

5.3	HR-TrailTrace Algorithm.	111
5.3.1	Description	112
5.3.2	Design Example.	115
5.3.3	Optimality	120
5.3.4	Time Complexity	123
5.4	Complete Study of Practical Cells.	124
5.5	Planar Cell Layout.	132
VI.	CELL ARRAY WIDTH AND HEIGHT MINIMIZATION	133
6.1	Layout Problem	133
6.2	HRM-TrailTrace Algorithm	135
6.2.1	Description	135
6.2.2	Design Example	137
6.2.3	Optimality	142
6.2.4	Time Complexity	148
6.3	Experimental Results	149
6.3.1	Commercial Circuits	149
6.3.2	Published Layouts	151
VII.	CONCLUSIONS	153
7.1	Contributions	153
7.2	Practical Applications and Extensions	155
	BIBLIOGRAPHY	157
	INDEX	167

PREFACE

The layout of an integrated circuit (IC) is the process of assigning geometric shape, size and position to the components (transistors and connections) used in its fabrication. Since the number of components in modern ICs is enormous, computer-aided-design (CAD) programs are required to automate the difficult layout process. Prior CAD methods are inexact or limited in scope, and produce layouts whose area, and consequently manufacturing costs, are larger than necessary. This book addresses the problem of minimizing exactly the layout area of an important class of basic IC structures called CMOS cells.

First, we precisely define the possible goals in area minimization for such cells, namely width and height minimization, with allowance for area-reducing reordering of transistors. We reformulate the layout problem in terms of a graph model and develop new graph-theoretic concepts that completely characterize the fundamental area minimization problems for series-parallel and nonseries-parallel circuits. These concepts lead to practical algorithms that solve all the basic layout minimization problems exactly, both for a single cell and for a one-dimensional array of such cells. Although a few of these layout problems have been solved or partially solved previously, we present here the first complete solutions to all the problems of interest.

We have implemented our algorithms by means of computer programs, and demonstrate that they efficiently generate minimum-area layouts for all possible cells of practical size. An analysis of these layouts permits us to make the first comprehensive evaluation of the quality and scope of prior layout methods. We show that most previous layout optimization algorithms are limited in scope, and cannot find optimal layouts for a large percentage of practical circuits. We also compare our optimal layouts to those of nonoptimal heuristic methods, and demonstrate that our exact algorithms produce significantly smaller cells.

This book will be of interest to designers of circuits or CAD tools, and others concerned with generating highly compact layouts for ICs. Chapter 1 introduces the subject of IC layout, and surveys traditional layout styles. Chapter 2 gives a brief tutorial on the cell layout problem, and evaluates the effectiveness of prior cell layout

techniques at minimizing area. Chapter 3 develops our general theory of cell layout and presents algorithms for generating minimum-width cells for series-parallel circuits; Chapter 4 generalizes these results for the nonseries-parallel case. Chapter 5 presents an algorithm that generates minimum-width and -height cells, and these results are extended to an array of cells in Chapter 6. Chapter 7 summarizes the book and suggests applications and extensions to this work.

Most of the material in this book was developed over the past few years as part of the first author's Ph.D. dissertation research at the University of Michigan. This research was supported by grants from the Office of Naval Research and the National Science Foundation, and by fellowships from the Burroughs (now part of Unisys) and Schlumberger Corporations. We wish to express our gratitude to these organizations. We also thank William P. Birmingham, Richard B. Brown, Philip J. Hanlon and Ronald J. Lomax of the University of Michigan for their suggestions.

LAYOUT MINIMIZATION OF CMOS CELLS

CHAPTER I

INTRODUCTION

We begin by discussing the general design problem for integrated circuits, and the role of the cell layout problem in relation to it. Next, we survey the popular cell layout styles for integrated circuits. The cell layout area minimization problem we address is defined and prior work on it is evaluated. We conclude with a discussion of our approach to exact layout optimization.

1.1 PROBLEM AND MOTIVATION

Since the introduction of the first commercial integrated circuit (IC) in 1961, the capacity of digital integrated circuits has been increasing at a rapid pace. Commercial IC chips in the 1960's had about 100 transistors. Through dramatic improvements in fabrication technology, very large scale integrated (VLSI) circuits with over ten million transistors are being fabricated at present, and the number of transistors per chip tends to double every year or two. The design of chips with such huge numbers of transistors presents major problems in design effort and cost. Computer-aided design (CAD) tools were developed to help designers cope with these problems. Such tools now play a key role in managing design complexity and improving designer productivity.

The design of a VLSI chip is a complicated process requiring many intermediate steps. Typically, several hierarchical levels of representation are used in the design process; see Fig. 1.1. One of the higher levels is the *register transfer level*, where the variables and the data operators represent the hardware registers and functional blocks of the data-processing section of a chip. The control section implements the sequencing of operations implied by the register transfer description. Next comes the *logic level*, at which the functional blocks are basic logic units called *gates*, such as the AND and NOR gates shown in Fig. 1.1(b); these gates implement the carry function of the adder in Fig. 1.1(a). At the *transistor level*, the logic gates are decomposed into transistor circuits. The NOR gate of Fig. 1.1(b) is translated into the transistor circuit of Fig. 1.1(c), where a *transistor* is a three-terminal device with

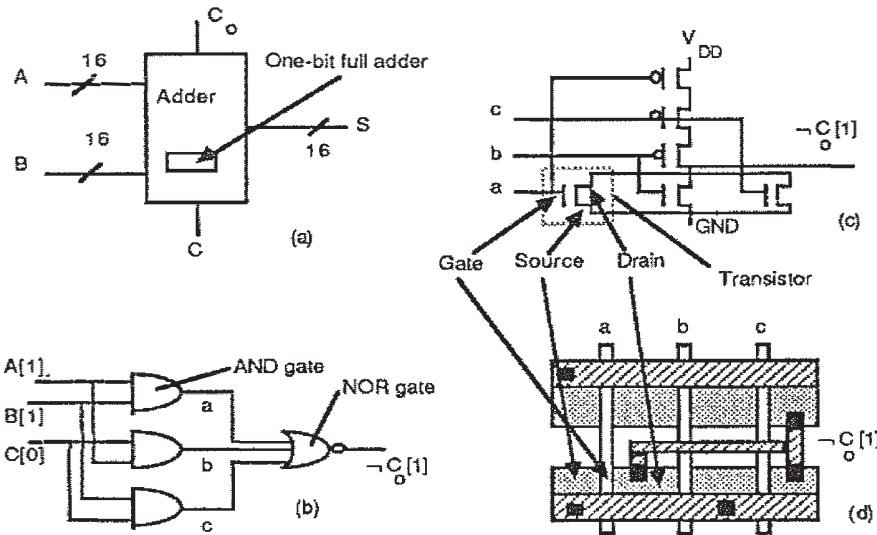


Fig. 1.1. Levels of design: (a) register transfer level (16-bit binary adder); (b) Logic level (carry circuit of a 1-bit adder); (c) transistor level (3-input NOR gate); (d) layout level (NOR gate in the CMOS functional cell style).

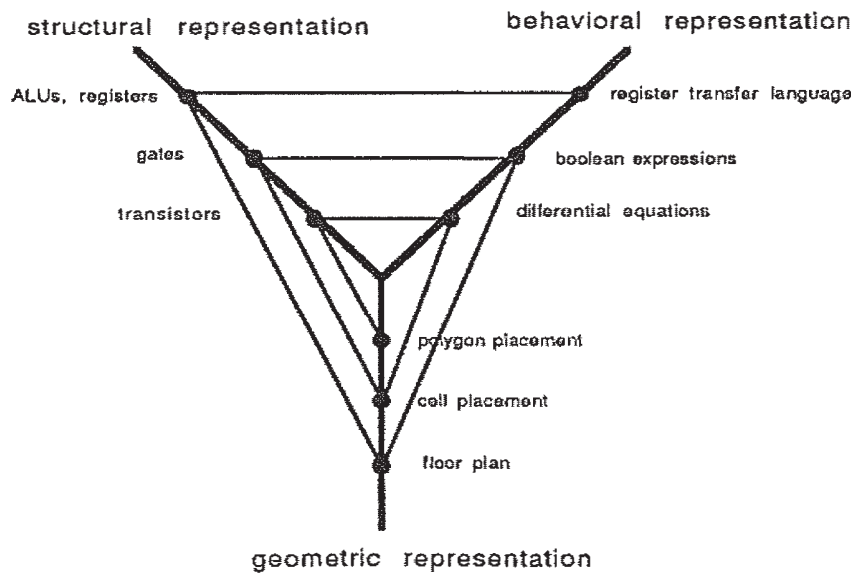


Fig. 1.2. Y chart representation of the integrated circuit design process [Ga82].

drain, source and gate terminals. The lowest design level is the *layout level*, where transistors are represented as geometric figures with dimensions such as length, width and position. The geometric representation or *layout* of a basic logic unit is called a *cell*. The layout of Fig. 1.1(d) is a cell implementing the NOR gate in Fig. 1.1(c). A large circuit may be laid out in the form of a set of interconnected cells. As in Fig. 1.1(d), the individual cells of the entire circuit are usually rectangular in shape. The layout design is transferred to mask-making facilities in this form for chip fabrication.

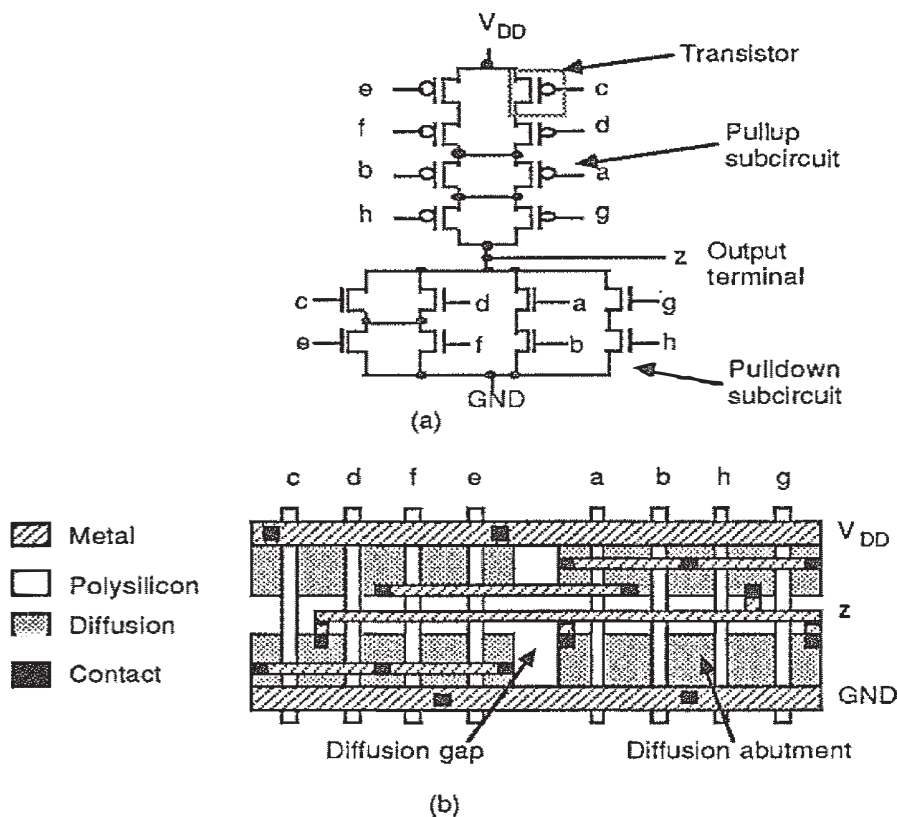


Fig. 1.3. A logic function implemented by a functional cell; (a) transistor circuit; (b) layout.

Each of the above design levels has behavioral, structural and geometric aspects. The Y chart in Fig. 1.2 illustrates these aspects, showing the order in which they are considered in a typical hierarchical, top-down design process [Ga85]. At the higher

levels, the behavioral aspects are more prominent, and the structural and geometric aspects are less important and less well defined. For example, at the register transfer level, the system is represented as registers and functional units such as an adder. These parts and their interconnection constitute the structure of the system. The behavior may be described by a register transfer language statement, such as

$$\text{register_A} := \text{register_B} + \text{register_C}.$$

Each of the functional units and interconnections at this level can be roughly sized, shaped, placed and oriented in what is called a floor plan [MC80]; see Fig. 1.2. At the transistor level, all structure is explicit; the parts are low-level design components, such as transistors and wires, and the logical behavior of each part is quite simple. At the layout level, the geometric figures composing each transistor and wire are assigned size and shape; each figure represents one of various materials used in fabricating an IC, such as diffusion, polysilicon and metal shown by different shadings in Fig. 1.3(b). Each of these materials occupies a different physical layer of the IC. At the layout level, structure and geometry are most prominent.

Each of the above design steps can be assisted or even be fully automated by CAD tools. Hardware description languages are needed at each level of representation to model the structure, behavior and geometry of the design as it is refined in top-down fashion. *Synthesis tools* are needed to translate the design from a higher level of representation, like the register transfer level, to a lower one, such as the layout level. *Analysis tools* are useful at each level of representation to analyze the quality of the design in terms of correctness and performance. Examples of CAD tools that aid IC designers are logic synthesizers, placement and routing programs, cell generators and simulators. A logic synthesizer can take a design described at the register transfer level and translate it into a set of logic (boolean) equations, accounting to some extent for speed, power and area requirements. These equations can be mapped directly into basic logic units such as AND, OR or NOT gates. A cell generator can produce a cell for each logic unit. The cells can be placed and routed on the chip by placement and routing tools. Simulation at the register transfer, gate and transistor levels can predict the performance of the design at each of these intermediate levels. The designer can use this information to decide whether the behavior of the design meets its specifications.

The *layout style* used in a chip design refers to the rules governing the transistor circuit structure (the circuit type), and the rules for mapping the circuit into the layout (the cell type), and has a profound effect on the amount of circuitry that fits on a chip of a given size. There are two major layout styles, which we discuss in detail in the next section. *Semi-custom* methods select manually designed cells from a cell library; these cells are well characterized in advance for speed and power. In

full-custom methods, on the other hand, a new type of cell is designed as needed for each logical unit. This can be done by human designers or by automatic cell generators, and requires simulation to analyze the performance of the cell.

This book is concerned with the translation of a design from the transistor circuit level to the layout level. Figure 1.3 shows the main concepts involved in this process. A transistor has three external terminals called the drain, source and gate; Figs. 1.1(c) and (d) show the correspondence between the transistor terminals and the layout. The conducting path between the drain and source terminals is open or closed depending upon the voltage level on the transistor's gate terminal; thus a transistor acts like an on-off switch. The transistor circuit of Fig. 1.3(a) is of a type called complementary MOS or CMOS, and uses two different types of transistors: p-channel MOS transistors in the pullup subcircuit and n-channel MOS transistors in the pulldown subcircuit. Each of these subcircuits has two designated terminals: the pullup subcircuit, whose designated terminals are the logical high voltage, referred to as V_{DD} or 1, and the output z ; and the pulldown subcircuit, whose designated terminals are the output z and the logical low voltage or ground, denoted as GND or 0. (The terminals to which the input signals $a, b, \dots, h$ are applied are not considered designated terminals.) The transistors composing the pullup and pulldown subcircuits create conducting paths connecting the output z either to V_{DD} or GND. Thus z implements a logical function that maps each combination of the input variables onto either 0 or 1.

An important layout task is to place and orient the transistors on the chip to minimize the layout area of a circuit. Figure 1.3(b) shows a cell implementing the transistor circuit of Fig. 1.3(a) in a very area-efficient layout style called a *functional cell* [Uv81]. The various types of shading indicate the materials used to fabricate the transistor terminals and the interconnections between these terminals. The drain and source terminals of a transistor are made from diffusion and the gate terminal is polysilicon. The pullup transistors of the circuit in Fig. 1.3(a) are placed side-by-side in the top diffusion row of the cell of Fig. 1.3(b) such that their drain and source terminals are adjacent, and the pulldown transistors are placed similarly in the lower row. Transistors in the two subcircuits sharing a common gate signal are placed in the same vertical column, which makes for easy routing of the signal between them using a polysilicon column. If the transistors are placed and oriented appropriately, the width of the cell can be reduced by arranging adjacent transistors to share the same drain/source terminal via diffusion abutment, as shown in Fig. 1.3(b). This obviates the need to have separate diffusion terminals which must be isolated by an area-wasting diffusion gap, as also illustrated in Fig. 1.3(b). A judicious placement and orientation of the transistors can result in fewer metal rows for interconnections, thus reducing cell height. Therefore, it is a basic goal of a layout generation tool to lay out the transistors of a circuit and their interconnections to minimize cell area.

We have chosen to address the topic of cell layout generation because it is so important in the overall chip design process. A significant reduction in the chip area required for a given circuit can be achieved by using full-custom layout, and a dramatic increase in productivity over hand layout can be realized by automatically generating these layouts. Layout generators exist for several full-custom layout styles; these styles are surveyed in the next section. Based on the level of recent research activity, the most promising of these styles is the functional cell. However, existing functional cell methods often produce layouts that have far from optimal area. This is due to the fact that most use heuristic or approximate approaches, rather than exact algorithms. An algorithmic layout methodology is needed that achieves minimal area in the functional cell layout style.

Therefore, the general problem being addressed in this book is that of generating minimum-area cell layouts of CMOS digital integrated circuits in the functional cell style. We present a complete solution to the problem of exact layout-area minimization under precisely specified conditions. We will demonstrate that exact area minimization of all practical-sized static CMOS circuits in the functional cell style is both computationally feasible and produces significantly smaller layouts than previous methods. We analyze the properties of the important class of dual planar complementary CMOS circuits, and present efficient algorithms for obtaining the optimal area layout of these circuits. These algorithms, which we have also programmed, are based on a complete theory of cell layout, which significantly extends prior work by Uehara and vanCleemput [Uv81] and others.

1.2 LAYOUT STYLES

Digital integrated circuits are often classified according to their layout styles, which are either semi-custom, such as gate arrays and standard cells, or full-custom. In this section we survey both semi-custom methods and the following full-custom styles: unstructured methods, programmable logic arrays, gate matrix methods, and functional cells.

1.2.1 Semi-Custom Layouts

Gate arrays constitute a popular approach to the design and layout of digital circuits [Ra80]. A conventional *gate array* consists of rows and columns of cells separated by routing areas for wires interconnecting the gates; see Fig. 1.4(a). These cells are groups of unconnected transistors that can be wired together to form various types of gates, such as AND (*), OR (+), NOT (−), NAND or NOR. Fig. 1.4(b) shows the structure of a CMOS cell, which has pMOS and nMOS transistors that are interconnected by metal wires (shown by the dotted lines) to form a two-input