Brewer, J.E., Zargham, M.R., Tragoudas, S., Tewksbury, S. “Integrated Circuits”
The Electrical Engineering Handbook
Ed. Richard C. Dorf
Boca Raton: CRC Press LLC, 2000

25

Integrated Circuits

Joe E. Brewer
Northrop Grumman Corporation

Medhi R. Zargham and Spyros Tragoudas
Southern Illinois University

Stuart Tewksbury
West Virginia University

25.1 Integrated Circuit Technology

Joe E. Brewer

Integrated circuit (IC) technology, the cornerstone of the modern electronics industry, is subject to rapid change. Electronic engineers, especially those engaged in research and development, can benefit from an understanding of the structure and pattern of growth of the technology.

Technology Perspective

A solid state IC is a group of interconnected circuit elements formed on or within a continuous substrate. While an integrated circuit may be based on many different material systems, silicon is by far the dominant material. More than 98% of contemporary electronic devices are based on silicon technology. On the order of 85% of silicon ICs are complementary metal oxide semiconductor (CMOS) devices.

From an economic standpoint the most important metric for an IC is the "level of functional integration." Since the invention of the IC by Jack Kilby in 1958, the level of integration has steadily increased. The pleasant result is that cost and physical size per function reduce continuously, and we enjoy a flow of new, affordable information processing products that pervade all aspects of our day-to-day lives. The historical rate of increase is a doubling of functional content per chip every 18 months.

For engineers who work with products that use semiconductor devices, the challenge is to anticipate and make use of these enhanced capabilities in a timely manner. It is not an overstatement to say that survival in the marketplace depends on rapid "design-in" and deployment.

For engineers who work in the semiconductor industry, or in its myriad of supporting industries, the challenge is to maintain this relentless growth. The entire industry is marching to a drumbeat. The cost of technology development and the investment in plant and equipment have risen to billions of dollars. Companies that lag behind face a serious loss of market share and, possibly, dire economic consequences.

Technology Generations

The concept of a technology generation emerged from analysis of historical records, was clearly defined by Gordon Moore in the 1960s, and codified as Moore's law. The current version of the law is that succeeding generations will support a four times increase in circuit complexity, and that new generations emerge at approximately 3-year intervals. The associated observations are that linear dimensions of device features change by a factor of 0.7, and the economically viable die size grows by a factor of 1.6.

Minimum feature size stated in microns (micrometers) is the term used most frequently to label a technology generation. "Feature" refers to a geometric object in the mask set such as a linewidth or a gate length. The "minimum feature" is the smallest dimension that can be reliably used to form the entity.

Figure 25.1 displays the technology evolution sequence. In the diagram succeeding generations are numbered using the current generation as the "0" reference. Because this material was written in 1996, the "0" generation is the 0.35 μm minimum feature size technology that began volume production in 1995.

An individual device generation has been observed to have a reasonably well-defined life cycle which covers about 17 years. The first year of volume manufacture is the reference point for a generation, but its lifetime actually extends further in both directions. As shown in Fig. 25.2, one can think of the stages of maturity as ranging over a linear scale which measures years to production in both the plus and minus directions. The 17-year life cycle of a single generation, with new generations being introduced at 3-year intervals, means that at any given time up to six generations are being worked on. This tends to blur the significance of research news and company announcements unless the reader is sensitive to the technology overlap in time.

To visualize this situation, consider Fig. 25.3. The top row lists calendar years. The second row shows how the life cycle of the 0.35 μm generation relates to the calendar. The third row shows the life cycle of the 0.25 μm generation vs. the calendar. Looking down any column corresponding to a specific calendar year, one can see which generations are active and identify their respective life cycle year.

generation 0	generation 1	generation 2	generation 3	generation 4	generation 5
1995	1998	2001	2004	2007	2010
0.35 μ	0.25 μ	0.18 μ	0.1 μ	0.07 μ	0.05 μ
production	development	research	research	research	research

FIGURE 25.1 Semiconductor technology generation time sequence.

			INDUSTRIAL RESEARCH				DEVELOPMENT					MANUFACTURING				
		UNIVERSITY RESEARCH					feasibility		productization							
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5

FIGURE 25.2 Life cycle of a semiconductor technology generation.

95	96	97	98	99	00	01	02	03	04	05	06	07	08	09	10	11
0.35 μm	1	2	3	4	5											
-3	-2	-1	0.25 μm	1	2	3	4	5								
-6	-5	-4	-3	-2	-1	0.18 μm	1	2	3	4	5					
-9	-8	-7	-6	-5	-4	-3	-2	-1	0.13 μm	1	2	3	4	5		
	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0.10 μm	1	2	3	4
				-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0.07 μm	1

FIGURE 25.3 Time overlap of semiconductor technology generations.

One should not interpret the 17-year life cycle as meaning that no work is being performed that is relevant to a generation before the 17-year period begins. For example, many organizations are conducting experiments directed at transistors with gate lengths smaller than 0.1 μm. This author's interpretation is that when basic research efforts have explored technology boundary conditions, the conditions are ripe for a specific generation to begin to coalesce as a unique entity. When a body of research begins to seek compatible materials and processes to enable design and production at the target feature size, the generation life cycle begins. This is a rather diffused activity at first, and it becomes more focused as the cycle proceeds.

National Technology Roadmap for Semiconductors

The National Technology Roadmap for Semiconductors (NTRS) is an almost 200-page volume distributed by the Semiconductor Industry Association (SIA). Focused on mainstream leading edge technology, the roadmap provides a common vision for the industry. It enables a degree of cooperative precompetitive research and development among the fiercely competitive semiconductor device manufacturers. It is a dynamic document which will be revised and reissued to reflect learning on an as-needed basis.

The NTRS is compiled by engineers and scientists from all sectors of the U.S. IC technology base. Industry, academia, and government organizations participate in its formulation. Key leaders are the Semiconductor Research Corporation (SRC) and SEMATECH industry consortia. The roadmap effort is directed by the Roadmap Coordinating Group (RCG) of the SIA.

The starting assumption of the NTRS is that Moore's law will continue to describe the growth of the technology. The overall roadmap comprises many individual roadmaps which address defined critical areas of semiconductor research, development, engineering, and manufacturing. In each area, needs and potential solutions for each technology generation are reviewed. Of course, this process is more definitive for the early generations because knowledge is more complete and the range of alternatives is restricted.

The NTRS document provides a convenient summary table which presents some of the salient characteristics of the six technology generations ranging from 1995 to 2010. That summary is reproduced (with minor variations in format) as Table 25.1.

TABLE 25.1 Overall Roadmap Technology Characteristics

	Year of First DRAM Shipment/Minimum Feature (μm)					
	1995/0.35	1998/0.25	2001/0.18	2004/0.13	2007/0.10	2010/0.07
Memory						
Bits/chip (DRAM/Flash)	64M	256M	1G	4G	16G	64G
Cost/bit @ volume (millicents)	0.017	0.007	0.003	0.001	0.0005	0.0002
Logic (high-volume microprocessor)						
Logic transistors/cm^2 (packed)	4M	7M	13M	25M	50M	90M
Bits/cm^2 (cache SRAM)	2M	6M	20M	50M	100M	300M
Cost/transistor @ volume (millicents)	1	0.5	0.2	0.1	0.05	0.02
Logic (low-volume ASIC)						
Transistors/cm^2 (auto layout)	2M	4M	7M	12M	25M	40M
Non-recurring engineering Cost/transistor (millicents)	0.3	0.1	0.05	0.03	0.02	0.01
Number of chip I/Os						
Chip to package (pads) high performance	900	1350	2000	2600	3600	4800
Number of package pins/balls						
Microprocessor/controller	512	512	512	512	800	1024
ASIC (high performance)	750	1100	1700	2200	3000	4000
Package cost (cents/pin)	1.4	1.3	1.1	1.0	0.9	0.8
Chip frequency (MHz)						
On-chip clock, cost performance	150	200	300	400	500	625
On-chip clock, high performance	300	450	600	800	1000	1100
Chip-to-board speed, high performance	150	200	250	300	375	475
Chip size (mm^2)						
DRAM	190	280	420	640	960	1400
Microprocessor	250	300	360	430	520	620

TABLE 25.1 (continued) Overall Roadmap Technology Characteristics

	Year of First DRAM Shipment/Minimum Feature (μm)					
	1995/0.35	1998/0.25	2001/0.18	2004/0.13	2007/0.10	2010/0.07
ASIC	450	660	750	900	1100	1400
Max number wiring levels (logic)						
On-chip	4–5	5	5–6	6	6–7	7–8
Electrical defect density (d/m^2)	240	160	140	120	100	25
Minimum mask count	18	20	20	22	22	24
Cycle time days (theoretical)	9	10	10	11	11	12
Maximum substrate diameter (mm)						
Bulk or epitaxial or SOI wafer	200	200	300	300	400	400
Power supply voltage (V)						
Desktop	3.3	2.5	1.8	1.5	1.2	0.9
Battery	2.5	1.8–2.5	0.9–1.8	0.9	0.9	0.9
Maximum power						
High performance with heatsink (W)	80	100	120	140	160	180
Logic without heatsink (W)	5	7	10	10	10	10
Battery (W)	2.5	2.5	3.0	3.5	4.0	4.5
Design and test						
Volume tester cost/pin ($K)	3.3	1.7	1.3	0.7	0.5	0.4
Number of test vectors (μP/M)	16–32	16–32	16–32	8–16	4–8	4
% IC function with BIST/DFT	25	40	50	70	90	90+

Related Topics

1.1 Resistors • 23.1 Processes

Further Information

The NTRS is available from the SIA, 181 Metro Drive, Suite 450, San Jose, CA 95110, telephone 408-436-6600, fax 408-436-6646. The document can also be accessed via the SEMATECH home page at <http://www.sematech.org>.

Information concerning the IC life cycle can be found in Larrabee, G. B. and Chatterjee, P. "DRAM Manufacturing in the 90's — Part 1: The History Lesson" and "Part 2: The Roadmap," Semiconductor International, pp. 84–92, May 1991.

25.2 Layout, Placement, and Routing

Mehdi R. Zargham and Spyros Tragoudas

Very large scale integrated (VLSI) electronics presents a challenge, not only to those involved in the development of fabrication technology, but also to computer scientists, computer engineers, and electrical engineers. The ways in which digital systems are structured, the procedures used to design them, the trade-offs between hardware and software, and the design of computational algorithms will all be greatly affected by the coming changes in integrated electronics.

A VLSI chip can today contain millions of transistors and is expected to contain more than 100 million transistors in the year 2000. One of the main factors contributing to this increase is the effort that has been invested in the development of computer-aided design (CAD) systems for VLSI design. The VLSI CAD systems are able to simplify the design process by hiding the low-level circuit theory and device physics details from the designer, and allowing him or her to concentrate on the functionality of the design and on ways of optimizing it.

A VLSI CAD system supports descriptions of hardware at many levels of abstraction, such as system, subsystem, register, gate, circuit, and layout levels. It allows designers to design a hardware device at an abstract level and progressively work down to the layout level. A layout is a complete geometric representation (a set of rectangles) from which the latest fabrication technologies directly produce reliable, working chips. A VLSI

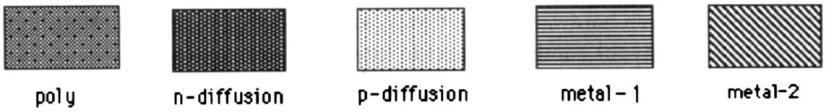

FIGURE 25.4 Different layers.

CAD system also supports verification, synthesis, and testing of the design. Using a CAD system, the designer can make sure that all of the parts work before actually implementing the design.

A variety of VLSI CAD systems are commercially available that perform all or some of the levels of abstraction of design. Most of these systems support a *layout editor* for designing a circuit **layout.** A layout-editor is software that provides commands for drawing lines and boxes, copying objects, moving objects, erasing unwanted objects, and so on. The output of such an editor is a design file that describes the layout. Usually, the design file is represented in a standard format, called Caltech Intermediate Form (CIF), which is accepted by the fabrication industry.

What Is Layout?

For a specific circuit, a layout specifies the position and dimension of the different layers of materials as they would be laid on the silicon wafer. However, the layout description is only a symbolic representation, which simplifies the description of the actual fabrication process. For example, the layout representation does not explicitly indicate the thickness of the layers, thickness of oxide coating, amount of ionization in the transistors channels, etc., but these factors are implicitly understood in the fabrication process. Some of the main layers used in any layout description are *n*-diffusion, *p*-diffusion, poly, metal-1, and metal-2. Each of these layers is represented by a polygon of a particular color or pattern. As an example, Fig. 25.4 presents a specific pattern for each layer that will be used through the rest of this section.

As is shown in Fig. 25.5, an *n*-diffusion layer crossing a poly layer implies an nMOS transistor, and a *p*-diffusion crossing poly implies a pMOS transistor.

Note that the widths of diffusion and poly are represented with a scalable parameter called *lambda.* These measurements, referred to as *design rules,* are introduced to prevent errors on the chip, such as preventing thin lines from opening (disconnecting) and short circuiting.

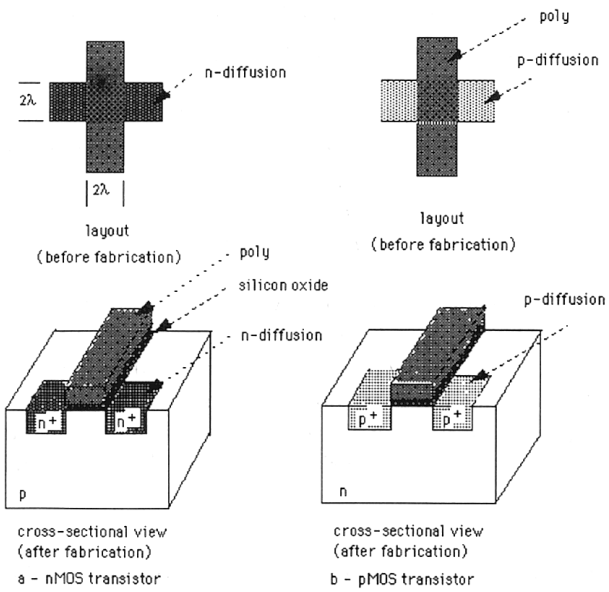

FIGURE 25.5 Layout and fabrication of MOS transistors.

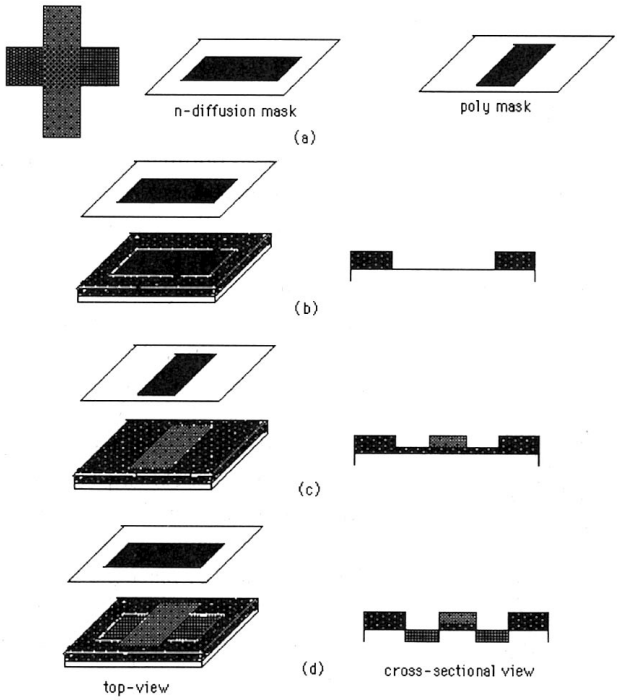

FIGURE 25.6 Fabrication steps for an nMOS transistor.

Implementing the design rules based on lambda makes the design process independent of the fabrication process. This allows the design to be rescaled as the fabrication process improves.

Metal layers are used as wires for connections between the components. This is because metal has the lowest propagation delay compared to the other layers. However, sometimes a poly layer is also used for short wires in order to reduce the complexity of the wire routing. Any wire can cross another wire without getting electrically affected as long as they are in different layers. Two different layers can be electrically connected together using *contacts.* The fabrication process of the contacts depends on types of the layers that are to be connected. Therefore, a layout editor supports different types of contacts by using different patterns.

From the circuit layout, the actual chip is fabricated. Based on the layers in the layout, various layers of materials, one on top of the others, are laid down on a silicon wafer. Typically, the processing of laying down each of these materials involves several steps, such as masking, oxide coating, lithography and etching [Mead and Conway, 1980]. For example, as shown in Fig. 25.6(a), for fabricating an nMOS transistor, first two masks, one for poly and one for *n*-diffusion, are obtained from the circuit layout. Next, the *n*-diffusion mask is used to create a layer of silicon oxide on the wafer [see Fig. 25.6(b)]. The wafer will be covered with a thin layer of oxide in places where the transistors are supposed to be placed as opposed to a thick layer in other places. The poly mask is used to place a layer of polysilicon on top of the oxide layer to define the gate terminals of the transistor [see Fig. 25.6(c)]. Finally, the *n*-diffusion regions are made to form the source and drain terminals of the transistor [see Fig. 25.6(d)].

To better illustrate the concept of layout design, the design of an inverter in the CMOS technology is shown in Fig. 25.7. An inverter produces an output voltage that is the logical inverse of its input. Considering the circuit diagram of Fig. 25.7(a), when the input is 1, the lower nMOS is on, but the upper pMOS is off. Thus, the output becomes 0 by becoming connected to the ground through the nMOS. On the other hand, if the input is 0, the pMOS is on and the nMOS is off, so the output must find a charge-up path through the pMOS to the supply and therefore becomes 1. Figure 25.7(b) represents a layout for such an inverter. As can be seen from this figure, the problem of a layout design is essentially reduced to drawing and painting a set of polygons. Layout editors provide commands for drawing such polygons. The commands are usually entered at the keyboard or with a mouse and, in some menu-driven packages, can be selected as options from a pull-down menu.

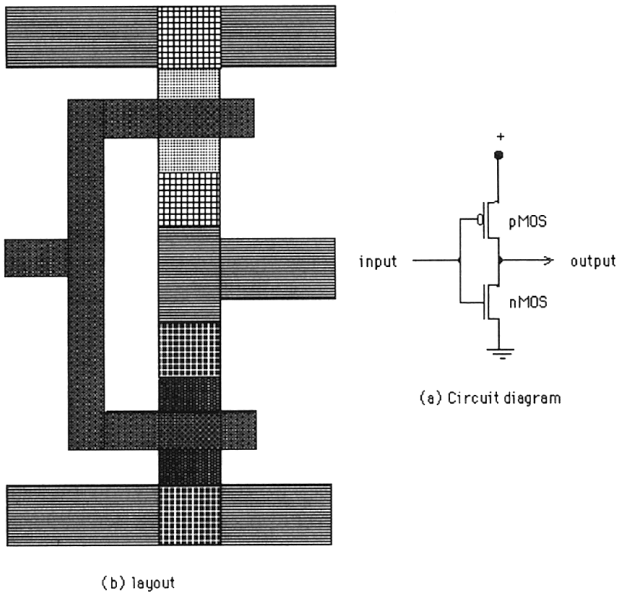

FIGURE 25.7 An inverter.

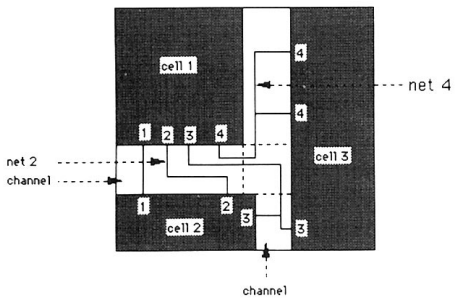

FIGURE 25.8 Placement and routing.,

In addition to the drawing commands, often a layout system provides tools for minimizing the overall area of the layout (i.e., size of the chip). Today a VLSI chip consists of a lot of individual cells, with each one laid out separately. A cell can be an inverter, a NAND gate, a multiplier, a memory unit, etc. The designer can make the layout of a cell and then store it in a file called the *cell library*. Later, each time the designer wants to design a circuit that requires the stored cell, he or she simply copies the layout from the cell library. A layout may consist of many cells. Most of the layout systems provide routines, called **floorplanning, placement** and **routing** routines, for placing the cells and then interconnecting them with wires in such a way that minimizes the layout area. As an example, Fig. 25.8 presents the placement of three cells. The area between the cells is used for routing. The entire routing surface is divided into a set of rectangular routing areas called channels. The sides of each channel consist of a set of terminals. A wire that connects the terminals with the same ID is called a net. The router finds a location for the wire segments of each net within the channel. The following sections classify various types of placement and routing techniques and provide an overview of the main steps of some of these techniques.

Floorplanning Techniques

The floorplanning problem in Computer Aided Design of Integrated Circuits is similar to that in Architecture and the goal is to find a location for each cell based on proximity (layout adjacency) criteria to other cells. We

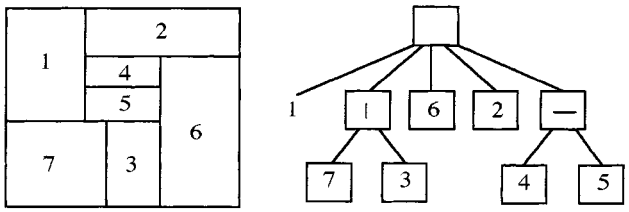

FIGURE 25.9 A hierarchical floorplan and its associated tree. The root node has degree 5. The internal node labeled with | indicates a vertical slicing. The internal node labeled with — indicates a horizontal slicing.

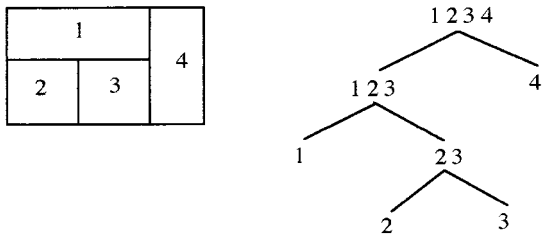

FIGURE 25.10 A sliceable floorplan and its associated binary tree.

consider rectangular floorplans whose boundaries are rectangles. It is desirable to obtain a floorplan that minimizes the overall area of the layout.

An important goal in floorplanning is the cell sizing problem where the goal is to determine the dimensions of variable cells whose area is invariant. All cells are assumed to be rectangular, and in the cell sizing problem the goal is to determine the width and height of each cell subject to predetermined upper and lower bounds on their ratio, and to their product being equal to its area, so that the final floorplan has optimal area.

One of the early approaches in floorplanning is the hierarchical, where recursive bipartition or partition into more than two parts is recursively employed and a floorplan tree is constructed. The tree simply reflects the hierarchical construction of the floorplan. Figure 25.9 shows a hierarchical floorplan and its associated tree. The partitioning problem and related algorithms are discussed extensively later in this section.

Many early hierarchical floorplanning tools insist that the floorplan be sliceable. A sliceable floorplan is recursively defined as follows: (a) a cell or (b) a floorplan that can be bipartitioned into two sliceable floorplans with either a horizontal or vertical line. Figure 25.10 shows a sliceable floorplan whose tree is binary.

Many tools that produce sliceable floorplans are still in use because of their simplicity. In particular, many problems arising in sliceable floorplanning are solvable optimally in polynomial time [Sarrafzadeh and Wong, 1996]. Unfortunately, sliceable floorplans are rarely optimal (in terms of their area), and they often result in layouts with very difficult routing phases. (Routing is discussed later in this section.) Figure 25.11 shows a compact floorplan that is not sliceable.

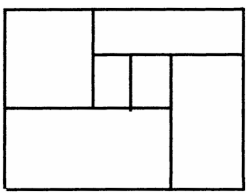

FIGURE 25.11 A compact layout that is not sliceable.

Hierarchical tools that produce nonsliceable floorplans have also been proposed [Sarrafzadeh and Wong, 1996]. The major problem in the development of such tools is that we are often facing problems that are intractable and thus we have to rely on heuristics in order to obtain fast solutions. For example, the cell sizing problem can be tackled optimally in sliceable floorplans [Otten, 1983 and Stockmeyer, 1983] but the problem is intractable for general nonsliceable floorplans.

A second approach to floorplanning is the rectangular dual graph. The idea here is to use duality arguments and express the cell adjacency constraints in terms of a graph, and then use an algorithm to translate the graph into a rectangular floorplan. A rectangular dual graph of a rectangular floorplan is a planar graph G = (V,E), where V is the set of cells and E is the set of edges, and an edge (C_1,C_2) is in E if and only if cells C_1 and C_2 are adjacent in the floorplan. See Fig. 25.12 for a rectangular floorplan and its rectangular dual graph G.

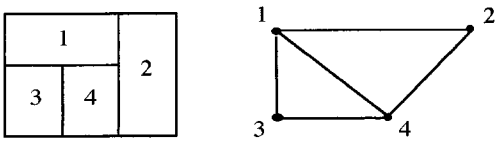

FIGURE 25.12 A rectangular floorplan and its associated dual planer graph.

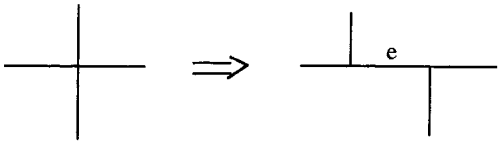

FIGURE 25.13 A cross junction can be replaced by 2 T-junctions.

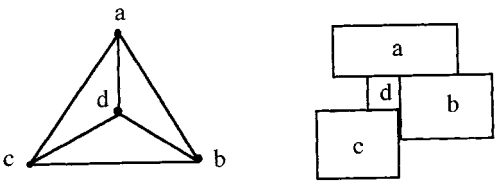

FIGURE 25.14 For a cycle of size 3 that is not a face we cannot satisfy all constraints.

Let us assume that the floorplan does not contain cross junctions. Figure 25.13 shows a cross junction. This restriction does not significantly increase the area of a floorplan because, as Fig. 25.13 shows, a cross junction can be replaced by two T-junctions by simply adding a short edge e.

It has been shown that in the absence of cross junctions the dual graph is planar triangulated (PT), and every T-junction corresponds to a triangulated face of the dual PT graph. Unfortunately, not all PT graphs have a rectangular floorplan. For example, in the graph of Fig. 25.14 we cannot satisfy the adjacency requirements of edges (a,b), (b,c) and (c,a) at the same time. Note that the later edges form a cycle of length three that is not a face. It has been shown that a PT graph has a rectangular floorplan if and only if it does not contain such cycles of length three. Moreover, a linear time algorithm to obtain such a floorplan has been presented [Sarrafzadeh and Wong, 1996]. The rectangular dual graph approach is a new method for floorplanning, and many floorplanning problems, such as the sizing problem, have not been tackled yet.

Rectangular floorplans can be obtained using simulated annealing and genetic algorithms. Both techniques are used to solve general optimization problems for which the solution space is not well understood. The approaches are easy to implement, but the algorithms have many parameters which require empirical adjustments, and the results are usually unpredictable.

A final approach to floorplanning, which unfortunately requires substantial computational resources and results to an intractable problem, is to formulate the problem as a mixed-integer linear programming (LP). Consider the following definitions:

W_i,H_i,R_i : width, height and area of cell C_i
X_i,Y_i : coordinates of lower left corner of cell C_i
X,Y : the width and height of the final floorplan
A_i,B_i : lower and upper bound for the ratio W_i/H_i of cell C_i
P_{ij}, Q_{ij} : variables that take 0/1 values for each pair of cells C_i and C_j

The goal is to find X_i,Y_i,W_i, and H_i for each cell so that all constraints are satisfied and XY is minimized. The latter is a nonlinear constraint. However, we can fix the width W and minimize the height of the floorplan as follows:

$$\min Y$$
$$X_i + W_i \le W$$
$$Y \ge Y_i + H_i$$

The complete mixed-integer LP formulation is [Sutanthavibul et al., 1991]:

$$\min Y$$
$$X_i, Y_i, W_i \ge 0$$
$$P_{ij}, Q_{ij} = 0 \text{ or } 1$$
$$X_i + W_i \le W$$
$$Y \ge Y_i + H_i$$
$$X_i + W_i \le X_j + W(P_{ij} + Q_{ij})$$
$$X_j + W_j \le X_i + W(1 - P_{ij} + Q_{ij})$$
$$Y_i + H_i \le Y_j + H(1 + P_{ij} - Q_{ij})$$
$$Y_j + H_j \le Y_i + H(2 - P_{ij} - Q_{ij})$$

When H_i appears in the above equations, it must be replaced (using first-order approximation techniques) by $H_i = D_i W_i + E_i$ where D_i and E_i are defined below:

$$W_{min} = \sqrt{R_i A_i}$$

$$W_{max} = \sqrt{R_i B_i}$$

$$H_{min} = \sqrt{R_i / B_i}$$

$$H_{max} = \sqrt{R_i / A_i}$$

$$D_i = (H_{max} - H_{min})/(W_{min} - W_{max})$$

$$E_i = H_{max} - D_i W_{min}$$

The unknown variables are X_i, Y_i, W_i, P_{ij}, and Q_{ij}. All other variables are known. The equations can then be fed into an LP solver to find a minimum cost solution for the unknowns.

Placement Techniques

Placement is a restricted version of floorplanning where all cells have fixed dimension. The objective of a placement routine is to determine an optimal position on the chip for a set of cells in a way that the total occupied area and total estimated length of connections are minimized. Given that the main cause of delay in a chip is the length of the connections, providing shorter connections becomes an important objective in placing a set of cells. The placement should be such that no cells overlap and enough space is left to complete all the connections.

All exact methods known for determining an optimal solution require a computing effort that increases exponentially with number of cells. To overcome this problem, many heuristics have been proposed [Preas and Lorenzetti, 1988]. There are basically three strategies of heuristics for solving the placement problem, namely, *constructive, partitioning,* and *iterative* methods. Constructive methods create placement in an incremental manner where a complete placement is only available when the method terminates. They often start by placing a *seed* (a seed can be a single cell or a group of cells) on the chip and then continuously placing other cells based on some heuristics such as size of cells, connectivity between the cells, design condition for connection lengths, or size of chip. This process continues until all the cells are placed on the chip. Partitioning methods divide the cells into two or more partitions so that the number of connections that cross the partition boundaries

is minimized. The process of dividing is continued until the number of cells per partition becomes less than a certain small number. Iterative methods seek to improve an initial placement by repeatedly modifying it. Improvement might be made by transforming one cell to a new position or switching positions of two or more cells. After a change is made to the current placement configuration based on some cost function, a decision is made to see whether to accept the new configuration. This process continues until an optimal (in most cases a near optimal) solution is obtained. Often the constructive methods are used to create initial placement on which an iterative method subsequently improves.

Constructive Method

In most of the constructive methods, at each step an unplaced cell is selected and then located in the proper area. There are different strategies for selecting a cell from the collection of unplaced cells [Wimer and Koren, 1988]. One strategy is to select the cell that is most strongly connected to already placed cells. For each unplaced cell, we find the total of its connections to all of the already placed cells. Then we select the unplaced cell that has the maximum number of connections. As an example consider the cells in Fig. 25.15. Assume that cells c_1 and c_2 are already placed on the chip. In Fig. 25.16 we see that cell c_5 has been selected as the next cell to be placed. This is because cell c_5 has the largest number of connections (i.e., three) to cells c_1 and c_2.

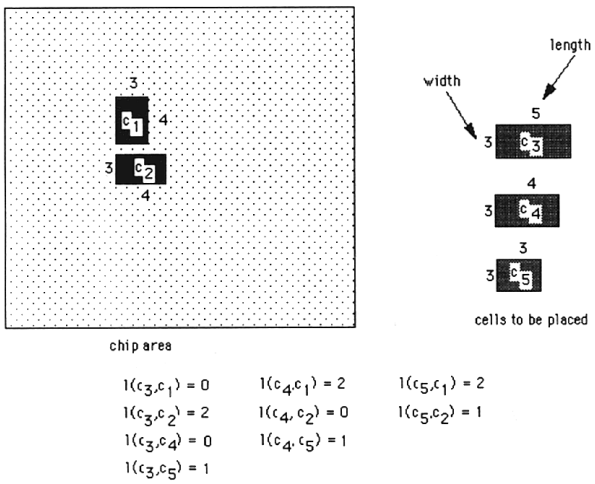

FIGURE 25.15 Initial configuration.

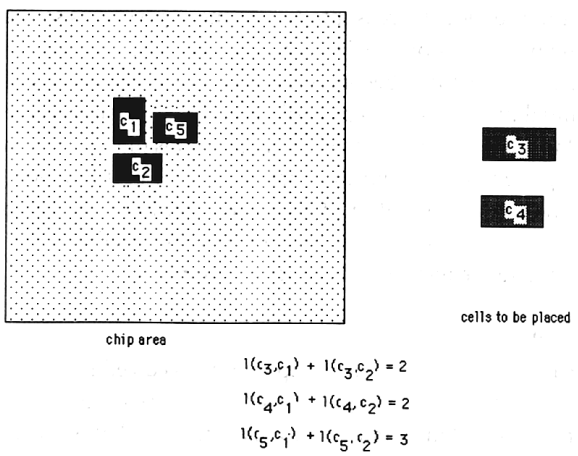

FIGURE 25.16 Selection based on the number of connections.

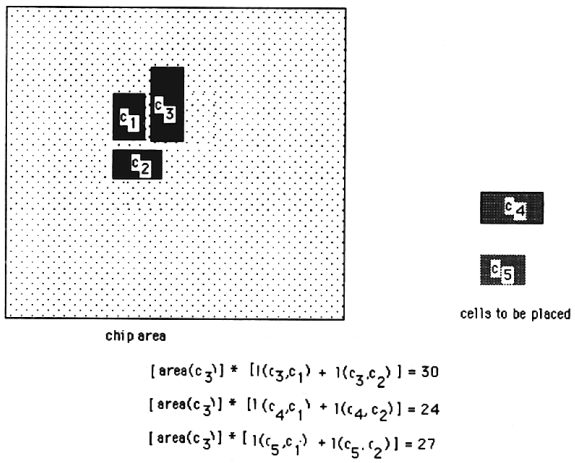

FIGURE 25.17 Selection based on the number of connections and area.

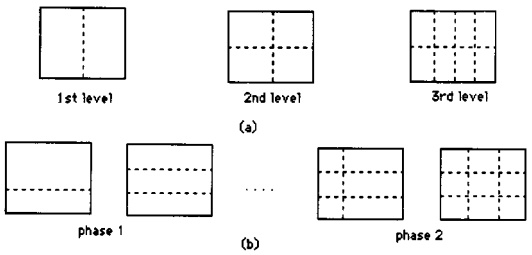

FIGURE 25.18 Partitioning.

The foregoing strategy does not consider area as a factor and thus results in fragmentation of the available free area; this may make it difficult to place some of the large unplaced cells later. This problem can be overcome, however, by considering the product of the number of connections and the area of the cell as a criteria for the selection. Figure 25.17 presents an example of such a strategy. Cell c_3 is selected as the next choice since the product of its area and its connections to c_1 and c_2 combine to associate with the maximum value.

Partitioning Method

The approaches for the partitioning method can be classified as quadratic and sliced bisection. In both approaches the layout is divided into two subareas, A and B, each having a size within a predefined range. Each cell is assigned to one of these subareas. This assignment is such that the number of interconnections between the two subareas is minimal. For example, Fig. 25.18 presents successive steps for the quadratic and sliced-bisection methods. As shown in Fig. 25.18(a), in the first step of the quadratic method the layout area is divided into two almost equal parts; in the second step the layout is further divided into four almost equal parts in the opposite direction. This process continues until each subarea contains only one cell. Similar to the quadratic method, the sliced-bisection also divides the layout area into several subareas.

The sliced-bisection method has two phases. In the first phase, the layout area is iteratively divided into a certain number of almost equal subareas in the same direction. In this way, we end up with a set of slices [see Fig. 25.18(b)]. Similarly, the second phase divides the area into a certain number of subareas; however, the slicing is done in the opposite direction.

Several heuristics have been proposed for each of the preceding partitioning methods. Here, for example, we emphasize the work of Fiduccia–Mattheyses [Fiduccia and Mattheyses, 1982], which uses the quadratic method. For simplicity, their algorithm is only explained for one step of this method. Initially the set of cells is randomly divided into two sets, A and B. Each set represents a subarea of the layout and has size equal to the area it represents. A cell is selected from one of these sets to be moved to the other set. The selection of the

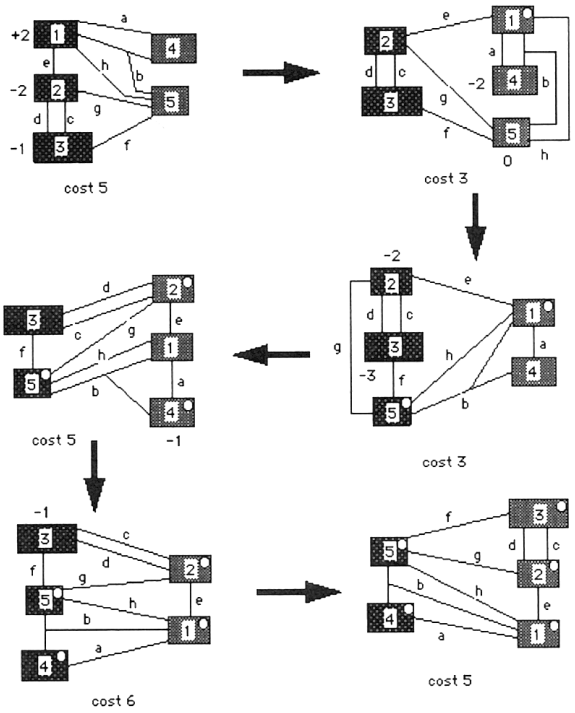

FIGURE 25.19 Illustration of a pass.

cell depends on three criteria. The cell should be free, i.e., the cell must have a minimum gain among the gains of all other free cells. A cell c has gain g if the number of interconnections between the cells of the two sets decreases by g units when c is moved from its current set to the other. Finally, the selected set's move should not violate a predefined balancing criterion that guarantees that the sizes of the two sets are almost equal. After moving the selected cell from its current set to the complementary set, it is no longer free. A new partition, which corresponds to the new instance of the two sets A and B, is created. The cost of a partition is defined as the number of interconnections between cells in the two sets of the partition. The Fiduccia–Mattheyses algorithm keeps track of the best partition encountered so far, i.e., the partition with the minimum cost. The algorithm will move the selected cell to the complementary set even if its gain is negative. In this case, the new partition is worse than the previous one, but the move can eventually lead to better partitions.

This process of generating new partitions and keeping track of the best encountered partition is repeated until no free cells are left. At this point of the process, which is called a pass, the algorithm returns the best partition and terminates. To obtain better partitions, the algorithm can be modified such that more passes occur. This can easily be done by selecting the best partition of a pass as the initial partition of the next pass. In this partition, however, all cells are free. The modified algorithm terminates whenever a new pass returns a partition that is no better than the partition returned by the previous pass. This way, the number of passes generated will never be more than the number of the interconnections in the circuit and the algorithm always terminates.

The balancing criterion in the Fiduccia–Mattheyses algorithm is easily maintained when the cells have uniform areas. The only way a pass can be implemented to satisfy the criterion is to start with a random initial partition in which the two sets differ by one cell, each time select the cell of maximum gain from the larger sized set, move the cell and generate a new partition, and repeat until no free cells are left.

In the example of Fig. 25.19, the areas of the cells are nonuniform. However, the assigned cell areas ensure that the previously described operation occurs so that the balancing criterion is satisfied. (The cell areas are omitted in this figure, but they correspond to the ones given in Fig. 25.15.) Figure 23.19 illustrates a pass of the Fiduccia–Mattheyses algorithm. During this pass, six different partitions are generated, i.e., the initial partition and five additional ones. Note that according to the description of the pass, the number of additional partitions equals the number of cells in the circuit.

Each partition consists of the cells of the circuit (colored according to the set to which they belong and labeled with an integer), the gain value associated with each free cell in the set from which the selected cell will be moved (this value can be a negative number), and the nets (labeled with letters). In the figure a cell that is no longer free is distinguished by an empty circle placed inside the rectangle that represents that cell.

The initial partition has cost 5 since nets *a*, *b*, *h*, *g*, *f* connect the cells in the two sets. Then the algorithm selects cell 1, which has the maximum gain. The new partition has cost 3 (nets *e*, *g*, *f*), and cell 1 is no longer free. The final partition has no free cells. The best partition in this pass has cost 3.

Iterative Method

Many iterative techniques have been proposed. Here, we emphasize one of these techniques called simulated annealing. Simulated annealing, as proposed by Kirkpatrick et al. [1983], makes the connection between statistical mechanics and combinatorial optimization problems. The main advantage with simulated annealing is its hill-climbing ability, which allows it to back out of inferior local solutions and find better solutions.

Sechen has applied simulated annealing to the placement problem and has obtained good solutions [Sechen, 1990]. The method basically involves the following steps:

BEGIN

1. Find an initial configuration by placing the cells randomly. Set the initial temperature, *T*, and the maximum number of iterations.
2. Calculate the cost of the initial configuration.
   A general form of the cost function may be: $Cost = c_1 * Area\ of\ layout + c_2 * Total\ interconnection\ length$ where c_1 and c_2 are tuning factors.
3. While (stopping criterion is not satisfied)
   {587
   a. For (maximum number of iteration)
      {
      1.Transform the old configuration into a new configuration.
      This transformation can be in the form of exchange of positions of two randomly selected cells or change of position of a randomly selected cell.
      2.Calculate the cost of the new configuration.
      3.If (new cost < old cost) accept the iteration, else check if the new iteration could be accepted with the probability: e ($|$new cost – old cost$|$ / *T*). There are also other options for the probability function.
      }
   b. Update the temperature.
      }

END

The parameter *T* is called *temperature;* it is initially set to a very large value, so that the probability of accepting "uphill" moves is very close to 1, that it is slowly decreasing toward zero, according to a rule called the cooling schedule. Usually, the new reduced temperature is calculated as follows:

$$\text{New temperature} = (\text{cooling rate}) \times (\text{Old temperature})$$

Using a faster cooling rate can result in getting stuck at local minima; however, a cooling rate that is too slow can pass over the possible global minima. In general, the cooling rate is taken from approximately 0.80 to 0.95.

Usually, the stopping criterion for the while-loop is implemented by recording the cost function's value at the end of each temperature stage of the annealing process. The stopping criterion is satisfied when the cost function's value has not changed for a number of consecutive stages.

Though simulated annealing is not the ultimate solution to placement problems, it gives very good results compared to the other popular techniques. The long execution time of this algorithm is its major disadvantage. Although a great deal of research has been done in improving this technique, substantial improvements have not been achieved.

Routing Techniques

Given a collection of cells placed on a chip, the routing problem is to connect the terminals (or ports) of these cells for a specific design requirement. The routing problem is often divided into three subproblems: global, detailed, and specialized routers. The global router considers the overall routing region in order to distribute the nets over the channels based on their capacities while keeping the length of each net as short as possible. For every channel that a net passes through, the net's id is placed on the sides of the channel. Once the terminals of each channel are determined, the detailed router connects all the terminals with the same id by a set of wire segments. (A wire segment is a piece of material described by a layer, two end-points, and a width.) The specialized router is designed to solve a specific problem such as routing of power and ground wires. These wires require special attention for two reasons: 1) they are usually routed in one layer in order to reduce the parasitic capacitance of contacts, and 2) they are usually wider than other wires (signal and data) since they carry more current.

Detailed routers further divide into general-purpose and restricted routers. The general-purpose routers impose very few constraints on the routing problem and operate on a single connection at a time. Since these routers work on the entire design in a serial fashion, the size of the problems they can attempt is limited. On the other hand, the restricted routers require some constraints on the routing problem, such as empty rectangular areas with all of the pins on the periphery. Because of their limited scope, these routers can do a better job of modeling the contention of nets for the routing resources and therefore can be viewed as routing the nets in parallel.

To reduce the complexity of restricted routers, this type of router often uses a rectangular grid on which trunks (horizontal wire segments) and branches (vertical wire segments) are placed on different layers. In other words, the layers supported by the technology are divided into two groups, horizontal and vertical. This is known as the Manhattan model. On the other hand, in a non-Manhattan model, the assignment of a layer to a vertical or horizontal direction is not enforced. Given the freedom of direction, assignment to layers reduces the channel width and vias in many cases; the latter model usually produces a better result than the former.

In the literature, many different techniques have been proposed for restricted routers. In general these techniques can be grouped into four different approaches: 1) algorithms (such as left-edge, maze, greedy, hierarchical); 2) expert systems; 3) neural networks; and 4) genetic algorithms [Zobrist, 1994; Sarrafzadeh, 1996; and Lengauer, 1990]. As an example, we consider here only one of the techniques, called a maze router which is widely known. The maze router can be used as a global router and/or detailed router. It finds the shortest rectilinear path by propagating a wavefront from a source point toward a destination point [Lee, 1969]. Considering the routing surface as a rectangular array of cells, the algorithm starts by marking the source cell as visited. In successive steps, it visits all the unvisited neighbors of visited cells. This continues until the destination cell is visited. For example, consider the cell configuration given in Fig. 25.20.

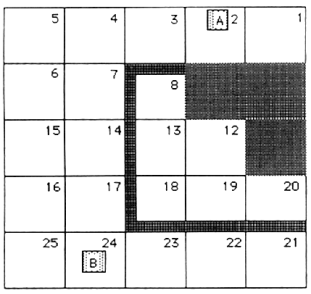

FIGURE 25.20 Initial configuration.

We would like to find a minimal-crossing path from source cell A (cell 2) to destination cell B (cell 24). (The minimal-crossing path is defined as a path that crosses over the fewest number of existing paths.) The algorithm begins by assigning the start cell 2 to a list, denoted as L, i.e., L consists of the single entry {2}. For each entry in L, its immediate neighbors (which are not blocked) will be added to an auxiliary list L′. (The auxiliary cell list L′ is provided for momentary storage of names of cells.) Therefore, list L′ contains entries {1,3}. To these cells a chain coordinate and a weight are assigned, as denoted in Fig. 25.21. For example, in cell 3, we have the pair (0, →), meaning that the chain coordinate is toward right and the cell weight is 0. The weight for a cell represents the number of wires that should be crossed in order to reach that cell from the source cell. The cells with minimum weight in list L′ are appended

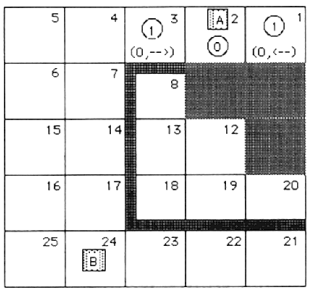

FIGURE 25.21 First step.

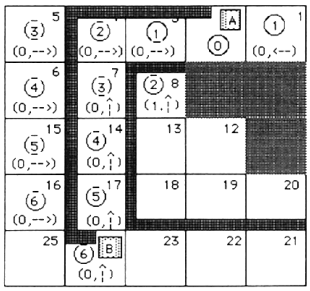

FIGURE 25.22 Final step.

to list L. Thus, cells 1 and 3 are appended to list L. Moreover, cell 2 is erased from list L. Appending the immediate neighbors of the cells in L to L′, we find that list L′ now contains entries 4 and 8. Note that cell 8 has a weight of 1; this is because a wire must be crossed in order to reach to this cell. Again the cells with minimum weight in list L′ are appended to list L, and cell 3 and cell 1 are erased from L. Now L contains entry {4}. The above procedure is repeated until it reaches to the final cell B. Then a solution is found by tracing the chain coordinated from cell B to cell A as shown in Fig. 25.22.

The importance of Lee's algorithm is that it always finds the shortest path between two points. Since it routes one net at a time, however, there is a possibility of having some nets unrouted at the end of the routing process. The other weak points of this technique are the requirements of a large memory space and long execution time. For this reason, the maze router is often used as a side router for the routing of critical nets and/or routing of leftover unrouted nets.

Defining Terms

Floorplanning: A floorplan routine determines an approximate position for each cell so that the total area is minimized.

Layout: Specifies the position and dimension of the different layers of materials as they would be layed on the silicon wafer.

Placement: A placement routine determines an optimal position on the chip for a set of cells with fixed dimensions in a way that the total occupied area and the total estimated length of connections are minimized.

Routing: Given a collection of cells placed on a chip, the routing routine connects the terminals of these cells for a specific design requirement.

Related Topic

23.1 Processes

References

C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," *Proceedings of the 19th Annual Design Automation Conference,* (July), pp. 175–181, 1982.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science,* vol. 220, no. 4598 (May), pp. 671–680, 1983.

C. Y. Lee, "An algorithm for path connections and its application," *IRE Transactions on Electronic Computers,* (Sept.), pp. 346–365, 1969.

T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout,* New York: John Wiley & Sons, 1990.

C. A. Mead and L. A. Conway, *Introduction to VLSI Systems,* Reading, Mass.: Addison-Wesley, 1980.

R. H. J. M. Otten, "Efficient floorplan optimization," *International Journal on Computer Design,* pp. 499–503, IEEE/ACM, 1983.

B. Preas and M. Lorenzetti, *Physical Design Automation of VLSI Systems,* Menlo Park, Calif.: Benjamin/Cummings, 1988.

M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design,* New York: McGraw-Hill, 1996.
C. Sechen, "Chip-planning, placement and global routing of macro-cell integrated circuits using simulated annealing," *International Journal of Computer Aided VLSI Design 2,* pp. 127–158, 1990.
L. Stockmeyer, "Optimal orientation of cells in slicing floorplan designs," *Information and Control,* 57 (2) pp. 91–101, 1983.
S. Sutanthavibul, E. Shargowitz, and J. B. Rosen, "An analytical approach to floorplan design and optimization," *IEEE Transactions on Computer Aided-Design,* 10 (6) pp. 761–769, 1991.
S. Wimer and I. Koren, "Analysis of strategies for constructive general block placement," *IEEE Transactions on Computer-Aided Design,* vol. 7, no. 3 (March), pp. 371–377, 1988.
G. W. Zobrist, Editor, *Routing, Placement, and Partitioning,* Ablex Publishing, 1994.

Further Information

Other recommended layout design publications include Weste and Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective,* Reading, Mass.: Addison-Wesley, 1988, and the book by B. Preas and M. Lorenzetti, *Physical Design Automation of VLSI Systems,* Menlo Park, Calif.: Benjamin/Cummings, 1988. The first book describes the design and analysis of a layout. The second book describes different techniques for development of CAD systems.

Another source is *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* which is published monthly by the Institute of Electrical and Electronics Engineers.

25.3 Application-Specific Integrated Circuits

S. K. Tewksbury

Introduction

Present-day silicon very large scale integration complementary metal-oxide semiconductor (**VLSI CMOS**) technologies can place and interconnect several million transistors (representing over a million gates) on a single integrated circuit (**IC**) approximately 1 cm square. Provided with such a vast number of gates, a digital system designer can implement very sophisticated and complex system functions (including full systems) on a single IC. However, efficient design (including optimized performance) of such functions using all these gates is a complex puzzle of immense complexity. If this technology were to have been provided to the world overnight, it is doubtful that designers could in fact make use of this vast amount of logic on an IC.

However, this technology instead has evolved over a long period of time (about three decades), starting with only a few gates per IC, with the number of gates per IC doubling consistently about every 18 months (a progression referred to as Moore's Law), and evolving to the present high gate densities per IC. Projections [The National Technology Roadmap for Semiconductors, 1994] of this evolution over the next 15 years, as shown in Table 25.2, promise continued dramatic advances in the amount of logic and memory which will be provided on individual ICs. Paralleling the technology evolution, computer-aided design (**CAD**) tools and electronics design automation (**EDA**) tools [Rubin, 1987; Sherwani, 1993; Hill and Peterson, 1993; Banerjee, 1994] have evolved to assist designers of the increasingly complex ICs. With these CAD tools, today's design teams effectively have an army of very "experienced" designers embedded in the tools, capable of applying the knowledge gained over the long and steady history of IC evolution. This prior experience captured in CAD/EDA tools includes the ability to convert a high-level description [Camposano and Wolf, 1991; Gajski et al., 1992] of a specific function (e.g., ALU, register, control unit, microcontroller, etc.) into an efficient physical implementation of that function.

Figure 25.23 is a photomicrograph of a contemporary, high-performance VLSI application-specific IC (**ASIC**) circuit, the ADSP-1060 (SHARC) digital signal processor (DSP) from Analog Devices, Inc. The right two thirds of the IC is 4 Mbit of SRAM, providing considerable on-chip memory. The DSP is on the left third of the IC. A 0.5-μm CMOS technology with two levels of metal was used, with a total of about 20 million transistors on the IC. The IC provides 120 MFLOP of performance.

TABLE 25.2 Prediction of VLSI Evolution by Semiconductor Industries Association

Year	1995	1998	2001	2004	2007	2010
Feature size (μm)	0.35	0.25	0.18	0.13	0.10	0.07
DRAM bits/chip	64M	256M	1G	4G	16G	64G
ASIC gates/chip	5M	14M	26M	50M	210M	430M
Chip size (ASIC) (mm^2)	450	660	750	900	1100	1400
Maximum number of wiring levels (logic)	4–5	5	5–6	6	6–7	7–8
On-chip speed (MHz)	300	450	600	800	1000	1100
Chip-to-board speed (MHz)	150	200	250	300	375	475
Desktop supply voltage (V)	3.3	2.5	1.8	1.5	1.2	0.9
Maximum power (W), heatsink	80	100	120	140	160	180
Maximum power (W), no heatsink	5	7	10	10	10	10
Power (W), battery systems	2.5	2.5	3.0	3.5	4.0	4.5
Number of I/Os	900	1350	2000	2600	3600	4800

Adapted from *The National Technology Roadmap for Semiconductors,* Semiconductor Industry Association, San Jose, Calif., 1994.

This section summarizes the design process, the gate-level physical design, and several issues which have become particularly important with today's VLSI technologies, with a focus on ASICs. An important and growing issue is that of testing, including built-in testing, design for testability, and related topics (see e.g., Jha and Kundu [1990] and Parker [1992]). This is a broad topic, beyond the scope of this section.

Primary Steps of VLSI ASIC Design

The VLSI IC design process consists of a sequence of well-defined steps [Preas and Lorenzetti, 1988; Hill et al., 1989; DeMicheli, 1994a and b] related to the definition of the functions to be designed; organization of the circuit blocks implementing these logic functions within the area of the IC; verification and simulation at several stages of design (e.g., behavioral simulation, gate-level simulation, circuit simulation [White and Sangiovanni-Vincentelli, 1987; Lee et al., 1993], etc.); routing of physical interconnections among the blocks, and final detailed placement and transistor-level layout of the VLSI circuit. This process can also be used hierarchically to design one of the blocks making up the overall IC, representing that circuit block in terms of simpler blocks. This establishes the "top-down" hierarchical approach, extendable to successively lower-level elements of the overall design.

These general steps are illustrated in Fig. 25.24(a), roughly showing the basic steps taken. A representative example [Lipman, 1995] of a contemporary design approach is illustrated in Fig. 25.24(b). Design approaches are continually changing and Fig. 25.24(b) is merely one of several current design sequences. Below, we summarize the general steps highlighted in Fig. 25.24(a).

A. *Behavioral Specification of Function:* The behavioral specification is essentially a description of the function expected to be performed by the IC. The design can be represented by schematic capture, with the designer representing the design using block diagrams. High-level description languages (**HDLs**) such as VHDL [Armstrong, 1989; Lipsett et al., 1990; Mazor and Langstraat, 1992] and Verilog [Thomas and Moorby, 1991] are increasingly used to provide a detailed specification of the function in a manner which is largely independent of the physical design of the function. VHDL and Verilog are "hardware description languages," representing designs from a variety of viewpoints including behavioral descriptions, structural descriptions, and logical descriptions. Figure 25.25(a) illustrates the specification of the overall function in terms of subfunctions (A(s), B(s), …, E(s)) as well as expansion of one subfunction (C(s)) into still simpler functions (c1, c2, c3, …, c6).

B. *Verification of Function's Behavior:* It is important to verify that the behavior specification of today's complex ICs properly represents the behavior desired. In the case of HDL languages, there may be software "debugging" of the "program" until the desired behavior is obtained, much as programming languages need to be debugged until correct operation is obtained. Early verification is important since any errors in the specification of the function will lead to an IC which is faulty as a result of the design, rather than of physical defects.

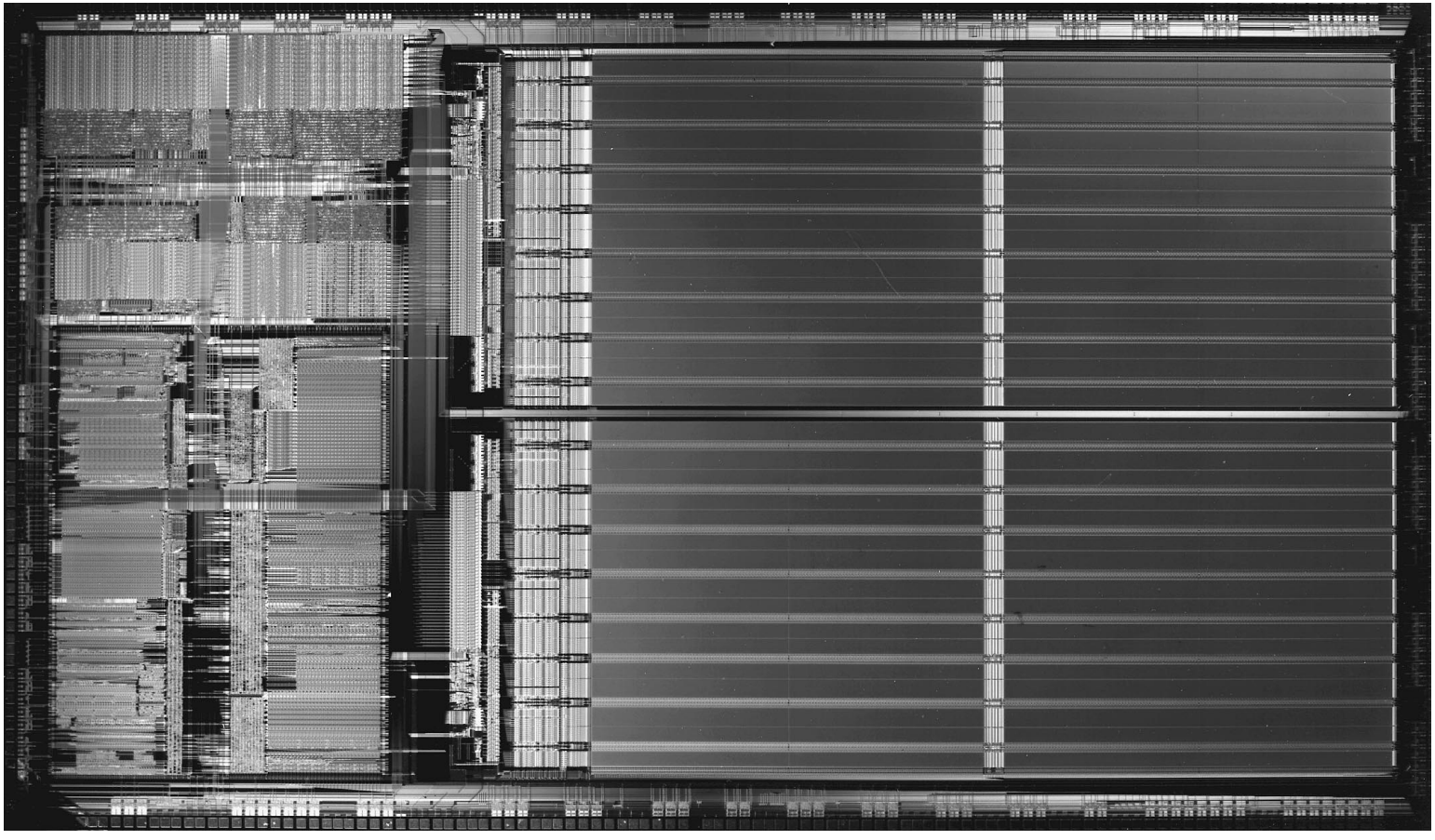

FIGURE 25.23 Photomicrograph of the SHARC DSP of Analog Devices, Inc. (Courtesy of Douglas Garde, Analog Devices, Inc., Norwood, Mass.)

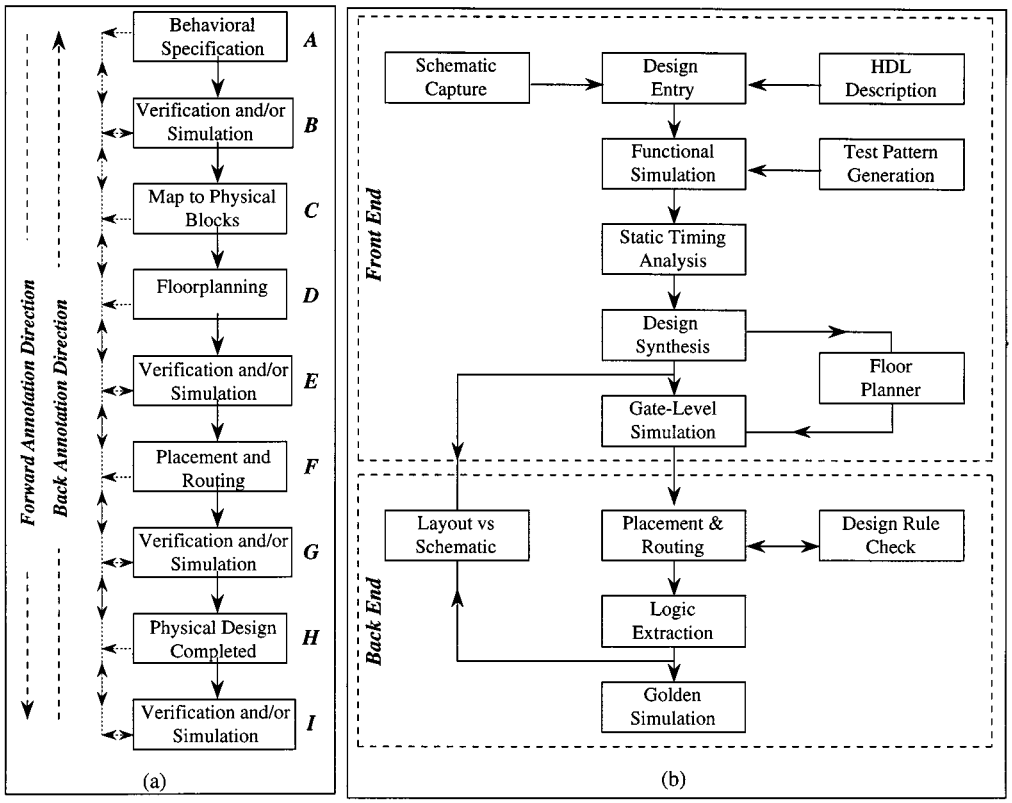

FIGURE 25.24 Representative VLSI design sequences. (a) Simplified but representative sequence. (b) Example design approach from recent trade journal [Lipman, 1995].

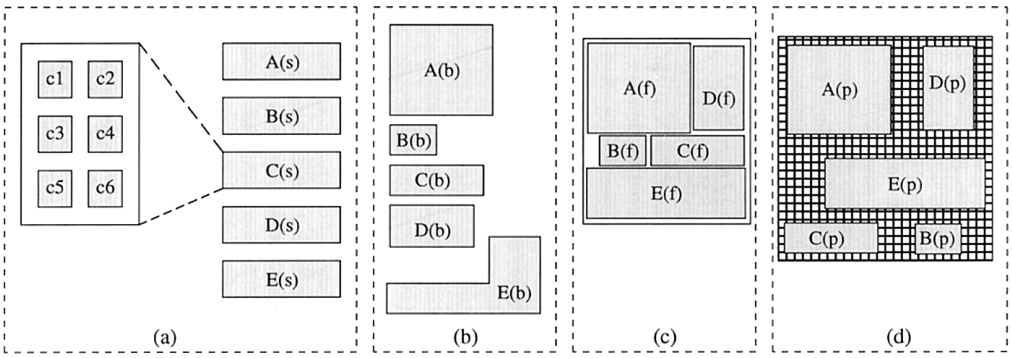

FIGURE 25.25 Circuit stages of design. (a) Initial specification (e.g., HDL, schematic, etc.) of ASIC function in terms of functions, with the next lower level description of function C illustrated. (b) Estimated size of physical blocks implementing functions. (c) Floorplanning to organize blocks on an IC. (d) Placement and routing of interconnections among blocks.

C. *Mapping of Logical Function into Physical Blocks:* Next, the logical functions, e.g., A(s), B(s), …, E(s) in Fig. 25.25(a), are converted into physical circuit blocks, e.g., blocks A(b), B(b), …, E(b) in Fig. 25.25(b). Each physical block represents a logic function as a set of interconnected gates. Although details of the physical layout are not known at this point, the estimated area and aspect ratio (ratio of height to width) of each circuit block is needed to organize these blocks within the area of the IC.

D. *Floorplanning:* Next, the individual circuit blocks must be compactly arranged to fit within the minimum area. Floorplanning establishes this organization, as illustrated in Fig. 25.25(c). At this stage, routing of interconnections among blocks has not been performed, perhaps requiring modifications to the floor plan after routing. During floor planning, the design of a logic function in terms of a block function can be modified to achieve shapes which better match the available IC area. For example, the block E(b) in Fig. 25.25(b) has been redesigned to provide a different geometric shape for block E(f) in the floor plan in Fig. 25.25(c).

E. *Verification/Simulation of Function Performance:* Given the floorplan, it is possible to estimate the average length of interconnections among the blocks (with the actual length not known until after interconnection routing in step F below. Signal timing throughout the IC is estimated, allowing verification that the various circuit blocks interact within timing margins.

F. *Placement and Routing:* When an acceptable floorplan has been established, the next step is to complete the routing of interconnections among the various blocks of the design. As interconnections are routed, the overall IC area expands to provide the area for the wiring, with the possibility that the original floorplan (which ignored interconnections) is not optimal. Placement [Shahookar and Mazumder, 1991] considers various rearrangements of the circuit blocks, without changing their internal design but allowing rotations, etc. For example, the arrangement of some of the blocks in Fig. 25.25(c) have been changed in the arrangement in Fig. 25.25(d).

G. *Verification/Simulation of Performance:* Following placement and routing, the detailed layout of the circuit blocks and interconnections has been established and more accurate simulations of signal timing and circuit behavior can be performed, verifying that the circuit behaves as desired with the interblock interconnections in place.

H. *Physical Design at Transistor/Cell Level:* As the above steps are completed, the design is moving closer toward a full definition of the ASIC circuit in terms of a set of physical masks precisely specifying placement of all the transistors and interconnections. In this step, that process is completed.

I. *Verification/Simulation of Performance:* Before fabricating the masks and proceeding with manufacture of the ASIC circuit, a final verification of the ASIC is normally performed. Figure 25.24(b) represents this step as "golden simulation," a process based on detailed and accurate simulation tools, tuned to the process of the foundry and providing the final verification of desired performance.

Increasing Impact of Interconnection Delays on Design

In earlier generations of VLSI technology (with larger transistors and wider interconnection lines/spacings), delays through the low-level logic gates greatly dominated delays along interconnection lines. This was largely the result of the lower resistance of the larger cross-section interconnections. Under these conditions, a single pass through a design sequence such as shown in Fig. 25.24 was often adequate. In particular, the placement of blocks on the ICs, although impacting the lengths of interconnections among blocks, did not have a major impact on performance since the gate delays within the blocks were substantially larger than interconnection delays between blocks. Under such conditions, the steps of floorplanning and of placement and routing focused on such objectives as minimum overall area and minimum interconnection area.

However, as feature sizes have decreased below about 0.5 μm, this condition has changed and current VLSI technology has interconnection delays substantially larger than logic delays. The increasing importance of interconnection delays is driven by several effects. The smaller feature size leads to interconnections with a higher resistance R^* per unit length and with a higher capacitance C^* per unit area (the capacitance increase also reflecting additional metal layers and coupling capacitances). For interconnection lines among high-level blocks (spanning the IC), the result is a larger RC time constant ($R^*LC \times {}^*L$), with L the line length. While interconnect delays are increasing, gate delays are decreasing. Figure 25.26 illustrates the general behavior on technology scaling to smaller features. A logic function F in a previous-generation technology requires a smaller physical area and has a higher speed in a later, scaled technology (i.e., a technology with feature sizes decreased). Although the intrablock line lengths decrease (relaxing the impact within the block of higher R^*C^*), the interblock lines continue to have lengths proportional to the overall IC size (which is increasing), with the larger R^*C^* leading to increased RC delays on such interconnections.

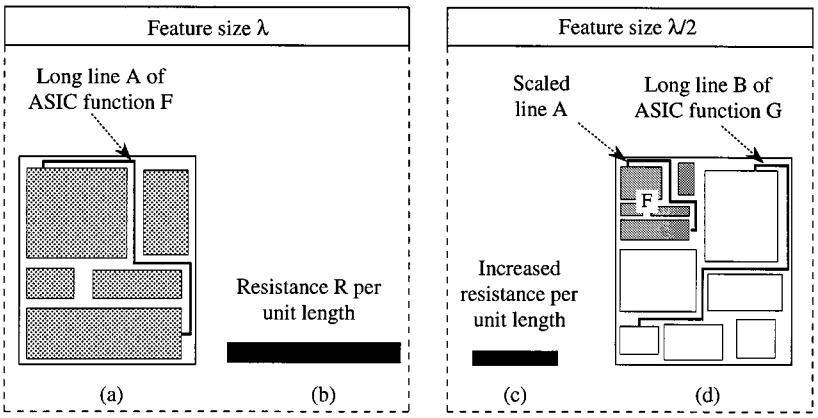

FIGURE 25.26 Interconnect lengths under scaling of feature size. (a) Initial VLSI ASIC function F with line A extending across IC. (b) Interconnection cross section with resistance R^* per unit length. (c) Interconnection cross section, in scaled technology, with increased resistance per unit length. (d) VLSI ASIC function G in scaled technology containing function F but reduced in size (including interconnection A) and containing a long line B extending across the IC.

As the interconnection delays have become increasingly dominant, the design process has evolved into an iterative process through the design steps, as illustrated by the back-and-forward annotation arrows in Fig. 25.24(a). Initial estimates of delays in step B need to be refined through back-annotation of interconnection delay parameters obtained after floorplanning and/or after placement and routing to reflect the actual interconnection characteristics, perhaps requiring changes in the initial specification of the desired function in terms of logical and physical blocks. This iterative process moving between the logical design and the physical design of an ASIC has been problematic since often the logical design is performed by the company developing the ASIC, whereas the physical design is performed by the company (the "foundry") fabricating the ASIC. CAD tools are an important vehicle for coordination of the interface between the designer and the foundry.

General Transistor Level Design of CMOS Circuits

The previous section has emphasized the CAD tools and general design steps involved in designing an ASIC. A top-down approach was emphasized, with the designer addressing successively more-detailed portions of the overall design through a hierarchical organization of the overall description of the function. However, the design process also presumes a considerable understanding of the bottom-up principles through which the overall IC function will eventually appear as a fully detailed specification of the transistor and interconnection structures throughout the overall IC [Dillinger, 1988; Weste and Eshraghian, 1993; Wolf, 1994; Rabaey, 1996; Kang and Leblebici, 1996].

Figure 25.27 illustrates the transistor-level description of a simple three-input, NAND gate. VLSI ASIC logic circuits are dominated by this general structure, with the **PMOS transistors** (making up the pull-up section) connected to the supply voltage V_{dd} and the **NMOS transistors** (making up the pull-down section) connected to the ground return *GND*. When the logic function generates a logic "1" output, the pull-up section is shorted through its PMOS transistors to *Vdd* leading to a high output voltage while the pull-down section is open, with no connection of *GND* to the output. When generating a logic "0" output, the pull-down section is shorted to *GND* while the pull-up section is open (no path to V_{dd}). Since the output is either a 1 or a 0, only one of the sections (pull-up or pull-down) is shorted, with no dc current flowing directly from V_{dd} to *GND* through the logic circuit pull-up and pull-down sections.

The PMOS transistors used in the pull-up section are fabricated with P-type source and drain regions on N-type substrates. The NMOS transistors used in the pull-down section, on the other hand, are fabricated with N-type source and drain regions on P-type substrates. Since a given silicon wafer is either N-type or P-type, a deep, opposite doping-type region must be placed in the silicon wafer for those transistors needing a substrate of the opposite type. The shaded regions in Fig. 25.27(a) represent the "substrate types" within which the

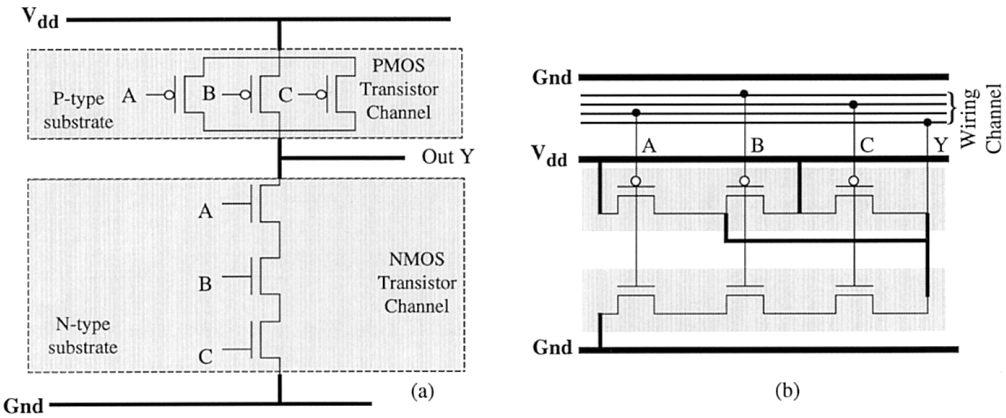

FIGURE 25.27 Transistor representation of three-input NAND gate. (a) Transistor representation without regard to layout. (b) Transistor representation using parallel rows of PMOS and NMOS transistors, with interconnections connected from a wiring channel.

transistors are fabricated. A deep doping of the desired substrate type is provided, with transistors fabricated within such deep, substrate doping "wells." To allow tight packing of transistors, large substrate doping wells are used, with a large number of transistors placed in each well.

Each logic cell must be connected to power (V_{dd}) and ground (*Gnd*), requiring that external power and ground connections to the IC be routed (on continuous metal lines to avoid resistive voltage drops) to each logic cell on the IC. Early IC technologies provided only a single level of metallization on which to route power and ground and an interdigitated layout, illustrated in Fig. 25.28(a), was adopted. Given this power and ground layout approach, channels of pull-up sections and channels of pull-down sections were placed between the power and ground interconnections, as illustrated in Fig. 25.28(b).

Bottom-up IC design is also hierarchical, with the designer completing detailed layout of a specific logic function (e.g., a binary adder), placing that detailed layout (a *cell*) in a library of physical designs and then reusing that library cell when other instantiations of the cell are required elsewhere in the IC. Figure 25.29 illustrates this cell-based approach, with cells of common height but varying width placed in rows between the power and ground lines. The straight power and ground lines shown in Fig. 25.29 allow tight packing of adjacent rows.

To achieve tight packing of cells within a row, adjacent cells (Fig. 25.29) are abutted against each other. Since metal interconnections are used within cells (and most basic cells are reused throughout the design), it is generally not possible to route intercell metal interconnections over cells. This constraint can be relaxed, as shown in Fig. 25.30(b), when additional metal layers are available, restricting a subset of the layers for intracell use and allowing over-the-cell routing [Sherwani et al., 1995] with the other layers.

When metal intercell interconnections cannot be safely routed over cells, interconnection channels must be provided between rows of logic cells, leading to the ***wiring channels*** above and/or below rows of logic cells as shown in Fig. 25.29. In this approach, all connections to and from a logic cell are fed from the top and/or bottom wiring channel. The width of the wiring channel is adjusted to provide space for the number of intercell interconnections required in the channel. Special cells providing through-cell routing can be used to support short interconnections between adjacent rows of cells. Given this layout style at the lowest level of cells, larger functions can be readily constructed, as illustrated in Fig. 25.30(a). Figure 25.29(b) illustrates interconnections (provided in the polysilicon layer under the metal layers) to the logic cells from the wiring channel. For classical CMOS logic cells, the same set of input signals are applied to both the pull-up and pull-down sections, as in the example in Fig. 25.27(b). By organizing the sequence of transistors along the pull-up and pull-down sections properly, inputs can extend vertically between a PMOS transistor in the pull-up section and a corresponding NMOS transistor in the pull-down section. Algorithms to determine the appropriate ordering of transistors evolved early in CAD tools.

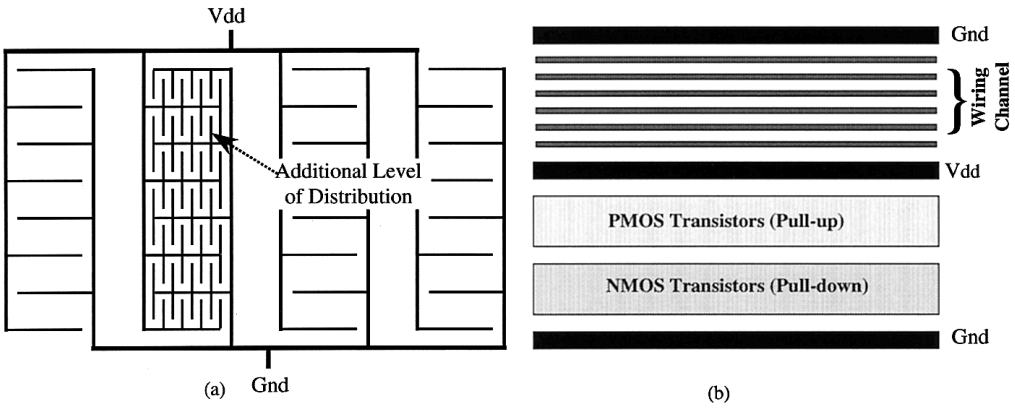

FIGURE 25.28 Power and ground distribution (interdigitated lines) with rows of logic cells and rows of wiring channels. (a) Overall power distribution and organization of logic cells and wiring channels. (b) Local region of power distribution network.

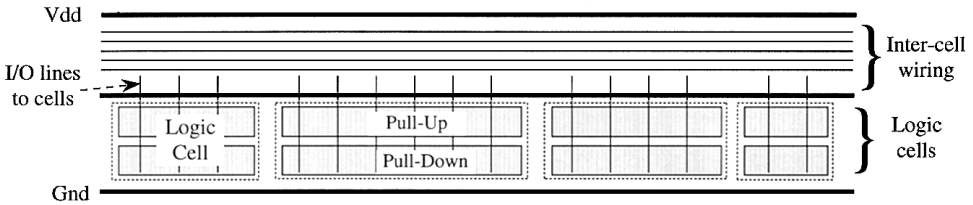

FIGURE 25.29 Cell-based logic design, with cells organized between power and ground lines and with intercell wiring in channels above (and/or below, also) the cell row.

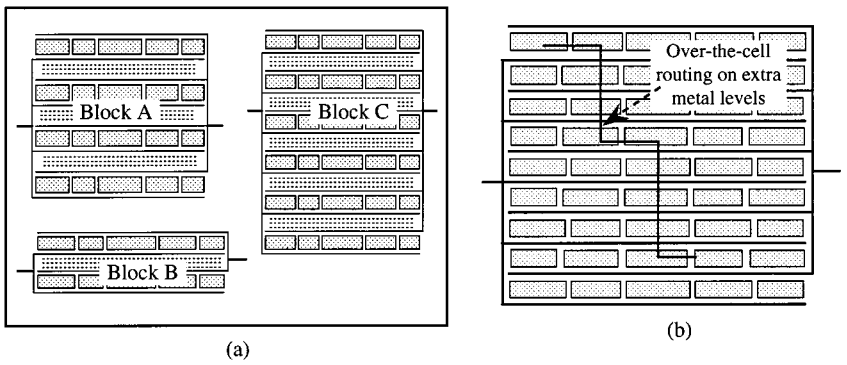

FIGURE 25.30 (a) Construction of larger blocks from cells with wiring channels between rows of cells within the block. (b) Over-the-cell routing on upper metal levels which are not used within the cells.

ASIC Technologies

Drawing on the discussion above, the primary ASIC technologies (gate arrays, sea-of-gate arrays, standard cell ASICs, ASICs with "megacells," and field-programmable gate arrays) can be easily summarized. For comparison, full custom VLSI is briefly described first.

Full Custom Design

In *full custom design,* custom logic cells are designed, starting at the lowest level (i.e., transistor-based cell design) and extending to higher levels (e.g., combinations of cells for higher-level functions) to create the overall IC function. Figure 25.31(a) illustrates the general layout at the cell level. The designer can exploit new cell designs

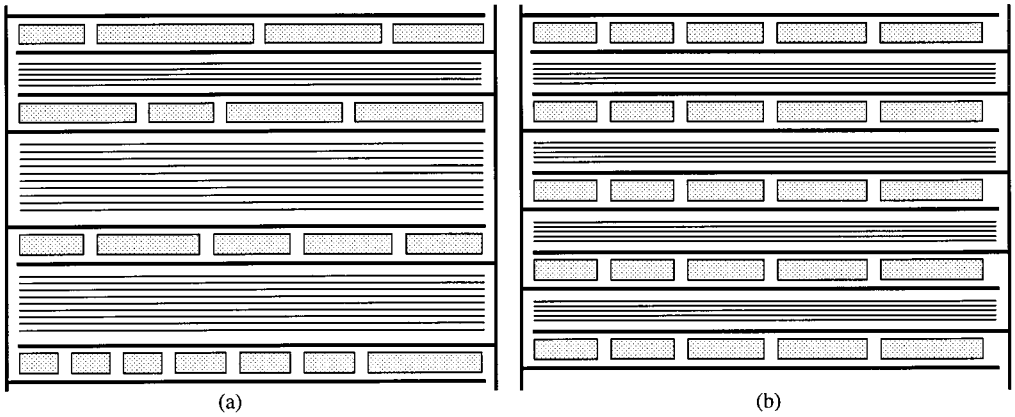

FIGURE 25.31 (a) Full custom layout (with custom cells) or standard cell ASIC layout (using library cells). (b) Gate array layout, with fixed width wiring channels (in prefabricated, up to metallization, wafers).

which improve the performance of the specific function being designed, can provide interconnections through wiring areas between logic cells to create compact functions, can use the full variety of CMOS circuit designs (e.g., dynamic logic, pass-transistor logic, etc.), can use cells previously developed privately for earlier ICs, and can use cells from a standard library provided by the foundry.

Standard Cell ASIC Technology

Consider a full custom circuit design completed using only predefined logic cells from a foundry-provided specific library and physical design processes provided by standard EDA/CAD tools. This approach, *standard cell design* [Heinbuch, 1987, *SCMOS Standard Cell Library,* 1989], is one of the primary ASIC technologies. A critical issue impacting standard cell ASIC design is the quality of the standard cell library provided by the foundry. By providing a rich set of library cells, the coordination between the logical design and the physical design is substantially more effective. Figure 25.31(a) also illustrates the general standard cell approach, using standard library cells of design-specified height (according to cells used), design-specified width logic rows, and varying width of the wiring channel to accommodate the number of interconnection wires determined during place and route.

Gate Array ASIC Technology

The *gate array technology* [Hollis, 1987] is based on partially prefabricated (up to but not including the final metallization layer) wafers with simple gate cells. Such *noncustomized* wafers are stockpiled and the ASIC designer specifies the final metallization layer added to *customize* the gate array. Gate array cells draw on the general cell design shown earlier, Fig. 25.27(b). Figure 25.32 illustrates representative non-customized transistor-level cells although different foundries use different physical layouts) with the dashed lines representing metal layers (including power and ground) added during the final metallization process.

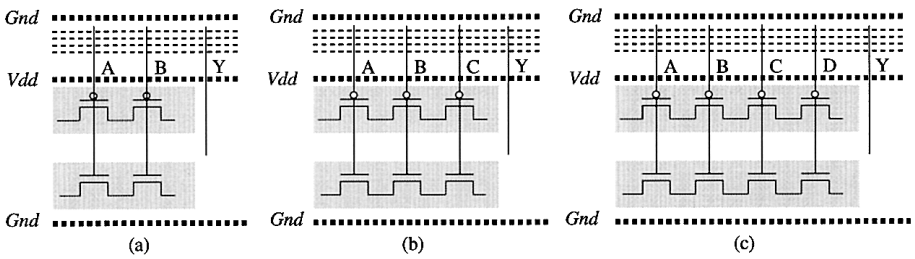

FIGURE 25.32 Basic noncustomized gate array cells (two input, three and four input examples). The dashed lines represent the power and ground lines, as well as interconnections in the wiring channel, which are placed on the IC during customization.

The ASIC designer's task is to translate the desired VLSI logic function into a physical design using the basic gate cells provided on the noncustomized IC. To avoiding the routing of intercell interconnections around long rows of logic cells, some of the "cells" along the row are feedthrough cells, allowing routing of interconnections to other logic cell rows. Included in the designer's toolset are library functions predefining the construction of more-complex logic functions (e.g., adders, registers, multipliers, etc.) from the gate cells on the noncustomized gate array IC.

The gate array technology shares the costs of masks among all ASIC customers and exploits high-volume production of the noncustomized wafers. On the other hand, construction of higher-level functions must adhere to the predefined positions and type of gate cells, leading to less-efficient and lower-performance designs than in the standard cell approach. In addition, the width of the wiring channel is fixed, limiting the number of parallel interconnections in the channel and imposing wasted area if all available wires are not used.

Sea-of-Gates ASIC Technology

The *sea-of-gates* technology also uses premanufactured, noncustomized gate arrays. However, additional metallization layers are provided, with lower-level metallization layer(s) used to program the internal function of the cells and the upper-level layer(s) used for over-the-cell routing [Sherwani et al., 95] of signals among cells, such as illustrated earlier in Fig. 25.30(b). This eliminates the need for wiring channels and feedthrough cells, leading to denser arrays of transistors.

CMOS Circuits Using Megacell Elements

The examples above have focused on low-level logic functions. However, as the complexity of VLSI ICs has increased, it has become increasingly important to include standard, high-level functions (e.g., microprocessors, DSPs, PCI interfaces, MPEG coders, RAM arrays, etc.) within an ASIC. For example, an earlier generation of microprocessor may offer the necessary performance and would occupy only a small portion of the area of a present-generation ASIC. Including such a standard microprocessor has the advantages of allowing the ASIC design to be completed more quickly as well as providing users with the microprocessor standard instruction set and software development tools. Such large cells are called *megacells*. As VLSI technologies advance and as standards increasingly impact design decisions, the use of standard megacells will become increasingly common.

Field-Programmable Gate Arrays: Evolving to an ASIC Technology

The *field-programmable gate array* (FPGA), like the gate array, places fixed cells on the wafer, and the FPGA designer constructs more-complex functions from these cells. However, the cells provided on the FPGA can be substantially more complex than the simple gates provided on the gate array. In addition, the term *field programmable* highlights the customizing of the ASIC by the user, rather than by the foundry manufacturing the FPGA. The *mask-programmable gate array* (MPGA) is similar to the FPGA (using more-complex cells than the gate array), but the programming is performed by addition of the metal layer by the FPGA manufacturer.

Figure 25.33 shows an example of cells and programmable interconnections for a representative FPGA technology [Actel, 1995]. The array of cells is constructed from two types of cell, which alternate along the logic cell rows of the FPGA. The combinational logic cell — "C-module" in Fig. 25.33(a) — provides a ROM-based lookup table (LUT) able to efficiently implement a complex logic function (with four data inputs and two control signals). The sequential cell (S-module) adds a flip-flop to the combinational module, allowing efficient realization of sequential circuits. The interconnection approach illustrated in Fig. 25.33(c) is based on (1) short vertical interconnections directly connecting adjacent modules, (2) long vertical interconnections extending through the overall array, (3) long horizontal interconnections extending across the overall array, (4) points at which the long vertical interconnections can be connected to cells, and (5) points at which the long vertical and horizontal lines can be used for general routing. The long horizontal and vertical lines are broken into segments, with programmable links between successive segments. The programmer can then connect a set of adjacent line segments to create the desired interconnection line. In addition, programmable connection points allow the programmer to make transitions between the long vertical lines and the long horizontal lines. By connecting various inputs the cell of an FPGA to either V_{dd} or *GND*, the cell can be "programmed" to perform one of its possible functions. The basic array is complemented by additional driver and other circuitry around the perimeter of the FPGA for interfacing to the "external world."

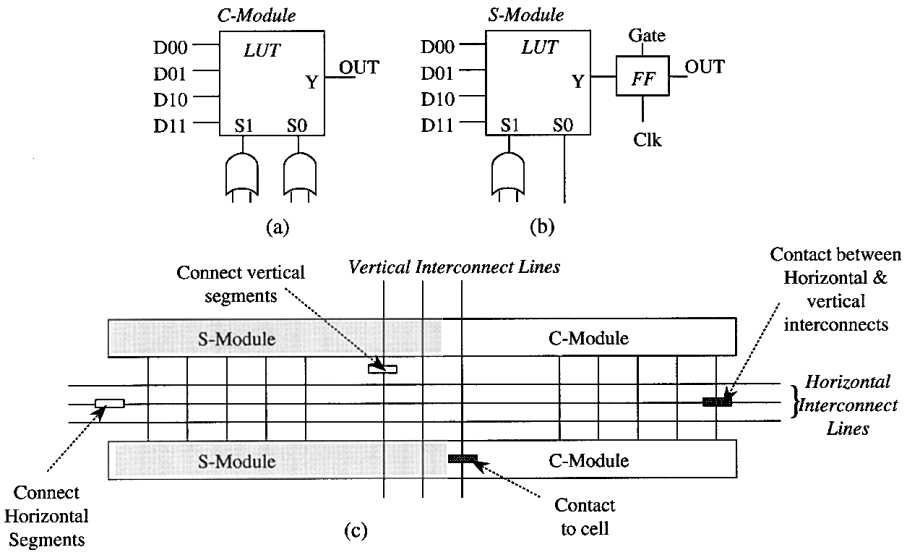

FIGURE 25.33 Example of FPGA elements (Actel FPGA family [Actel, 1995]). Combinational cells (a) and sequential cells (b). (c) Programmable wiring organization.

Different FPGA manufacturers have developed different basic cells, seeking to provide the most useful functionality in the cells for generation of overall FPGA functions. Cells range from fine-grained cells consisting of basic gates through medium-grained cells providing more complex programmable functions to large-grained cells. Different FPGA manufacturers also provide different approaches to the programming step. FPGA programming technologies include one-time programming or multiple-time programming capabilities with the programming either nonvolatile (i.e., programming is retained when power is turned off) or volatile (i.e., programming is lost when power is turned off). Physical programming includes antifuse (fuse) approaches in which the programming nodes are normally off (on) and are "blown" into a permanently on (off) state, providing one-time, nonvolatile programming. Electrical switches also can be used for programming, with an electrical control signal setting the state of the switch. The state control signal can be provided by an EPROM (one-time, nonvolatile), an EEPROM (multiple-time, nonvolatile), or an SRAM (multiple-time, volatile), with different approaches having different advantages and disadvantages.

Interconnection Performance Modeling

Accurate estimation of signal timing is increasingly important in contemporary VLSI ASICs and will become even more important as feature sizes decrease further. Higher clock rates impose tighter timing margins, requiring more accurate modeling of the signal delays. In addition, more-sophisticated models are required for smaller feature size VLSI.

Perhaps of greatest impact is the rapid increase in the importance of interconnect delay relative to gate delay. In the earlier 1 μm VLSI technologies, typical gate delays were about six times the average interconnection delays. For the 0.5 μm technologies, gate delays had decreased while interconnect delays had increased, becoming approximately equal. At 0.3 μm, the decreasing gate delay and increasing interconnect delays have led to average interconnect delays about six times greater than typical gate delays. Accurate estimation of signal delays early in the design is therefore increasingly difficult, since the designer does not have a detailed knowledge of interconnection lengths and nearby lines, which can cause coupling noise, until much of the design has been completed. As the design proceeds and the interconnection lengths become better specified, parameters related to the signal performance can be fed back (back-annotated) to the earlier design steps, allowing the design to be adapted to reflect necessary changes to achieve desired performance.

In earlier VLSI technologies, a linear delay model was adequate, representing the overall delay τ from the input to one cell (cell A in Fig. 25.34) to the input to the connected cell (cell B in Fig. 25.34) by an analytic

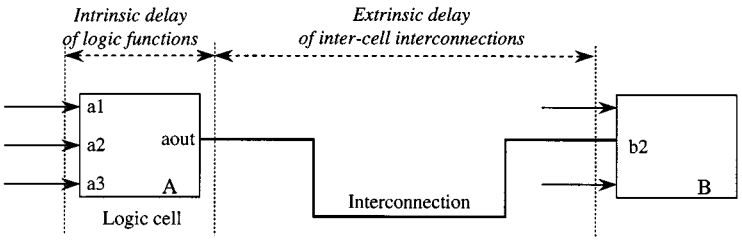

FIGURE 25.34 Logic cell delay (intrinsic delay) and intercell interconnection delay (extrinsic delay).

form such as $\tau = \tau(0) + k1 \cdot C(\text{out}) + k2 \cdot t(s)$, where $\tau(0)$ is the intrinsic (internal) delay of the cell with no loading, $C(\text{out})$ is the output capacitance seen by the output driver of the cell, $t(s)$ is the **rise/fall time** of the signal, and the parameters $k1$ and $k2$ are constants. In the case of deep submicron CMOS technologies, the overall delay must be divided into the **intrinsic delay** of the gate and the **extrinsic delay** of the interconnect, each having substantially more-complex models than the linear model.

Factors impacting the intrinsic delay of gates include the following, with the input and output signals referring to logic cell A in Fig. 25.34.

1. A 0-to-1 change in an input to cell A may cause a different delay to the output of cell A than a 1-to-0 change in that input.
2. Starting from the time when the input starts to change, slower transition times lead to a longer delays before the threshold voltage is reached, leading to longer delays to the output.
3. Once the input passes the threshold voltage, a slower changing input may lead to a longer delay to the output transition.
4. The delay from a change in a given input to a change in the output may depend on the state of other inputs to the circuit.

These are merely representative examples of the more-complex behavior seen in the gates as the feature sizes decrease. Together, those complexities lead to a nonlinear delay model, which is typically implemented as a LUT, rather than with an analytic expression.

The models used for interconnections [Tewksbury, 1994] have also changed, reflecting the changing interconnection parameters and increasing clock rates. The three primary models are as follows.

Lumped RC Model: If the rise/fall times of the signal are substantially greater than the round-trip propagation delay of the signal, then the voltage and current are approximately constant across the length of the interconnection. The interconnection is modeled as a single lumped resistance and a single lumped capacitance. The signal does not incur a propagation delay, and at all points along the line the signal has the same rise/fall times.

Distributed RC Model: If the line is sufficiently long, the signal sees a decreasing resistance and capacitance toward the destination as the signal traverses the line. To represent this changing *RC,* the distributed RC model divides the overall line into shorter segments, each of which can be represented by the lumped RC model above. The transfer function of the overall line is then the product of the transfer functions of the sections. The propagation delay is negligible, though the rise/fall times increase as the signal propagates toward the far-end gates (significant if the line is tapped along its length to drive multiple gates).

Distributed RLC Model: As the rise/fall times become shorter, the relative contributions of capacitance and inductance change. In particular, the impedance of the capacitance is inversely proportional to frequency while that of the inductance is proportional to frequency. At sufficiently high data rates, the inductance effects become significant. In this case, the signal is delayed as it propagates toward the far-end gates, with the rise/fall times increasing along the line. Different terminal points of a net will see the signal at different times with different rise/fall times.

Given the wide range of lengths of signal interconnections, all three models above are relevant; the lumped RC model suitable for short interconnections, the distributed RC model for low-to-moderate speed signals on longer-length interconnections, and the distributed RLC model for high-speed signals on longer interconnections.

Accurate modeling of signals propagating on an interconnection requires detailed knowledge of the capacitance C^* and inductance L^* per unit length along the length of the line. As additional metal layers have been provided, capacitance to neighboring lines (on different or the same metal layer) has become increasingly important, even exceeding the capacitance to ground in some cases. Extraction of accurate interconnect delay parameters may require the use of three-dimensional field solvers, with two-dimensional analysis used for less-accurate modeling of signal behavior.

In addition to the effects noted above, crosstalk (increasingly problematic for buses whose parallel lines run long distances) and reflections (of increasing importance as the signal frequencies increase) degrade signals. This broad range of effects impacting signal delay, distortion, and noise have made *signal integrity* an increasingly important issue in VLSI design. Signal integrity effects also appear on the "dc" power and ground lines, due to large transient currents caused by switching gates and switching drivers of output lines.

Clock Distribution

Signals are increasingly distorted not only by long line lengths, but also by the higher clock frequency in present-day VLSI circuits, with the combination of long lines and high clock rates of particular concern. Present-day VLSI circuits include a vast number of flip-flops (often as registers) distributed across the area of the VLSI circuit. *Synchronous ASICs* use a common clock, distributed to each of these flip-flops. With the clock signal being the longest interconnection on the VLSI circuit and the highest-frequency signal, design of the clock distribution network is critical for highest performance.

Complex synchronous ASICs are designed assuming that all flip-flops are clocked simultaneously. *Clock skew* is the maximum difference between the times of clock transitions at any two flip-flops of the overall VLSI circuit. The clock network must deliver clock signals to each of the flip-flops within the margins set by the allowed clock skew, margins which are substantially less than the clock period. For example, part of the 2-ns clock period of a high-speed VLSI circuit operating with a 500-MHz clock is consumed by the rise/fall times of the signals appearing at the input to the flip-flop and by the specified setup and hold times of the flip-flop. The result is that the clock must be applied to the flip-flop within a time interval small as compared with the clock period.

The distance over which the clock signal can travel before incurring a delay greater than the clock skew defines *isochronous* regions (illustrated in Fig. 25.35(a) as shaded regions) within the IC. If the external clock can be provided to such regions with zero clock skew, then clock routing within the isochronous region is not critical. Figure 25.35(a) illustrates the H-tree approach, whose clock paths have equal lengths to terminal points, ideally delivering clock pulses to each of the terminal points (leaf nodes) of the tree simultaneously (zero skew). In a real circuit, precisely zero clock skew is not achieved since different network segments encounter different environments of data lines coupled electrically to the clock line segment.

In Fig. 25.35(a), a single buffer drives the entire H-tree network, requiring a large area buffer and wide clock lines toward the connection of the clock line to the external clock signal. Such a large buffer can account for up to 30% or more of the total VLSI circuit power dissipation. Figure 25.35(b) illustrates a distributed buffer approach, with a given buffer only having to drive those clock line segments to the next level of buffers. In this case, the buffers can be smaller and the clock lines can be narrower. The 300-MHz DEC Alpha microprocessor, for example, uses an H-tree clock distribution network with multiple stages of buffering extending to the final legs of the H-tree network. Another approach to relax the clock distribution problem uses multiple input/output (I/O) pins for the clock. In this case, a number of smaller H-trees can be driven separately, one starting at each clock I/O pin.

The constraint on clock timing is a bound on clock skew, not a requirement for zero clock skew. In Fig. 25.35(c), the clock network uses multiple buffers but allows different path lengths consistent with clock skew margins. For tight margins, an H-tree can be used to deliver clock pulses to local regions in which distribution proceeds using a different buffered network approach such as that in Fig. 25.35(c).

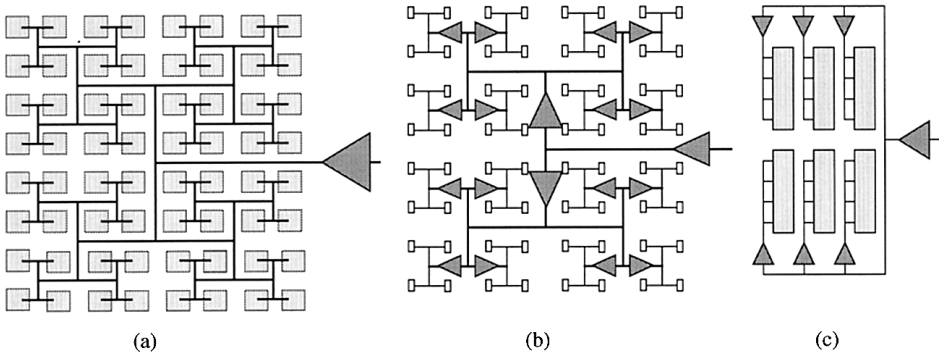

FIGURE 25.35 H-tree clock distribution. (a) Example of single driver and isochronous (shaded) regions. (b) Example of distributed drivers. (c) Example of clock distribution with unequal line lengths but within skew tolerances.

Other approaches for clock distribution are gaining in importance. For example, a lower frequency clock can be distributed across the VLSI circuit, with phase-locked loops (PLLs) used to multiply the clock rate at various sites on the circuit. In addition to multiplying the low clock rate, the PLL can also adjust the phase of the high rate clock, correcting clock skew which may have been introduced in the routing of the low rate clock to that PLL. Another approach is to generate a local clock at a local register when the register input data changes. This *self-timed circuit* approach leads to *asynchronous circuits* but can be quite effective for register-transfer logic (generating a local clock for a large register).

Power Distribution

Present-day VLSI ASICs consume considerable power, unless specifically designed for battery-operated, low-power portable electronics [Chandrakasan and Broderson, 1995]. A 40-W IC operating at 3.3 V requires a current of 12 A, with currents increasing in future generations of high-power VLSI (due not only to the higher power dissipation but also to lower V_{dd}).

Voltage and ground lines must be properly sized to prevent the peak current density exceeding the level at which the power lines will be physically "blown out," leading to rapid and catastrophic failure of the circuit. An equally serious problem is gradual deterioration of a voltage or ground line, eventually leading to failure as a result of *electromigration*. Electromigration failure affects both signal and power lines, but is particularly important in power lines because of the constant direction of the current. As current flows through an aluminum interconnection, the average force exerted on the metal atoms by the electrons leads to a slow migration of those atoms in the direction of electron flow, causing the line to migrate in the direction of the electrons (opposite to the current flow). In regions of the metal line where discontinuities occur (e.g., at the naturally occurring grain boundaries), a void can develop, creating an open in the line. Fortunately, there is a current density threshold level (about 1 mA/μm) below which electromigration is insignificant. Notably, copper, in addition to having a lower resistivity than aluminum, has greater resistance to electromigration. Accurate estimates of power dissipation due to logic switching within logic blocks of the ASIC are also necessary to assess thermal heating within the IC.

Another issue in power distribution concerns *ground bounce* (or *simultaneous switching noise*), which is increasingly problematic as the number of ASIC I/O data pins increases. Consider M output lines switching simultaneously to the 1 state, each of those lines outputting a current transient I(out) within a time t(out). (If M output lines switch simultaneously to the 0 state, then a corresponding input current transient is produced.) The net output current $M \cdot I$(out) is fed through the V_{dd} pin (returned to ground in the case of outputs switching to 0). With an inductance L associated with the V_{dd} pin, a transient voltage $DV \approx L \cdot M \cdot I(\text{out})/t(\text{out})$ is imposed on V_{dd}. A similar effect occurs on the ground connection for outputs switching to 0. With a total output current of 200 mA/ns and a ground pin inductance of 5 nH, the voltage transient is about 1 V. The voltage transient propagates through the IC, potentially causing logic blocks to fail to produce the correct outputs. The transient voltage can be reduced by reducing the power line inductance L, for example by replacing

the single V_{dd} and *Gnd* pins by multiple V_{dd} and *Gnd* pins, with *K* voltage pins reducing the inductance by a factor of *K*.

With power distribution and power line noise problems growing in importance, EDA/CAD tools are rapidly evolving to provide the designer with early estimates and final accurate assessments of various measures of current and of power dissipation.

Analog and Mixed-Signal ASICs

One of the exciting ASIC areas undergoing rapid development is the addition of analog integrated circuits [Geiger et al., 1990; Ismail and Fiez, 1994; Laker and Sansen, 1994] to the standard digital VLSI ASIC, corresponding to *mixed-signal* VLSI. **Mixed-signal ICs** allow the IC to interact directly with the real physical (and analog) world. Library cells representing various analog circuit functions supplement the usual digital circuit cells of the library, allowing the ASIC designer to add needed analog circuits within the same general framework as the addition of digital circuit cells. Automotive electronics is a representative example, with many sensors providing analog information which is converted into a digital format and analyzed using microcomputers or other digital circuits. Mixed-signal library cells include A/D and D/A converters, comparators, analog switches, sample-and-hold circuits, etc., while analog library cells include op amps, precision voltage sources, and phase-locked loops.

As such mixed-signal VLSI ASICs evolve, EDA/CAD tools will also evolve to address the performance and design issues related to analog circuits and their behavior in a digital circuit environment. In addition, analog high-level description languages (AHDLs) are being developed to support high-level specifications of both analog and mixed-signal circuits.

Summary

For about three decades, microelectronics technologies have been evolving, starting with primitive digital logic functions and evolving to the extraordinary capabilities available in present-day VLSI ASICs. This evolution promises to continue for at least another decade, leading to VLSI ICs containing complex systems and vast memory on a single IC. ASIC technologies (including the EDA/CAD tools which guide design to a final IC) deliver this complex technology to the systems designers, including those not associated with a company having a microfabrication facility. This delivery of a highly complex technology to the average electronic systems designer is the result of a steady migration of specialized skill to very powerful EDA/CAD tools which control the complexity of the design process and the result of a need to provide a wide variety of electronics designers with access to technologies which earlier had been available only within large, vertically integrated companies.

Defining Terms

ASIC: Application-specific integrated circuit — an integrated circuit designed for a specific application.

CAD: Computer-aided design — software programs which assist the design of electronic, mechanical, and other components and systems.

CMOS: Complementary metal-oxide semiconductor transistor circuit composed of PMOS and NMOS transistors.

EDA: Electronics design automation — software programs which automate various steps in the design of electronics components and systems.

Extrinsic Delay: Also called *point-to-point delay,* the delay from the transition of output of a logic cell to the transition at the input of another logic cell.

HDL: High-level description language — a software "language" used to describe the function performed by a circuit (or collection of circuits).

IC: Integrated circuit — a normally silicon substrate in which electronic devices and interconnections have been fabricated.

Intrinsic delay: Also called *pin-to-pin delay,* the delay between the transition of an input to a logic cell to the transition at the output of that logic cell.

Mixed-signal ICs: Integrated circuits including circuitry performing digital logic functions as well as circuitry performing analog circuit functions.

NMOS transistor: A metal-oxide semiconductor transistor which is in the on state when the voltage input is high and in the off state when the voltage input is low.

PMOS Transistor: A metal-oxide semiconductor transistor which is in the off state when the voltage input is high and in the on state when the voltage input is low.

Rise/(fall) time: The time required for a signal (normally voltage) to change from a low (high) value to a high (low) value.

V_{dd}**:** The supply voltage used to drive logic within an IC.

VLSI: Very large scale integration — microelectronic integrated circuits containing large (presently millions) of transistors and interconnections to realize a complex electronic function.

Wiring channel: A region extending between the power and ground lines on an IC and dedicated for placement of interconnections among logic cells.

Related Topic

79.1 IC Logic Family Operation and Characteristics

References

Actel, 1995. *Actel FPGA Data Book and Design Guide,* Sunnyvale, Calif.: Actel Corp., 1995.

J. R. Armstrong, *Chip-Level Modeling with VHDL,* Englewood Cliffs, N.J.: Prentice-Hall, 1989.

P. Banerjee, 1994. *Parallel Algorithms for VLSI Computer-Aided Design,* Englewood Cliffs, N.J.: Prentice-Hall.

R. Camposano, and W. Wolf (Eds.), *High Level VLSI Synthesis,* Norwell, Mass.: Kluwer Academic Publishers, 1991.

A. Chandrakasan and R. Broderson, *Low Power Digital CMOS Design,* Norwell, Mass.: Kluwer Academic Publishers, 1995.

G. De Micheli, *Synthesis of Digital Circuits,* New York: McGraw-Hill, 1994a.

G. De Micheli, *Synthesis and Optimization of Digital Circuits,* New York: McGraw-Hill, 1994b.

T. E. Dillinger, *VLSI Engineering,* Englewood Cliffs, N.J.: Prentice-Hall, 1988.

D. Gajski, N. Dutt, A. Wu, and S. Lin, *High-Level Synthesis: Introduction to Chip and Systems Design,* Norwell, Mass.: Kluwer Academic Publishers, 1992.

R. L. Geiger, P. E. Allen, and N. R. Strader, *VLSI Design Techniques for Analog and Digital Circuits,* New York: McGraw-Hill, 1990.

D. V. Heinbuch, *CMOS Cell Library,* New York: Addison-Wesley, 1987.

F. J. Hill, and G. R. Peterson, *Computer Aided Logical Design with Emphasis on VLSI,* New York: John Wiley & Sons, 1993.

D. Hill, D. Shugard, J. Fishburn, and K. Keutzer, *Algorithms and Techniques for VLSI Layout Synthesis,* Norwell, Mass.: Kluwer Academic Publishers, 1989.

E. E. Hollis, *Design of VLSI Gate Array ICs,* Englewood Cliffs, N.J.: Prentice-Hall., 1987.

M. Ismail, and T. Fiez, *Analog VLSI: Signal and Information Processing,* New York: McGraw-Hill, 1994.

N. Jha, and S. Kundu, *Testing and Reliable Design of CMOS Circuits,* Norwell, Mass.: Kluwer Academic Publishers, 1990.

S.-M. Kang, and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design,* New York: McGraw-Hill, 1996.

K. R. Laker, and W. M. C. Sansen, *Design of Analog Integrated Circuits and Systems,* New York: McGraw-Hill, 1994.

K. Lee, M. Shur, T. A. Fjeldly, and Y. Ytterdal, *Semiconductor Device Modeling for VLSI,* Englewood Cliffs, N.J.: Prentice-Hall, 1993.

J. Lipman, "EDA tools put it together," *Electronics Design News* (EDN), Oct 26, 1995, pp. 81–92.

R. Lipsett, C. Schaefer, and C. Ussery, *VHDL: Hardware Description and Design,* Norwell, Mass.: Kluwer Academic Publishers, 1990.

S. Mazor, and P. Langstraat, *A Guide to VHDL,* Norwell, Mass.: Kluwer Academic Publishers, 1992.

The National Technology Roadmap for Semiconductors, San Jose, Calif.: Semiconductor Industry Association, 1994.
K. P. Parker, *The Boundary-Scan Handbook,* Norwell, Mass.: Kluwer Academic Publishers, 1992.
B. Preas, and M. Lorenzetti, *Physical Design Automation of VLSI Systems,* Menlo Park, Calif.: Benjamin-Cummings, 1988.
J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective,* Englewood Cliffs, N.J.: Prentice-Hall, 1996.
S. Rubin, *Computer Aids for VLSI Design,* New York: Addison-Wesley, 1987.
SCMOS Standard Cell Library, Center for Integrated Systems, Mississippi State University, 1989.
K. Shahookar, and P. Mazumder, "VLSI placement techniques," *ACM Computing Surveys,* vol. 23(2), pp. 143–220, 1991.
N. A. Sherwani, *Algorithms for VLSI Design Automation,* Norwell, Mass.: Kluwer Academic Publishers, 1993.
N. A. Sherwani, S. Bhingarde, and A. Panyam, *Routing in the Third Dimension: From VLSI Chips to MCMs,* Piscataway, N.J.: IEEE Press., 1995.
S. Tewksbury (Ed.), *Microelectronic Systems Interconnections: Performance and Modeling,* Piscataway, N.J.: IEEE Press, 1994.
D. E. Thomas, and P. Moorby, *The Verilog Hardware Description Language,* Norwell, Mass.: Kluwer Academic Publishers, 1991.
N. H. E. Weste, and K. Eshraghian, *Principles of CMOS VLSI Design,* New York: Addison-Wesley, 1993.
J. White, and A. Sangiovanni-Vincentelli, *Relaxation Methods for Simulation of VLSI Circuits,* Norwell, Mass.: Kluwer Academic Publishers, 1987.
W. Wolf, *Modern VLSI Design: A Systems Approach,* Englewood Cliffs, N.J.: Prentice-Hall, 1994.

Further Information

The *Institute for Electrical and Electronics Engineers, Inc.* (IEEE) publishes several professional journals which describe the broad range of issues related to contemporary VLSI circuits, including the *IEEE Journal of Solid-State Circuits* and the *IEEE Transactions on Very Large Scale Integration Systems.* Other applications-related journals from the IEEE cover VLSI-related topics. Representative examples include the *IEEE Transactions on Signal Processing,* the *IEEE Transactions on Computers,* the *IEEE Transactions on Image Processing,* and the *IEEE Transactions on Communications.* Several conferences also highlight VLSI circuits, including the *IEEE Solid-State Circuits Conference,* the *IEEE Custom Integrated Circuits Conference,* and several others.

Commercial software tools and experiences related to ASIC designs are changing rapidly, but are well covered in several trade journals including *Integrated System Design* (the Verecom Group, Los Altos, Calif.), *Computer Design* (PennWell Publishing Co., Nashua, N.H.), *Electronics Design News* (Cahners Publishing Co., Highlands Ranch, Colo.), and *Electronic Design* (Penton Publishing Inc., Cleveland, Ohio).

There are also many superb books covering the many topics related to IC design and VLSI design. The references for this chapter consist mainly of books, all of which are well-established treatments of various aspects of VLSI circuits and VLSI design automation.