CHAPTER 12

# VLSI Models of Computation

The electronics revolution initiated by the invention of the transistor by Schockley, Brattain, and Bardeen in 1947 accelerated with the invention of the integrated circuit in 1958 and 1959 by Jack Kilby and Robert Noyce. An **integrated circuit** contains wires, transistors, resistors, and other components all integrated on the surface of a **chip**, a piece of semiconductor material about the size of a thumbnail. And the revolution continues. The number of components that can be placed on a semiconductor chip has doubled almost every 18 months for about 40 years. Today more than 10 million of them can fit on a single chip. Integrated circuits with very large numbers of components exhibit what is known as very large-scale integration (VLSI). This chapter explores the new models that arise as a result of VLSI.

As the size of the electronic components decreased in size, the area occupied by wires consumed an increasing fraction of chip area. In fact, today some applications devote more than half of their area to wires. In this chapter we examine VLSI models of computation that take this fact into account. Using simulation techniques analogous to those employed in Chapter 3, we show that the performance of algorithms on VLSI chips can be characterized by the product $AT^2$, where $A$ is the chip area and $T$ is the number of steps used by a chip to compute a function. We relate $AT^2$ to the planar circuit size $C_{\mathrm{p},\Omega}(f)$ of a function $f$, a measure that plays the role for VLSI chips that circuit size plays for FSMs. The $AT^2$ measure is the direct analog of the measure $C_\Omega(\delta, \lambda)T$ for the finite-state machine that was introduced in Chapter 3, where $C_\Omega(\delta, \lambda)$ is the size of a circuit to simulate the next-state and output functions of the FSM. We also relate the measure $A^2T$ to $C_{\mathrm{p},\Omega}(f)$.

# 12.1 The VSLI Challenge

The design of VLSI chips represents an enormous intellectual challenge akin to that of constructing very large programs. They each involve the assembly of millions of elements, instructions in the case of software, and electronic components in the case of chips. The design and implementation of VLSI chips is also challenging because it involves many steps and many technologies. In this section we provide a brief introduction to this process as preparation for the introduction of the VLSI models and algorithms that are the principal topics of this chapter.

## 12.1.1 Chip Fabrication

A VLSI chip consists of a number of conducting, insulating, and doped layers that are placed on a semiconductor substrate. (A **doped layer** is created on the surface of the substrate by infusing small concentrations of impurities into the semiconductor. This is called **doping**.) The layers are created using **masks**, templates with open regions through which ionizing radiation is projected onto the surface of the semiconductor. The radiation changes the chemical properties of a previously deposited photosensitive material so that the exposed regions can be washed away with a solvent. The material that is now exposed can be doped or removed. Doping is used to create transistors and wires. A removal step is used when a metallic layer has been previously deposited from which sections are to be removed, leaving wires. A chip may have several layers of wires separated by layers of insulating material in addition to the doped layers that form transistors and wires. The layout of a NAND gate is shown schematically in Fig. 12.1, in which the shadings of rectangles and annotations identify to a chip designer the types of materials used to realize the gate.

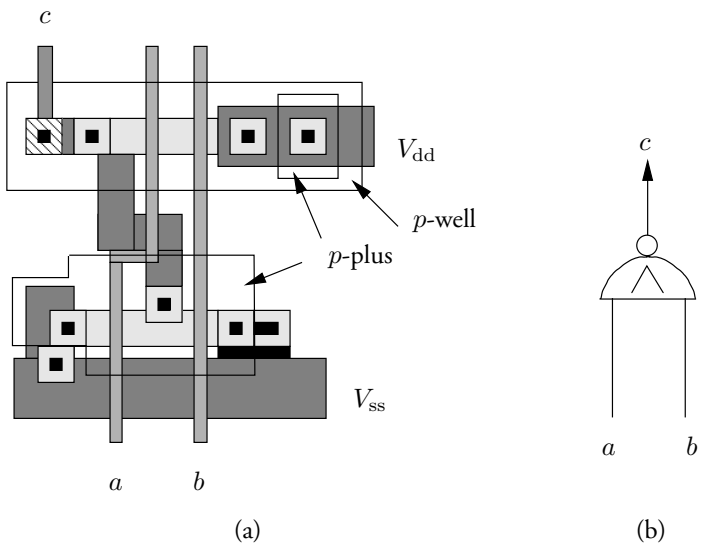



**Figure 12.1** The schematic layout of a NAND gate and its logical symbol.

Geometric design rules specify the amounts of overlap of and separation between metal and dopant rectangles that are needed to guarantee the desired electrical and electronic properties of a VLSI circuit. If wires are too thin, electrons, which move through them at very high speeds, can cause excess heating as well as dislodge atoms and create an open circuit (this is called **metal migration**), especially at points at which a wire bends to descend into a well created during chip fabrication. Similarly, if wires are too close, an error in registration of masks may cause short circuits between wires. Also, since transistors are constructed through the doping and overlaying of insulating and conducting materials, if the regions defining a transistor are too small, it will not behave as expected.

The geometric design rules for a particular chip technology can be quite complex. For the purpose of analysis they are simplified into a few rules concerning the width and separation of rectangles, the amount of area required for contacts between wires on layers separated by insulation, and the size of the various rectangular regions that form gates and transistors. As suggested by this discussion, a VLSI chip is **quasiplanar**; that is, its components lie on a few layers, which are separated by insulation except where contacts are made between layers.

### 12.1.2 Design and Layout

Many tools and techniques have been developed to address the complexity of chip layout. Typically these tools and techniques use abstraction; that is, they decompose a problem into successively lower level units of increasing complexity. At each level the number of units involved in a design is kept small so that the design is comprehensible.

The design of a VLSI chip begins with the specification of its functionality at the **functional** or **algorithmic level**. Either a function or an algorithm is given as the starting point. An algorithm is then produced and translated into a specification at the **architectural level**. At this level a chip is specified in terms of large units such as a CPU, random-access memory, bus, floating-point unit, and I/O devices. (The material of Chapters 3 and 4 is relevant at this level.) After an architectural specification is produced, design commences at the **logical level**. Here particular methods for realizing architectural units are chosen. For example, an adder could be realized either as a ripple or a carry-lookahead adder depending on the stated speed and cost objectives. (The material of Chapter 2 applies at this level.)

At the **gate level**, the next level in the design process, a technology, such as NMOS and CMOS, is chosen in which to realize the transistors and wires. This involves specifications of widths for wires, the number of layers of metal, and other things. If new transistor layouts are used, their physics is often simulated to determine their electrical properties.

At the next level, the **layout level**, a gate-level design is translated into physical positions for modules, gates, and wires. Often at this level a rough layout is produced manually, after which automatic routing and compaction algorithms are invoked to route wires between modules and squeeze out the unnecessary area. Space must be reserved on each layout for **I/O pads**, rectangular regions large enough to connect external wires. They serve as **ports** through which data is read and written. Because these wires and pads are very large by comparison with the wires on the chip, there is a practical limit on the number of I/O ports on a chip. A port can be both an input and an output port.

Once a layout is complete it is usually simulated logically, that is, at the level of Boolean gates. Parts of it may also be simulated electrically, a much more time-consuming process given the much lower level of detail that it entails.

After a chip has been fabricated it is then tested. Because the testing process for a complete chip cannot be exhaustive, due to the number of configurations that are possible, subunits are often isolated and tested. Testing circuitry is often built into a chip to simplify the testing process.

Because the design, layout, simulation, and testing of VLSI chips is complex and error prone, **computer-aided design (CAD)** tools have been developed. CAD is very large subject beyond the scope of this book. Instead, we limit our attention in this chapter to the performance of VLSI chips.

## 12.2 VLSI Physical Models

Of all the parameters that affect the performance of a VLSI chip, its **area** is one of the most important. Equally important are the **width of** and **separation between wires**, both of which are directly related to area. Area is important for two reasons. First, a larger area means a chip can have more computing elements and do more work. Also, more area means a chip can have more I/O ports to facilitate data movement on and off the chip.

Unfortunately, the area of a chip has a practical limit due to imperfections that occur in the chip manufacturing process. A single very small piece of dust or a dislocation in the crystalline semiconductor substrate, each of which can be large by comparison with the dimensions of components, can destroy a chip. As a consequence, only a small fraction (the **yield**) of the chips resulting from a fabrication process work. The rest must be discarded.

The yield of a chip is very sensitive to its size. If the number of faults per unit area is $F$, with very high probability a fault occurs if the area $A$ of a chip exceeds $1/F$. As $F$ is reduced by improvements in the manufacturing process, the area of any one chip can increase. However, if $F$ is fixed, so is the value of $A$ at which an economical yield is possible. ($F$ has not decreased much over time.) To make chip manufacture economical, dozens of chips are manufactured together on a circular wafer of 4 to 8 inches in diameter. The wafer is then sliced into individual chips. If the die size is chosen correctly, a fixed fraction of the chips on a wafer will work. The importance of testing becomes evident in light of these observations.

Because the area of a chip has a practical upper limit, the width and separation of wires determine the number of components that can be placed on a chip. As mentioned above, the technology for chip manufacture places a lower limit on these parameters as well as the area of chip components.

To simplify our modeling and analysis, we assume that the minimal width and separation of wires is $\lambda$ (the **minimum feature size**) and that each gate, memory cell, port, and pair of crossing wires has area $\lambda^2$. There is no great loss in assuming a single number for wire width and separation and one number for the minimal area of components because in practice the width and separation of wires of different kinds and the area of components are all small multiples of common values. The only component for which these assumptions are weak is the pads for I/O ports, which are generally very much larger than $\lambda^2$. It is important to be cognizant of this fact in drawing conclusions.

Since chips are quasiplanar, we assume that **each chip has at most** $\nu \geq 1$ **layers** on which wires can reside but that there is only one layer of gates. Also, since wires are rectangular, it is impractical for them to meet at angles that are not close to 0 or 45 degrees. In fact, wires are usually rectilinear, that is, run horizontally and vertically. Thus, we assume that **wires are rectilinear**.

To complete the physical modeling of chips we recognize three types of **transmission model**, the synchronous, transmission-line, and diffusion models. The **synchronous model** assumes that one unit of time is needed to transmit a bit across a wire, independent of its length. This is a good model when the switching time of gates is large by comparison with the time to transmit data through a wire or when wires are short, a situation that prevails for most designs. When it does not prevail, the unit of transmission time can be increased so that it does apply. The **transmission-line model** assumes that the time to transmit a bit across a wire is proportional to its length (see Problems 12.1 and 12.2), whereas the **diffusion model** assumes it is quadratic in its length. The models apply to VLSI chip technologies at different wire lengths. The synchronous, transmission-line, and diffusion models apply to wires that are short, medium-length, and long, respectively.

Although we do not examine **energy consumption** in this chapter, the type of gate used can have a large impact on the amount of energy consumed during a computation. NMOS transistors consume energy all the time, whereas CMOS transistors consume energy only when they change their state.

When the area of I/O pads and gates are comparable, the placement of the pads on a VLSI chip can have a big impact on the area occupied by a chip. For example, if the chip realizes a tree and its $n$ leaves (and their pads) are placed on the boundary of a convex region, as noted in Problem 12.3, the chip must have area proportional to $n \log n$. However, as shown in Section 12.5.1, when its leaves can be placed anywhere, there is a layout for a tree (known as the H-tree) that has area proportional to $n$. If the I/O pads are much larger than the gates, the impact of their placement is diminished.

## 12.3 VLSI Computational Models

We assume that a VLSI chip implements a finite-state machine instantiated as a clocked sequential machine. (A chip could also model an analog computer rather than a digital one, a topic not discussed in this book.) Although every FSM is eventually realized from two-input gates, binary memory cells, and wires carrying binary values (see Section 3.1), chips are generally designed around an aggregate model for data. That is, if operations are done on integers, the wires associated with an integer travel together on the chip surface. Although the time required for an operation on data depends on the size of alphabet from which the data is drawn and on the complexity of the operation itself, we simplify the analysis by assuming that one unit of time is taken. A more sophisticated analysis takes these factors into account.

To be concrete we let the states of an FSM be represented as tuples over a set $X$ of binary $b$-tuples. We also assume that gates realize functions $\{h : X^2 \mapsto X\}$ and that memory cells hold one value of $X$. We recognize a **logic circuit over the set** $X$ as the graph of a straight-line in which the operations are drawn from a basis $\{h : X^2 \mapsto X\}$. This model is used to study problems defined over non-binary alphabets, such as matrix multiplication and the discrete Fourier transform over rings.

We continue to use the notation $\lambda$ for the minimum feature size of a VLSI chip even though we now allow data to be treated as values in the set $X$. When the set $X$ is big, it will be important to make use of its size in accounting for the area occupied by wires and gates, an issue that we ignore in this chapter.

Computation **time** in the synchronous model is the number of steps executed by a chip. This is the same measure of time used for finite-state machines. Computation **time** in the

other models is the elapsed time in seconds, which is approximated by the number of steps multiplied by the length of the longest step. This time is generally a function of the area of the chip and the problem for which the chip is designed.

Another measure of time, but one that is given only a cursory examination, is the **period** $P$ of a VLSI chip. This is the time between successive inputs to a pipelined chip, one designed to receive a new set of inputs while the previous inputs are propagating through it. Pipelining is illustrated in Section 12.5.1 on H-trees and Section 11.6 on block I/O.

In this chapter **we assume that VLSI chips compute a single function** $f : X^n \mapsto X^m$, a perfectly general assumption that allows any FSM computation to be performed. While this allows the VLSI chip to be a CPU or a RAM, to convey ideas we limit our attention to functions that are simply defined, such as matrix multiplication and the discrete Fourier transform.

The variables of the function computed by a VLSI chip are supplied via its I/O ports. A single port can receive the values of multiple variables but at different time instances. Also, the value of a variable can be supplied at multiple ports, either in the same time step or in multiple time steps. However, the outputs of a function computed by a chip are supplied once to an output port. As noted above, a port can be either an input or output port or serve both purposes, but not in the same time step.

As with the FSM, we cannot allow either the time or the I/O port at which data is received as input or is supplied as output to be data-dependent. To do otherwise is to assume that an external agent not included in the model is performing computations on behalf of the user. We can expect misleading results if this is allowed. Thus, we assume that each I/O operation is **where-** and **when-oblivious**; that is, where an input or output occurs is data-independent, as are the times at which the I/O operations occur.

For many VLSI computations it is important that the input data be read once by the chip even if it may be convenient to read it multiple times. (These are called **semellective** or **read-once** computations.) For example, if a chip is connected to a common bus it may be desirable to supply the data on which the chip operates once rather than add hardware to the chip to allow it to request external data. However, in other situations it may be desirable to provide data to a chip multiple times. Such computations are called **multilective**. Multilective computations must be where- and when-oblivious.

If a multilective VLSI algorithm reads its $n$ input variables $\beta\mu n$ times but only $\mu n$ times when multiple inputs of a variable (at multiple time steps) at one I/O port are treated as a single input, then the algorithm is **$(\beta, \mu)$-multilective**.

## 12.4 VLSI Performance Criteria

As stated in Theorem 7.4.1, the product $pT_p$ of the time, $T_p$, and the number of processors, $p$, in a parallel network of RAM processors to solve a problem cannot be less than the serial time, $T_s$, on a serial RAM with the same total storage capacity for that problem. Applying this result to the VLSI model, since the number of processors of any given size that can be placed on a chip of area $A$ is proportional to $A$, it follows that the product $AT$ of area with the time $T$ for a chip to complete a task cannot be less than the serial time to compute the same function using a single processor; that is, $AT = \Omega(T_s)$.

In the next section we show that the matrix-vector multiplication and prefix functions can be realized optimally with respect to the $AT$ measure. This holds because these problems have

low complexity. For problems of higher complexity, such as $n \times n$ matrix-matrix multiplication, we cannot achieve $AT$-optimality because stronger lower bounds apply. In particular, both $AT^2$ and $A^2T$ must grow as $n^4$ for this problem, as we show. $AT$, $AT^2$ and $A^2T$ are the only measures of VLSI performance considered in this chapter.

## 12.5 Chip Layout

In this section we describe and discuss layouts for a number of important graphs and problems. These include balanced binary trees, multi-dimensional meshes, and the cube-connected cycle.

### 12.5.1 The H-Tree Layout

H-trees are embeddings of binary trees that use area efficiently. Let $H_k$ be an H-tree with $4^k$ leaves. Figure 12.2 shows the H-tree $H_2$ with 16 darkly shaded squares that can be viewed either as subtrees or leaves. The lightly shaded regions are internal vertices of the binary tree. Leaves often perform special functions that are not performed by internal vertices whereas internal vertices of a tree often perform the same function. Each quadrant of the tree shown in Fig. 12.2 can be viewed as the H-tree $H_1$ on four subtrees or leaves.

The layout of $H_k$ is recursively defined as follows: replace each of the four leaves of $H_{k-1}$ with a copy of $H_1$. Thus, $H_2$ in Fig. 12.2 is obtained by replacing each leaf in $H_1$ with a copy of $H_1$.

We now derive an upper bound on the area of an H-tree under the assumption that each vertex is square, leaf vertices occupy area $b^2$, and the separation between leaf vertices is $c$. If $S(k)$ is the length of a side of $H_k$, then $S(1) = 2b + c$. Also, from the recursive construction of $H_k$ the following recurrence holds:

$$S(k) = 2S(k-1) + c$$

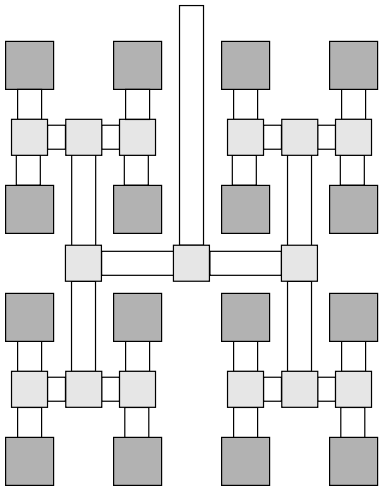


**Figure 12.2** The H-tree $H_2$ containing 16 subtrees (or leaves).

The solution to this recurrence is $S(k) = (b + c)2^k - c$ as the reader can verify. Since $H_k$ has $n = 4^k$ leaves and area $A_n = (S(k))^2$, it follows that an $n$-vertex H-tree has area $A_n \leq n(b + c)^2$.

To appreciate the importance of the H-tree construction, observe that its leaves are interior to the layout. Given the usual drawing of a binary tree one is tempted to place its leaves along the boundary of a chip. If this boundary is convex, the area of a binary tree on $n$ leaves must be at least proportional to $n \log n$. (See Problem 12.3.)

**MATRIX-VECTOR MULTIPLICATION ON AN H-TREE** We now describe an algorithm based on an H-tree that multiplies an $n \times n$ matrix $A$ with an $n$-vector $\boldsymbol{x}$, $n = 2^k$, by forming the $n$ inner products of the $n$ rows of $A$ with $\boldsymbol{x}$. (Matrix-vector multiplication is defined in Section 6.2.2.) This algorithm assumes that one unit of time is taken to store one piece of data and to perform an addition or multiplication on data.

On the first time step of our algorithm the components of the vector $\boldsymbol{x}$ are supplied in parallel to the $n$ leaves of the tree and stored there. On the second time step components of the first row of $A$ are also provided in parallel to the leaves. In the third time step the product of corresponding components of $\boldsymbol{x}$ and the first row of $A$ are multiplied. In $k = \log_2 n$ additional time steps these products are added in the H-tree and the result supplied as output. In the next two steps the second row of $A$ is supplied as input and its components multiplied by those of $\boldsymbol{x}$. After $k$ additional steps these products are summed and the result generated as output. This process is repeated for each of the remaining rows of $A$. This algorithm is semellective.

Since we treat the time to add and multiply as the basis for measuring the time required by this H-tree, each inner product requires $O(\log n)$ time and the $n$ inner products require $O(n \log n)$ time. However, if each addition vertex in this tree can also store its result (thereby causing a slight increase in area), a new row of $A$ can be supplied to the H-tree in each unit of time (we say the **period** of the computation is $P = 1$) because a series of partial results can move through the tree in parallel. This is an example of pipelining. In this case the time to perform the $n$ inner products is $O(n + \log n) = O(n)$. If pipelining is not used, this matrix-vector multiplication algorithm does not make the best use of area and time, as we now show.

Even without pipelining there exists an $AT$ optimal algorithm for matrix-vector multiplication. Let $n$ be such that $n/\log_2 n$ is a power of 4. Decompose each row of $A$ as well as $\boldsymbol{x}$ into $(\log_2 n)$-tuples. This is equivalent to representing the $n \times n$ matrix $A$ by a $n \times (n/\log_2 n)$ matrix $B$ whose entries are $1 \times \log_2 n$ matrices (equivalently, $(\log_2 n)$-vectors) and to representing $\boldsymbol{x}$ by an $(n/\log_2 n)$-vector $\boldsymbol{y}$ whose components are $(\log_2 n)$-vectors.

We implement this computation on an H-tree with $O(n/\log n)$ area. To compute the inner product of $A$'s $j$th row with $\boldsymbol{x}$, sequentially supply to each H-tree leaf the components of one $(\log_2 n)$-vector of $\boldsymbol{y}$ and the corresponding vector in the $j$th row of $B$. Supply the individual components of these $(\log_2 n)$-vectors in alternate cycles. After a leaf vertex receives the corresponding components of $A$ and $\boldsymbol{x}$, it multiplies them and adds the result to its running sum. Upon completion of an inner product of two $(\log_2 n)$-vectors, the leaf vertices make their values available to be added in the H-tree in $O(\log n)$ steps. After $n$ of these operations, all $n$ inner products of $A\boldsymbol{x}$ are computed.

This algorithm uses $T = O(n \log n)$ time but only has area $A = O(n/\log n)$. Thus, its area–time product satisfies $AT = O(n^2)$, which is optimal since each of the $n^2 + n$

components of $A$ and $\boldsymbol{x}$ must be read. This algorithm is multilective because it supplies each component of $\boldsymbol{x}$ $n$ times.

**PREFIX COMPUTATION ON AN H-TREE** The H-tree is also an effective way to do a prefix computation. Prefix computations (let $\odot$ be the associative operator) are naturally executed on trees. A tree-based prefix computation is described in Problem 7.31. One datum enters the root of the tree; the rest travel up from the leaves. When implemented on an H-tree, this algorithm uses area $O(n)$ on $n$ inputs and time $O(\log n)$, giving an $AT$ product of $O(n \log n)$. This algorithm is semellective.

This algorithm can be converted into an $AT$-optimal algorithm using a technique similar to that used above. We subdivide the input $n$-tuple $\boldsymbol{x}$ into $(\log_2 n)$-tuples, of which there are $(n/\log_2 n)$, and serially form the associative combination of the $(\log_2 n)$ components of each tuple using $\odot$ in $(\log_2 n)$ steps. We then perform the prefix computation on these $(n/\log_2 n)$ results. To complete the computation, for $1 \leq j \leq (n/\log_2 n) - 1$ we reread each of the original $(\log_2 n)$-tuples in parallel and add the $(j-1)$st result (the zeroth result is 0) to the first component of the $j$th $(\log_2 n)$-tuple, and then serially perform a prefix computation on these new $(\log_2 n)$-tuples.

We increase $(n/\log_2 n)$ to the next power of 4 (adding inputs whose corresponding outputs are ignored) and embed the tree of Fig. 7.23 directly into an H-tree. The initial associative combination of $(\log_2 n)$-tuples and the final prefix computation on $(\log_2 n)$-tuples are done at vertices of the H-tree that are I/O vertices of the prefix tree. This algorithm takes time $O(\log n)$ on the initial and final phases as well as on the prefix computation. Since the area of the layout is $O(n/\log_2 n)$ and every one of the $n$ inputs must be read, its area–time product, $AT$, is $O(n)$ which is optimal. This algorithm is multilective since each input is supplied twice.

## 12.5.2 Multi-dimensional Mesh Layouts

As explained in Section 7.5, many important problems can be solved with systolic arrays. If the cells of one- and two-dimensional systolic arrays are of fixed size and quasiplanar, they can be embedded directly onto a chip with area proportional to the number of cells. Applying the results of Theorems 7.5.1, 7.5.2, and 7.5.3 we have the following facts concerning the area and time for three important problems when realized by such arrays.

| *Problem* | *Dimensions* | *Area* | *Time* |
|---|---|---|---|
| $n \times n$ Matrix-Vector Multiplication | 1D | $O(n)$ | $O(n)$ |
| Bubble Sort of $n$ items | 1D | $O(n)$ | $O(n)$ |
| Batcher's Odd-Even Sorting of $n$ items | 1D | $O(n)$ | $O(n)$ |
| $\sqrt{n} \times \sqrt{n}$ Matrix-Matrix Multiplication | 2D | $O(n)$ | $O(\sqrt{n})$ |

Fully normal algorithms for problems such as shifting, summing, broadcasting, and fast Fourier transform on $n = 2^{2d}$ inputs can each be done in $O(\log n)$ steps on the $n$-vertex hypercube or the canonical cube-connected cycles network on $n$ vertices. From Theorems 7.7.4 and 7.7.5 these problems can also be solved in $O(n)$ and $O(\sqrt{n})$ steps, respectively, on $n$-vertex one- and two-dimensional systolic arrays. We summarize these facts in Figure 12.3.

| *Problem* | *Dimensions* | *Area* | *Time* |
|---|---|---|---|
| Shifting of $n$-vector | 1D | $O(n)$ | $O(n)$ |
| | 2D | $O(n)$ | $O(\sqrt{n})$ |
| Summing $n$ items | 1D | $O(n)$ | $O(n)$ |
| | 2D | $O(n)$ | $O(\sqrt{n})$ |
| Broadcasting to $n$ locations | 1D | $O(n)$ | $O(n)$ |
| | 2D | $O(n)$ | $O(\sqrt{n})$ |
| $n$-point FFT | 1D | $O(n)$ | $O(n)$ |
| | 2D | $O(n)$ | $O(\sqrt{n})$ |

**Figure 12.3** Area vs. time performance of VLSI algorithms for four problems.

In Section 12.6 we show that shifting of an $n$-vector, the $n$-point FFT, and $n \times n$ matrix-matrix multiplication each require area $A$ and time $T$ satisfying $AT^2 = \Omega(n^2)$. Consequently, the 2D algorithms cited above for these problems are optimal to within a constant factor.

In the next section we now show that every normal algorithm can be implemented on the cube-connected cycles (CCC) network in time $T$ satisfying $\Omega(\log n) \leq T \leq O(\sqrt{n})$ and that the CCC network can be embedded in the plane using area $A = O(n^2/T^2)$. In Theorems 12.7.2 and 12.7.3 we show that these implementations are optimal up to constant multiplicative factors with respect to area and time for the three problems mentioned above.

### 12.5.3 Layout of the CCC Network

In Section 7.7.6 we describe the realization of a fully normal algorithm on the canonical CCC network. The realization extends directly from the canonical CCC network to a general $(k, d)$-CCC network in which there are $2^d$ cycles and $2^k$ vertices on each cycle. (See Fig. 12.4.)

A fully normal algorithm is simulated on the CCC network by giving the processors on the $j$th cycle, $0 \leq j \leq 2^d - 1$, the addresses $i + j2^k$ where $0 \leq i \leq 2^k - 1$. The cycles are treated as 1D arrays and used to simulate a normal algorithm on the first $k$ dimensions exactly as is done in Section 7.7.6. These simulations are done in parallel after which the swaps across the higher-order $d$ dimensions are simulated by first rotating the leading element on each cycle to the first of the inter-cycle edges. After executing one swap, each cycle is advanced one step so that the second elements on each cycle are aligned with the first of the high-order dimensions. At this point the first elements on each cycle are aligned with the edge associated with the second of the high-order dimensions. Thus, while swaps are done between the second elements on each cycle across the first of the high-order dimensions, swaps occur between leading elements along the second of the high-order dimensions. This rotating and swapping is done until all cycle elements have been swapped across all high-order dimensions.

This algorithm performs $O(2^k)$ steps on the cycles to perform swaps across low-order dimensions and align the cycles for swaps at higher dimensions. An additional $O(d)$ steps are used to perform swaps on the $d$ high-order dimensions. Thus, the number of steps used by this algorithm, $T$, satisfies $T = O(2^k + d)$. The number of processors used in $(k, d)$-CCC network, $n$, satisfies $n = 2^{d+k}$.

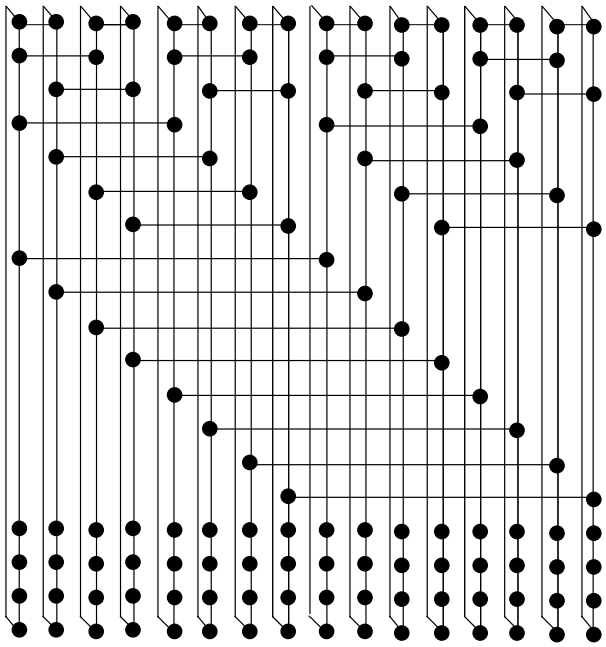


**Figure 12.4** An embedding of a $(k, d)$-CCC network in the plane for $k = 3$ and $d = 4$. The $2^d$ columns represent cycles of length $2^k \geq d$. For $1 \leq j \leq d$, the $j$th vertex on each cycle is connected to the $j$th vertex on another cycle.

Figure 12.4 shows a layout of a $(3, 4)$-CCC network. A layout for a general $(k, d)$-CCC network, $2^k \geq d$, can be developed following this pattern. Place each cycle of length $2^k$ in a column. Use $2^d - 1$ rows to make connections between columns. These rows are divided into $d$ sets. The first set, consisting of one row, connects adjacent columns. The second set, containing two rows, connects every other column. The $j$th set, containing $2^{j-1}$ rows, connects every $2^j$th column. The number of rows used for these connections is $1 + 2 + 4 + \cdots + 2^{d-1} = 2^d - 1$. Since $d$ processors are used in each column to make these connections, each column contains $2^k - d \geq 0$ processors not connected to other columns. (These are suggested by the lightly shaded vertices.) It follows that this layout has $2^d + 2^k - (d + 1)$ rows and $2^{d+1}$ columns. If a wire is assumed to have the same width as a processor, the layout has area $A = 2^{d+1}(2^d + 2^k - (d + 1))$.

Recall that $n = 2^{d+k}$ and $2^k \geq d$ or $k \geq \log_2 d$. It follows that $T = \Theta(2^k + d) = \Theta(2^k)$. Since $k \geq \log_2 d$, $T = \Omega(d) = \Omega(\log n)$. Also, when $k \leq d$, $2^{2k} \leq n$ and $T \leq O(\sqrt{n})$. We summarize this result below.

**THEOREM 12.5.1** *Every fully normal algorithm for a $n$-processor hypercube can be implemented on a CCC network whose VLSI layout has area $A$ and uses time $T$ satisfying the following bound for $\Omega(\log n) \leq T = O(\sqrt{n})$.*

$$AT^2 = O(n^2)$$

This result can be applied to any of the fully normal algorithms described in Section 7.6 and the Beneš permutation network discussed in Section 7.8.2.

## 12.6 Area–Time Tradeoffs

The $AT^2$ measure encountered in the last section is fundamental to VLSI computation. This is established by deriving a lower bound on $AT^2$ in terms of the planar circuit complexity, $C_{\mathrm{p},\Omega}(f)$, of the function $f$ computed by a VLSI chip of area $A$ in $T$ steps. A similar result is derived for the product $A^2T$. The planar circuit size of $f$ is the size of the smallest memoryless planar circuit for $f$. The measures $AT^2$ and $A^2T$ are the sizes of two different memoryless planar circuits that compute the same mapping from inputs to outputs as a VLSI chip of area $A$ that executes $T$ steps.

### 12.6.1 Planar Circuit Size

We now formally define planar circuit size and show how it relates to the standard circuit size measure.

**DEFINITION 12.6.1** *A* **planar circuit** *over the set $X$ is a logic circuit over the set $X$ that has been embedded in the plane in such a way that gates do not overlap but edges may cross. A planar circuit is* **semellective** *if there is a unique vertex at which each input variable is supplied. Otherwise, the planar circuit is* **multilective**.

*The* **size of a planar circuit** *is the number of inputs, edge crossings, and gates drawn from a basis $\Omega = \{h : X^2 \mapsto X\}$ that the circuit contains. The* **planar circuit size** *of a function $f : X^n \mapsto X^m$ over $\Omega$, $C_{\mathrm{p},\Omega}(f)$, is the size of the smallest planar circuit for $f$ over the basis $\Omega$.*

*A multilective circuit of order $\mu$, $\mu \geq 1$, for a function $f : \mathcal{B}^n \mapsto \mathcal{B}^m$ has $\mu n$ input vertices. The size of the smallest multilective planar circuit of order $\mu$ for $f$ is denoted $C^{(\mu)}_{\mathrm{p},\Omega}(f)$. If the planar circuit is semellective, the planar circuit size of $f$ is denoted $C^{(1)}_{\mathrm{p},\Omega}(f)$ or $C_{\mathrm{p},\Omega}(f)$ when confusion is not likely.*

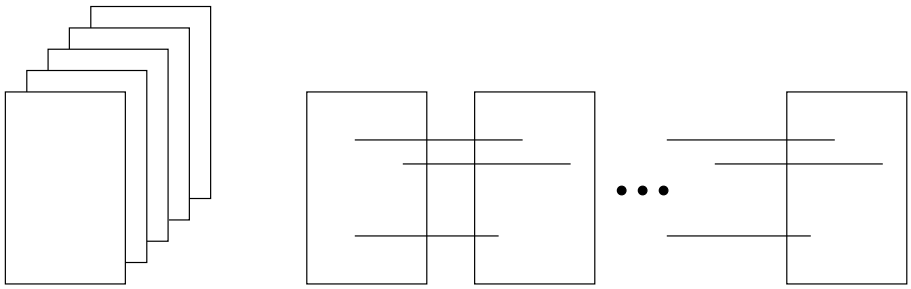
Every binary function has a planar circuit. To see this, observe that every function has a circuit, which is a graph, and that every graph has a planar embedding with edge crossings. The planar circuit size of a function is at worst quadratic in its standard circuit size, as we now show.

**LEMMA 12.6.1** *The (multilective) planar circuit and standard size of $f : \mathcal{B}^n \mapsto \mathcal{B}^m$ relative to the basis $\Omega$ are in the following relationship where $r$ is the fan-in of $\Omega$.*

$$C_\Omega(f) + n \leq C_{\mathrm{p},\Omega}(f) \leq r^2 C^2_\Omega(f)/2 + C_\Omega(f) + n$$

**Proof** The first inequality follows because the planar circuit size measure includes inputs, crossings, and gates, whereas the circuit size measure includes only gates.

Consider an embedding of a standard circuit for $f$ containing $C_\Omega(f)$ gates. In such an embedding it is not necessary for any two edges to intersect more than once because if they violate this condition the edge segments between any two successive crossings can be swapped so that these two crossings can be eliminated. Since every gate has at most $r$ inputs, a minimal standard circuit for $f$ has at most $rC_\Omega(f)$ edges connecting gates. It follows that



(a) (b)

**Figure 12.5** Two simulations of a $T$-step VLSI chip computation by a planar circuit.

the number of crossings does not exceed $r^2 C_\Omega(f)^2/2$ because there are at most $\binom{q}{2}$ ways of forming pairs drawn from a set of size $q$ and $q = rC_\Omega(f)$. Combining this with the number of inputs and gates, we have the desired upper bound. ■

In Section 12.7 we show that $f^{(n)}_{\text{cyclic}}$ nearly meets the upper bound of Lemma 12.6.1. That is, the planar circuit size of this function is nearly quadratic in its standard circuit size.

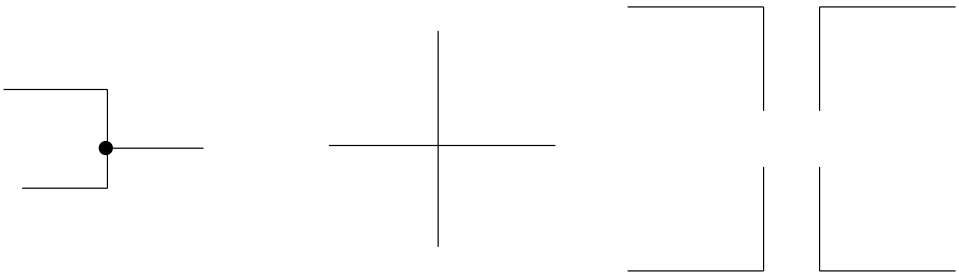
## 12.6.2 Computational Inequalities

We now show that every VLSI chip computation can be simulated by planar circuits of size $O(AT^2)$ and $O(A^2T)$. The simulation is patterned on the simulations of Chapter 3; that is, the loop that constitutes the computation by the chip with memory is unwound to create a planar circuit. Instead of passing the outputs of the next-state/output circuit to binary memory cells they are passed to another copy of the circuit.

Figure 12.5 shows two simulations of a $T$-step VLSI chip computation by a planar circuit. The first is obtained by placing $T$ copies of the chip one above the other and supplying the state output of one copy to the state input of the next copy. The second is simulated by placing $T$ copies of the chip side by side and running wires from the state output of one chip to the state input of the next. We convert each of these memoryless circuits to planar circuits and bound the number of inputs, crossings and gates they contain. Recall that we assume that wires are rectilinear; that is, they run only horizontally and vertically.

Since the number of wire layers on a single chip is bounded, it does not hurt to assume that the centerlines of parallel wires on different planes are displaced slightly. (It is bad practice to overlap wires because one wire can induce currents in the other.) Now make the width of wires and the area of gates infinitesimal. (Wires are shrunk to their centerline.) As shown in Fig. 12.6(a), each two-input gate is replaced by an infinitesimal vertex connected by a straight-line to its output and the two connections from its inputs are made by wires that contain bends (two wires touch). This converts a single chip to a planar graph with wires that touch or cross. (See Fig. 12.6(b) and (c)).

We now bound $n_{\text{w}}$, the number of wires, and $n_{\text{g}}$, the number of gates on a chip of area $A$. Since each wire has width $\lambda$ and length at least $\lambda$ and each gate occupies area $\lambda^2$, $n_{\text{w}}$ and



(a) (b) (c)

**Figure 12.6** (a) The result of shrinking a physical gate to a point. (b) A crossing of two wires, and (c) four types of connection between two wires.

$n_g$ satisfy the following bounds.

$$n_w \leq A/\lambda^2$$
$$n_g \leq A/\lambda^2$$

Because each point of crossing or touching of wires occupies area at least $\lambda^2$, the number of points at which wires cross and touch on each of the $\nu$ layers of a chip that has area $A$ is at most $A/\lambda^2$. As shown in Fig. 12.6(a), when gates are made infinitesimal two additional bends are created at the point at which the output wire touches the gate. This can be viewed as adding four wire bends per gate. Since the number of gates is at most $A/\lambda^2$, we have the following bound on $n_{cr}$, the number of wire crossings and touchings.

$$n_{cr} \leq (\nu + 4)A/\lambda^2$$

Consider the first of the two simulations. $T$ layers of one chip are placed one above the other. To expose overlapping wires, displace all layers to the northeast by an infinitesimal amount. Every pair of wires that cross or meet has the potential to introduce crossings, as suggested in Fig. 12.7(a) and (b). The maximum number of crossings that can be introduced per touching or crossing of wires is $T^2$. Since the number of input vertices is $O(AT)$, this provides an upper bound of $O(AT^2)$ on the number of inputs, gates, and crossings of the resultant planar circuit.

Now consider the second simulation. $T$ copies of one chip are laid side-by-side and the layout of each chip opened and at most $n_w$ parallel wires inserted to make connections to adjacent chips. Since there are $n_w$ wire segments on a single chip, at most $n_w^2$ new wire crossings are introduced on one chip. Thus, the number of inputs, gates, and crossings in this layout is $O(AT + n_w^2 T) = O(A^2T)$.

The following theorem, which is an application of Theorem 3.1.1 to the VLSI model, summarizes the above results. It makes use of the fact the planar circuit size of a function $f$ computed by a VLSI chip of the kind described above is no larger than that of the planar circuits just constructed. This theorem demonstrates the importance of the measures $AT^2$ and $A^2T$ as characterizations of the complexity of VLSI computations. It also shows that lower bounds on the performance of VLSI chips can be obtained in terms of the planar circuit size of the functions computed by them.

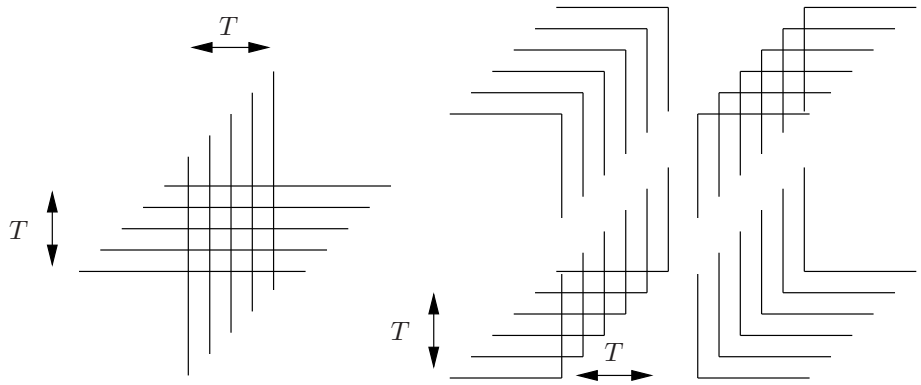



**Figure 12.7** Crossings obtained by translating infinitesimally to the northeast $T$ copies of (a) one crossing and (b) the four possible connections between two wires.

**THEOREM 12.6.1** *Let $f_M^{(T)}$ be the function computed by a VLSI chip that realizes the FSM $M$ in $T$ steps. The planar circuit size over a basis $\Omega = \{h : X^2 \mapsto X\}$ of any function $f$ computed by $M$ in $T$ steps satisfies the following inequalities:*

$$C_{\mathrm{p},\Omega_0}(f) = O(AT^2)$$
$$C_{\mathrm{p},\Omega_0}(f) = O(A^2T)$$

*If $M$ is multilective of order $\mu$, then $C_{\mathrm{p},\Omega}(f)$ is replaced by $C^{(\mu)}_{\mathrm{p},\Omega}(f)$.*

It is important to note that these relationships between planar circuit size and the measures $AT^2$ and $A^2T$ hold for all functions computed by VLSI algorithms, both multi-output functions and predicates.

In the next section we develop the planar separator theorem that is used in the next section to derive lower bounds on the planar circuit size of important problems.

## 12.6.3 The Planar Separator Theorem

The **planar separator theorem** applies to graphs $G = (V, E)$ for which a non-negative cost function $c$ is defined on $V$. The cost of $V$, denoted, $c(V)$, is the sum of the costs of every vertex in $V$. The theorem states that the vertices of every planar graph $G$ on $N$ vertices can be partitioned into three sets, $A$, $B$, and $C$ such that no edge connects a vertex in $A$ with one in $B$, the cost of vertices in $A$, $c(A)$, and those in $B$, $c(B)$, satisfy $c(A), c(B) \leq 2c(V)/3$ and $C$ contains at most $4\sqrt{N}$ vertices.

The following lemma uses the concept of the **spanning tree** of a graph, a tree that contains every vertex of a connected graph $G$. It shows the existence of a cycle that divides a planar graph into an "inside" and an "outside" containing about the same number of vertices. The **radius** of a rooted spanning tree is the number of edges on the longest path from the root to a vertex. (See Problem 12.8 for an illustration of the following lemma.)

**LEMMA 12.6.2** *Let $G = (V, E)$ be a finite connected planar graph. Let $c$ be a non-negative cost function defined on $V$ and let $c(V)$ be the total cost of all vertices in $V$. If $G$ has a rooted spanning tree of radius $r$, then $V$ can be partitioned into sets $A$, $B$, and $C$ such that $c(A), c(B) \leq 2c(V)/3$, no edge joins a vertex of $A$ with one of $B$, and $C$ contains at most $2r + 1$ vertices.*

**Proof** Since the lemma is true if the cost of any vertex exceeds 1/3, assume the converse. Let $G = (V, E)$ be embedded in the plane. A **face of a planar graph** is a region bounded by vertices and edges that does not contain any other vertices and edges. The **external face** of a finite planar graph is the face of unbounded area. Since $G$ is finite, it has an external face. A **triangular planar graph** is a planar graph in which each face is a **triangle**. If a planar graph is not triangular, it can be made triangular by choosing one vertex on the boundary of each face and adding an edge between it and every other vertex on this face to which it does not already have an edge. Without loss of generality we assume that $G$ is triangular.

Let $T$ be the spanning tree of radius $r$ postulated in the lemma. Each edge $e$ in $E$ not on $T$ defines a unique cycle $\xi(e)$ of length at most $2r + 1$. The cycle divides $V$ into three sets, vertices on $\xi(e)$, and vertices on each side of $\xi(e)$. Let $c_1(e)$ and $c_2(e)$ be the cost of vertices on either side. (The side with the larger cost is called the **inside of the cycle**.) We claim that for some $e$ not on $T$ the larger of $c_1(e)$ and $c_2(e)$ is more than $2c(V)/3$. We suppose the larger is no more than $2c(V)/3$ and establish a contradiction.

Let $e = (x, y)$ be an edge not on $T$ such that $\mu(e) = \max(c_1(e), c_2(e))$ is smallest and for all other $e^*$ such that $\mu(e^*) = \mu(e)$ the inside of $\xi(e)$ has the fewest faces. In case of ties, let $e$ be chosen arbitrarily. We show the assumption that $\mu(e) > 2c(V)/3$ is false.

Consider the triangle containing the edge $e = (x, y)$ on the side of the cycle $\xi(e)$ that has largest cost. Let $z$ be the third vertex in this triangle. $z$ is on the spanning tree because every vertex is on the tree. We consider two cases for $z$: (a) either edge $(x, z)$ or $(y, z)$ is in $T$ and (b) neither edge is in $T$.

In case (a) without loss of generality, let $(y, z)$ be in $T$. There are two subcases to consider: (a1) $z$ is on $\xi(e)$ (see Fig. 12.8(a)) and (a2) it is not on $\xi(e)$ (see Fig. 12.8(b)). In (a1) the edge $e' = (x, z)$ cannot be a tree edge since $T$ contains no cycles unless the cycle consists of just the vertices $x$, $y$, and $z$, which is impossible since the inside of $\xi(e)$ contains

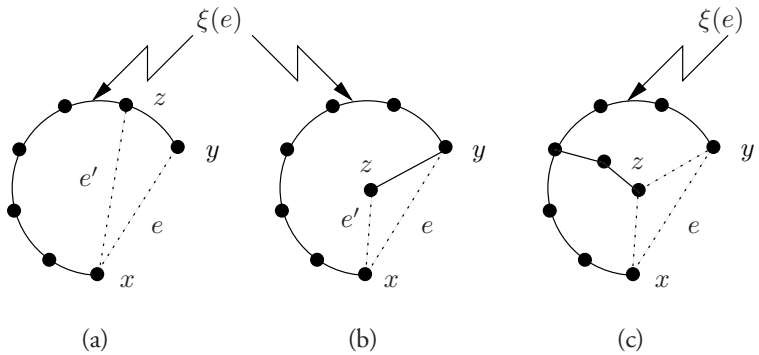



**Figure 12.8** A non-tree edge $e = (x, y)$ in a triangular planar graph with spanning tree $T$ defines a cycle $\xi(e)$. The triangle containing $e$ on the larger side of $\xi(e)$ contains a third vertex $z$. In (a) and (b) $(y, z)$ is on $T$, whereas in (c) neither $(x, z)$ nor $(y, z)$ is on $T$. In (a) $(y, z)$ is on $\xi(e)$, whereas in (b) it is not.

at least one vertex. But $\xi(e')$ includes the same set of vertices of $V$ inside it (and has the same cost) as does $\xi(e)$, although it has fewer faces, contradicting the choice for $e = (x, y)$.

In case (a2) the edge $e' = (x, z)$ is a non-tree edge since $T$ contains no cycles. The inside of $\xi(e')$ contains no more cost and one less face than $\xi(e)$. If the cost inside $\xi(e')$ is greater than the cost outside, $e'$ would have been chosen instead of $e$. On the other hand, if the cost inside $\xi(e')$ is at most the cost outside, since the latter is equal to the cost outside $\xi(e)$, which is at most $c(V)/3$, the cost inside $\xi(e')$ is at most $c(V)/3$. However, this contradicts the assumption that $\mu(e^*) > 2c(V)/3$ for all edges $e^*$.

Consider the case (b) in which neither edge $(x, z)$ nor $(y, z)$ is in $T$. (See Fig. 12.8(c).) The edges $(x, z)$ and $(y, z)$ each define a cycle contained within $\xi(e)$. Without loss of generality assume that the cycle defined by $(x, z)$ has more cost on the inside of $\xi(e)$ than does the cycle defined by $(y, z)$. Because the cost of vertices on the inside of the original cycle is more than $2c(V)/3$, the cost inside and on $\xi((x, z))$ is more than $c(V)/3$. Thus, the cost outside $\xi((x, z))$ is less than or equal to $2c(V)/3$. If the cost inside $\xi((x, z))$ is also less than or equal to $2c(V)/3$, we have a contradiction. If greater than $2c(V)/3$, $\xi((x, z))$ is a cycle with fewer faces for which $\mu((x, z)) > 2c(V)/3$, another contradiction. ■

The following theorem uses Lemma 12.6.2 together with a spanning tree constructed through a breadth-first traversal of a connected planar graph to show the existence of a small separator that divides the vertices into approximately two equal cost parts.

**THEOREM 12.6.2** *Let $G = (V, E)$ be an $N$-vertex planar graph having non-negative vertex costs summing to $c(V)$. Then, $V$ can be partitioned into three sets, $A$, $B$, and $C$, such that no edge joins vertices in $A$ with those in $B$, neither $A$ nor $B$ has cost exceeding $2c(V)/3$, and $C$ contains no more than $4\sqrt{N}$ vertices.*

**Proof** We assume that $G$ is connected. If not, embed it in the plane and add edges as appropriate to make it connected. Assume that it has been triangulated, that is, every face except for the outermost is a triangle.

Pick any vertex (call it the root) and perform a breadth-first traversal of $G$. This traversal defines a **BFS spanning tree $T$ of $G$**. A vertex $v$ has level $d$ in this tree if the length of the path from the root to $v$ has $d$ edges. There are no vertices at level $q$ where $q$ is the level one larger than that of all vertices. Let $R_d$ be the vertices at level $d$ and let $r_d = |R_d|$.

The reader is asked to show that there is some level $m$ such that the cost of vertices at levels below and above $m$ each is at most $c(V)/2$. (See Problem 12.9.) Let $l$ and $h$, $l \leq m \leq h$, be levels closest to $m$ that contain at most $\sqrt{N}$ vertices. That is, $r_l \leq \sqrt{N}$ and $r_h \leq \sqrt{N}$. There are such levels because level 0 contains a single vertex and there are none at level $q$.

The vertices in $G$ are partitioned into the following five sets: a) $L = \bigcup_{d<l} R_d$, b) $R_l$, c) $M = \bigcup_{l<d<h} R_d$, d) $R_h$, and e) $H = \bigcup_{h<d} R_d$. Since $L$ and $H$ are subsets of the sets of vertices with levels less than and more than $m$, $c(L), c(H) \leq c(V)/2$. Also, by construction, $r_l, r_h \leq \sqrt{N}$. If $R_l = R_h = R_m$ (which implies that $M$ is empty and $l = h = m$), let $A = L$, $B = H$, and $C = R_l = R_h$. Then, $C$ is a separator of size at most $\sqrt{N}$ and the theorem holds. If $l \neq h$, then $h - l - 1 \geq 0$. Since each of the $h - l - 1$ levels between $l$ and $h$ has at least $\sqrt{N} + 1$ vertices, it follows that $h - l - 1 \leq \sqrt{N} - 1$ because these levels cannot have more than $N - 1$ vertices altogether.

Consider the subgraph of $G$ consisting of the vertices in $M$ and the edges between them. Add a new vertex $v_0$ to replace the vertices in $L \cup R_l$ and add an edge from $v_0$ to each of

the vertices at level $l+1$. This operation retains planarity and the resulting graph remains triangulated because adjacent vertices on $R_{l+1}$ have an edge between them. Also, it defines a spanning tree $T^*$ consisting of $v_0$, the new edges, and the projection of the original spanning tree to the vertices in $M$. $T^*$ has radius at most $\sqrt{N}$.

Apply Lemma 12.6.2 to $T^*$ giving $v_0$ zero cost. This lemma identifies three sets of vertices, $A_0$, $B_0$ and $C_0$, from which we delete $v_0$ and adjacent edges. Since $c(M) \leq c(V)$, it follows that there are no edges between vertices in $A_0$ and $B_0$, $c(A_0), c(B_0) \leq 2c(V)/3$, and $|C_0| \leq 2\sqrt{N}$. Let $C = C_0 \cup R_l \cup R_h$. It follows that $|C| \leq 4\sqrt{N}$.

Each of the four sets $A_0$, $B_0$, $L$, and $H$ has cost at most $2c(V)/3$. If any one of them has cost more than $c(V)/3$, let it be $A$ and let $B$ be the union of the remaining sets. If none of them has cost more than $c(V)/3$ vertices, order the sets by size and let $A$ be the union of the fewest of these sets whose cost is at least $c(V)/3$ vertices. This procedure insures that $A$ has cost between $c(V)/3$ and $2c(V)/3$ which implies that $B$ satisfies the same condition as $A$ and the theorem is established. ■

The preceding version of the planar separator theorem only guarantees that the vertices of a planar graph are divided into two sets whose costs are nearly balanced and a small separator. It does not insure that the number of vertices in the two sets are balanced. The following lemma remedies this situation. We leave its proof to the reader. (See Problem 12.10.)

**LEMMA 12.6.3** *Let $G = (V, E)$ be an $N$-vertex planar graph having non-negative vertex costs summing to $c(V)$. Then $V$ can be partitioned into three sets, $A$, $B$, and $C$, such that no edge joins vertices in $A$ with those in $B$, neither $A$ nor $B$ has cost exceeding $7c(V)/9$, $|A|, |B| \leq 5N/6$, and $C$ contains no more than $K_1\sqrt{N}$ vertices, where $K_1 = 4(\sqrt{2/3}+1)$.*

This new result can be applied to show that the vertices of a planar graph can be partitioned into many sets each having about the same cost and such that a small set of vertices can be removed to separate each set from all other sets. This result is also left to the reader. (See Problem 12.11.)

**LEMMA 12.6.4** *Let $G = (V, E)$ be an $N$-vertex planar graph and let $c$ be a non-negative cost function on $V$ with total cost of $c(V)$. Let $P \geq 2$. There are constants $2P/3 \leq q \leq 3P$ and $K_2 = 4(\sqrt{2/3}+1)/(1-\sqrt{5/6})$ such that $V$ can be partitioned into $q$ sets, $A_1, A_2, \ldots, A_q$ such that for $1 \leq i \leq q$*

$$c(V)/(3P) \leq c(A_i) \leq 3c(V)/(2P)$$

*and there are sets $C_i$, $|C_i| \leq K_2\sqrt{N}$, and $B_i = V - A_i - C_i$ such that no edges join vertices in $A_i$ with vertices in $B_i$.*

## 12.7 The Performance of VLSI Algorithms

Using Theorem 12.6.1 and Lemma 12.6.4, we now derive lower bounds on $AT^2$ and $A^2T$ for individual functions by deriving lower bounds on their planar circuit size. In the following section we derive lower bounds to the planar circuit size for multi-output functions using the $w(u, v)$-flow property of these functions. In Section 12.7.2 we set the stage for deriving lower bounds on the planar circuit size of predicates.

### 12.7.1 The Performance of VLSI Algorithms on Functions

The $w(u, v)$-flow property of functions is introduced in Section 10.4.1 and applied to the study of space–time tradeoffs in the pebble game. In this section we use this property to derive lower bounds on the semellective planar circuit size of multi-output functions.

**DEFINITION 12.7.1** *A function $f : X^n \mapsto X^m$ has a* $\boldsymbol{w(u, v)}$**-flow** *if for all subsets $U_1$ and $V_1$ of its $n$ input and $m$ output variables with $|U_1| \geq u$ and $|V_1| \geq v$ there is some assignment to variables not in $U_1$ (variables in $U_0$) such that the resulting subfunction $h$ of $f$ that maps input variables in $U_1$ to output variables in $V_1$ (the other outputs are discarded) has at least $|X|^{w(u,v)}$ points in the image of its domain. (Note that $w(u, v) \geq 0$.)*

A lower bound on planar circuit size of a function $f$ is now derived from its $w(u, v)$-flow property. For some functions the parameter $P$ will need to be large for $w(u, v) > 0$, as is seen Lemma 12.7.1.

**THEOREM 12.7.1** *Let $f : X^n \mapsto X^m$ have a $w(u, v)$-flow. Then its semellective planar circuit size must satisfy the following lower bound for $u \geq n(1 - 3/2P)$, $v \geq m/(3P)$, and $P \geq 2$, where $K_2 = 4(\sqrt{2/3} + 1)/(1 - \sqrt{5/6})$.*

$$C_{\mathrm{p},\Omega}(f) \geq \frac{w^2(u, v)}{4K_2^2}$$

**Proof** Consider a minimal semellective planar circuit for $f : X^n \mapsto X^m$ on $n$ inputs containing $N = C_{\mathrm{p},\Omega}(f)$ inputs, gates, and crossings. We apply the version of the planar separator theorem given in Lemma 12.6.4 to this circuit by assigning unit weight to each input vertex and zero weight to all other vertices. For any integer $P \leq |V|$ we conclude that the inputs, gates, and crossings of this circuit can be partitioned into $q$ sets $\{A_1, A_2, \ldots, A_q\}$, for $2P/3 \leq q \leq 3P$, such that each set has at least $n/(3P)$ and at most $3n/(2P)$ input vertices. Since the average number of output vertices in these sets is $m/q$, at least one set, call it $A_1$, has at least the average of output vertices or at least $m/3P$ vertices. Let $U_0$ and $V_1$ be the sets of inputs and outputs in $A_1$, respectively. Then, $n/(3P) \leq |U_0| \leq 3n/(2P)$ and $|V_1| \geq m/3P$.

For some assignment of values to variables in $U_0$, there are at least $|X|^{w(u,v)}$ values for the outputs in $V_1$ when $u = n - |U_0| \geq n(1 - 3/2P)$ and $v = |V_1| \geq m/(3P)$. But all of the values assumed by the outputs in $V_1$ must be assumed by the inputs, gates, and crossing wires of the separator. Since at most two wires cross, a separator $C$ of size $|C|$ has at most $2|C|$ inputs, gates, and wires each of which can have at most $|X|$ values. Thus, if $C_1$, the separator for $A_1$, has a size satisfying $2|C_1| < w(u, v)$, a contradiction results and the output variables in $V_1$ cannot assume $|X|^{w(u,v)}$ values. It follows that $|C_1| \geq w(u, v)/2$. Since $C_1 \leq K_2\sqrt{N}$, this implies that $N \geq w^2(u, v)/(2K_2)^2$, the desired conclusion. ■

We apply this general result to $(\alpha, n, m, p)$-independent functions and matrix multiplication. A function is $(\alpha, n, m, p)$-independent (see Definition 10.4.2) if it has a $w(u, v)$-flow satisfying $w(u, v) > (v/\alpha) - 1$ for $n - u + v \leq p$, where $n - u \geq 0$.

**LEMMA 12.7.1** *Let $f : X^n \mapsto X^m$ be $(\alpha, n, m, p)$-independent. Then for $P \geq (m/3 + 3n/2)/p$ and $m \geq 2\alpha$, $f$ has semellective planar circuit size satisfying the following lower bound:*

$$C_{p,\Omega}(f) \geq \frac{m^2}{144(\alpha P)^2 K_2^2}$$

**Proof** $f$ has a $w(u, v)$-flow satisfying $w(u, v) > (v/\alpha) - 1$ for $n - u + v \leq p$. When $u \geq n(1 - 3/2P)$, $n - u + v \leq p$ is satisfied if $v \leq p - 3n/(2P)$. Since we also require that $v \geq m/(3P)$, this implies that $P \geq (m/3 + 3n/2)/p$. Also, $v/\alpha - 1 \geq v/2\alpha$ if $v \geq 2\alpha$. Substituting $m/3P$ for $v$, we have the desired conclusion. ■

In Section 10.5 we have shown that many functions are $(\alpha, n, m, p)$-independent. We summarize these results below.

| *Name* | *Function* | *Independence Property* |
|---|---|---|
| Wrapped convolution | $f^{(n)}_{\text{wrapped}} : \mathcal{R}^{2n} \mapsto \mathcal{R}^n$ | $(2, 2n, n, n/2)$ |
| Cyclic shift | $f^{(n)}_{\text{cyclic}} : \mathcal{B}^{n+\lceil \log n \rceil} \mapsto \mathcal{B}^n$ | $(2, n + \lceil \log n \rceil, n, n/2)$ |
| Integer multiplication | $f^{(n)}_{\text{mult}} : \mathcal{B}^{2n} \mapsto \mathcal{B}^{2n}$ | $(2, 2n, n, n/2)$ |
| $n$-point DFT | $F_n : \mathcal{R}^n \mapsto \mathcal{R}^n$ | $(2, n, n, n/2)$ |

It follows that for each case Lemma 12.7.1 holds when $P \leq m/(6\alpha)$. Thus, each of the $(\alpha, n, m, p)$-independent function has a planar circuit size that is quadratic in $n$, its number of inputs. The following theorem results from this observation and Theorem 12.6.1.

**THEOREM 12.7.2** *The area $A$ and time $T$ required to compute $f^{(n)}_{\text{wrapped}} : \mathcal{R}^{2n} \mapsto \mathcal{R}^n$, $f^{(n)}_{\text{cyclic}} : \mathcal{B}^{n+\lceil \log n \rceil} \mapsto \mathcal{B}^n$, $f^{(n)}_{\text{mult}} : \mathcal{B}^{2n} \mapsto \mathcal{B}^{2n}$, and $F_n : \mathcal{R}^n \mapsto \mathcal{R}^n$ on a semellective VLSI chip satisfy the following bounds:*

$$AT^2, A^2T = \Omega(n^2)$$

*The $AT^2$ lower bound can be achieved up to a constant multiplicative factor for each of these functions for $\Omega(\log n) \leq T \leq \sqrt{n}$.*

**Proof** From Theorem 12.5.1 we know that any fully normal algorithm can achieve the $AT^2 = O(n^2)$ for $\Omega(\log n) = T = O(\sqrt{n})$ on an embedded CCC network. Since cyclic shift and FFT are shown to be fully normal (see Section 7.7), we have matching upper and lower bounds for them. From Problem 12.13 we have that the wrapped convolution can be realized with matching bounds on $AT^2$ over the same range of values for $T$. The same statement applies to integer multiplication (see Problem 12.16). ■

In Section 12.6.1 we said that we would exhibit a function whose planar circuit size is nearly quadratic in its standard circuit size. This property holds for the cyclic shifting function because, as shown in Section 2.5.2, $f^{(n)}_{\text{cyclic}} : \mathcal{B}^{n+\lceil \log n \rceil} \mapsto \mathcal{B}^n$ has circuit size no larger than $O(n \log n)$, whereas from the above its planar circuit size is $\Theta(n^2)$.

The cyclic shift function is also an example of a function for which most of the chip area is occupied by wires when $T = O(\sqrt{n/\log n})$, because in this case the area is $\Omega(n \log n)$ but the number of gates needed to realize it is $O(n \log n)$.

Lower bounds on $AT^2$ and $A^2T$ also exist for matrix multiplication. From Lemma 10.5.3 we know that the matrix multiplication function $f^{(n)}_{A\times B} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ has a $w(u,v)$-flow, where $w(u,v) \geq (v-(2n^2-u)^2/4n^2)/2$. Using this we have the following lower bound on the planar circuit size of this function.

**THEOREM 12.7.3** *The area $A$ and time $T$ required to compute the matrix multiplication function $f^{(n)}_{A\times B} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ with a semellective VLSI algorithm satisfies the following lower bound:*

$$AT^2, A^2T = \Omega(n^4)$$

*The $AT^2$ lower bound can be met to within a constant multiplicative factor.*

**Proof** Apply Theorem 12.7.1 to matrix multiplication by replacing the number of input variables $n$ by $2n^2$ and the number of output variables $m$ by $n^2$. The $w(u,v)$-flow function has value

$$w(u,v) = (v-(2n^2-u)^2/4n^2)/2 \geq \frac{n^2}{2}\left(\frac{1}{3P} - \left(\frac{3}{2P}\right)^2\right)$$

The right-hand side is maximized when $P = 14$ and has value greater than $n^2/163$, from which the conclusion follows.

As shown in Section 7.5.3, two $n\times n$ matrix can be multiplied with area $A = O(n^2)$ and time $T = n$, which meets the lower bound up to a multiplicative factor. Other near-optimal solutions also exist. (See Problem 12.15.) ■

## 12.7.2 The Performance of VLSI Algorithms on Predicates

The approach taken above can be extended to predicates, functions whose range is $\mathcal{B}$. Again we derive lower bounds on the size of the smallest planar circuit for a function. However, since the flow of information from inputs to outputs is at most one bit, we must find some other way to measure the amount of information that must be exchanged between the two halves of a planar circuit. An extension of the communication complexity measure introduced in Section 9.7.1 serves this purpose.

The communication complexity measure of Section 9.7.1 assumes that two players exchange bits to compute the value of a Boolean function $f : \mathcal{B}^n \mapsto \mathcal{B}$. The input variables of $f$ are partitioned into two sets $U$ and $V$ and assigned to two players. Given this partition, the players choose a protocol (a scheme for alternating the transmission of bits from one to the other) by which to decide the value of $f$ for every input $n$-tuple of $f$. The bits of each $n$-tuple are partitioned between the two players according to the division of the $n$ input variables between the sets $U$ and $V$. The players then use their protocol to determine the value of $f$. The **communication complexity** $C(U,V)$ of this game is the minimum over protocols of the maximum over input $n$-tuples of the number of bits exchanged by the players to compute $f$ given the partition of the input variables into sets $U$ and $V$. This measure and its associated game are naturally extended to predicates $f : X^n \mapsto \mathcal{B}$, whose variables assume values over the set $X$. Players now exchange values drawn from the set $X$.

We can derive a lower bound on planar circuit size by applying the planar separator theorem. Since this theorem partitions the input variables into three sets, $A$, $B$, and a separator $C$, where $A$ and $B$ contain at most two-thirds of the total number of input vertices, it is natural

to extend the standard communication complexity measure to the following VLSI communication complexity measure for functions $f : X^n \mapsto \mathcal{B}$.

**DEFINITION 12.7.2** *The* **VLSI communication complexity** *of a predicate* $f : X^n \mapsto \mathcal{B}$, $CC_{\text{vlsi}}(f)$, *is the minimum of the communication complexity* $C(U, V)$ *over all partitions* $(U, V)$ *of the variables of* $f$ *into two sets of size at most* $2n/3$.

The following theorem, which is left as an exercise (see Problem 12.17), summarizes the result of applying the VLSI communication complexity measure $CC_{\text{vlsi}}(f)$ together with the planar separator theorem to derive a lower bound on the semellective planar circuit size of predicates.

**THEOREM 12.7.4** *Let* $f : X^n \mapsto \mathcal{B}$ *have VLSI communication complexity* $CC_{\text{vlsi}}(f)$. *Then, the following bounds hold for the computation of* $f$ *by a semellective VLSI chip with area* $A$ *in* $T$ *steps.*

$$(CC_{\text{vlsi}}(f))^2 = O(AT^2), O(A^2T)$$

Note that in a planar circuit all the information passed from each side of the separator to the other is sent simultaneously, whereas in the communication game players alternate in sending values drawn from the set $X$. Because more freedom is granted to players in the communication game (each player can choose data to send based on responses previously received from the other player), a lower bound on communication complexity is a lower bound on the amount of information that must be exchange in a planar circuit.

A number of techniques have been developed to derive lower bounds on the planar circuit size of predicates. One of these uses the pigeonhole principle (also known as a **crossing-sequence argument**) to derive lower bounds for predicates that are $w(u, v)$-separated. This new property is similar to the $w(u, v)$-flow property of multi-output functions. It is defined below.

**DEFINITION 12.7.3** *A function* $f : X^n \mapsto \mathcal{B}$ *is* $w(u, v)$*-separated if its variables can be permuted and partitioned into three sets* $U$, $V$, *and* $Z$, $|U| \geq u$ *and* $|V| \geq v$, *such that there is some value* $\boldsymbol{z}$ *for variables in* $Z$ *and values* $\boldsymbol{u}_i$ *and* $\boldsymbol{v}_i$, $1 \leq i \leq |X|^{w(u,v)}$, *for variables in* $U$ *and* $V$, *respectively, such that the following holds:*

$$f(\boldsymbol{u}_i, \boldsymbol{v}_j, \boldsymbol{z}) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

This definition can be applied to predicates that are associated with multi-output functions. These functions are defined below.

**DEFINITION 12.7.4** *The characteristic predicate* $p_f : X^{(n+m)} \mapsto \mathcal{B}$ *of* $f : X^{(n)} \mapsto X^{(m)}$ *is defined below.*

$$p_f(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{y} = f(\boldsymbol{x}) \\ 0 & \text{otherwise} \end{cases}$$

It is straightforward to show that the characteristic predicate of a function that has a $w(u, v)$-flow is $w(u, v)$-separated. (See Problem 12.18.) As a consequence, quadratic lower

bounds exist on the semellective planar circuit size of the characteristic predicates of the convolution, cyclic shift, integer multiplication, discrete Fourier transform, matrix multiplication functions, and many others.

## 12.8 Area Bounds

We now derive lower bounds on the area used by semellective VLSI chip algorithms for a variety of functions. For the functions considered here, these bounds are linear in their number of variables. As explained in the Chapter Notes, not all functions are amenable to the type of analysis presented in this section.

The technique used to derive area lower bounds is similar to that used in Section 10.4.2 to derive lower bounds on the exchange of space for time in the pebble game. If a chip has many I/O ports, it has large area. On the other hand, if it has a small number of ports, the inputs to the function computed are received over many cycles. If the function has a large $w(u, v)$-flow, by direct analogy with the pebble game, the area must be large to insure that enough information be stored between cycles.

**THEOREM 12.8.1** *Let $\beta \geq 1$. If $f : X^n \mapsto X^m$ has a $w(u, v)$-flow, every chip computing $f$ requires area $A = \Omega(\min((m/2\beta), w(u, v)))$, where $u = n(1 - 1/\beta)$ and $v = (m/4\beta)$.*

**Proof** If the chip has $\pi$ I/O pads or can store $S$ values over the alphabet $X$, it has area $A \geq \lambda^2 \min(\pi, S)$. Fix $\beta \geq 1$. Its value is chosen later to provide a strong lower bound. If $\pi \geq m/2\beta$, we are done. Thus, we show that $S \geq w(u, v)$ when $\pi < m/2\beta$.

Let the VLSI algorithm have $T$ time steps and let $h_i \leq \pi$ outputs be generated on the $i$th time step, $1 \leq i \leq T$. Create $q$ intervals of consecutive time steps as follows: The first interval contains the first $k_1$ time steps, where $k_1$ is such that the total number of outputs produced during the first $k_1$ steps is as large as possible without exceeding $m/\beta$. Successive intervals are created in the same way, namely by grouping consecutive later time steps to satisfy the same requirement on the number of outputs produced. For all intervals except possibly the last, the number of outputs produced is at least $(m/\beta) - \pi + 1 > (m/2\beta)$. If the last interval contains fewer than $(m/2\beta)$ outputs, redistribute the elements in the last two intervals, of which there are at least $(m/\beta) - \pi + 2 \geq (m/2\beta) + 2$, so that each has at least $(m/4\beta) + 1$ outputs. It follows that the number of intervals, $q$, satisfies $\beta \leq q \leq 4\beta$.

We now examine the inputs read during intervals. Since there are $n$ inputs to be read and each is read once, the average number read per interval is $n/q$ which is at most $n/\beta$. It follows that there is some interval $I$ in which at least $(m/4\beta) + 1$ outputs are pebbled and at most $n/\beta$ inputs are read.

Fix the inputs that are read during $I$. The remaining inputs, of which there are at least $u = n(1 - 1/\beta)$, are free to vary. The number of outputs produced during $I$ is at least $v = (m/4\beta)$. Since $f$ has a $w(u, v)$-flow, if $S < w(u, v)$, the $v$ outputs, whose values are determined by the values stored on the chip at the beginning of $I$, cannot assume all their values. It follows that $S \geq w(u, v)$, which is the desired conclusion. ■

We now apply this bound to $(\alpha, n, m, p)$-independent functions. Later we apply it to the matrix multiplication function.

**THEOREM 12.8.2** *Let $f : X^n \mapsto X^m$ be $(\alpha, n, m, p)$-independent. It requires area $A = \lambda^2((mp/(n + m/4)\alpha) - 1)$ when realized by a semellective VLSI algorithm.*

**Proof** We apply Theorem 12.8.1 with $u = n(1 - 1/\beta)$ and $v = (m/4\beta)$. Because $f$ is $(\alpha, n, m, p)$ independent, $w(u, v) > v/\alpha - 1$ for $n - u + v \leq p$. Since $n - u = n/\beta$ and $v = (m/4\beta)$, this implies that $\beta \geq (n + m/4)/p$. The lower bound of Theorem 12.8.1 then is the smaller of $(m/2\beta)$ and $(m/4\alpha\beta) - 1$. Since we are free to choose $\beta$, we choose it to make the smaller of the two as large as possible. In particular, we set $\beta = (n+m/4)/p$, which provides the desired result. ■

Because all of the $(\alpha, n, m, p)$-independent functions listed in Theorem 12.7.2 have $n$, $m$, and $p$ proportional to one another, each requires area $A = \Omega(n)$, as stated below. It follows that the lower bound $AT^2 = \Omega(n^2)$ for these problems cannot be achieved to within a constant multiplicative factor if $T$ grows more rapidly with $n$ than $\sqrt{n}$.

**COROLLARY 12.8.1** *The functions* $f^{(n)}_{\text{wrapped}} : \mathcal{R}^{2n} \mapsto \mathcal{R}^n$, $f^{(n)}_{\text{cyclic}} : \mathcal{B}^{n+\lceil \log n \rceil} \mapsto \mathcal{B}^n$, $f^{(n)}_{\text{mult}} : \mathcal{B}^{2n} \mapsto \mathcal{B}^{2n}$, *and* $F_n : \mathcal{R}^n \mapsto \mathcal{R}^n$ *each require area* $A = \Omega(n)$ *when realized by a semellective VLSI algorithm.*

A similar result applies to matrix multiplication.

**THEOREM 12.8.3** *The area $A$ required to compute the matrix multiplication function* $f^{(n)}_{A\times B} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ *with a semellective VLSI algorithm satisfies* $A = \Omega(n^2)$

**Proof** We apply Theorem 12.8.1 with $n$ and $m$ replaced by $2n^2$ and $n^2$, respectively. Since $u = 2n^2(1 - 1/\beta)$ and $v = (n^2/4\beta)$, the lower bound on $w(u, v)$-flow for matrix multiplication function satisfies the following

$$w(u, v) = (v - (2n^2 - u)^2/4n^2)/2 \geq \frac{n^2}{2}\left(\frac{1}{4\beta} - \frac{1}{\beta^2}\right)$$

The lower bound is a positive multiple of $n^2$ if $\beta > 4$ and largest for $\beta = 8$, from which the desired conclusion follows. ■

# Problems

### VLSI COMPUTATIONAL MODELS

12.1 Assume the I/O ports are on the periphery of a convex chip. In the speed-of-light model show that if $p$ such ports all have paths to some point on the chip, then the time for data supplied to each port to reach that point is $\Theta(p)$.

12.2 Under the assumptions of Problem 12.1, derive a lower bound on the time to compute a function $f$ on $n$ inputs under the additional assumption that there is a path on the chip from the port at which each variable arrives to the port at which $f$ is produced.

**Hint:** Show that the time required is at least the sum of the number of cycles needed to read all $n$ inputs and the time for data to travel across the chip. State these times in terms of $p$ and choose $p$ to maximize the smaller of these two lower bounds.

### CHIP LAYOUT

12.3 Show that every layout of a balanced binary tree on $n$ leaves in which the root and the leaves are placed on the boundary of a convex region has area proportional to $n \log n$.

**Hint:** Consider an inscribed quadrilateral defined by the longest chord and a chord perpendicular to it.

12.4 The $n \times n$ mesh-of-trees network, $n = 2^r$, is described in Problem 7.4. Give an area-efficient layout for an arbitrary graph in this family of graphs and derive an expression for its area.

12.5 Let $n = 2^k$. As suggested in Fig. 12.9, the $n \times n$ **tree of meshes** $T_n$ is a binary tree in which each vertex is a mesh and the meshes are decreasing in size with distance from the root. The edges between vertices are bundles of parallel wires. The root vertex is an $n \times n$ mesh, its immediate descendants are $n/2 \times n$ meshes, and their immediate descendants are $n/2 \times n/2$ descendants, and so on.

The depth-$d$, $n \times n$ mesh of trees, $T_{n,d}$, is $T_n$ that has been truncated to vertices at distance $d$ or less from the root.

Determine the area of an area-efficient layout of the tree $T_{n,d}$.

### COMPUTATIONAL INEQUALITIES

12.6 Use the results of Problem 12.11 to extend Theorem 12.7.1 to multilective planar circuits of order $\mu$.

12.7 Further extend the results of Problem 12.6 to $(\beta, \mu)$-multilective VLSI algorithms by showing that, at the expense of a small increase in $AT^2$ and $A^2T$, multiple inputs of a variable at the same I/O port can be treated as a single input, thereby possibly reducing the multilective order of the corresponding planar circuit. This implies that if multiple copies of each variable are read at a single port, then the semellective planar circuit size is a lower bound to both $AT^2$ and $A^2T$.

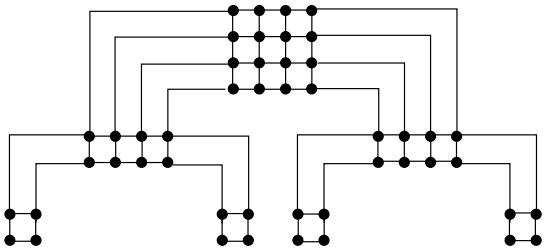


**Figure 12.9** The $4 \times 4$ tree of meshes, $T_4$.

### THE PLANAR SEPARATOR THEOREM

12.8 The **pizza pie graph** $G = (V, E)$ has $n = |V| - 1$ vertices that are uniformly spaced points on a circle as well as a vertex at the center of the circle. $E$ consists of the arcs between vertices on the circle and edges between the central vertex and vertices on the circle.

When $n = 12$, triangulate $G$ by adding edges between vertices on its external face. Illustrate Lemma 12.6.2 by choosing a cost function $c$ and constructing two sets whose cost at most $2\,c(V)/3$ and a separator containing at most three vertices.

12.9 In a spanning tree for a graph $G = (V, E)$ the level of a vertex is the length of the path from the root to it. Given a non-negative cost function on the vertices of $G$ totaling $c(V)$, show there is some level $m$ such that the cost of vertices at levels less than and more than $m$ each is at most $c(V)/2$.

12.10 (**Two-Cost Planar Separator Theorem**) Let $G = (V, E)$ be an $N$-vertex planar graph having non-negative vertex costs summing to $c(V)$. Show that $V$ can be partitioned into three sets, $A$, $B$, and $C$, such that no edge joins vertices in $A$ with those in $B$, neither $A$ nor $B$ has cost exceeding $7c(V)/9$, $|A|$ and $|B|$ contain at most $5N/6$ vertices, and $C$ contains no more than $K_1\sqrt{N}$ vertices, where $K_1 = 4(\sqrt{2/3} + 1)$.

**Hint:** Apply the planar separator theorem twice. The first time use it to partition $V$ into two sets of about the same size and a separator. If each of the two sets has cost at most $2c(V)/3$, the result holds. If not, make a second application of the planar separator theorem to the set with larger cost. Show that it is possible to combine sets to simultaneously meet both the size and cost requirements.

12.11 Let $G = (V, E)$ be an $N$-vertex planar graph and let $c$ be a non-negative cost function on $V$ with total cost $c(V)$. Let $P \geq 2$. Show there are constants $2P/3 \leq q \leq 3P$ and $K_2 = 4(\sqrt{2/3} + 1)/(1 - \sqrt{5/6})$ such that $V$ can be partitioned into $q$ sets, $A_1, A_2, \ldots, A_q$ such that for $1 \leq i \leq q$

$$c(V)/(3P) \leq c(A_i) \leq 3c(V)/(2P)$$

and there are sets $C_i$, $|C_i| \leq K_2\sqrt{N}$, and $B_i = V - A_i - C_i$ such that no edges join vertices in $A_i$ with vertices in $B_i$.

**Hint:** When $P = 2$, use the result of Problem 12.10 and combine the vertices of the separator with the other two sets to satisfy the necessary conditions. When $P > 2$, subdivide any set with cost exceeding $c(V)/P$ into two sets and a separator using the two-cost planar separator theorem. Assign vertices of the separator to these two sets to keep the cost in balance.

### THE PERFORMANCE OF VLSI ALGORITHMS

12.12 Show that the function defined by the product of three square matrices has a semellective planar circuit size that is quadratic in its number of variables and that it can be realized by a VLSI chip with $AT^2$ that meets the semellective planar circuit size lower bound.

12.13 Show that the wrapped convolution function $f^{(n)}_{\text{wrapped}} : \mathcal{R}^{2n} \mapsto \mathcal{R}^n$, can be realized as an embedded CCC network on a VLSI circuit with area $A$ and time $T$ satisfying $AT^2 = \Theta(n^2)$ for $\Omega(\log n) \leq T \leq \sqrt{n}$.

12.14 Design a VLSI chip for $n \times n$ matrix multiplication that achieves $AT^2 = n^4 \log^2 n$ for $T = O(\log n)$.

**Hint:** Represent each matrix as a $2 \times 2$ matrix of $(n/2) \times (n/2)$ matrices and use the standard algorithm that performs eight multiplications of $(n/2) \times (n/2)$ matrices. A multiplier has one side longer than the other. Place the long side of the $(n/2) \times (n/2)$ matrix multiplier at right angles to the long side of the $n \times n$ matrix multiplier. Apply this rule to the recursive construction of the multiplier.

12.15 Show that an algorithm of the kind described in Problem 12.14 can be combined with a mesh-based matrix multiplication algorithm of the kind described in Section 7.5.3 to produce a family of algorithms that achieve the lower bound on $n \times n$ matrix multiplication for $\Omega(\log n) \leq T \leq n$.

12.16 Devise a VLSI chip for $n$-bit integer multiplication function chip that uses area $A$ and time $T$ efficiently.

**Hint:** Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote binary numbers. Recursively form the product of these integers as the sum of two products, that of $\boldsymbol{x}$ with the high-order $(n/2)$ bits of $\boldsymbol{y}$ and that of $\boldsymbol{x}$ with the low-order $(n/2)$ bits of $\boldsymbol{y}$. Use carry-save addition where possible.

12.17 Give a proof of Theorem 12.7.4.

12.18 Show that the characteristic predicate of a function that has a $w(u, v)$-flow is $w(u, v)$-separated.

#### AREA BOUNDS

12.19 Show that any VLSI algorithm that realizes a superconcentrator on $n$ inputs requires area $\Theta(n)$.

## Chapter Notes

Mead and Conway wrote an influential book [212] that greatly simplified the design rules for VLSI chips and made VLSI design accessible to a large audience. Ullman [338] summarized the status of the field around 1984 and Lengauer [192] addressed the VLSI layout problem. Lengauer has also written a survey paper [193] that provides an overview of the theory of VLSI algorithms as of about 1990. The three transmission models described in Section 12.2 reflect the analysis of Zhou, Preparata, and Khang [371].

Thompson [325] obtained the first important tradeoff results for the VLSI model of computation. He demonstrated that under a suitable model a lower bound of $AT^2 = \Omega(n^2)$ could be derived for the discrete Fourier transform, a result he subsequently extended to sorting [326]. Generalizations of this model were made to convex chips [59], compact plane regions [194], and other closely related models [201]. Vuillemin [354] extended the models to include pipelining. Chazelle and Monier [67] introduced the transmission-line model described in Problems 12.1 and 12.2. For a discussion of other models that take into account the effects of distributed resistance, capacitance and inductance, see [40] and [371].

Systolic algorithms, which make good use of area and time, were popularized by Kung [176] and others (see, for example, [103,121,178,179,180,189]). The H-tree featured in Section 12.5.1 is due to Mead and Rem [213]. Prefix computations are discussed in Chapter 2. The cube-connected cycles network (its layout is given in Section 12.5.3) and the efficient

realization of normal algorithms are due to Preparata and Vuillemin [261], as explained in Chapter 7. Lengauer [192] provides an in-depth treatment of algorithms for VLSI chip layout.

Most authors prefer to derive lower bounds on $AT^2$ by partitioning the planar region occupied by chips [59,194,325]. In effect, they employ a physical version of the planar separator theorem. The characterization of VLSI lower bounds in terms of planar circuit complexity introduced by Savage [287] reinforces the connection between memoryless and memory-based computation explored in Chapter 3 but for planar computations by VLSI chips. It also provides an opportunity to introduce the elegant planar separator theorem of Lipton and Tarjan [202]. Lipton and Tarjan [203] developed quadratic lower bounds on the planar circuit size of shifting and matrix multiplication before the connection was established between VLSI complexity and planar circuit size. Improving upon results of [287], McColl [208] and McColl and Paterson [209] show that almost all Boolean functions on $n$ variables require a planar circuit size of $\Omega(2^n)$ and that this lower bound can be achieved for all functions to within a constant multiplicative factor close to 1. Turán [335] has shown that the upper bound of Lemma 12.6.1 is tight by exhibiting a family of functions of linear standard circuit size whose planar circuit size is quadratic.

Abelson [1] and Yao [365] studied communication complexity with fixed partitions. Yao [366] and Lipton and Sedgewick [201] made explicit the implicit connection between VLSI communication complexity and the derivation of the $AT^2$ lower bounds. (See also [235], [12], and [193] for a discussion of the conditions under which lower bounds can be derived on the VLSI communication complexity measure.)

Many authors have contributed to the derivation of semellective lower bounds for particular functions. Among these are Thompson [325,326,327,328], who obtained bounds of the form $AT^2 = \Omega(n^2)$ for the DFT and sorting, as did Abelson and Andreae [3] and Brent and Kung [59] for integer multiplication, JáJá and Kumar [148] for a variety of problems, Bilardi and Preparata [41] for sorting, Savage for matrix multiplication, inversion, and transitive closure [288] and binary integer powers and reciprocals [287], and Vuillemin for transitive functions [354] (see Problem 10.22). These authors generally show that the lower bounds for functions can be met either to within a small multiplicative constant factor.

Good VLSI designs have been given by Baudet, Preparata, and Vuillemin [31] for convolution, Guibas and Liang [122] for systolic stacks, queues, and counters, and Kung and Song [182] and Kung, Ruane, and Yen [181] on 2D convolution. Also, Luk and Vuillemin [206] give an optimal VLSI integer multiplier and Mehlhorn has provided optimal algorithms for integer division and square rooting [216] whose range of optimality has been extended by Mehlhorn and Preparata [218]. Preparata [257] has given a mesh-based optimal VLSI multiplier for large integers and Preparata and Vuillemin have given optimal algorithms for multiplying square [259] and triangular matrices [260]. C. Savage [283] has given a systolic algorithm for graph connectivity.

Lower bounds for the semellective computation of predicates by VLSI algorithms have been derived by Yao [366] for graph isomorphism, by Lipton and Sedgewick [201] for the recognition of context-free languages, pattern matching, and binary integer factorization testing, and by Savage [287] for the characteristic predicates of multi-output functions.

Hochschild [133], Kedem and Zorat [162,163], Savage [289,290], and Turán [336] have developed lower bounds on performance of multilective VLSI algorithms. Savage has explored multilective planar circuit size [290], giving a multi-output function with a $\Omega(n^{4/3})$ lower

bound. Turán [336] exhibits a function and a predicate with $\Omega(n^{3/2}\log n)$ and $\Omega(n\log n)$ lower bounds to their multilective planar circuit size, respectively. The $w(u,v)$-flow and $w(u,v)$-separated properties used in Section 12.7 were introduced in [290].

Lower bounds on the area of chips have been explored by a number of authors. Yao [366] examined addition; Baudet [30] studied functions that do not have a large information flow; Heintz [130] derived bounds for matrix-matrix multiplication; Leighton [190] introduced and used the crossing number of a graph to derive area bounds; Siegel [308] derived bounds for sorting; and Savage [287] examined functions with many subfunctions. Bilardi and Preparata [42] have generalized arguments of [30] and [151] to derive stronger area–time lower bounds for functions, such as prefix, for which the information flow arguments give weak results. Lower bounds on the area of multilective chips were obtained by Savage [290], Hromkovič [141,142], and Ďuriš and Galil [92].

Models for 3D VLSI chips, which are not yet a reality, have been introduced by Rosenberg [281,282] and studied by Preparata [262].