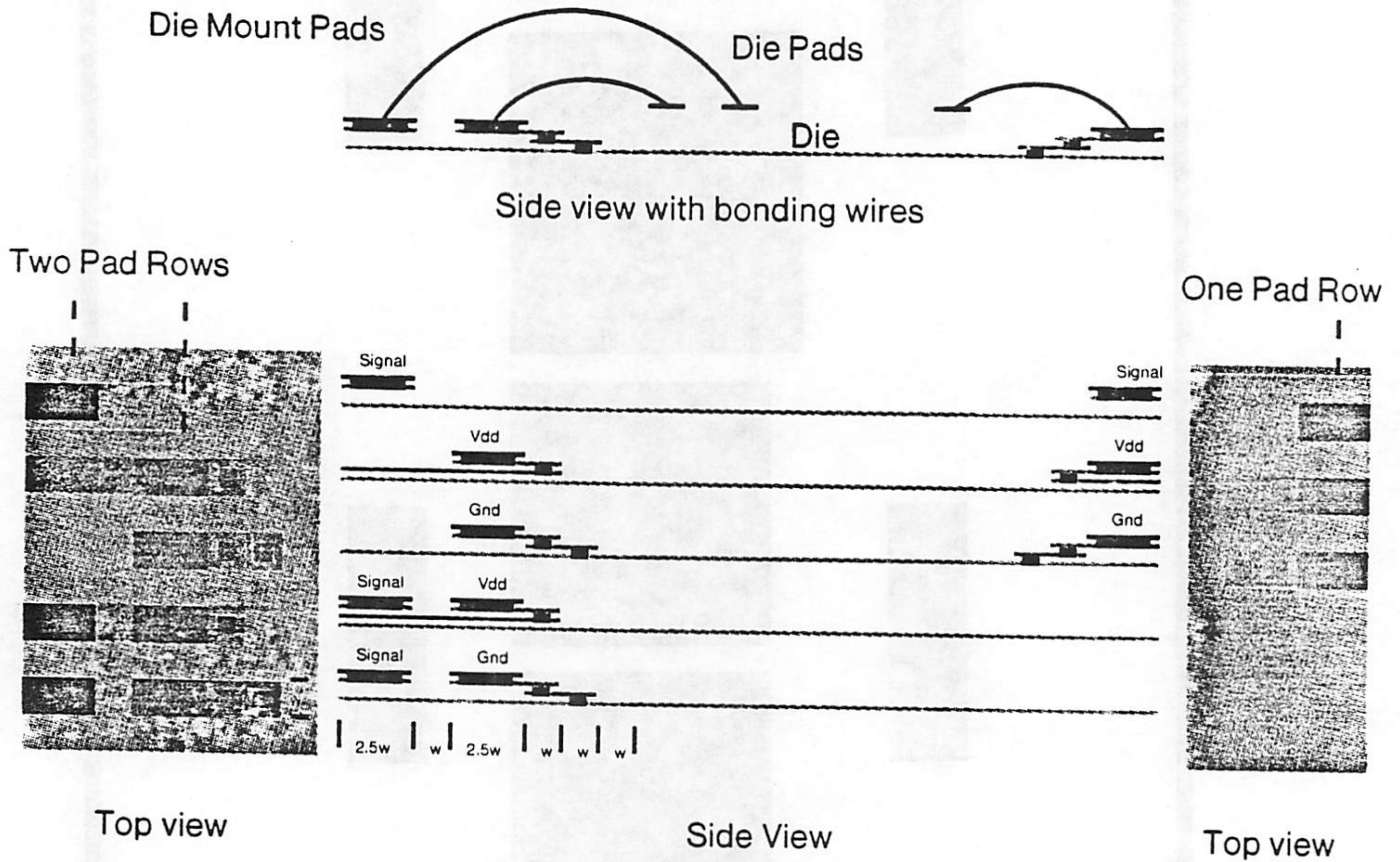
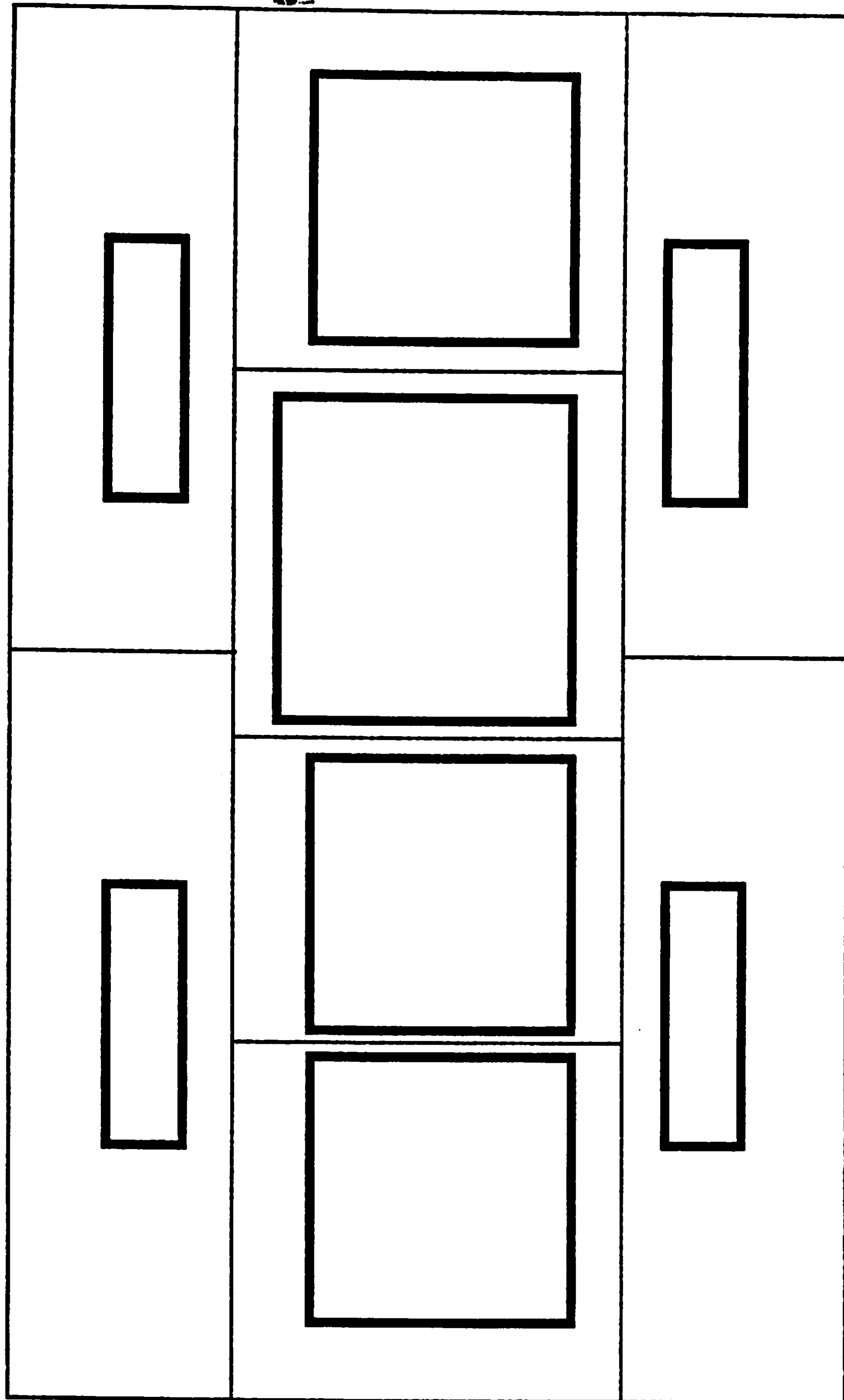


Figure 6. This shows the detailed construction of the die mount patterns. A typical value of w is 0.1 mm. Taps to the V_{dd} and Ground planes are automatically inserted as required. Signal pins are touch to the periphery of the pattern for routing.



Channel Intersection Graph

Figure 7. This Channel Intersection Graph corresponds to the layout of Figure 1. The red lines represent the routing channels.

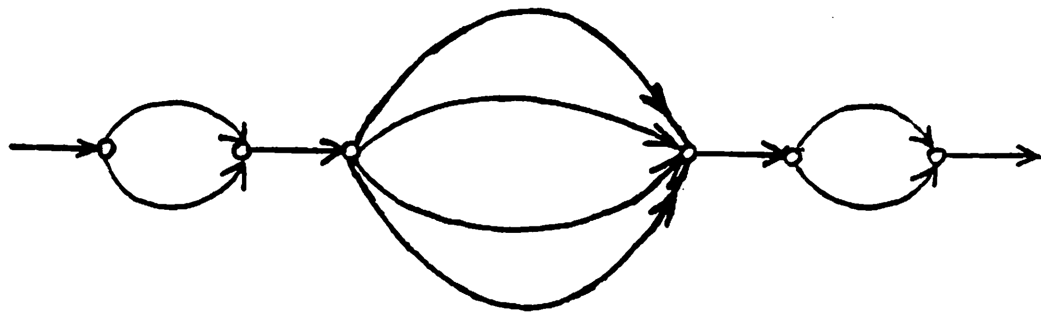


Figure 8. This horizontal channel position graph corresponds to the layout of Figure 1 and is used to compute horizontal dimensions of the layout.

```
FOR each net DO
    save current global information for this net;
    remove global route information for this net;
    compute ranges of maximum track density along all channels;
    determine channel widths based on maximum track densities;
    assign coordinates to channel position graphs (Geometerize);
    identify critical CIG arcs (CriticalArcs), and update their arc weights;
    insert pins of this net into CIG, and reroute it (ShortestPath);
    compute new hybrid area (Size);
    IF new hybrid area is reduced THEN accept;
    ELSE IF new hybrid area unchanged THEN
        IF number of critical channels is reduced THEN accept;
        ELSE IF number of critical channels unchanged THEN
            IF length of maximum density ranges along critical channels is reduced THEN
                accept;
            ELSE reject;
        ELSE reject;
    ELSE reject;
    IF accept THEN update hybrid area;
    ELSE IF reject THEN restore global route information for this net;
END FOR;
```

Figure 9: The global route improvement algorithm reroutes nets to reduce layout area. Topological model procedure names are shown in parenthesis.