A MULTIPLE-ROW-BASED LEAF CELL GENERATOR

by

SATISH KRISHNA RAO, B.Eng.

A THESIS

IN

COMPUTER SCIENCE

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

Approved

August, 1990

505
T3
1990
NO.89
cop.2.

ACKNOWLEDGMENTS

I could not have had this opportunity without the sacrifices from my dear mother and father. I sincerely thank them for everything. My brothers and sister are my best friends and thank them for always being there.

I thank Dr. Lakhani for providing me all the help I needed with the research, and for making me a better student and a better person with his wonderful courses. I am grateful to Dr. William M. Marcy, and Dr. Chao for their help with my research work.

I have been lucky to have a great bunch of friends, both at home in India, and in Lubbock, who made going to school fun and helped me during hard times. My heartfelt thanks to everybody in the Department of Computer Science at Tech, especially to my teachers. I appreciate Ms. Birgit for taking so much trouble with the corrections.

CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Over the past few years, integrated circuit (IC) technology has grown to such an extent that IC chips containing over 100,000 devices are now quite common. To cope with the enormous complexity in designing chips of such magnitude, many methodologies and design algorithms have been developed. Good Computer Aided Design (CAD) tools are now necessary to reduce the design cost and the complexity. Therefore, there is increased interest in the automation of the entire chip design process.

This chapter describes a cell design methodology using the Complimentary Metal Oxide Silicon (CMOS) technology. The different options and their advantages and disadvantages are also given. Later, it also describes a new approach for cell generation that we propose to study. The expected improvements of our method are stated.

1.1 CMOS Technology

Over the past few years, CMOS technology has played an increasingly important role in the world of integrated circuit (IC) industry. A Metal-Oxide-Silicon (MOS) structure is created by superimposing several layers

of conducting, insulating, and transistor forming materials. After a series of processing steps, a typical structure might consist of levels called diffusion, polysilicon (poly), and metal that are separated by insulating layers. CMOS technology provides two types of transistors, an n-type transistor (nMOS) and a p-type transistor (pMOS). These are fabricated in silicon by using either negatively doped silicon that is rich in electrons or positively doped silicon that is rich in holes (the dual of electrons). For the n-transistor (p-transistor), the structure consists of a section of p-type (n-type) silicon separating two diffused areas of n-type (p-type) silicon. The area separating the n (p) regions is capped with a sandwich consisting of an insulator and a conducting electrode called the "gate." The transistor also has two additional connections, which are designated as the "drain" and the "source," these being formed by the n (p in the case of a p-transistor) diffused regions. The gate is connected to the control input and it controls the flow of the electrical current between the drain and the source terminals. The terminals are physically equivalent and the name assignment depends on the direction of the current flow. Because of their inherent characteristics, the p- and n-transistors are ON and OFF for complementary values of the gate signal. An n-transistor (p-transistor) is almost

perfect when a logic '0' (logic '1') voltage is to be passed from an output to an input but not when passing a logic '1' (logic '0'). By combining an n- and a p-transistor in parallel, a switch which passes both levels of logic is constructed. By using combinations of p- and n-transistors, CMOS combinational gates may be constructed. In a CMOS gate, there is duality between the connection graph of n-transistors and the connection graph of p-transistors, i.e., if p-transistors are connected in parallel, then the corresponding n-transistors are connected in series and vice versa. Thus when we mention a single row of transistors, we mean one row each of n-and p-transistors.

1.2 Chip Design Process

A typical design begins with a behavioral description of the chip. This description is then decomposed into functional blocks necessary to implement the behavior. Each functional block is then given a set of performance objectives that must be satisfied by the remaining steps of the design process. Next, logic descriptions of each of the functional blocks are developed. A logic designer is responsible for this step. The descriptions are then passed on to the circuit designer who, in conjunction with the layout experts, generates the mask artwork ensuring that each block meets its performance objectives. The

generation of the layout is the most time- consuming and error prone step. Recent studies have established that the cost of fabricating an integrated circuit is an exponential function of the area. Furthermore, area-efficient circuit layouts lead to better packaging density and improved circuit performance.

Automation of each of the above phases is continuously evolving at many CAD tool development centers. The ultimate goal of a CAD system is to integrate all the phases so that a mask-level description can be generated directly from the behavioral description of the chip. A system with this capability is called a silicon compiler. Figure 1.1 shows this chip design process.

1.3 Chip Layout

Chip layout, or circuit layout, is a transformation of the components of an electrical circuit into equivalent geometric mask features. The geometric masks should be area-efficient and at the same time must satisfy a large number of independent, geometric constraints. These constraints are imposed by the process technology and are known as the design rules. The design rules specify the minimum allowable values for certain widths, separations, extensions and overlaps of geometric objects patterned on various layers. Constructing an area-efficient layout,

without violating any design rules over a large number of objects in the layout, is a very difficult task. A number of algorithms have been developed to automate the layout process. The different design approaches which are followed in obtaining a chip can be broadly classified under the following two categories: semi-custom chip design, and full-custom chip design.

1.4 Semi-custom Chip Design

In this category, the designer starts with a partially pre-fabricated standard layout of transistors and develops the complete chip layout by making the required interconnections between the existing transistors. Two of the popular methods that use the semi-custom design approach and their salient features are described below.

1.4.1 Gate Arrays

Gate arrays can be categorized by a design that uses a large number of identical "sites," each site consisting of a number of n- and p-transistors. The designer just has to make the interconnections between the transistors.

The gate arrays are quite popular because of the following reasons: i) little time is needed to obtain the required layout (turn-around time); ii) the user has to

make the metal connections only; iii) sometimes the full-custom chips are taken out of the market forcing the user to use semi-custom chips.

The disadvantages of gate arrays are as follows: i) the transistors exist regardless of whether they play a role in the circuit or not. So, if they are not used, they contribute to wastage of the chip area; ii) the layout style is very much constrained.

1.4.2 Standard Cells

In the standard cells approach, the designer can use a set of predefined logic/circuit cells (libraries) to complete a design. The predefined cells can be placed anywhere in the chip area.

The standard cells approach has two major advantages: i) the turn-around time is low, and ii) less area is wasted when compared to gate arrays. The major disadvantage of this approach is that the overall hierarchy of the circuit is flattened to the leaf-cell level (described later in this chapter). Then, the leaf-cells are generated using the standard library of cells. In other words, it is like writing a program using only the available standard packages as opposed to writing a custom program which is more efficient. Further optimization at higher levels is not possible.

1.5 Full-custom Chip Design

Under this approach, the complete layout is developed from scratch. Custom design offers the best utilization of the chip area, without ignoring the performance requirements. A number of systems which attempt to automate custom chip design have been developed recently and many of them follow symbolic design methodology. Symbolic design systems hide the physical geometric constraints (e.g., design rules, mask layer) from the designer and focus on the structural intent of the design. This thesis is concerned with the development of a layout system that follows symbolic design methodology.

1.5.1 Symbolic Design Methodology

Symbolic design process begins with a hierarchical floorplan which specifies the spatial distribution of logic in a very abstract form. The elements of the floorplan, at the lowest level of hierarchy, are called the leaf-cells. These leaf-cells define the physical layout of symbolic primitives (transistors, contacts, wires, etc.). A leaf-cell designer places these symbols on a grid to realize a circuit. The experience and skill of the designer are crucial for developing good layout topologies, since his task is limited to obtaining a relative placement and

interconnection of symbols without worrying about the design rules. The leaf-cells are then symbolically connected to obtain composite cells. Composite cells contain instances of other cells only (leaf-cells and composite cells). Composite cells are further combined to form larger modules. Different modules contain cells which are structurally unrelated. For example, a PLA controller and a data path would normally be designed as two separate modules. These modules are then connected to each other and to the I/O modules (pads) using some place and route techniques. Basically this method reduces to a top-down circuit partitioning and a bottom-up layout implementation.

1.5.2 Leaf-cell Compaction

The task of converting a hierarchical symbolic layout to a mask layout can be divided into a number of steps, all of which should be automated to alleviate the possibility of human errors. The typical steps are as follows: After a leaf-cell has been symbolically designed, a leaf-cell compaction program is used to convert the symbols to mask elements and to pack them as close to each other as allowed by the design rules. After compacting all the leaf-cells given in the floorplan individually, cells are connected. The terminals between the adjacent cells are, as specified by the net lists, aligned or pitch-

matched. Cells are aligned so that the distance between two adjacent terminals on a cell boundary is made equal to the distance of their mates (the terminals on the boundary of adjoining cell). It may involve restretching of the compacted cells. After this step, the mask-level representation is generated for the pitch-matched cells. Note that each leaf-cell is compacted only once and different mask versions are generated if different pitch-matching constraints are present.

1.5.3 Cell Assembly

The function of cell assembly is to combine the compacted leaf-cells as specified in the floorplan, into a larger structure (called a module) and to make inter-cell connections that are specified symbolically. The cell assembler performs many functions: i) it enforces proper separation between cells to avoid any inter-cell design rule violation; ii) it routes wires to interconnect terminals, if necessary, and iii) it reduces the area needed for the assembled cell. Terminals on two adjoining cells can be interconnected either by keeping all the routing wires straight (pitch-matching) or by jogging some of the wires.

Two of the popular methods for full-custom leaf-cell generation are i) the gate-matrix style, and ii) the Row-based layout style.

1.5.4 The Gate-matrix Style

The salient feature of this style is that it has a fixed layout style in which poly-silicon (poly) wires run vertically carrying the gate signals; and rows of p-diffusion (P) and n-diffusion (N) layers are laid out horizontally. Transistors are formed at the intersection of diffusion rows with the poly columns. Metal wire segments are placed horizontally to connect terminals of transistors placed in P and N rows. Horizontal metal wire segments that run across the whole layout are used for power and ground signals. Figure 1.2a shows a circuit that computes the complement of the function F = A + (A. B) + (B. C. D) and its gate-matrix representation is shown in Fig. 1.2b.

The major advantages of this style are: i) more flexibility is available to the designer under fixed style, when compared to gate-arrays and standard cells, and ii) each column can have more than one transistor, i.e., sharing of gate signals is possible.

This style suffers from the following: i) since the height of the cell (maximum height of any column in the virtual grid) depends on the maximum number of transistors in any column, and if a few columns have a comparatively larger number of transistors than the others, it will

result in an unacceptable amount of unused area or white space; ii) the width of the cell layout depends upon the number of gate signals in the circuit. Therefore, it is difficult to maintain the aspect ratio of the cell.

1.5.5 The Row-based Style

In the single Row-based style, transistors are laid in two rows, one containing the pMOS and the other containing the nMOS transistors only. The pMOS and nMOS transistors which share the same gate signals are paired generally and are placed such that their gate signals are aligned vertically. Horizontal metal segments are used to connect terminals of different transistors. Power and ground signals run across the circuit in horizontal metal wires. Since there is only one row of nMOS and pMOS transistors, the height of the cell is generally small. Hence, the primary goal of a Row-based style generator, is to reduce the width of the cell by reducing the number of breaks in the diffusion strip of the two rows. The minimization is achieved by ordering the transistor pairs so that there is maximum abutment of transistor terminals. The major difference between the gate-matrix style and the Row-based style is that in this style, only one row of pMOS and nMOS transistors per cell is used. Figure 1.2c shows

the layout obtained using this approach for the circuit given in Fig. 1.2a.

Advantages of the Row-based approach are as follows: i) the steps involved in the cell generation are simple, and ii) it provides good abutment of transistors thus reducing the width of the cell. Major disadvantages of this method are: i) only one pair of transistors (n and p) can be present, i.e., sharing of gate signals is not possible; ii) minimization of width using abutments is the sole optimization goal, i.e., they perform one-dimensional optimization, and iii) each cell is developed without giving consideration to the other cells. Due to this, inter-cell routing can become very complicated. The disadvantages of the Row-based approach are further illustrated with examples in the next section.

1.6 The Thesis Problem

A detailed discussion on some of the existing Row-based cell generation algorithms is given in chapter 2. This section provides a brief summary of Row-based cell generation methods and proposes an extension of this style.

1.6.1 Our Approach

After reviewing the strengths and drawbacks of the two styles--the gate-matrix style and the single Row

based style--it is clear that we need to develop a layout style which incorporates the strengths of both styles and at the same time, avoids the common weaknesses. As a result, we propose a multiple Row-based layout style. Our objective is to i) minimize the width of the cell by maximizing abutment as in the case of the single Row-based approach, and ii) at the same time promote sharing of common gate signals between successive rows as it occurs in the case of the gate-matrix method. Figure 1.3 shows the proposed layout style.

In the Row-based approach, the given circuit is partitioned into blocks, and then each block is realized separately in the form of a row of transistors. Then the transistors belonging to adjacent rows are connected by using a channel router. Naturally, each row is developed without considering the impact of the placement of the devices on the entire cell. As a result, the possibility of sharing gate signals among adjacent rows can not be considered. In general, it leads to complicated inter-row routing which could be very costly area-wise. This fact is illustrated by an example. Figure 1.4a shows a full adder circuit, and Fig. 1.4b shows the cell developed by following the single Row-based approach. The circuit is divided into two rows, and by using the Eulerean paths (explained in chapter 2), the placement of the transistors in each row

is obtained independently. After the two rows are laid out the common gate signals are routed. The area of the resulting layout is 160 square units, where the area is defined as the number of columns multiplied by the number of rows on the virtual grid.

Figure 1.4c shows yet another layout of the same circuit which requires an area of 128 square units. The gain in the area is obtained by developing the two rows together and by keeping the inter-row routing in mind. It shows that multiple Row-based style for cell generation is advantageous over the single Row-based style.

Most of the existing Row-based cell generators do not consider the density of the intra-cell routing directly in the consideration of the complexity of their algorithms. Due to this reason, they may not produce optimal layouts of even some simple circuits. This fact is illustrated by the following example. Figures 1.5a and 1.5b show the gate-level and transistor-schematic representation of a 4x2 AOI (and-or-invert) [TTL 7454] logic gate. In the associated graph (a, c, d, g, h, f, d, b) is an Eularian path for both n- and p-graphs, and therefore, many Row-based cell generators will produce the layout shown in Fig. 1.5c. Although this layout contains only one contiguous diffusion strip in each row, it needs seven tracks for making intra-cell connections. However, the layout shown

in Fig. 1.5d has a break in the diffusion strips but requires only four tracks. This illustrates the fact that a layout which has the minimum width does not always produce a layout of minimum area.

In this research we consider a synthesizer that follows a multiple Row-based layout style. We were motivated for this research by the fact that in custom circuit environment, by following the multiple Row-based style, a designer can align the gates of transistors that share common signals. Presently, most cell synthesis algorithms are targeted towards the development of standard cell libraries, where the single Row-based style provides maximum flexibility. The advantage shown by the previous example has not been exploited in the standard cell environment because most logic partitioning tools which partition the logic into rows do not consider this possibility. Present partitioning tools which are based on clustering or min-cut approaches, assign the transistors in the same row and connect them by a horizontal wire. We assume a partitioning tool which would divide the circuit into different rows in such way that the sharing of gate signal among different rows is enhanced. (The goals of the partitioning tool are described in more detail in chapter 2.)

We observed that there are many circuits which are neither fully structured so that they could be realized

as a PLA, ROM, etc., nor are they fully random as seen in the controllers of many finite state machines. Data paths of many processors usually contain such circuits or components. For example, comparators, multiplexer, ALU function generators, etc. (see chapter 2 [10]), contain sufficient regularity, and at the same time have a certain amount of randomness. The characteristics of these circuits are: i) several input signals are common to many logic gates; ii) there exist multiple instances of several components which can be synthesized independently, as it is done in the standard cell environment, but at the same time the whole circuit can be generated as a single module, and iii) the circuits contain multiple levels of combinatorial logic gates which produce only a few output signals. Since most present-day cell generators can handle cells with no more than 100 transistors, their users will divide the circuits into smaller components and will then use standard channel routers to interconnect the layouts of the components. Undoubtedly, if the whole circuit is generated as a single module, it will be area-wise more efficient.

For this research, we propose to formulate an optimization problem for cell generation that not only considers the minimization of the number of diffusion strips but also concentrates on the wirability and on the track density of the routing channels. We propose to

develop a CAD tool which will run in two phases: i) placement, and ii) net realization. The first phase will obtain the placement of transistors in each row. During the second phase, the placement may be modified to make room for nets connecting non-adjacent rows. We propose to consider a cost function that will consider the width of the diffusion strips, the number of routing tracks, and the wire length. This cost function will be used to guide the search through the problem space. We plan to use the simulated annealing technique for solving the optimal placement problem.

1.7 Organization of the Thesis Report

Chapter 2 presents a survey of literature related to the Row-based cell generation methods. Existing techniques and results that have motivated our research are explained in detail. It contains a complete formulation of the multiple Row-based approach. It also gives a description of the input and output specifications, the layout model for the multiple Row-based style and the routing channels.

Chapter 3 describes the use of simulated annealing technique used for optimization of the cost function of the layout. It also explains the method of generation of n- and p-graphs for the given circuit, obtaining Euler

paths, the concept of delayed binding and the data structures used for storing the circuit information.

Algorithms developed for intra- and inter-row routing and the data structures developed for our cell generator are given in chapter 4.

Chapter 5 contains the results obtained and comparisons with the single Row-based approaches. Some conclusions and our contributions to Row-based cell synthesis are also listed.

The appendix describes the software developed for the CAD tool. Examples of its usage are also presented. The input description and the method of viewable output generation are described.

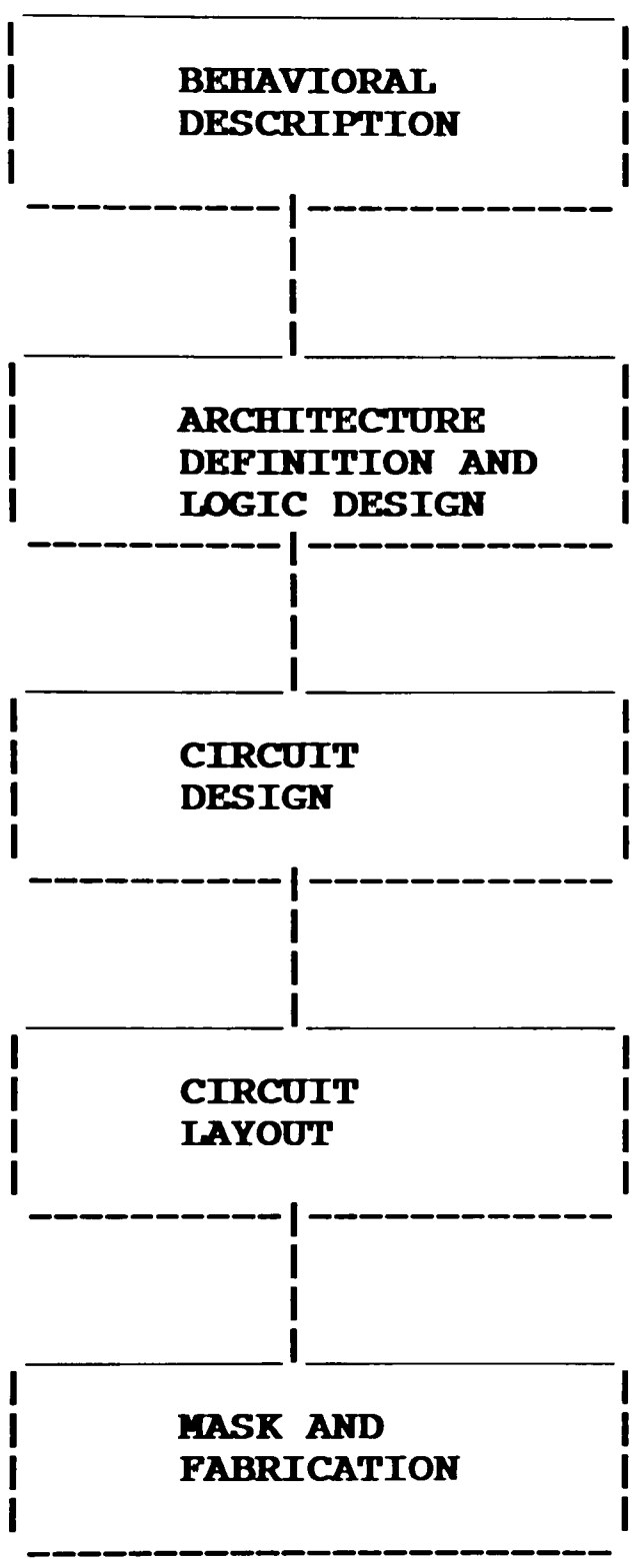


Fig. 1.1 The Chip Design Process

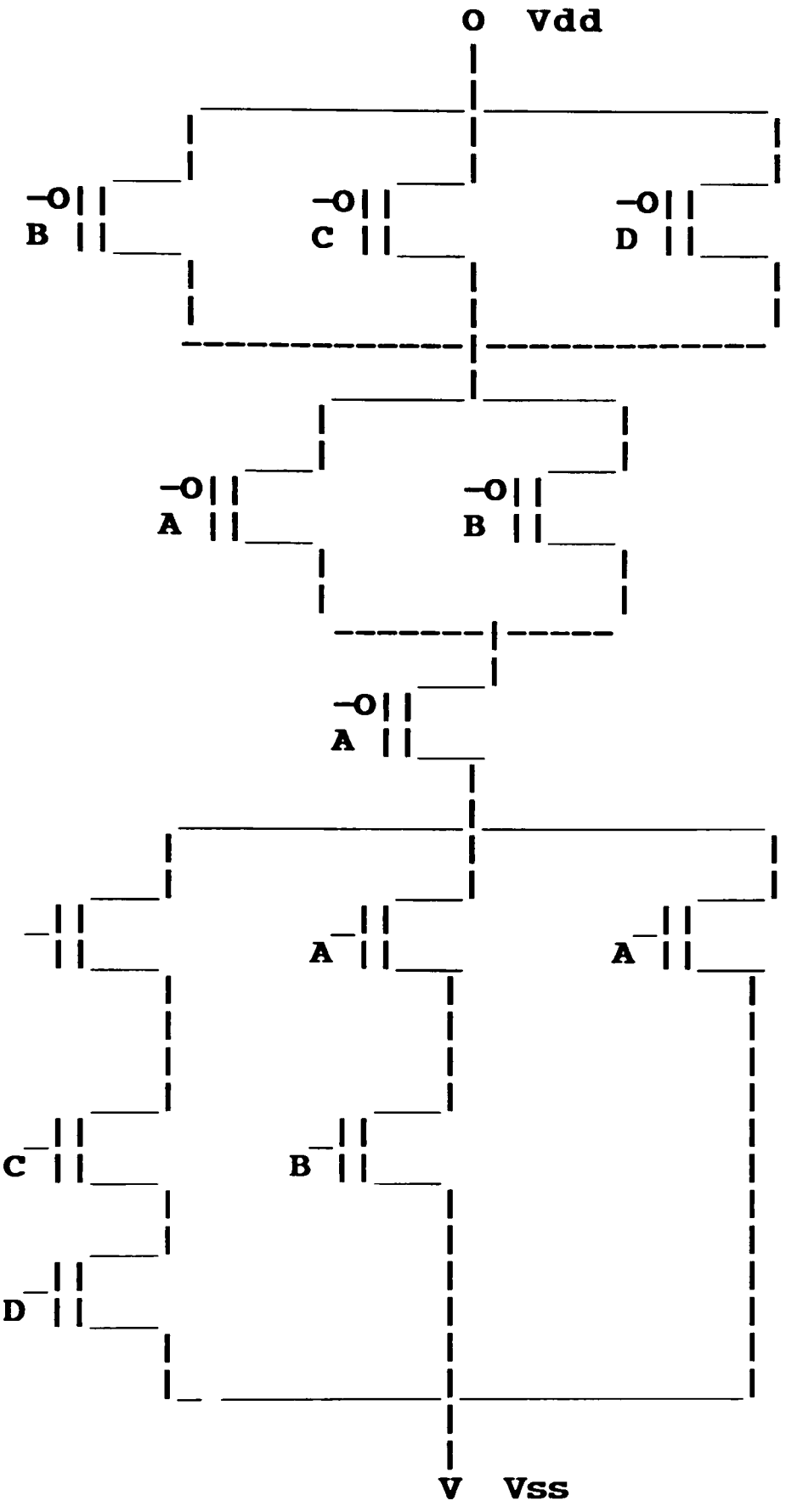


Fig. 1.2 The Layout Styles

(a) $F = \overline{(A + (A * B) + (B * C * D))}$

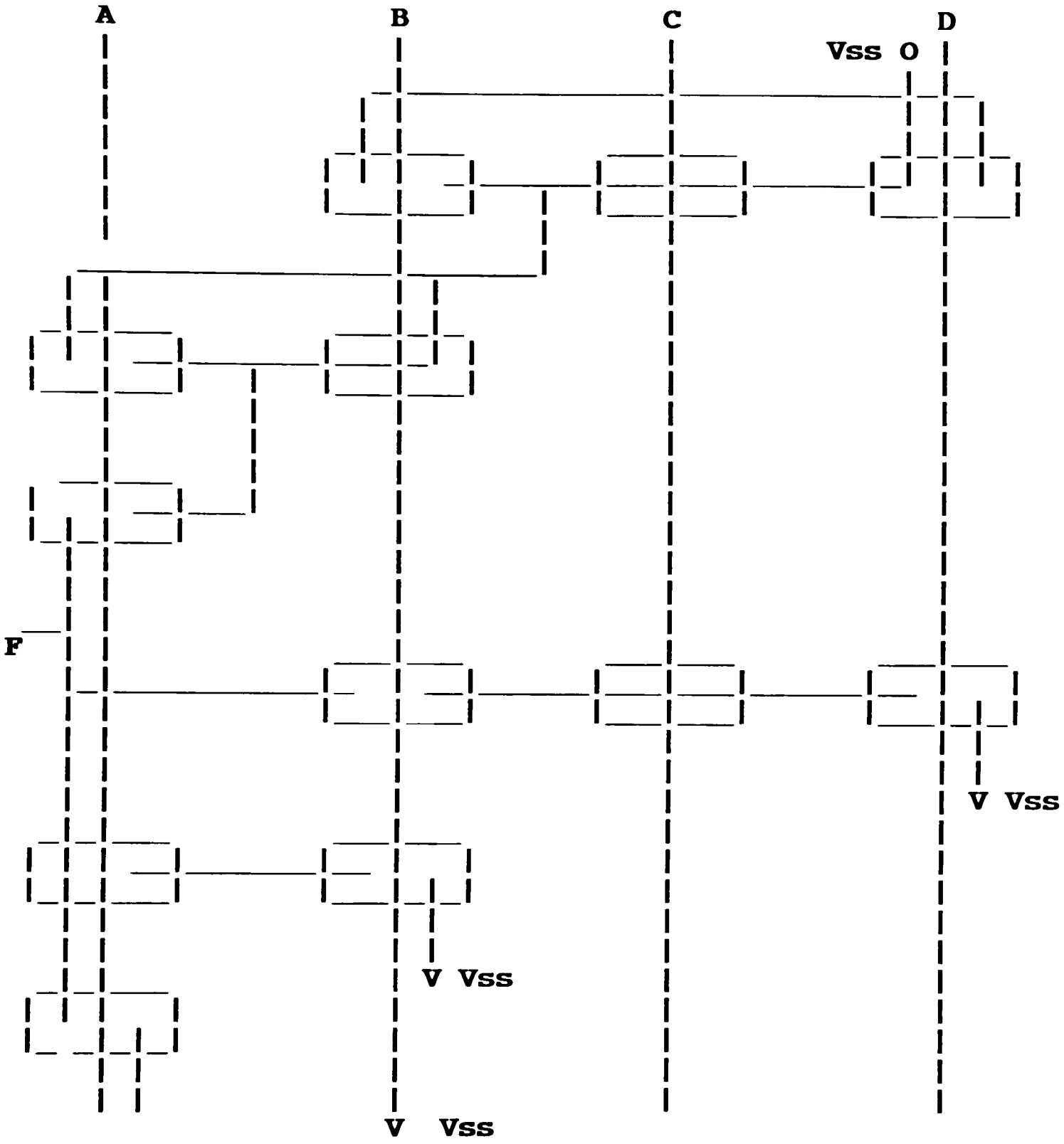


Fig. 1.2 (cont'd.)
(b) Gate-matrix Layout

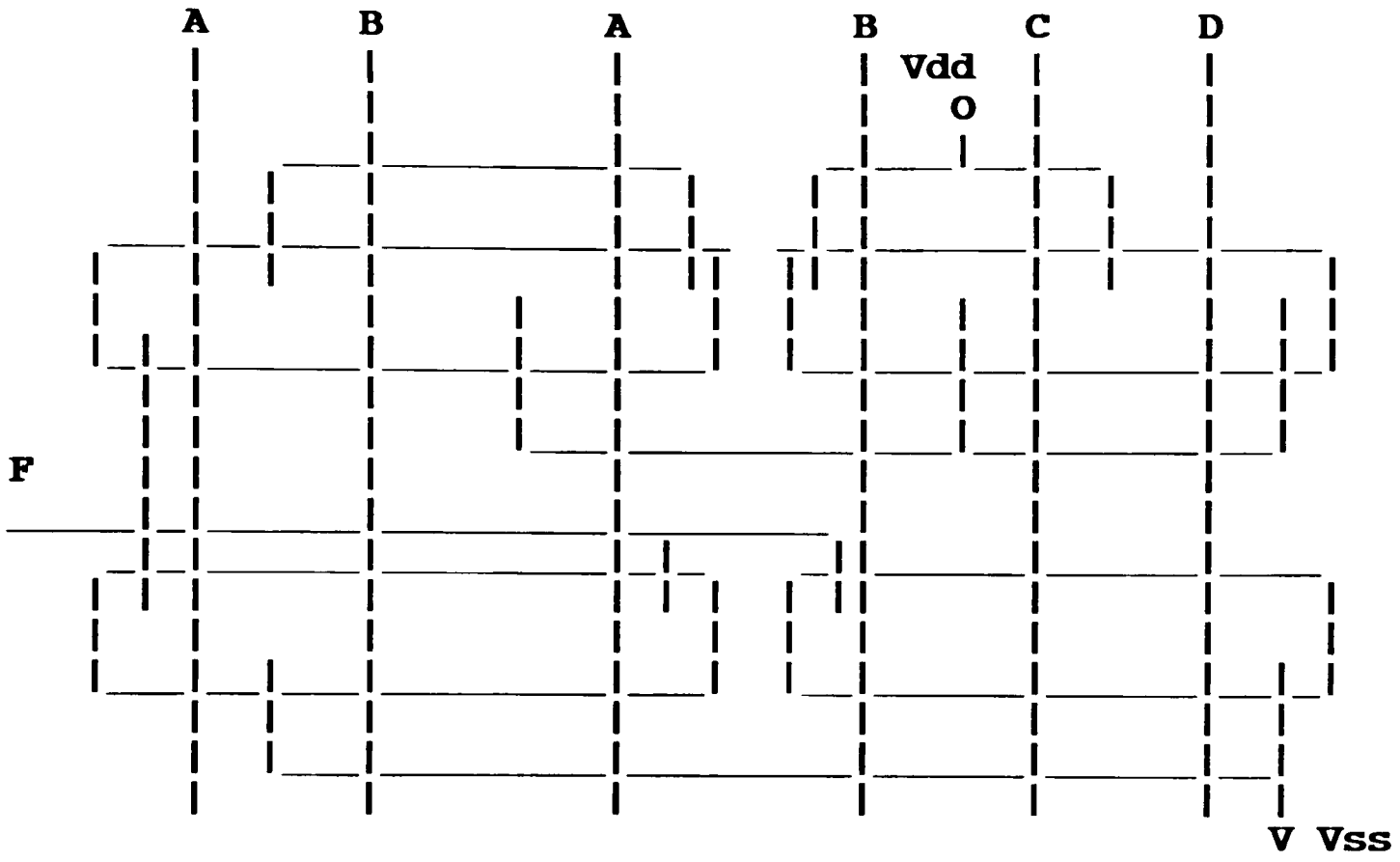


Fig. 1.2 (cont'd.)
(c) Row-based Layout

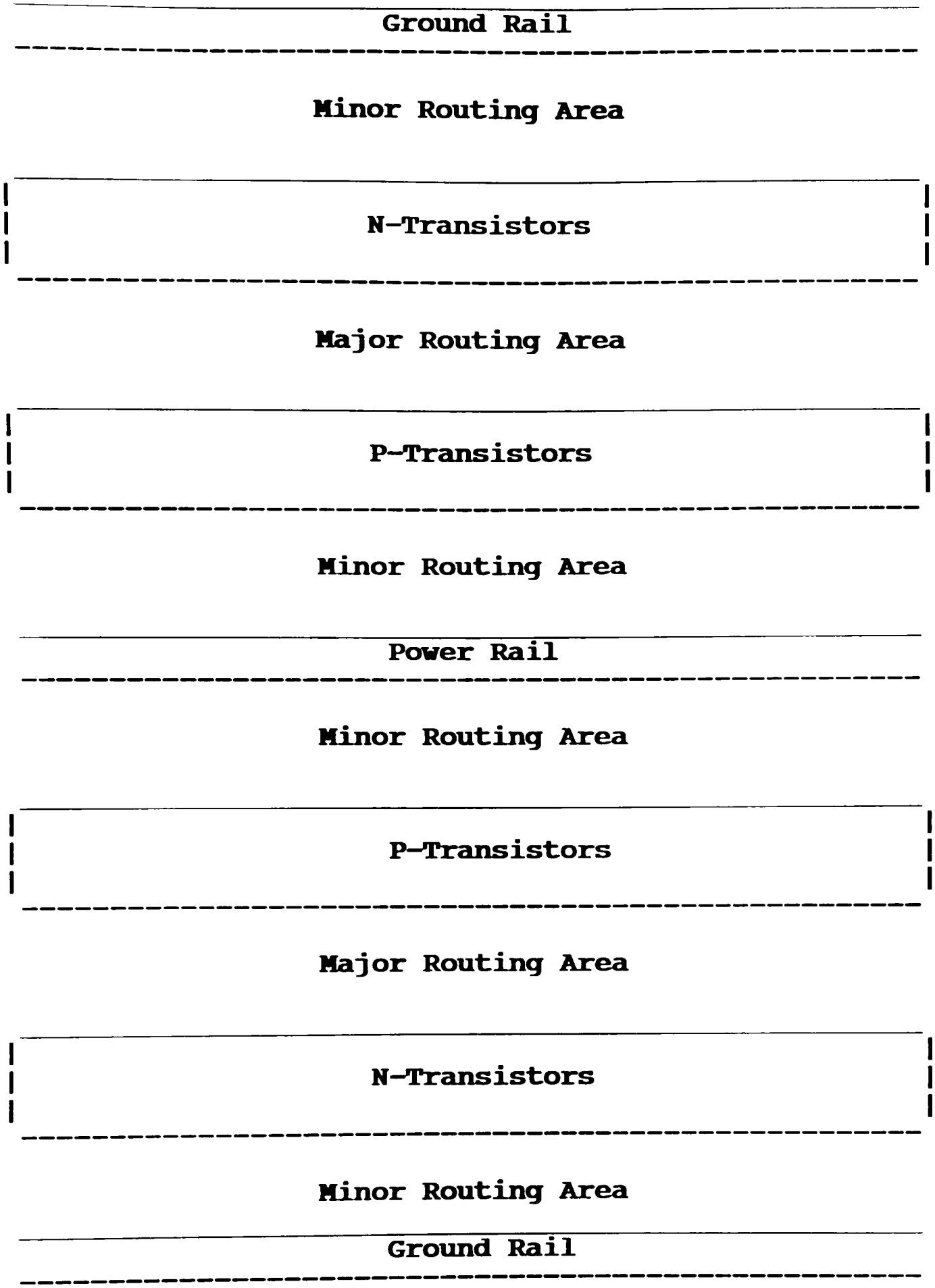


Fig. 1.3 The Multiple Row-based Layout Style

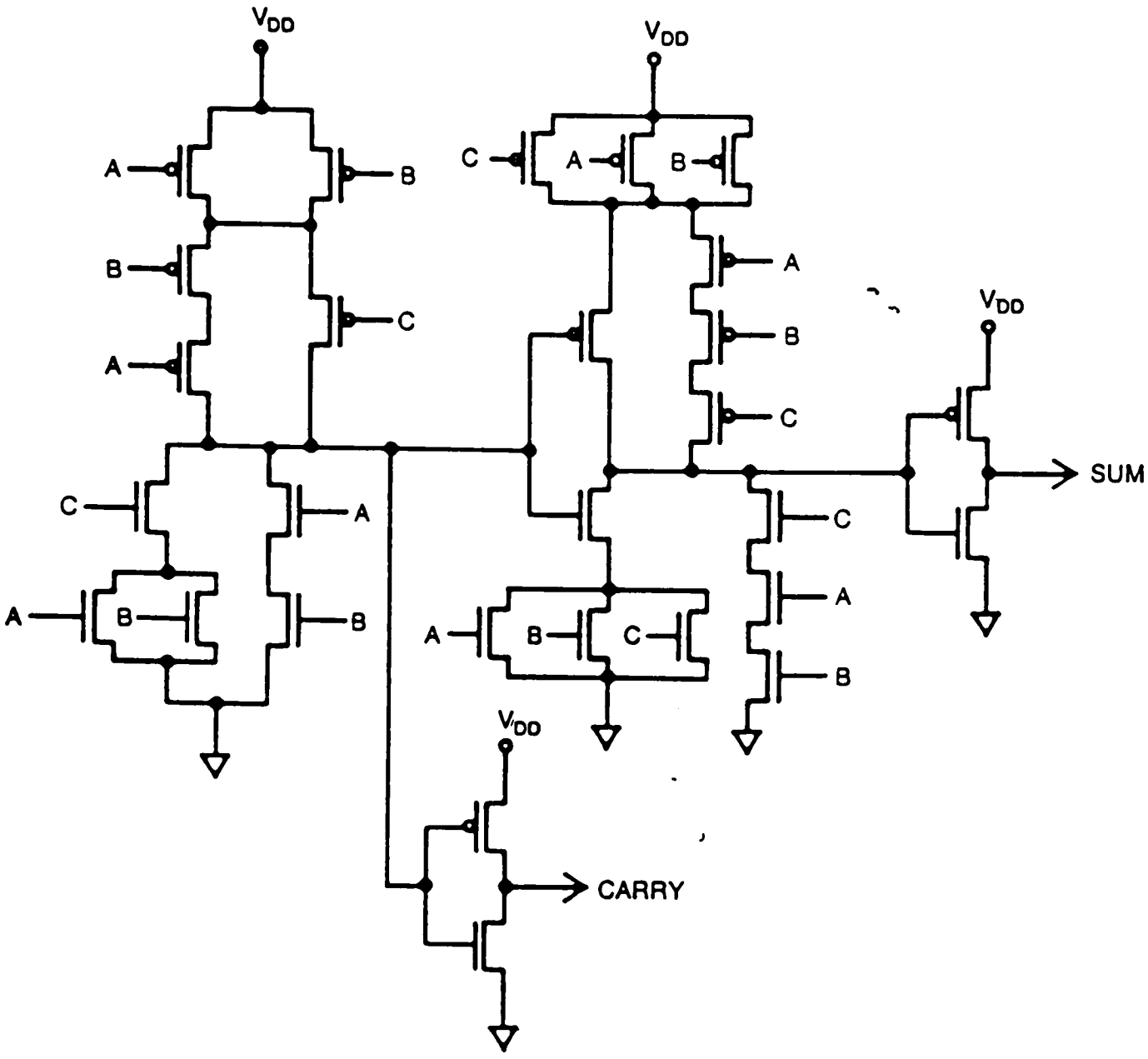


Fig. 1.4 A Full Adder
(a) Schematic

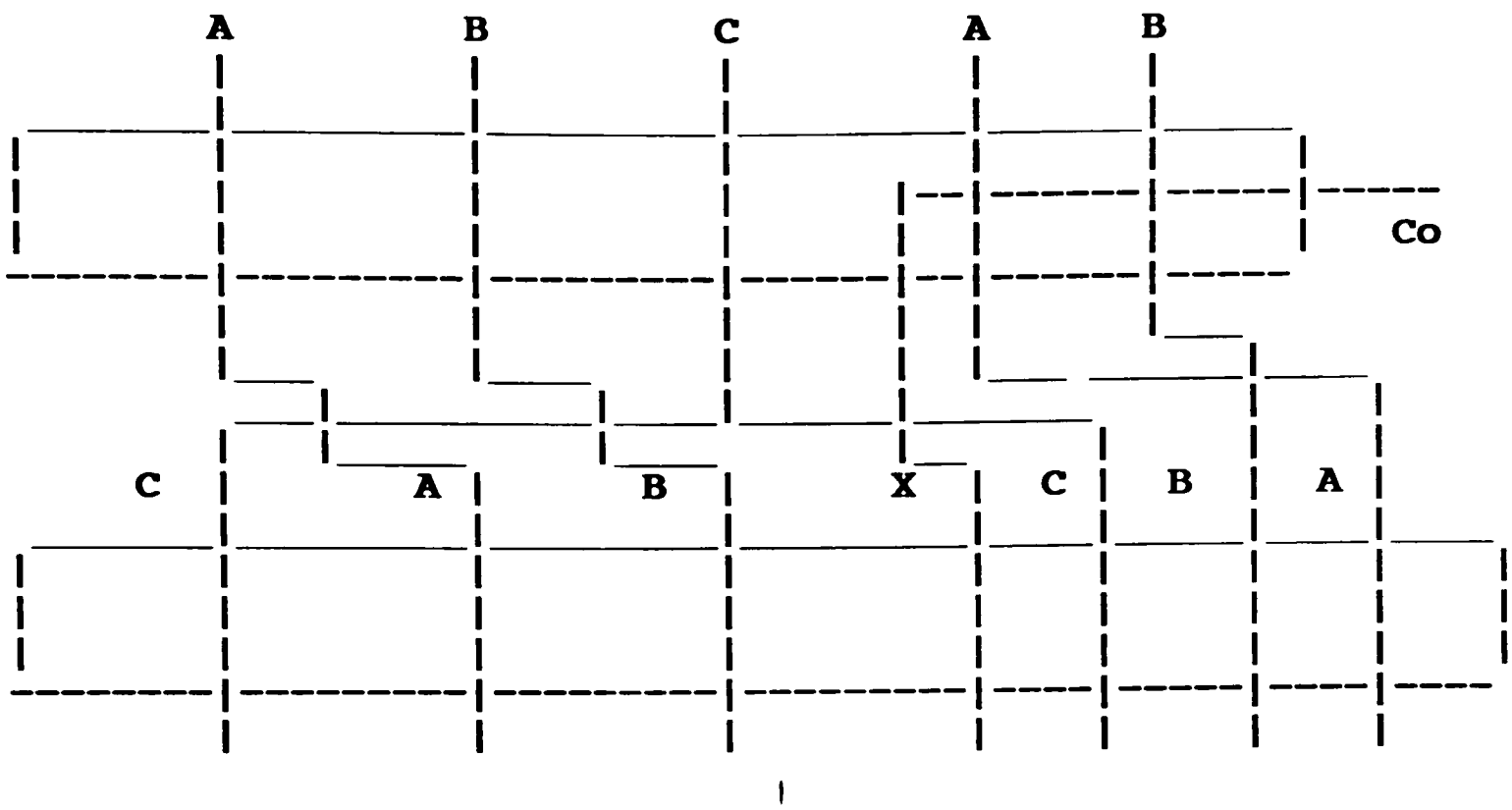


Fig. 1.4 (cont'd)
(b) Ignoring Inter-row Routing

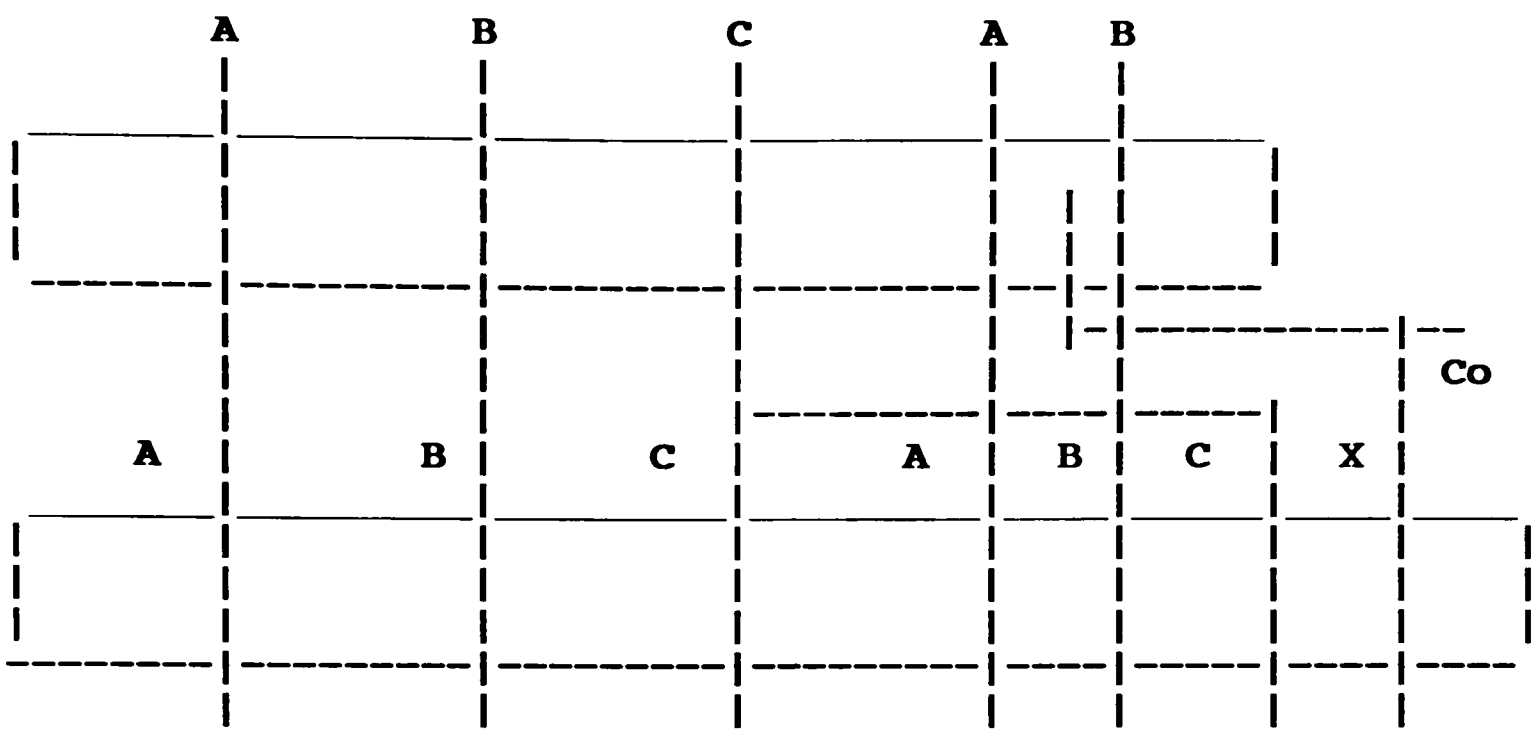


Fig. 1.4 (cont'd)
(c) Considering Inter-row Routing

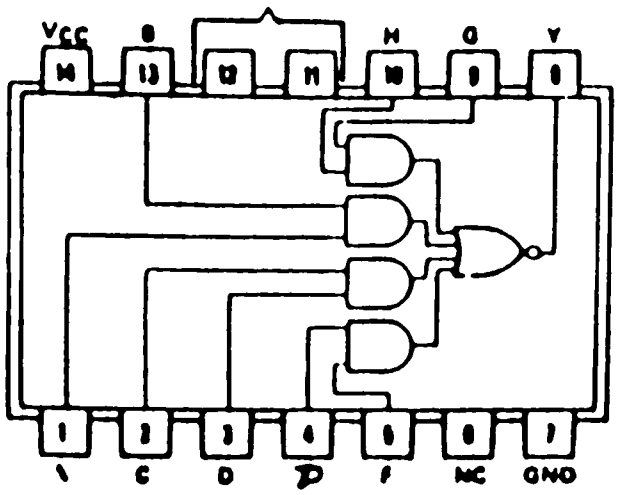


Fig. 1.5 4 x 2 AOI TTL 7454
(a) Chip Design

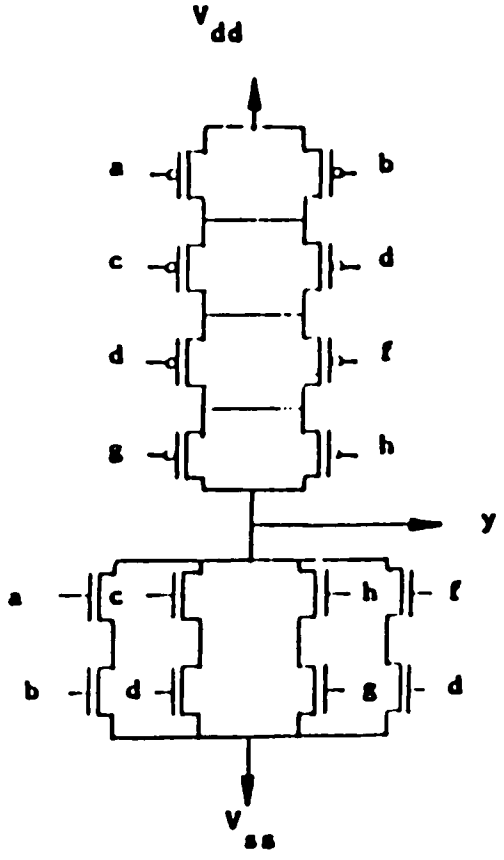


Fig. 1.5 (cont'd.)
(b) Schematic

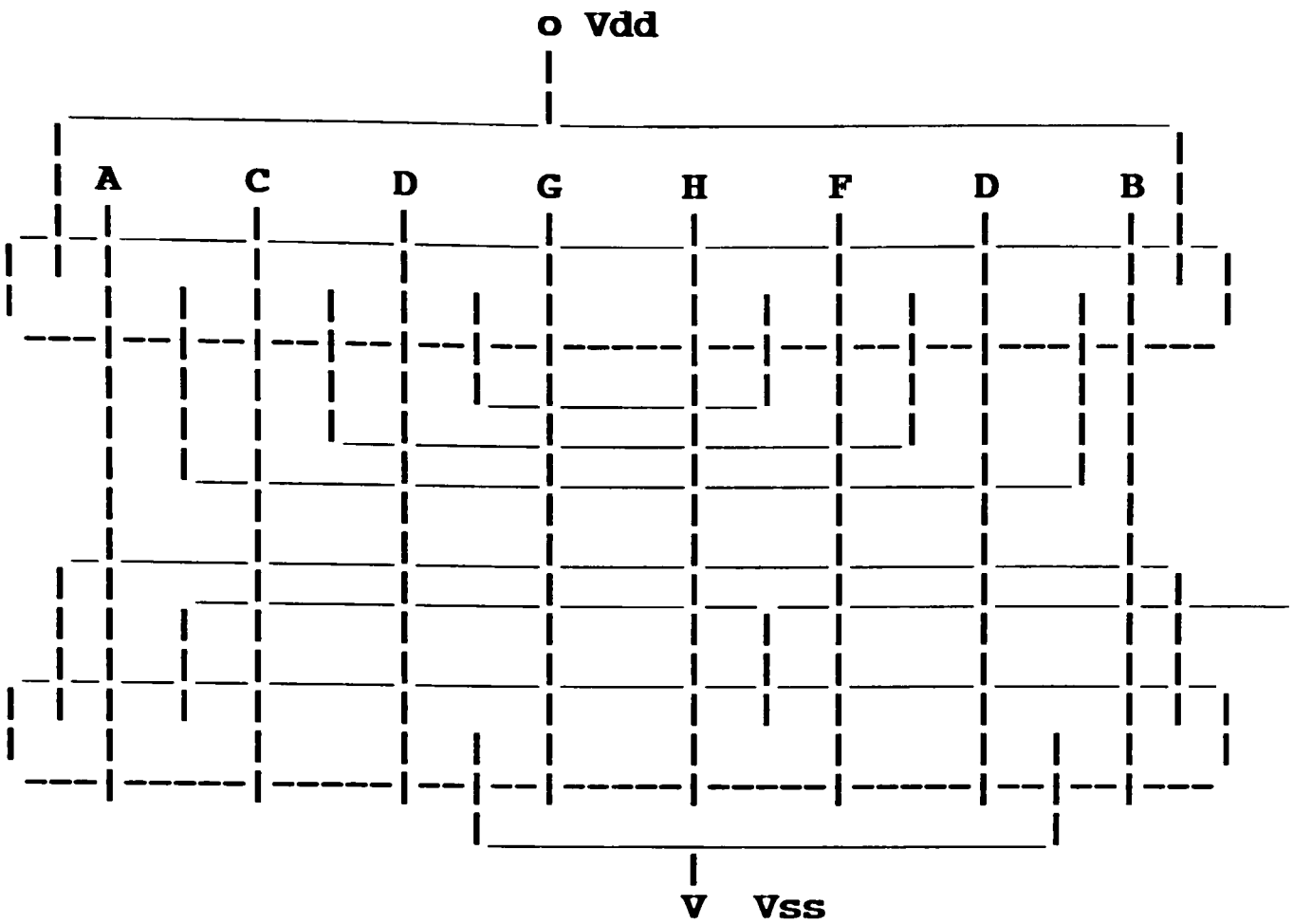


Fig. 1.5 (cont'd.)
(c) A Seven Track Layout

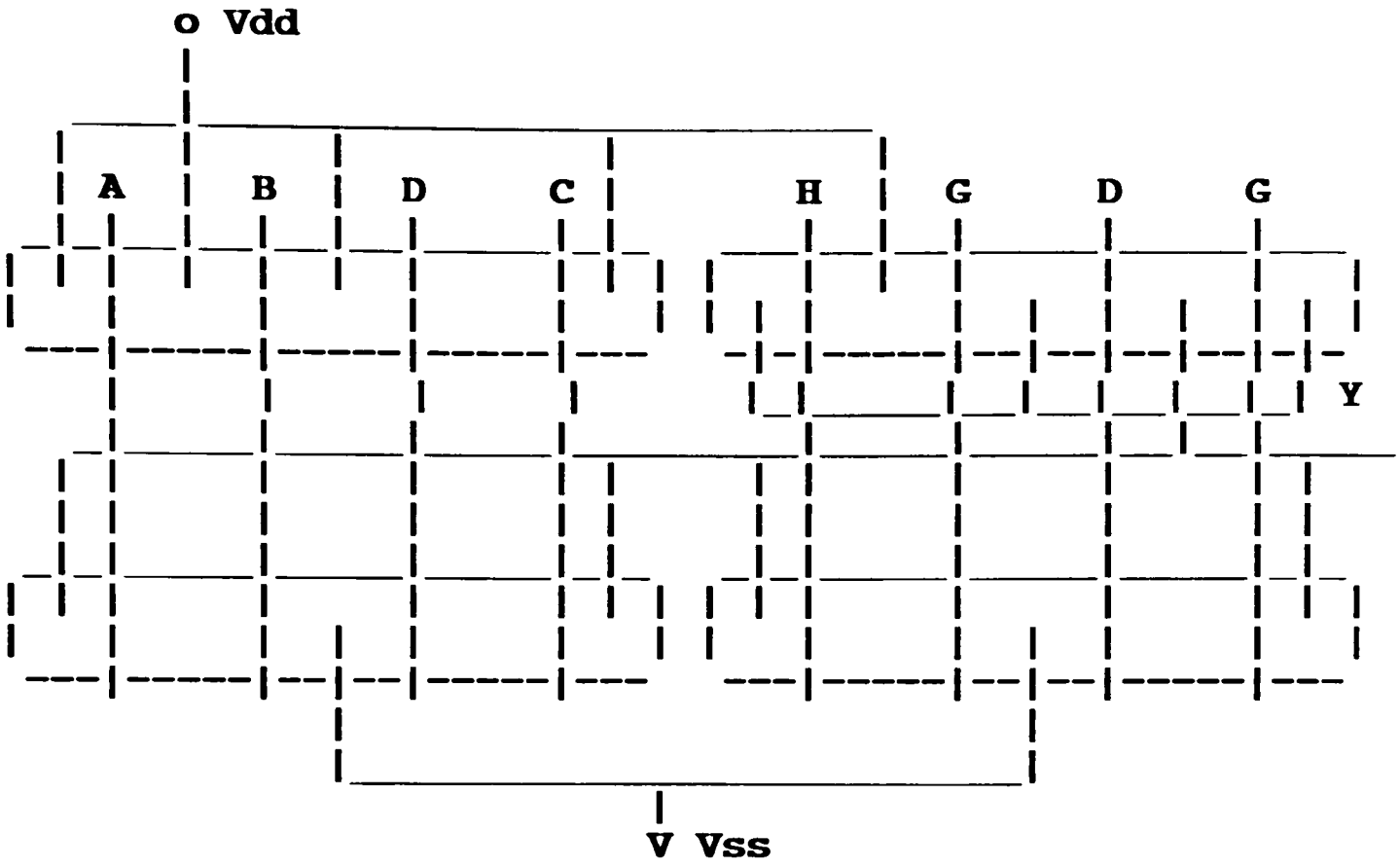


Fig. 1.5 (cont'd.)
(d) Four Track Layout

CHAPTER II

LITERATURE SURVEY AND PROBLEM FORMULATION

Automatic cell synthesis based on the Row-based approach has received a lot of attention from the custom and standard-cell development researchers. This chapter provides the necessary material to understand the work done by researchers in the field of Row-based cell generation. Discussion of the work in this area starting with the pioneer work of Uehara and Van Cleemput [1] up to the very recent work by Chen and Hou [15] involving delayed binding is provided. The main advantages and disadvantages of these methods are pointed out. The popular algorithm for finding the Euler paths in a graph and the simulated annealing technique for optimization are introduced.

This chapter also gives the complete formulation of the research problem. The specifications of input and output languages are given, and the proposed layout style and the placement model are described.

2.1 Euler Paths in Row-based Cell Design

The important step in the Row-based cell synthesis is finding the placement of transistors which occupy

the minimum width. This is done by finding the Euler paths for the circuit as described below.

The CMOS circuit is converted to a graph where i) the verticies in the graph are the source/drain connections, and ii) the edges in the graph are transistors that connect particular source-drain verticies. Two graphs, one for the n-logic tree and one for the p-logic tree, result. Figure 2.1a (Chapter 5 [10]) shows a simple circuit and Fig. 2.1b shows its graph transformation. The connection edges in the graphs mirror the series-parallel connection of the transistors in the circuits. Each edge is named with the gate signal name for that particular transistor. Thus, for instance, the p-graph has four verticies: Z, V1, V2, and Vdd. It has four edges, representing the four transistors in the p-logic structure. Transistor A (signal A connected to gate) is an edge from vertex Z to V2. The other transistors are similarly arranged in Fig. 2.1b. Note that the graphs are dual of each other as the p- and n-trees are the dual of each other. If two edges are adjacent in the p- or n-graph, then they may share a common source-drain connection and may be connected by an abutment. Furthermore, if there exists a sequence of edges (containing all edges) in the p-graph and n-graph that have identical labeling, then the gate may be designed with no breaks. This path is known as an Euler path. The main

features of the algorithm are as follows:

1) Find all Euler paths that cover the graph.

2) Find p- and n-Euler paths that have identical labeling (a labeling is an ordering of the gate labels on each vertex).

3) If 2) is not found, then break the gate in the minimum number of places to achieve 2) by separate Euler paths.

In the example shown in Fig. 2.1a, the original graph with a possible Euler path is shown in Fig. 2.1c. The sequence of gate signal labels in the Euler path is <A,B,C,D>. Note that the graph for the n- and p-graphs allows this labeling. To complete a layout, the transistors are arranged in the ordering of the labeling, n- and p-transistors in parallel rows, as shown in Fig. 2.2. Vertical polysilicon lines complete the gate connections. Metal routing wires complete the layout.

2.2 Literature Survey

In their seminal paper [1], Uehara and Van Cleemput suggested a layout paradigm for CMOS circuits that is geared towards minimizing the diffusion required in the artwork. Their primary concern was to chain both the p-type and n-type transistors in such a way that their source or drain ports abut as much as possible, thereby also minimize

the need for additional inter-cell routing. In order to make the most of this method, good algorithms are needed to generate chains that maximize the occurrences of diffusion ports adjacencies, and then find the smallest number of chains that realize the circuit.

The functionality of such an algorithm is to map a circuit diagram (topology) into a layout (geometry). Normally, the circuit given is a graph, describing the connectivity of the transistors. The layout is produced as a stick diagram which can then be instantiated into mask data by a circuit compactor. Their algorithm was based on the minimization of width based upon the Euler paths in the circuit. They ensured the existance of the Euler path by making the number of inputs to all the gates in the circuit odd. They inserted pseudo inputs (transistors) whenever they came across a gate with even inputs and proposed methods to reduce the number of breaks generated due to these inputs. The algorithm suggested requires that the n-part and the p-part of the circuit graph be mutual duals. The contribution of factors other than the width is not considered towards the efficiency of the layout.

Notable among the research that followed the aforementioned work, was that of Wimer et al. [4]. In this work the constraints placed on the graph earlier were removed. A general algorithm that could handle arbitrary

graphs and produce optimal results was developed. This was done by allowing p- and n-type transistors to share a column (i.e., have the same position in the chain), not only when they are complementary, but also when they have a source or drain port in common. Their technique resulted in longer chains of transistors.

One of the major drawbacks of their approach was that they assumed that the minimum width of the cell would always result in good layout. Contribution of many of the other factors, such as the number of tracks, wire length, etc., were not considered.

Shirashi et al. [7] developed a diffusion island couple algorithm which considers the wire length also in the cell synthesis. The basic idea is that, until all logic gates are recognized, preference is given for minimization of the number of diffusion separations. After the recognition of all the logic gates, reduction of total wire length is given preference. Circuits which consisted of an unequal number of p- and n-transistors and circuits with transmission gates could also be synthesized using their algorithm.

The researchers mentioned earlier assumed that minimum width cell will result in a cell which has the optimum area. It was illustrated in chapter 1 that in many cases a cell with the minimum width may result in

larger heights causing a larger area because of the complicated interconnections. This fact was noted by Stauffer and Nair [5] and was exploited in their work. They proposed algorithms for cell synthesis which were based on the relaxation approach. They start with an initial layout and move towards an optimal placement by introducing series of local perturbations generated in some systematic manner.

In a CMOS complex gate, permutation of some transistors may not affect the realized logic function, but may affect the layout significantly. This concept of postponing the binding of transistors until we see some clear advantage is known as "delayed binding" and it was adopted by Chen and Hou [15] in their work. Failure to use this concept will result in binding decisions before sufficient global information is available, thus over-constraining the layout optimization and producing less than real optimal results.

It is apparent from the above discussion that if the methods discussed earlier are used, we will have a layout which comprises of a single row of n- and p- transistors. But in the real world there will be very few situations which call for such a layout. Cells of different aspect ratios (different shapes) will be required

rather than those with a single row. Under such circumstances, the entire circuit is divided into several subcircuits and several cells are obtained by applying the algorithms described above. But each cell is synthesized completely independent of the others and the required logic is realized by performing the routing between different cells. This will cause an increase in the area of the layout because of the complications in routing that are introduced.

In the methods described earlier, minimization of width is obtained by eliminating as many breaks in the diffusion rows as possible. Inspite of that, there will be a large number of breaks which are a natural consequence of the logic that is implemented by the circuit. These breaks do not play any role in the layout other than contributing to the width of the cell.

These layouts do not consider the possiblity of using metal strips for interconnection in the same track as the diffusion layer. This could result in considerable saving in the height.

None of the methods described, with the exception of that of Chen and Hou, try to reduce the number of tracks that are used for interconnection or the wire length. They also do not consider the routing of power and ground signals in the minimization phase.

In our proposed method the placement procedure itself is based upon the aspect ratio of the cell. Because of this we will be able to consider the inter-row routing problem and the optmization possible in that area during the placement stage.

The drawback of having breaks in the diffusion layer as described earlier is turned into an advantage in our proposed method. We use these breaks for the purpose of inter-row routing of gate signals. We identify the Euler paths which require a lesser number of metal tracks than the other possible Euler paths to reduce the height of the cells.

2.3 Simulated Annealing

Research in the area of combinatorial optimization aims at developing efficient techniques for finding minimum or maximum values of a function of very many independent variables. This function, usually called the cost function or objective function, represents a quantitative measure of the "goodness" of some complex system. The cost function detailed configuration of the many parts of that system.

Statistical mechanics is the study of the behavior of complex systems consisting of a large number of

interacting atoms in thermal equilibrium at a finite temperature. In particular, the atomic states at any temperature T satisfy the Boltzmann's distribution, namely, the probability that a system is in a given state "r" is given by exp [E(r) / Kb T] where E(r) is the energy associated with state r, and Kb is the Boltzmann's constant. Hence, under equilibrium, the most probable states at any given temperature are those with the lowest energy.

The analogy between a combinatorial optimization problem and the problem of determining the lowest-energy ground state of a physical system with many interacting atoms was first observed by Kirkpatrick et al. [12]. It is well known that to bring a fluid to a highly ordered, low-energy state (for example, in growing a single large crystal), a process called careful annealing can be employed. We first melt the system by heating it up to some high temperature, and then cool it slowly, spending a long time at temperatures in the vicinity of the freezing point. At each temperature during the annealing process, slow cooling enables the system to achieve equilibrium. If the temperature is lowered too quickly, the system does not have sufficient time to achieve equilibrium, and the resulting configuration might have many defects in the form of high-energy, meta-stable, locally optimal structures. Intuitively, fast cooling in physical systems corresponds to an

iterative improvement scheme in combinatorial optimization. Kirkpatrick et al. suggested that better results to combinatorial optimization problems can be obtained by simulating the annealing process of physical systems. The generalized algorithm developed by them is given below.

```
begin
  S := Initial solution S0;
  T := Initial temperature T0;
  while (stopping criterion is not satisfied) do
    begin
       while (not yet in equilibrium) do
          begin
             S' := Some random neighboring solution of S;
             del := COST (S') - COST (S);
             Prob := min (1, exp (- del/ T));
             if random (0,1) <= Prob then S := S';
          end ;
        update T;
       end;
      output best solution;
     end.
```

As seen in the above algorithm, in each step, a small perturbation of the configuration is chosen at random and the resulting change in the cost of the system "del,"

is computed. The new configuration is accepted with probability 1 if del <=0, and with probaility exp [-del / Kb T] if del > 0.

2.4 Problem Formulation

2.4.1 Pre-processing of the Circuit

We assume that the circuit has been partitioned into several rows using a tool based upon certain heuristics which take the following factors into consideration:

1) The number of transistors in each row:

If we divide the circuit such that there is a large difference in the number of transistors in different rows, it will result in a cell in which we have some rows which are much wider than the others generating white (unused) space. But at the same time, if we divide the circuit concentrating only on the number of transistors and break up the circuit without considering the logic that is being implemented, we will have more breaks in the rows than called for and also will have a more complicated inter-row routing.

2) The number of common gate signals:

One of the salient features of our algorithm is the sharing of a gate signal (a column) by transistors belonging to different rows. So, it would be a good idea

to divide the circuit such that the transistors using the same gate signal are distributed in different rows.

2.4.2 Input Specification

The given circuit will be expressed in terms of logic gates such as AND, OR, NAND, NOR, NOT and TG (Transmission gate). Thus the circuit will contain the same number of p- and n-transistors but they need not be graphical duals of each other. Specification of input for a cell implementing the logic, shown in Fig. 2.3 [10], is shown below. The circuit has been divided into three rows based on the principles stated earlier.

```
ROWS 3
OUTPUT A-, B-
INTERMEDIATE a, a-, b, b-, s, d, d-
OP a {phi1} OP a- <a> OP a- {phi2} OP A- ( [ 1, a- ], [ b-,
s] )
# /* end of first row */
OP b {phi1} OP b- <b> OP sq- [ ( [ H, b-, a, d-], [s, (d,
[d-, H-])]), R-] OP sq- {phi2} OP s <sq->
#
OP d {d} OP dq- [b-, (d, [a-, b], [b-, a])] OP dq- {phi2}
OP d <dq->
#
```

The complete description of the input specification language is given in the appendix.

2.4.3 Layout Style

Figure 1.3 shows the layout model that we propose to adopt for the cell generator. The cell will consist of several pairs of rows of n- and p-diffusion layers laid out horizontally. Poly-silicon (poly) wires running vertically generate transistors when they intersect with diffusion layers. A single layer of metal running horizontally will be used for making interconnections. Short vertical strips of p- and n-diffusion will be used to make the vertical connection from the diffusion strips to the metal layer. Horizontal metal wires running across the cell will carry power and ground signals. Separate channels will be identified for intra-and inter-row routing and designated as the major and minor channels, respectively. The major channel will also be used for routing the outputs generated to the pins and the minor channel for carrying the power and ground signals. The input pins will be at the top and bottom; the output pins on one of the corners.

If we have three rows in the cell, then the first row will have n-layer followed by p-layer, the second row will have p-layer followed by n-layer, the third row will have n-layer followed by p-layer. Vss above the n-layer of the first row and in between the n-layers of the second and the third rows. Vdd will be in between the p-layers of the

first and the third rows, and also below the p-layer of the third row.

2.4.4 Output Specification

The cell generator will produce a complete symbolic representation of the cell in ABCD (A Better Circuit Description) language. ABCD is a textual language and was developed to describe the virtual grid representation of the cells [11]. The layout can then be displayed/plotted using graphics tools such as AUTOCAD or the Berkely tools on the Sun workstations. The cell can then be compacted using a virtual grid compactor and be assembled with other cells to produce a complete module [14].

2.4.5 Placement Model

As stated earlier, the input circuit will be partitioned into functional cells so that one or more functional cells can be realized in each row of the layout floor. Then the transistor level schematic of each functional cell is obtained. Transistors are then placed within each row. A placement is called legal if it is possible to orient every transistor such that the diffusion connection between every two consecutive transistors corresponds to a legal interconnecting net. For example, the

following is a legal placement of transistors for the circuit shown in Fig. 2.3. The circuit is divided into three rows based on the heuristics discussed earlier in the chapter, and the gate level description of the three rows is shown below.

{TG1} * <a> {TG2} * M1 * {TG2} * M2

{TG1} * <b> M3 * {TG2} * <sq->

{TG1} * {TG1} * <d> M4 {TG2} * <dq>,

where M denotes a macro and the macros are given below:

M1 is 1 a- * b- sq * *

M2 is sq a- * b- 1 * *

M3 is H b- a d- * H- d- * d * s * R- *

M4 is a- b * b- a * dq H *.

The placement has been obtained using Uehara's and Van Cleemput's algorithm [1]. { } denotes a transmission gate, < > denotes an inverter, a- denotes the complement of a, * denotes a break (or a dummy transistor) and M# denotes a macro. Any transistor, including a dummy, can be legally placed next to a dummy transistor. Note that each row denotes a pair of rows of n- and p-transistors.

For a given placement of transistors, the algorithms for finding the additional dummy transistors required to legalize the placement is given in [5], but it is applicable only for two rows of transistors. Note that

each break between two transistors in series represents a horizontal wire segment (this will be realized either in metal or diffusion depending on whether the dummy gate is used for routing a gate signal), and a break between two transistors in parallel represents an actual break. One of the goals is to find a placement so that the total number of breaks in all the rows is minimal and that the breaks are evenly distributed among the rows. A legal placement with all breaks in one single row is by far inferior compared to a placement which evenly distributes the breaks among the rows because in the latter case the breaks can be used for routing.

2.4.6 Track Density of Channels

Although the goal of the placement procedure is to improve the wirability, it affects the routing in an indirect way. Our goal is to consider the wirability directly. In our cell generator, we propose to implement it by fixing specific channels for different types of nets even before the placement procedure is started. We will compute an approximate cost for every ordering of the transistor.

In order to estimate the layout area, we will compute the area of each routing channel. In Row-based cell synthesis, the major portion of the nets are the

horizontal segments which are laid out in metal layer. If we assume that each wire is of the same width, the intra-cell routing can be conceptualized as standard-cell channel routing. Hence, we assume that each channel is divided into horizontal tracks, and the count of the number of tracks needed to place all horizontal wire segments is sufficient to estimate the width of the channel.

2.4.7 Goal of the Research

The goal of the research is to develop a cell generator based on a new layout style. We propose to develop a new layout style, an input language and placement and routing procedures based on the new layout.

The goal of the cell generator is to optimize the layout area. Since the actual layout area depends upon the routing of the nets, it is very difficult to compute it during the placement phase. Instead, we will develop a cost function to estimate the area and strive at minimizing it. We will estimate the value of the cost function with different placements and find an optimal placement by developing algorithms based on the simulated annealing technique described earlier.

2.5 Summary

In this chapter we presented a survey of the existing Row-based cell generation methods, concentrating

mainly on their strengths and weaknesses. We also pointed out some of the improvements that we have incorporated in our method of cell synthesis which uses a new layout style.

Also in this chapter, the thesis problem was formulated and the input and output specifications were given. The objective of the research along with the outline of the solution procedure were given. The next two chapters give a more detailed description of the placement and the routing procedures.

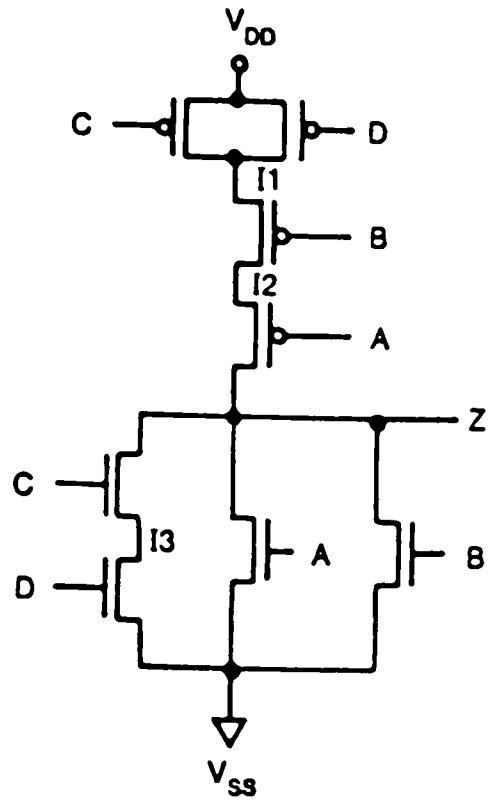


Fig. 2.1 Euler Paths

(a) $\overline{Z = A + B + (C * D)}$

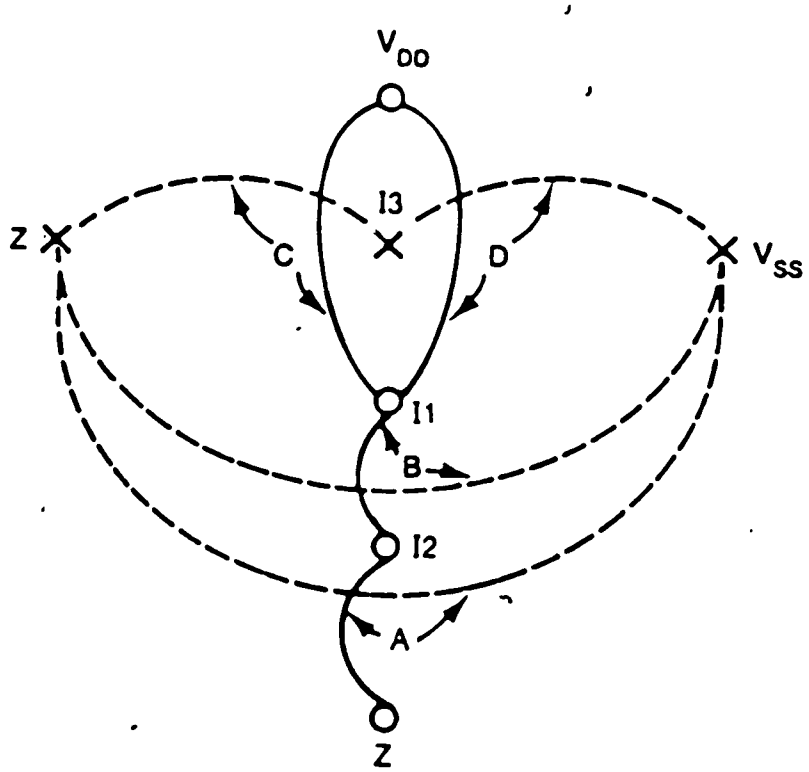


Fig. 2.1 (cont'd.)
(b) n- and p-graphs

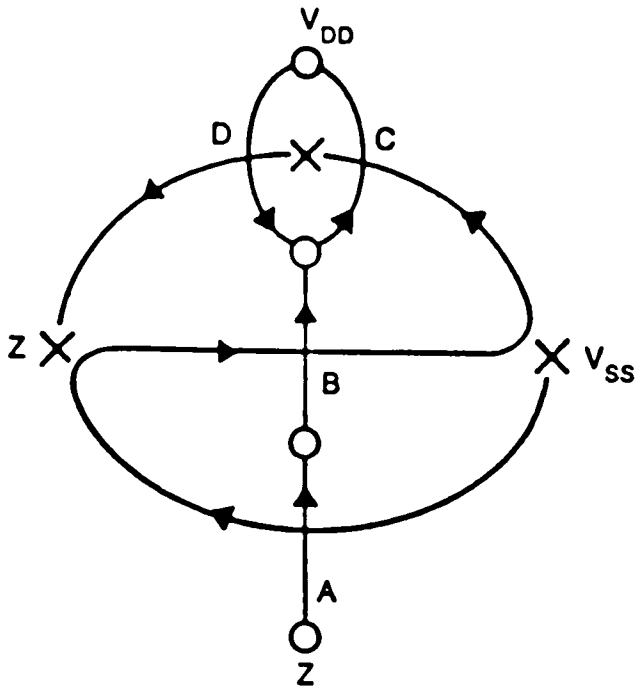


Fig. 2.1 (cont'd.)
(c) Euler Paths in the Graphs

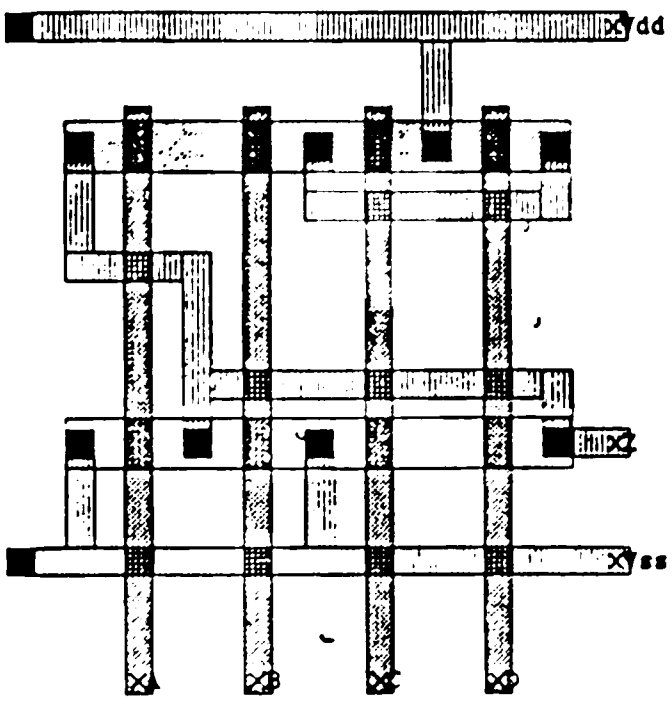


Fig. 2.1 (cont'd.)
(d) The Resulting Layout

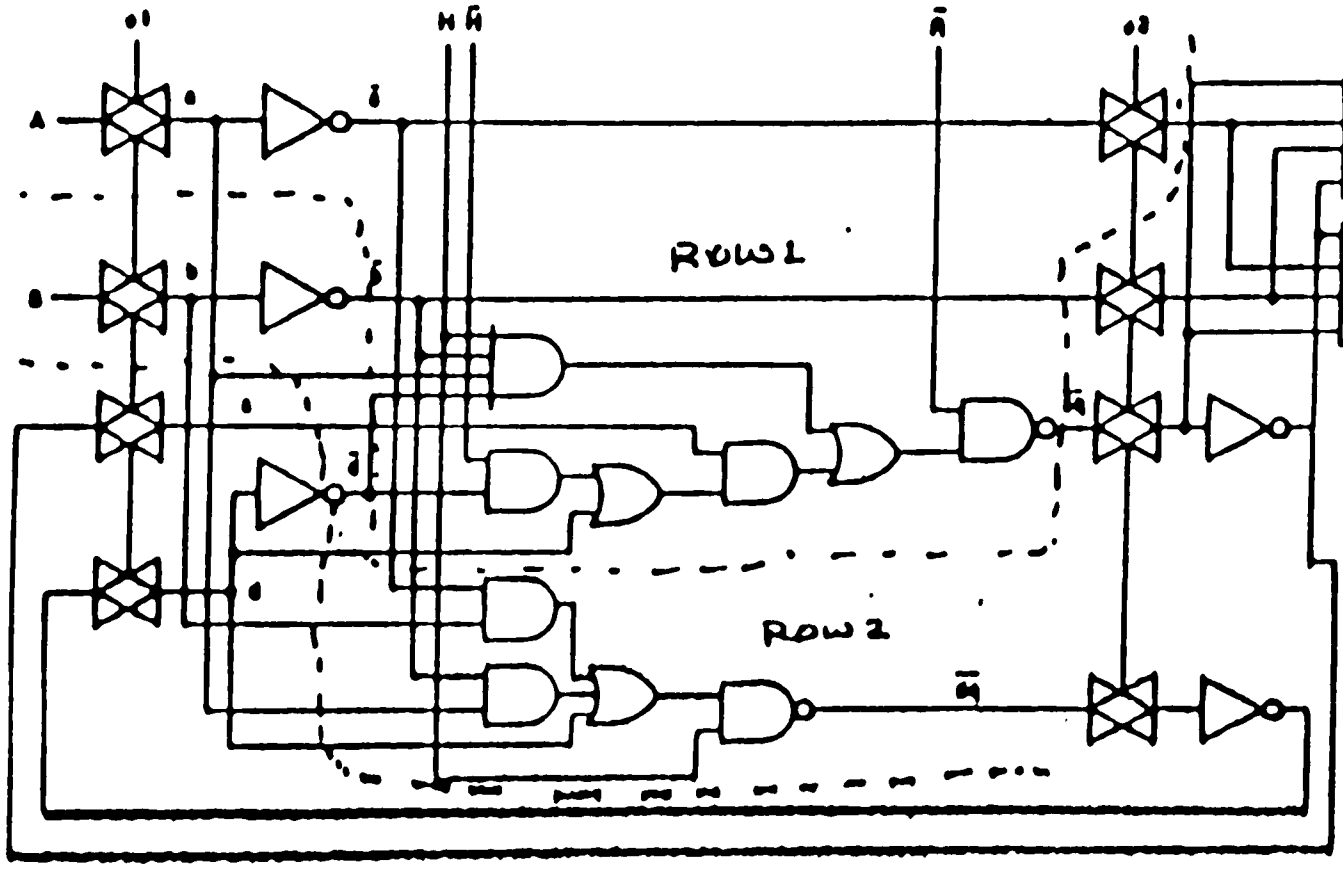


Fig. 2.2 Partition of a Circuit for 3-Rows Layout

CHAPTER III

PLACEMENT

In this chapter, we give a description of the simulated annealing algorithm used and the motives behind the move generator and the cost function. We also describe the placement scheme followed by the algorithm.

3.1 Simulated Annealing Algorithm

The generic simulated annealing algorithm was described in chapter 2. We now describe the same algorithm when applied to the placement of transistors within each row of the cell.

```
begin
  S := Initial placement of transistors;
  T := Initial temperature T0;
  while (stopping criterion is not satisfied) do
    begin
       while (not yet in equilibrium) do
          begin
             S' := Some random neighboring legal placemet
                   of S;
             del := COST (S') - COST (S);
             Prob := min (1, exp (- del/ T));
             if random (0,1) <= Prob then S := S';
```

```
        end;
      T := T * Boltzman's constant;
    end;
  output best solution;
end.
```

As seen in the algorithm, we start with an initial placement of transistors in each row, and a high temperature (some high value). We calculate the goodness of the placement by computing the value of the cost function. Then we change the placement by using a move generator and recalculate the cost. The new move is accepted or rejected based on a probability function. This procedure is continued until we reach the equilibrium state, which is based upon the number of downhill moves (moves which increase the cost) and the number of moves made without appreciable change in the cost. We permit downhill moves so that our solution does not get stuck at a local minimum. In the begining, due to high "temperature," moves which cause large amounts of change are accepted. Then the temperature is decreased, which causes the algorithm to be more selective about the moves. This process is continued until we reach the stopping criterion--which is again based on factors similar to the equilibrium condition and also a good value of the cost function which was determined based on the area of the cell generated by other popular methods.

3.1.1 Input Specification

We have developed a language which describes the circuit in each row of the cell. It also specifies the number of rows in the cell, the outputs from the cell, and the other attributes required by the front-end parser. The input specification language has been given in chapter 2 and it is explained in detail in the appendix.

3.1.2 Initial Placement

We obtain an Euler path for each of the rows specified using the algorithm developed by Uehara and Van Cleemput [1]. We ensure that there exists an Euler path by inserting a pseudo transistor whenever a gate has an even number of inputs. The set of these paths is the input to the simulated annealing algorithm.

3.1.3 Search Space

We know that Euler paths in the graphs provide the placement of transistors which will result in the minimum width of the row. But there can be several such paths in each graph. The set of all the combinations of Euler paths possible in each row is the search space for our algorithm.

3.1.4 Legal Moves

The simulated annealing algorithm depends on a set of legal moves through the search space to arrive at an optimal solution. These are the changes introduced in the existing legal placement of transistors in each row which results in another legal placement.

3.1.4.1 Delayed Binding

In a CMOS complex gate, permutation of some transistors may not affect the realized logic function, but may affect the layout significantly. Sometimes it is advantageous to delay the binding of the position of a device within the circuit. This concept known as delayed binding is used to generate the legal moves by the cell generator.

Given a sequence of transistors (an Euler path), a legal move can be defined as flipping of the sub-sequence of the given sequence to obtain a new sequence, but not affecting the logic function that is realized.

Using this concept, we define our moves which are based on the flipping of a sequence of transistors within a row. Supposing the sequence is a,b,c,d,e,f,g,h,k,l and m (Fig. 3.1a), flipping transistors 2 through 6 will give us a new sequence a,f,e,d,c,b,g,h,k,l and m (Fig. 3.1b).

3.1.5 Cost Function

The goal of the cell generator is to optimize the layout area. Since the actual routing area depends upon the routing of nets, it is very difficult to compute it during the placement phase. Instead, we use a cost function to estimate the area. Let T_i denote the number of transistors in the i^{th} pair of rows, and let d_i denote the sum of the track densities of the major channel in the i^{th} pair and the minor channel that separates the i^{th} row from the $(i+1)^{th}$ row. Let B_i denote the number of dummy transistors, including those needed for routing of nets between non-adjacent pairs of rows. Ignoring the difference in the number of breaks in the p- and n-rows of a pair, we can then estimate the width of the i^{th} pair by $(B_i + T^i)$. If w denotes an estimate of the width of the diffusion and power lines in a pair of rows, then $(B_i + T_i) * (d_i + w)$ is an estimation of the area for the i^{th} block. Since B_i is much smaller compared to T_i and it contributes heavily in the quality of the layout, we use the following as the cost function:

$C(b, a, d) = bB^2 + a(\max B_i - \min B_i)^2 + d\ \mathrm{sum}(B^i, d_i)$

where sum () is taken over all pairs of rows, and b, a, and d are user-defined constants.

3.1.6 Annealing Schedule

At first we allow flipping of entire gates which introduces large amounts of perturbance in the cost of the layout. Then as the temperature decreases, we allow moves only within a gate so that the degree of change is not as large as at higher temperatures. A random number generator is used for this purpose. We calculate the cost for any move only if it is accepted, and make the necessary changes in the data structures. We keep track of the best possible layout at any point. A good result can be obtained by experimenting with different values of temperature and the stopping criterion.

3.1.7 Stopping Criterion

This is based on the number of downhill moves, the number of moves made without considerable change in the cost, and by comparing the resulting area with the area obtained by other methods. These are specified as parameters to the annealing algorithm so that the user can experiment with the algorithm for a good result.

3.2 Placement Phase

The flips (or the moves) that are used to move in the search space can contribute to reduction in the area of

the cells due to several reasons. Some of the important reasons are discussed in the following paragraphs.

3.2.1 Reduction in Width

1) Several dummy gates which appear consecutively can be replaced by only one dummy gate. This can be achieved by flipping.

2) Any dummy gate which appears on the left or the right edge of a row does not contribute to the width. By using flips we can remove a number of breaks.

3.2.2. Reduction in Height

1) We can align the gate signals in different rows, thus eliminating several tracks needed for its routing.

2) We can also align gate signals in the middle row with dummy gates in row1 and row3, thus reducing the routing complexity and the number of tracks involved.

3.2.3 Reduction of Intra-row Routing Tracks

The number of tracks used for intra-row routing can be reduced by reducing the degree of the nodes in the n- and p-graphs of the circuit [15]. This actually causes some of the tracks to merge, thus reducing their number.

3.3 Data Structures Used

This section describes the data structure used to store the n- and p-diffusion rows of the cell. Figures 3.2 and 3.3 give a pictorial description of the same.

The n- and p-transistors for each row are stored in a two-dimensional array of structures. The row number is used as an index into the array (zeroth row is not used). The position of the transistor within the row specifies the second index of the array. The following information is associated with each cell in the array (Fig. 3.2).

1) The gate signal for the transistor--different symbols are used to specify dummy transistors which can be either in series or in parallel.

2) The number indicating the drain voltage for the transistor (a vertex in the graph). We do not store the source voltage because in our layout style the source of one transistor is at the same potential as the drain of the next transistor (we start with a dummy transistor in the begining of each row).

3) The X co-ordinate for the transistor which is used for generating the ABCD description for the cell.

4) In case of dummy transistors, a flag indicating whether it is already used for routing or not. This information is needed during the routing phase.

5) A flag indicating whether the transistor is at the left or the right edge of the line of diffusion. This information is used during the ABCD generation phase for ignoring the dummy transistors. Figure 3.3 shows several rows made up of the cells described earlier. The major and minor routing channels are represented by arrays.

3.4 Summary

In this chapter the simulated annealing technique adopted for our layout scheme was described. The legal moves which guide the annealing, the cost function used and also the data structure used to hold the layout information were explained.

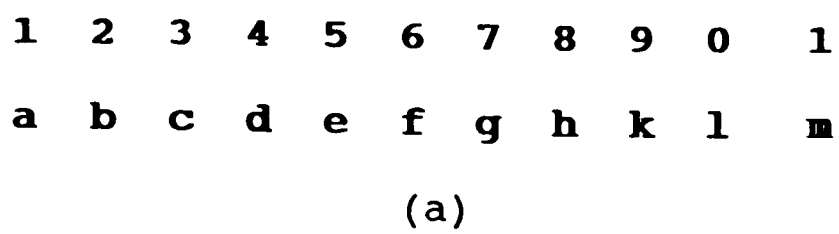


Fig. 3.1 Legal Moves
(a) Original Placement
(b) After Flipping 2 through 6

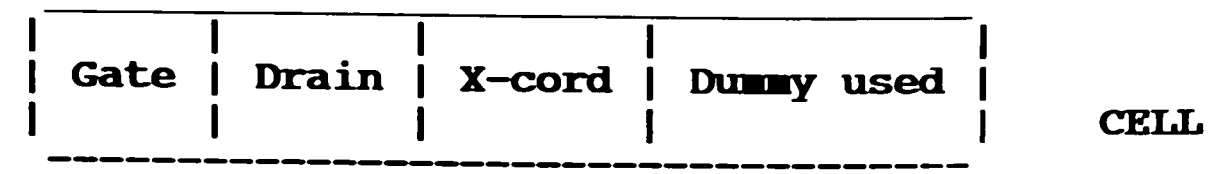


Fig. 3.2 Each Cell in a Row

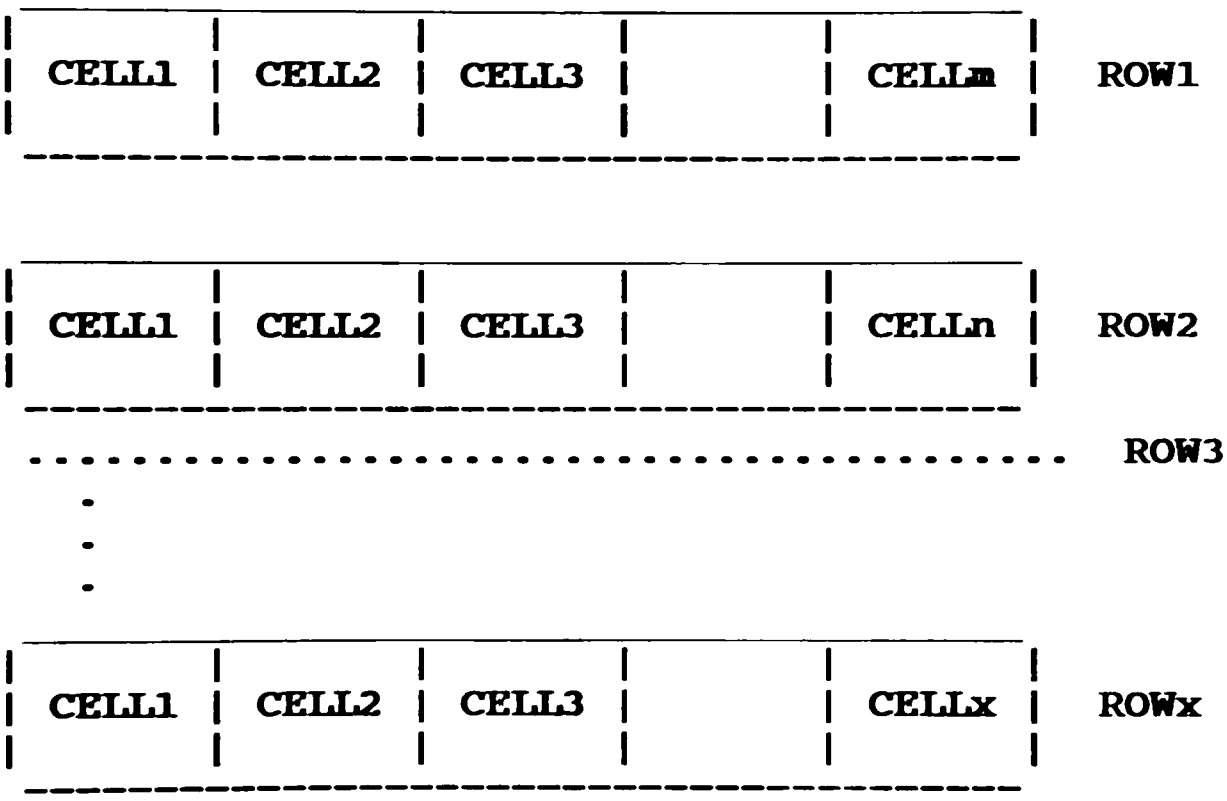


Fig. 3.3 x Number of Rows

CHAPTER IV

ROUTING

The role of the major and the minor routing channels in our layout was explained in chapters 2 and 3 (see Fig. 1.3). This chapter describes the inter- and intra-row routing performed within those channels and identifies the different types of routes in each channel.

4.1 Intra-row Routing

Intra-row routing is the routing used to make the inter-connection between different points on the same diffusion row--either n or p. The points could be on the same diffusion island or on different islands but within the same diffusion row.

A channel is designated for each diffusion row in which the routing is done. Within this row the left-edge first algorithm, described in the next paragraph, is used for track allocation. This ensures that we use the least possible number of tracks for intra-row routing.

4.1.1 The Left-edge First Algorithm

The left-edge first channel routing algorithm is due to Hashimoto and Stevens [18]. The objective of

channel routing is to electrically connect corresponding pins on the top and bottom of a rectangular region (called the channel) using a minimum number of horizontal grid lines (called tracks). Figure 4.1a shows a channel routing problem. Two layers are assumed--polysilicon for the vertical tracks and metal for the horizontal tracks. The left-edge first algorithm attempts to maximize the placement of the metal wires in each track. The edges are sorted on the position of their leftmost endpoints. The sorted list for Fig. 4.1.a is (1, 2, 3, 4, 5, 6, 7). The algorithm then selects the first edge in this list (which is edge 1, the leftmost edge) and places it in the lower left corner of the channel, forming track 1. This edge is deleted from the list. Next, the algorithm picks up an edge that does not overlap with edge 1, and is also closest to edge 1. Edge 6 is chosen and is placed on track 1. This is continued until no more edges can be placed on track 1. The algorithm then goes through the rest of the edges in the sorted list. Figure 4.1b shows the track selections made by the left-edge first algorithm which is a minimal track assignment (a total of four tracks is shown).

This algorithm is adopted with very few changes for the horizontal track allocation during the intra-diffusion routing stage. In case of the nets that we are dealing with, the nets are already sorted on the position

of their leftmost endpoints. We just have to traverse the n- or p-layer from the left edge, and we encounter the sorted list of nets. We then allocate tracks as explained earlier. To connect the nets on n- and p-side, we use a flexible approach based upon the delayed binding principle (as explained later).

One exception to the above is the case where the net that we are routing is an output net. Since the output net runs all the way to the corner to the output pin, we allocate an entire track for it initially. It can be observed that these nets can share a track with another net only if it lies to the left. After the final placement, if possible, these output nets are made to share a track with other nets. But for the initial calculation a separate track is considered.

4.1.2 Inter-diffusion Routing

Inter-diffusion routing is the routing of the nets between n- and p-regions of the same row. The following types of nets are identified:

4.1.2.1. Gate Signals

Since we consider only Euler paths within the circuit graph, for a given row, the n- and p-transistors

are always aligned. So the gate signal routing between the n- and p-regions of the same row is always straight.

4.1.2.2 N & P Output Connection

The following algorithm is used to choose the best position to connect the outputs generated on the n- and p-diffusion strips of the same row:

```
If n or p output node has a net
   If both n and p output nodes have a net
     If they do not overlap
       Extend the n net by one grid position
     else  /* overlap */
        Choose a point along the net where no new
          column is added (if possible)
else  /* n or p has a net */
   if the output net encompasses the output node
        on the counterpart side new column need not
          be added
   else  /* output node not encompassed */
     extend the output net in the necessary
     direction. If the n/p output is also an output
     of the circuit and the vertical connection
     point is to the right, then extending the net
     is not necessary.
```

4.1.2.3 Duplicate Gate Signals

If there is a multiple occurrence of any gate signal within any row of the cell, then the signals are all connected together. There is no separate channel allocated for this. An interesting method is used for track allocation for this connection. Supposing the gate signal 'x' occurs at grid positions 1-5-9-14, then gate interconnection is achieved in three different stages by using a greedy algorithm. First the connection between the points 1 and 5 is considered and is routed using either of the channels that are allocated earlier for intra-diffusion routing. If it is possible, sharing of the track with another net is done. Then the connection between the points 5 and 10 is performed in a similar fashion without worrying about the other stages of the interconnection. A different track, possibly in a different channel (within the same row), might be found. Figure 4.2 illustrates this point. This technique could result in considerable saving.

4.1.2.4 Intermediate Output to Gate Signal Connection

If any occurrence of the gate signal lies within either the p/n output nets, then we just use a contact at the output nets' point of intersection. Otherwise either n

or p, whichever is closer, is extended to make the connection. This does not result in a track as the net has already been allocated a track. This is illustrated in Fig. 4.3.

4.2 Inter-row Routing

The routing of various types of signals between different rows of the cell is designated as the inter-row routing. The types of signals that require inter-row routing are: i) common gate signals, and ii) intermediate outputs that are used as gate signals.

4.2.1. Common Gate Signals

4.2.1.1 Between R1/ R3 and R2

If there is a single occurrence of the common gate signal, either in row1/row3, it is routed in the inter-row routing channel. If there is either a single or multiple occurrence of the signal in row1/row3, the closest occurrence in both the rows is chosen as the routing point (because the common signal in row1/row3 should be connected, too). This type of routing is shown in Fig. 4.4.

4.2.1.2 Intermediate Output to Gate Signal Routing

This routing is also done based on the same heuristics as described above. If the intermediate output

generated in any row is used in the same row as a gate signal, then this routing need not be explicitly done because the common gate signals between different rows would be connected as described earlier.

If there is a multiple occurrence of the signal in row2 and either single or multiple occurrence of the signal in row1/row3, the points having the least distance between them are chosen. When there are multiple occurrences of the same signal in row1/row3, it seems as if it would be a good idea to see if the point in row2 that is to be connected lies within the net interconnecting the multiple occurrences in row1/row3 and then run a straight through poly wire to make the connection. It would be possible only if there were a dummy gate at that position.

4.2.1.3 Common Gate Signals between R1 & R3

If the signal is present in row2 also, then it would already be connected. Otherwise we try to run a straight_through route via row2 if we find a dummy gate. If neither of the choices are applicable, we make the connection going around the row2 diffusion strip (see Fig. 4.5). The reason why these signals are connected is to enable the user to use the pins either at the top or the bottom to make connections from the outside world.

4.3 Routing to Output Pin

The n- or p-net, whichever is closer to the side where the pins are located, is extended all the way. As mentioned earlier, the output nets are allocated separate tracks so that they can be extended easily.

4.4 Vdd-Vss Routing

We have vdd-vss-vdd-vss running across the cell as shown in Fig. 1.3. This makes the power and ground connections easy.

4.5 Routing of Gate Signals

If the gate signal under consideration is present in either row1 or row3, or if it is generated internally in the same or in a different row, the method of routing is described earlier. If it does not fall into either of the categories, we use the following approach:

1) Look for a dummy gate in the row above or below it. Choose the closest one. We assume an unlimited number of dummy gates at either end of the row. Because of this supposing we have a situation where we have a gate in row2 which has to be routed, and we find that the closest dummy gate is 10 horizontal grids to the right, but the left corner is only four horizontal grids away (Fig. 4.4), we will be able to route from the left corner of the cell.

2) If there are no dummy gates, the following criteria is used to choose the position of the pin:

a. Choose the row (row1/ row2) that has the lesser number of transistors. This results in elimination of white space in the cell. Allocate the pins on the right end until the width of this row is the same as that of the widest row in the cell.

b. Once the number of transistors in row1 and row3 becomes equal, we start allocating pins on either sides of row1 and row3. The first pin will be allocated to the left of row1, the next to the left of row2. This causes the new column that was added to be shared by the new gate signals. Similarly, the next two pins are allocated to the right of row1 and row3. In this case the new column is shared by the two new gate signals and also the two rows added (by the two new signals to the left of row1 and row3) are also used for routing. It can be noted that due to traversal order, we would not cause crossing of the wires. Figure 4.6 illustrates the different gate signal routing methods.

4.6 Data Structures Used

The data structures used to store each row were given in chapter 3 and they also store a considerable amount of information necessary for routing. The other

important data structure used is the array of structures which stores the net-lists. Figure 4.7a shows a net which is comprised of terminals at positions 1,3,5,7, and 9. Its representation is shown in Fig. 4.7b. The occurrence field indicates that there are "occurrence - 1" occurrences of this particular node in the circuit. Node-pos is an array containing the positions at which the node in consideration occurs. The structure also stores the net length, track number, x and y co-ordinates and other control information. There is a pair of arrays of structures for each row in the cell.

4.7 Summary

In this chapter the inter- and intra-row routing schemes, and the algorithms and the heuristics used to implement them were described. The different types of nets that are routed in these channels were identified and the method of their routing was described. The data structure used to capture the net-list information was described.

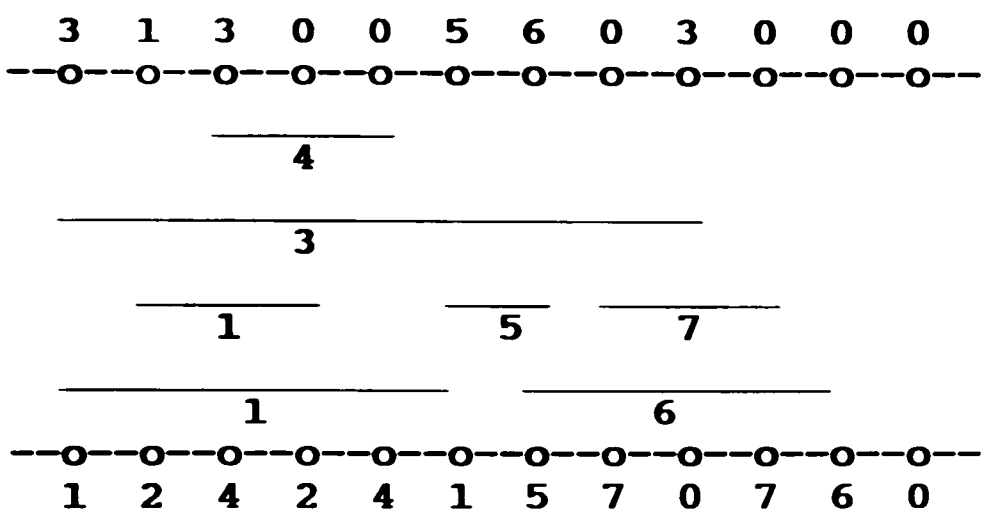


Fig. 4.1 (cont'd.)
(b) Track Assignment by Left-Edge First

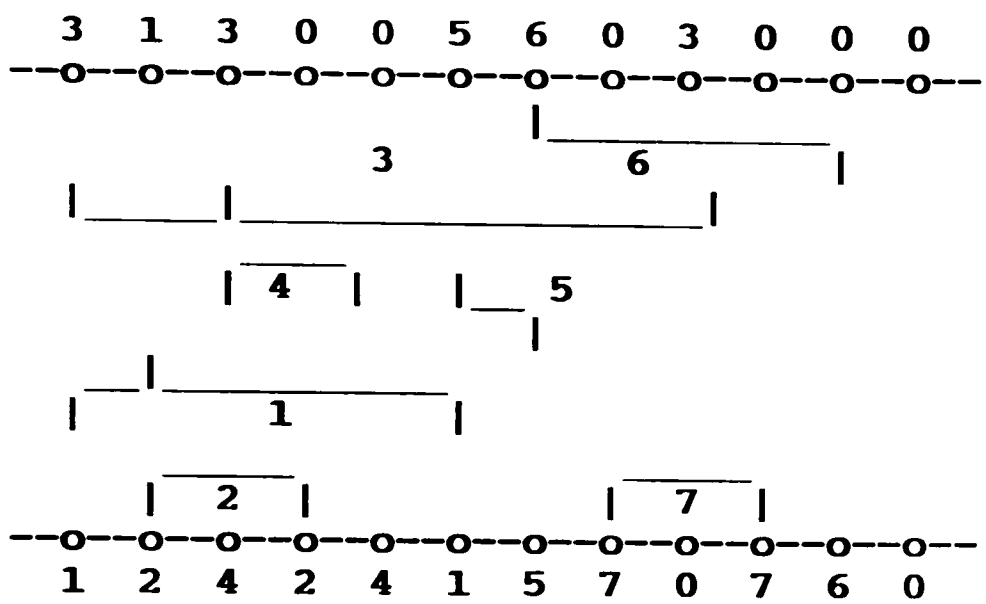


Fig. 4.1 The Left-Edge First Channel Router
(a) The Channel Routing Problem

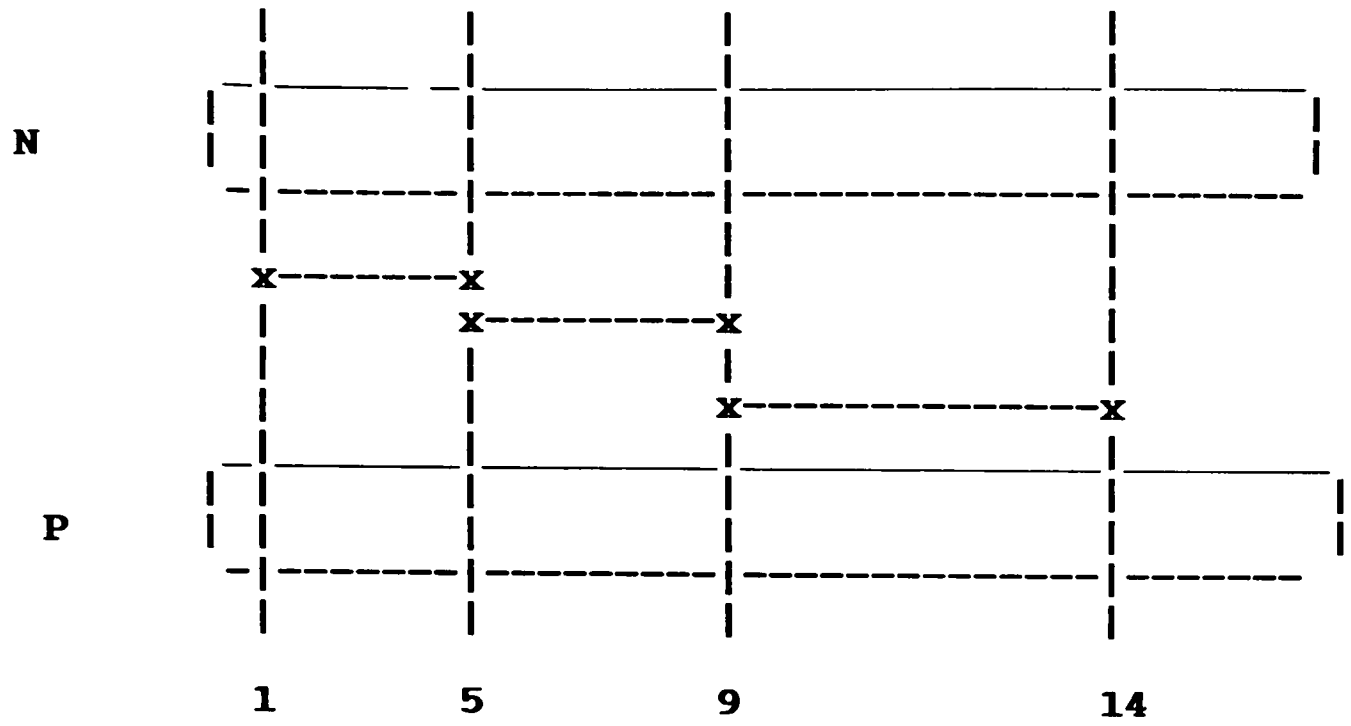


Fig. 4.2 Duplicate Gate Signals

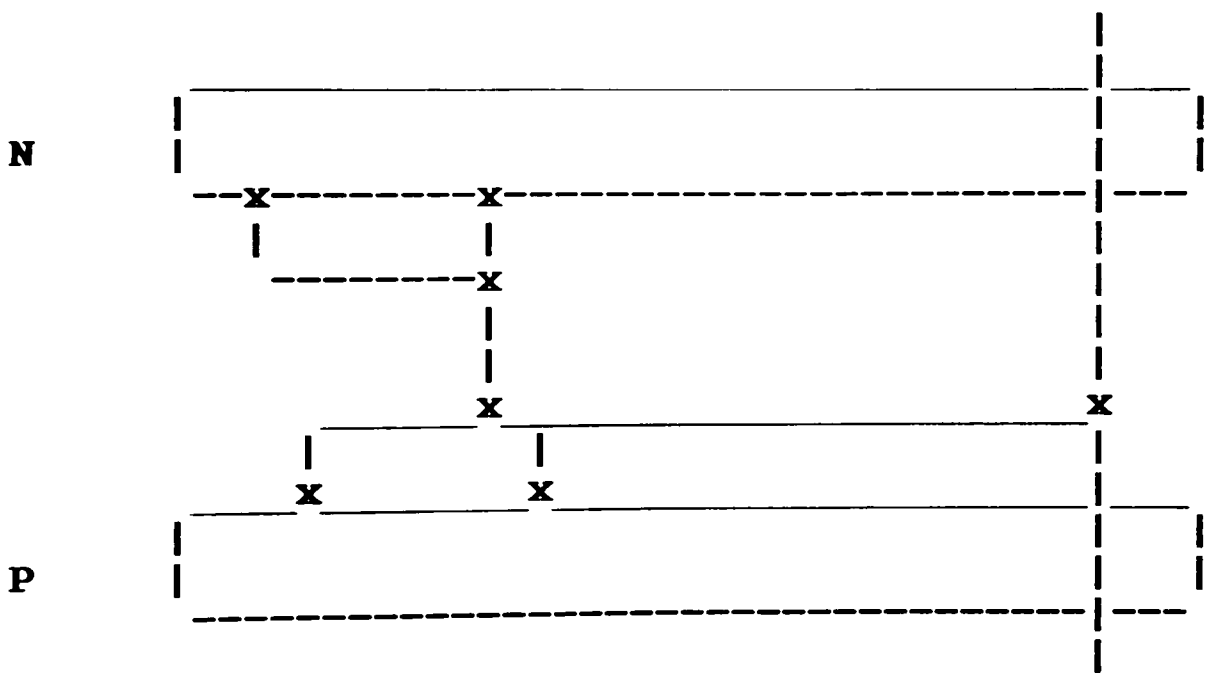


Fig. 4.3 Intermediate Output to Gate Signal Connection

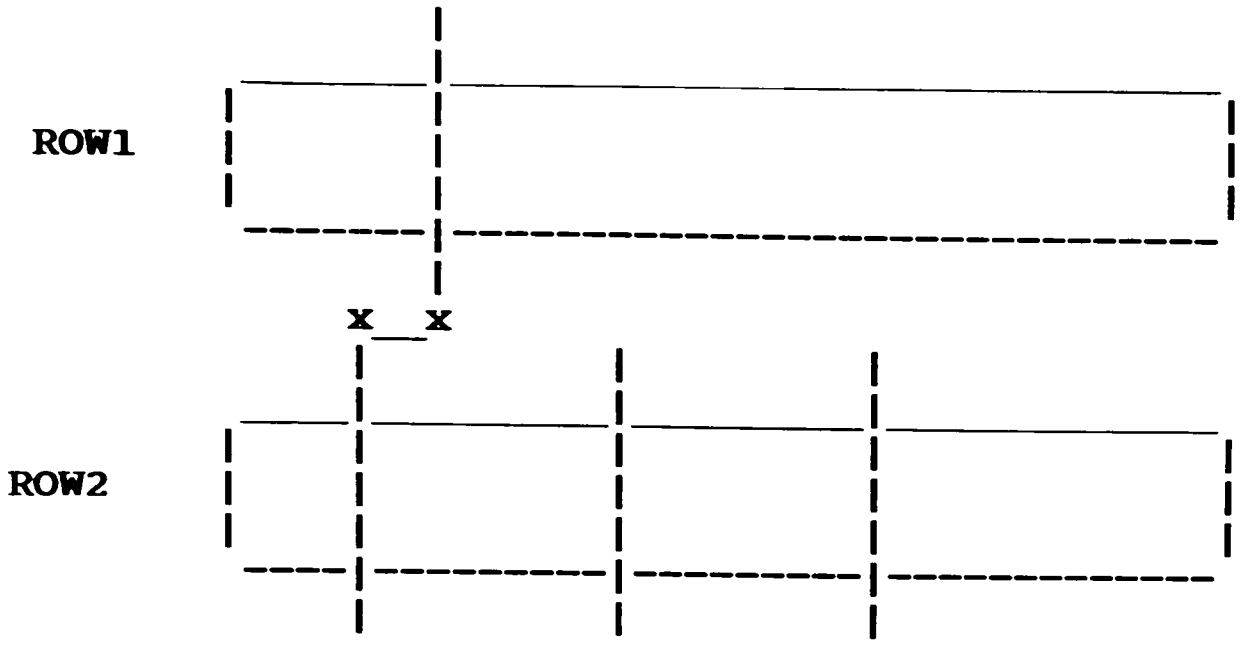


Fig. 4.4 Common Gate Signals between R1/R3 and R2

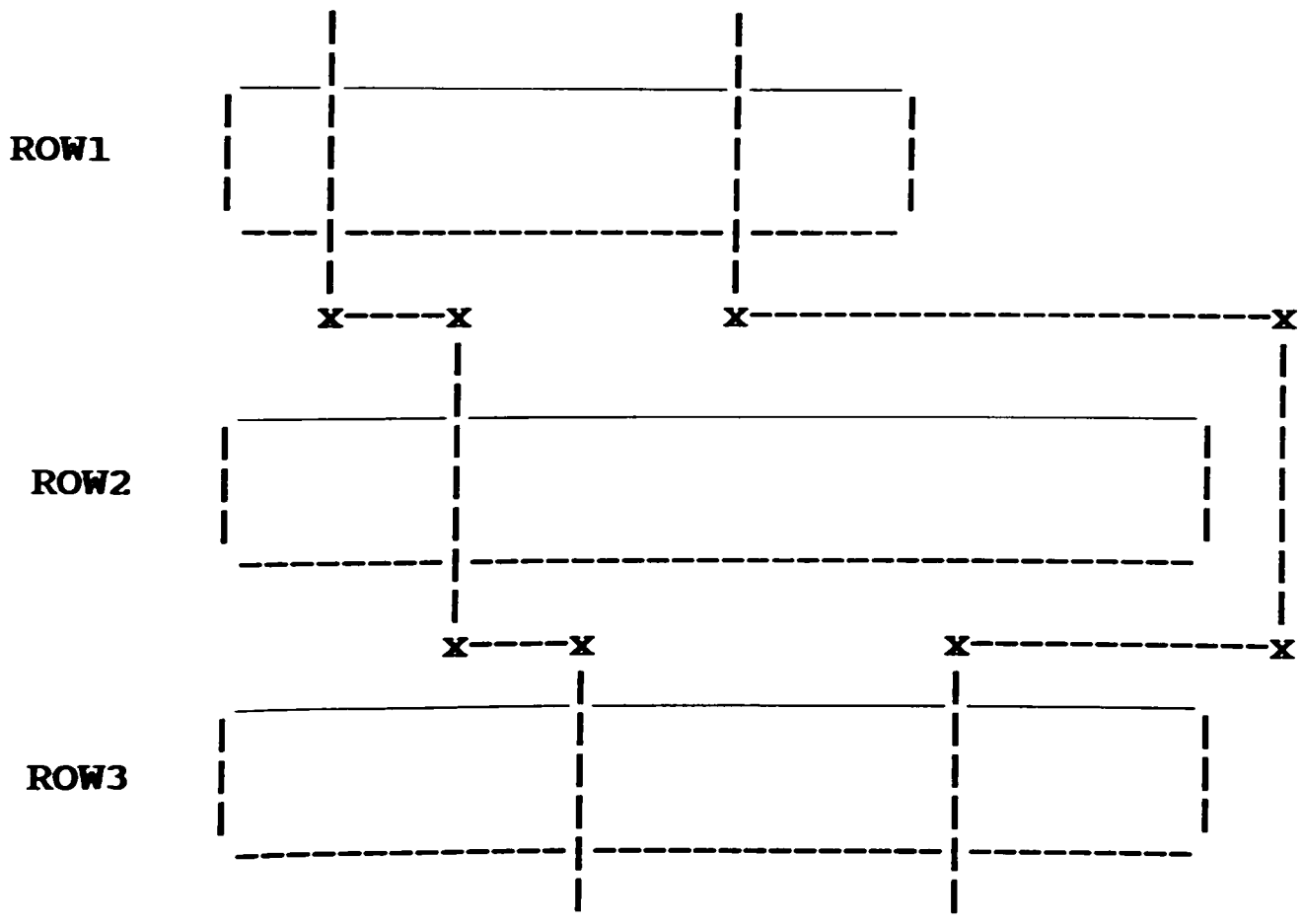


Fig. 4.5 Common Gate Signals between R1 and R3

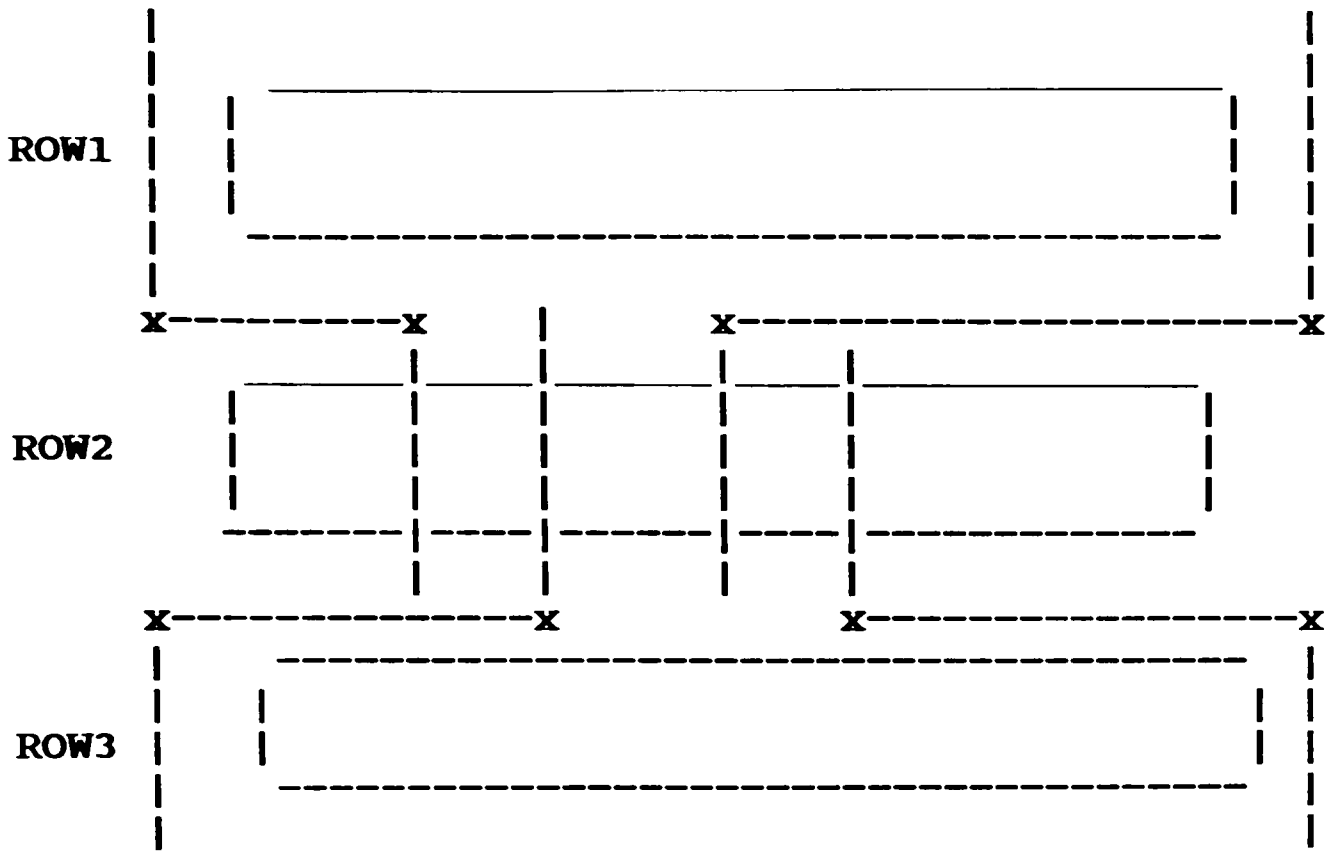


Fig. 4.6 Routing Gate Signals

Postion	1	2	3	4	5	6	7	8	9	10
Occurence	x		x		x		x		x	

Fig. 4.7 A Net

Occurence	6
node-pos	1 | 3 | 5 | 7 | 9 |
net-lt	8
track	1
Output-node	True
pow-gnd	False
x-start	10
x-end	90
y-cord	120

Fig. 4.8 Structure Representing the Net in Fig. 4.7

CHAPTER V

CONCLUSIONS AND RESULTS

5.1 Contributions

In this thesis, we have extensively discussed, formulated, and extended the method of Row-based cell generation. The goal of the study was to develop algorithms that would produce cell layouts requiring minimum silicon area, by considering multiple rows of transistors and by optimizing in both the dimensions simultaneously. Our algorithms consider the cell layout generation as a two-dimensional area optimization problem instead of two one-dimensional optimization problems as considered in many existing systems. In most systems, this problem is divided into two phases--placement and routing--which are carried out separately. We have combined them to a certain extent.

We are not aware of any system that has considered multiple rows of transistors simultaneously. This certainly is the most important feature of our cell generator. Another important feature is that our cost function includes the complexity of routing during the placement phase. Therefore, our cell generator tries for a reasonable compromise between the width and the height reduction to arrive at the "best" possible layout.

We measure the layout area in terms of both the width, which is considered to be proportional to the maximum of the number of transistor terminals (source, drain, or gate) in each row, and the height, which is considered to be proportional to the number of tracks used for routing. We minimize the width by considering Euler paths in the graphs and by eliminating the breaks between two consecutive transistors wherever possible. The height is minimized through reduction of the routing complexity by considering the routing problem during the placement of transistors. The quality of the circuit is improved by minimizing the total wire length in the cell, which also reduces the total resistance of the circuit.

The two-dimensional area optimization problem is solved by using the technique of simulated annealing. Simulated annealing is an iterative technique and it manipulates the Euler paths for each of the rows in the cell during every iteration until it finds a near optimum solution. As stated in earlier chapters, our cell generator has two phases. The placement phase obtains a placement of transistors. This phase is algorithmic and its goals were summarized above. The second phase--the routing--realizes the connections between the transistors. This phase follows certain styles for routing the nets but there are many problems which could not be formalized in the form of a

layout model. These are very local and are therefore solved using ad hoc methods. To limit the complexity of these problems, we limited the number of rows to be no more than three in a cell. It helped us to route the global connections to the boundary of cells.

An important feature of our cell generator is that during the placement, it ensures that the routing will be realizable. In many systems, if the router fails, the system is run again to find an alternative placement.

Our tool considers circuits which can include non-dual gates such as the transmission gates. Thus dynamic logic can be included in the circuit.

In order to be able to view the symbolic layout, we also developed a translator that takes the output of the cell generator and produces a textual description of the cell layout. Since we possess a virtual grid compactor, written for circuits described in ABCD language [11], we decided to use ABCD as the language to describe the layout. By using this translator and the compactor, we can produce compacted cells which can then be used by a cell assembler [14].

5.2 Future Work

In this section, we suggest some of the ways to improve the present implementation.

1) For the experiments and testing of our CAD tool, we chose certain circuits and partitioned them by hand in a manner that would yield optimal layout. Since the goal of this thesis was to study layout optimization and cell generation, we assumed that the partitioning heuristics can be automated. Since the existing circuit partitioning tools were developed with the single Row-based cell generator in mind; these tools are not usable with our cell generator. We suggest that a partitioning tool which can exploit the advantages of multiple Row-based cells be developed. This tool should have the features described in the earlier chapters.

2) The simulated annealing implementation used requires a stopping criterion to end its search for an optimal value. Our stopping criterion was based primarily on the results obtained by other methods for the same circuit and on intuition. We did not evaluate the effect of this criterion on the layout area and believe that better heuristics should be developed for the same to improve the usefulness of our tool.

3) Our cell generator does not allow the user to specify the positions of any of the input/output pins. In some of the cases the pin orientation is a desirable feature. Modifications should be made to the algorithms to accommodate the same.

5.3 Results

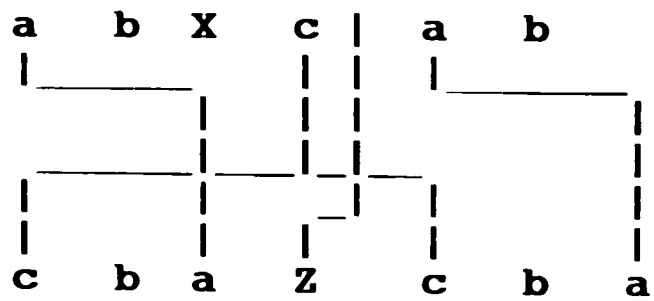
We describe the results of our experiments in the following paragraphs. A number of very small to very large circuits were chosen for testing. Results of the test and layouts obtained by using our tool are discussed below by considering two circuits. To show the advantage of the multiple Row-based layout style, we also discuss the area complexity of the layout and compare the area of these layouts with the area of symbolic layouts which use only one row of transistors. In order not to clutter the layouts, we have shown only the inter-row connections and have avoided the intra-row connections which connect transistors of the same type in a row. Furthermore, we take only inter-row connections into the computation of the area since only these connections play a significant role in the area computation. We express the dimensions of the layout in terms of the number of columns of transistors and the number of rows occupied by wire segments which connect the transistors.

Our first example is a full adder and is shown in Fig. 1.4a. A layout of this circuit which uses the single Row-based style, and is obtained by using the algorithm

5.1a. The dimension of the layout is five rows and 14 columns; the resulting area is 70 sq. units. Figure 5.1b shows the layout that uses the multiple Row-based style. The dimensions of this layout are two rows and 13 columns--resulting in an area of 26 sq. units. The saving of the area for this circuit is 60% and the results are tabulated in Fig. 5.1c.

Figure 5.2.a shows a 10-line to 4-line Priority encoder [10]; Fig. 5.2.b shows its layout obtained by the single Row-based style and Fig. 5.2.c shows the layout obtained using our multiple Row-based style. The first layout has seven rows and 33 columns, thus requiring an area of 231 sq. units. The second layout has five rows and 33 columns, resulting in an area of 165 sq. units--a saving of 28.5%; Fig. 5.2d tabulates the results. Figure 5.2e shows the layout obtained by using our cell generator. It shows all the inter- and intra-row connections.

The inverters and transmission gates are not shown as they are not accommodated in the single Row-based approach. The description of the steps to arrive at this layout is given in the appendix.



(a)

a b c a b

a b c a b Z

(b)

Style	# rows	# columns	Area
Single row-based	5	14	70
Multiple row-based	2	13	26

Area saving = 60%

(c)

Fig. 5.1 Results for the Full Adder
(a) Using Single Row-based Style Layout *
(b) Using Multiple Row-based Style Layout *
(c) Table Showing the Results

* Each name in the row is a gate signal

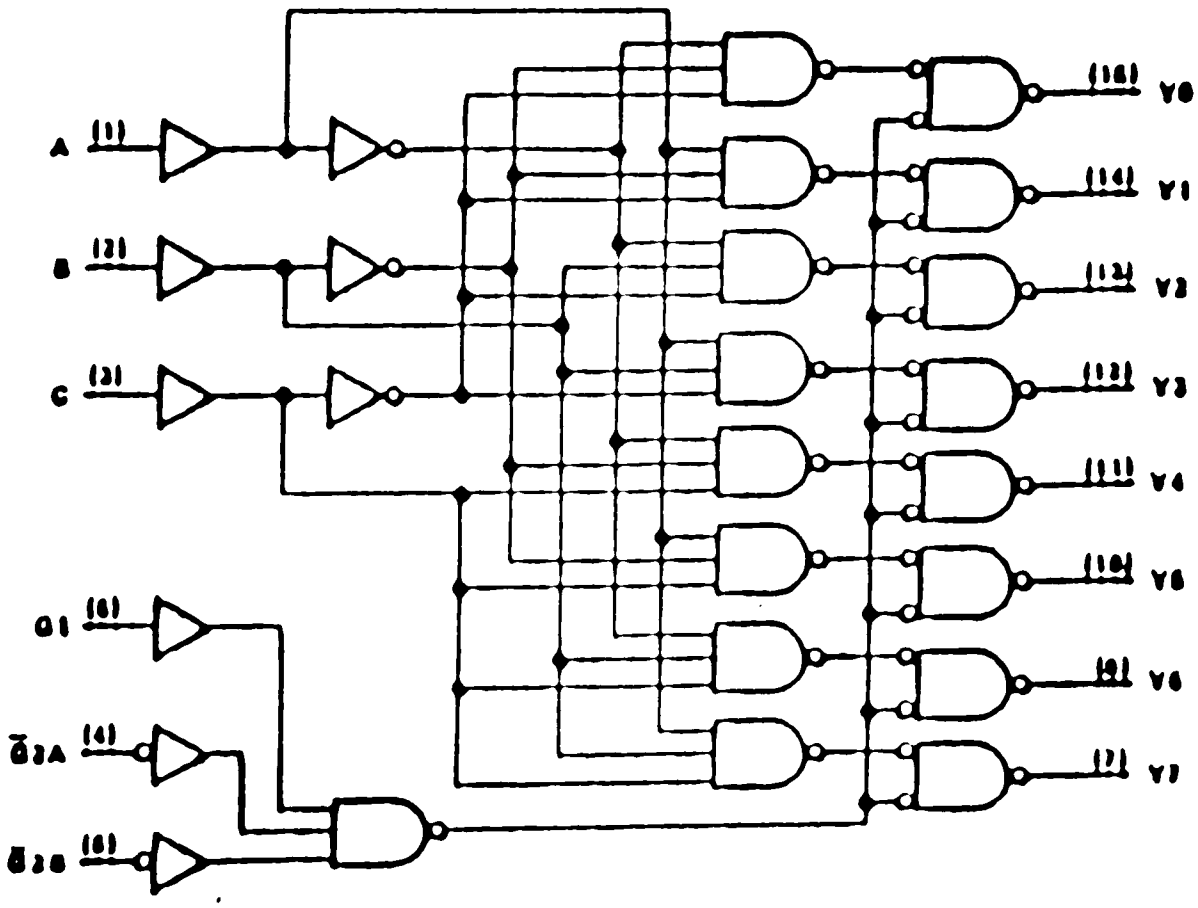
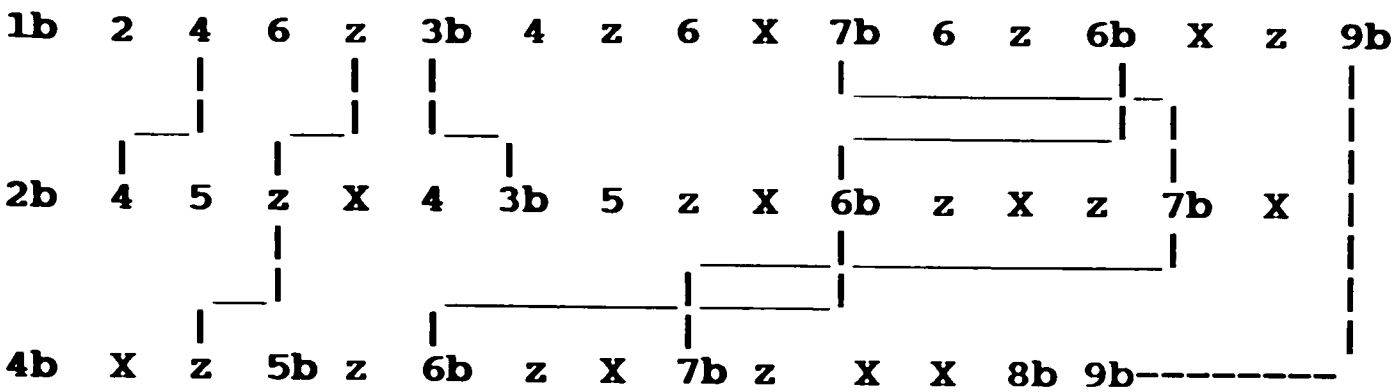
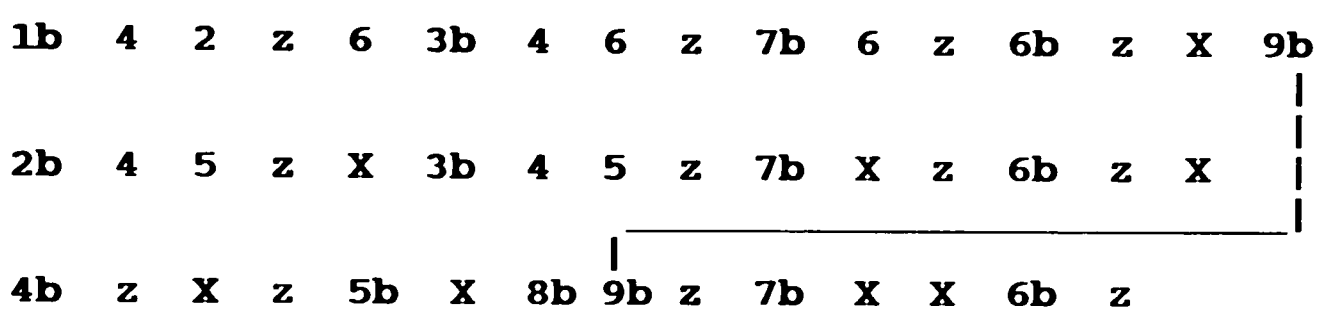


Fig. 5.2 Priority Encoder
(a) Input Circuit



(b)



(c)

Style	# rows	# columns	Area
Single row-based	7	33	231
Multiple row-based	5	33	165

Area saving = 28.57%

(d)

Fig. 5.2 (cont'd.)
(b) Using Single Row-based Style Layout *
(c) Using Multiple Row-based Style Layout *
(d) Table Showing the Results

* Each name in the row is a gate signal

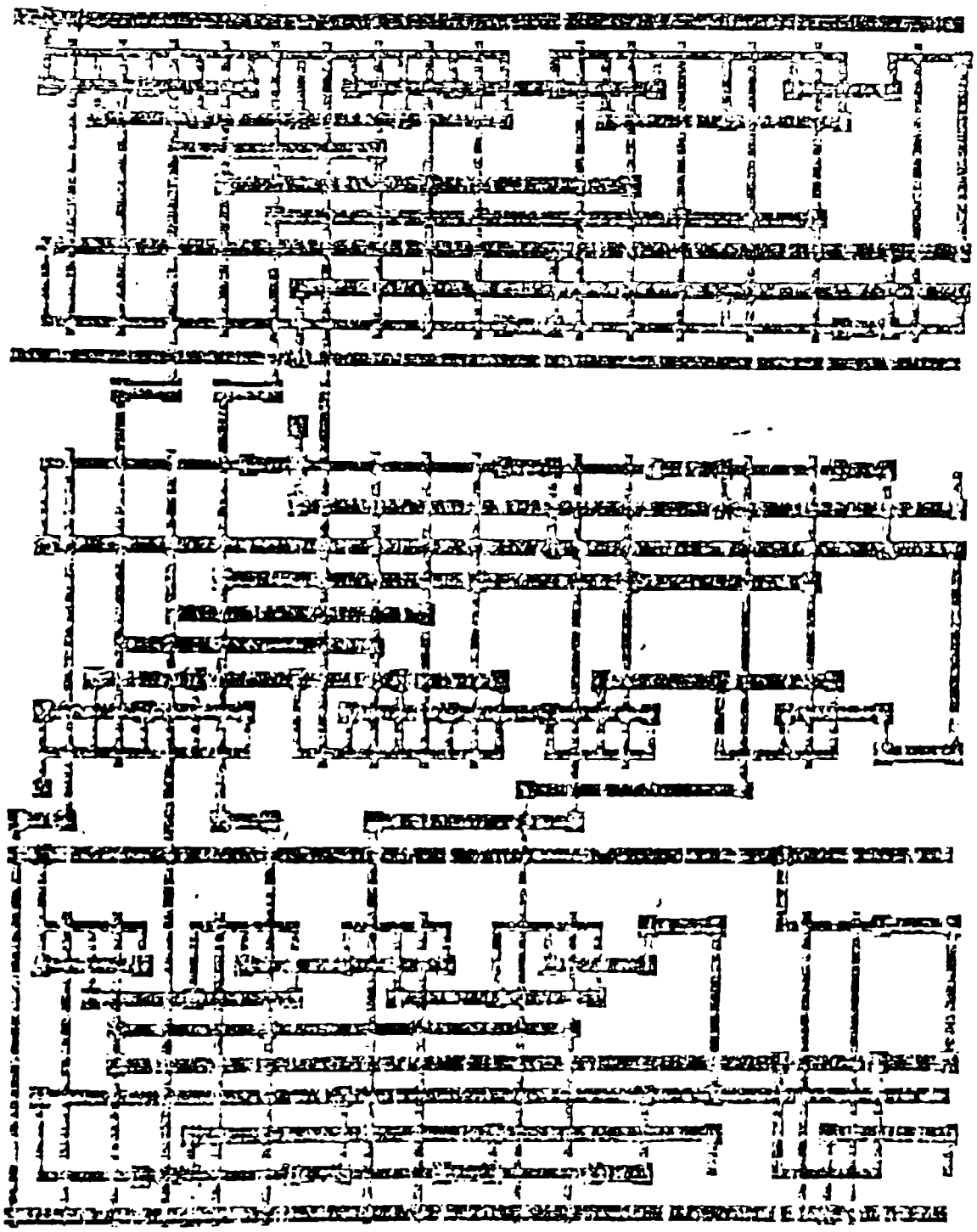


Fig. 5.2 (cont'd.)
(e) Multiple Row-Based Layout

BIBLIOGRAPHY

1. T. Uehara, and W.M. Van Cleemput, "Optimal Layout of CMOS Functional Arrays," IEEE Trans. Computers, vol. C-30, pp.305-312, May 1981.

2. P.W. Kollaritsch, and N.H.E. Weste, "TOPOLOGIZER: an Expert System Translator of Transistor Connectivity to Symbolic Cell Layout," IEEE Trans. on Solid-state Circuits, vol. SC-20, pp. 799-804, June 1985.

3. D.D. Hill, "Sc2: A Hybrid Automatic Layout System," Proc. of the International Conf. on Computer-Aided Design, pp. 172-174, 1985.

4. S. Wimer, R.Y. Pinter, and J.A. Feldman, "Optimal Chaining of CMOS Transistors in a Functional Cell," IEEE Trans. Computer-Aided Design, vol. CAD-6, pp. 795-801, September 1987.

5. A. Stauffer, and R. Nair, "Optimal CMOS Cell Transistor Placement: A Relaxation Approach," Proc. International Conf. Computer-Aided Design, pp. 364-367, 1988.

6. Y-L. Lin, and D. Gajski, "LES: A Layout Expert System," Proc. DAC, pp. 672-678, June 1987.

7. Y. Shirashi, J. Sakemi, M. Kutsuwada, A. Tsukizoe, and T. Satoh, "A High Packing Density Module Generator for CMOS Logic Cells," 25th ACM/IEEE DAC, 1988.

8. C.Y.R. Chen, and S. Chow, "The Layout Synthesizer: An Automatic Netlist-to-Layout System," Proc. ACM/IEEE Design Automation Conf., 1989.

9. N. Weste, and K. Eshraghian, Principles of CMOS VLSI Design, Addison-Wesley, 1985.

10. High-Speed CMOS Logic, Texas Instruments, 1988.

11. A. Rosenberg, "ABCD: A Better Circuit Description Language," MCNC Technical Report May 1983.

12. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 44, pp. 671-680, May 1983.

13. D. Dutt, "Optimization Techniques For Cell Assembly," M.S. Thesis, Texas Tech University, 1988.

14. D. Dutt, and G. Lakhani, "Optimization in Automatic Cell Assembly," IEEE International Conf. Computer-Aided Design, 1988.

15. C.Y.R. Chen, and C.Y. Hou, "A New Layout Optimization Methodology for CMOS Complex Gates," ICCAD, Nov. 1988.

16. K. Kannappan, "Optimization Techniques For PLA Folding," M.S. Thesis, Texas Tech University, 1989.

17. G. Lakhani and S. Rao, "A Multiple Row-Based Layout Generator For CMOS Cells," ISCAS, 1990.

18. A. Hashimoto, and J. Stevens,"Wire Routing by Optimizing Channel Assignment Within Larger Apertures," Proc. of 8th Design Automation Workshop, 1971.

APPENDIX

MARC: A MULTIPLE ROW-BASED CELL GENERATOR

In this appendix, the organization of MARC source code, and the input and output files are described.

A.1 Organization of MARC

MARC was written in "C" (about 8,000 lines of code) under the UNIX environment on SUN workstations. The following is a list of the header and source files that make up the cell generator.

sym.h--contains the definitions of all the global data structures.

lexx--contains the (LEX) lexical analyzer.

parser--contains the YACC (Yet Another Compiler Compiler) description, the code for building the n- and p-graphs, and part of the routing.

row.c--the main program; calls the other programs and contains the code for simulated annealing.

ab.c--generates the abcd code and also does part of the routing.

gatetrack.c--assigns tracks to horizontal nets.

pr_trail.c--prints n- and p-graphs.

A.2 A Manual for MARC

NAME

marc

SYNOPSIS

marc input-file output-file

marc is the executable program, input-file is the name of the file containing the input description (which is discussed later in this section), and output-file is the name of the file which will store the description of the cell in the form of ABCD language.

A.2.1 Input File Description

The encoder in Fig. 5.2a is divided into three rows as shown. The input is described as shown below:

```
ROWS  3
OUTPUTS A, B, C
INTERMEDIATE x
#
OP C ([4b,x], [5b,x], [6b,x],[7b,x]) OP x (8b, 9b)
#
OP B ([2b,4,5,x], [3b,4,5,x], [6b,x], [7b,x])
#
OP A ( [1b,2,4,6,x], [3b, 4, 6, x], [7b, 6, x], [6b, x], 9b)
&
```

ROWS--indicates the number of rows (valid numbers are 1, 2, or 3).

OUTPUTS--indicates the names of the output pins.

INTERMEDIATE--indicates the names of the outputs generated

intermediately, but not requiring an output pin.

#--signals the beginning of a new row.

OP--indicates the name of the output signal generated by the following gate.

(a,b,c)--represents an OR gate with 3 inputs--a,b, and c.

[a,b,c]--represents an AND gate with 3 inputs--a, b, and c.

&--signals the end of the input description.

A.2.3 Output File Description

MARC produces two output files--one containing the ABCD description of the cell and the other containing the description of the input and output pins. The name of the former is chosen by the user and the latter file is called the "pinfile." The ABCD language is completely described in [11]. The pinfile contains the position of the pin, its orientation (north, south, east, or west), and the layer in which it appears.

The layout given in the ABCD file can be plotted using AUTOCAD or the Berkely tools.

A.2.4 Help Facility

On-line documentation is available in the form of the subject.HLP files (where subject is the name of some of the important steps such as input, plotting, etc.), and the README file.

A.3 Limitations of MARC

In the present version of MARC the number of rows is limited to three; transmission gates are not supported. Some of the variables such as the number of transistors per gate, number of gates per row, the number of tracks per channel, etc., are given in the "sym.h" file and can be changed according to the needs.

PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Texas Tech University, I agree that the Library and my major department shall make it freely available for research purposes. Permission to copy this thesis for scholarly purposes may be granted by the Director of the Library or my major professor. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my further written permission and that any user may be liable for copyright infringement.

Disagree (Permission not granted)	Agree (Permission granted)
	Salustio Courtinho Rao
Student's signature	Student's signature
	05-26-1990
Date	Date