Portions from Application-Specific Integrated Circuits Copyright © 1997 by Addison Wesley Longman, Inc.

INTRODUCTION TO ASICs

An ASIC (pronounced "a-sick"; bold typeface defines a new term) is an application-specific integrated circuit -at least that is what the acronym stands for. Before we answer the question of what that means we first look at the evolution of the silicon chip or integrated circuit (IC).

Figure 1.1(a) shows an IC package (this is a pin-grid array, or PGA, shown upside down; the pins will go through holes in a printed-circuit board). People often call the package a chip, but, as you can see in Figure 1.1(b), the silicon chip itself (more properly called a die) is mounted in the cavity under the sealed lid. A PGA package is usually made from a ceramic material, but plastic packages are also common.

FIGURE 1.1 An integrated circuit (IC). (a) A pin-grid array (PGA) package. (b) The silicon die or chip is under the package lid.

The physical size of a silicon die varies from a few millimeters on a side to over 1 inch on a side, but instead we often measure the size of an IC by the number of logic gates or the number of transistors that the IC contains. As a unit of measure a gate equivalent corresponds to a two-input NAND gate (a circuit that performs the logic function, F = A * B). Often we just use the term gates instead of gate equivalents when we are measuring chip size-not to be confused with the gate terminal of a transistor. For example, a 100 k-gate IC contains the equivalent of 100,000 two-input NAND gates.

The semiconductor industry has evolved from the first ICs of the early 1970s and matured rapidly since then. Early small-scale integration (SSI) ICs contained a few (1 to 10) logic gates-NAND gates, NOR gates, and so on-amounting to a few tens of transistors. The era of medium-scale integration (MSI) increased the range of integrated logic available to counters and similar, larger scale, logic functions. The era of large-scale integration (LSI) packed even larger logic functions, such as the first microprocessors, into a single chip. The era of very large-scale integration (VLSI) now offers 64-bit microprocessors, complete with cache memory and floating-point arithmetic units-well over a million transistors-on a single piece of silicon. As CMOS process technology improves, transistors continue to get smaller and ICs hold more and more transistors. Some people (especially in Japan) use the term

ultralarge scale integration (ULSI), but most people stop at the term VLSI; otherwise we have to start inventing new words.

The earliest ICs used bipolar technology and the majority of logic ICs used either transistor-transistor logic (TTL) or emitter-coupled logic (ECL). Although invented before the bipolar transistor, the metal-oxide-silicon (MOS) transistor was initially difficult to manufacture because of problems with the oxide interface. As these problems were gradually solved, metal-gate n -channel MOS (nMOS or NMOS) technology developed in the 1970s. At that time MOS technology required fewer masking steps, was denser, and consumed less power than equivalent bipolar ICs. This meant that, for a given performance, an MOS IC was cheaper than a bipolar IC and led to investment and growth of the MOS IC market.

By the early 1980s the aluminum gates of the transistors were replaced by polysilicon gates, but the name MOS remained. The introduction of polysilicon as a gate material was a major improvement in CMOS technology, making it easier to make two types of transistors, n -channel MOS and p -channel MOS transistors, on the same IC-a complementary MOS (CMOS , never cMOS) technology. The principal advantage of CMOS over NMOS is lower power consumption. Another advantage of a polysilicon gate was a simplification of the fabrication process, allowing devices to be scaled down in size.

There are four CMOS transistors in a two-input NAND gate (and a two-input NOR gate too), so to convert between gates and transistors, you multiply the number of gates by 4 to obtain the number of transistors. We can also measure an IC by the smallest feature size (roughly half the length of the smallest transistor) imprinted on the IC. Transistor dimensions are measured in microns (a micron, 1 m m, is a millionth of a meter). Thus we talk about a 0.5 m m IC or say an IC is built in (or with) a 0.5 m m process, meaning that the smallest transistors are 0.5 m m in length. We give a special label, l or lambda , to this smallest feature size. Since lambda is equal to half of the smallest transistor length, l ª 0.25 m m in a 0.5 m m process. Many of the drawings in this book use a scale marked with lambda for the same reason we place a scale on a map.

A modern submicron CMOS process is now just as complicated as a submicron bipolar or BiCMOS (a combination of bipolar and CMOS) process. However, CMOS ICs have established a dominant position, are manufactured in much greater volume than any other technology, and therefore, because of the economy of scale, the cost of CMOS ICs is less than a bipolar or BiCMOS IC for the same function. Bipolar and BiCMOS ICs are still used for special needs. For example, bipolar technology is generally capable of handling higher voltages than CMOS. This makes bipolar and BiCMOS ICs useful in power electronics, cars, telephone circuits, and so on.

Some digital logic ICs and their analog counterparts (analog/digital converters, for example) are standard parts , or standard ICs. You can select standard ICs from catalogs and data books and buy them from distributors. Systems manufacturers and designers can use the same standard part in a variety of different microelectronic systems (systems that use microelectronics or ICs).

With the advent of VLSI in the 1980s engineers began to realize the advantages of designing an IC that was customized or tailored to a particular system or application rather than using standard ICs alone. Microelectronic system design then becomes a matter of defining the functions that you can implement using standard ICs and then implementing the remaining logic functions (sometimes called glue logic) with one or more custom ICs . As VLSI became possible you could build a system from a smaller

number of components by combining many standard ICs into a few custom ICs. Building a microelectronic system with fewer ICs allows you to reduce cost and improve reliability.

Of course, there are many situations in which it is not appropriate to use a custom IC for each and every part of an microelectronic system. If you need a large amount of memory, for example, it is still best to use standard memory ICs, either dynamic random-access memory (DRAM or dRAM), or static RAM (SRAM or sRAM), in conjunction with custom ICs.

One of the first conferences to be devoted to this rapidly emerging segment of the IC industry was the IEEE Custom Integrated Circuits Conference (CICC), and the proceedings of this annual conference form a useful reference to the development of custom ICs. As different types of custom ICs began to evolve for different types of applications, these new ICs gave rise to a new term: application-specific IC, or ASIC. Now we have the IEEE International ASIC Conference , which tracks advances in ASICs separately from other types of custom ICs. Although the exact definition of an ASIC is difficult, we shall look at some examples to help clarify what people in the IC industry understand by the term.

Examples of ICs that are not ASICs include standard parts such as: memory chips sold as a commodity item-ROMs, DRAM, and SRAM; microprocessors; TTL or TTL-equivalent ICs at SSI, MSI, and LSI levels.

Examples of ICs that are ASICs include: a chip for a toy bear that talks; a chip for a satellite; a chip designed to handle the interface between memory and a microprocessor for a workstation CPU; and a chip containing a microprocessor as a cell together with other logic.

As a general rule, if you can find it in a data book, then it is probably not an ASIC, but there are some exceptions. For example, two ICs that might or might not be considered ASICs are a controller chip for a PC and a chip for a modem. Both of these examples are specific to an application (shades of an ASIC) but are sold to many different system vendors (shades of a standard part). ASICs such as these are sometimes called application-specific standard products (ASSPs).

Trying to decide which members of the huge IC family are application-specific is tricky-after all, every IC has an application. For example, people do not usually consider an application-specific microprocessor to be an ASIC. I shall describe how to design an ASIC that may include large cells such as microprocessors, but I shall not describe the design of the microprocessors themselves. Defining an ASIC by looking at the application can be confusing, so we shall look at a different way to categorize the IC family. The easiest way to recognize people is by their faces and physical characteristics: tall, short, thin. The easiest characteristics of ASICs to understand are physical ones too, and we shall look at these next. It is important to understand these differences because they affect such factors as the price of an ASIC and the way you design an ASIC.

1.1 Types of ASICs

1.2 Design Flow

1.3 Case Study

1.4 Economics of ASICs

1.5 ASIC Cell Libraries

1.6 Summary

1.7 Problems

1.8 Bibliography

1.9 References

1.1 Types of ASICs

ICs are made on a thin (a few hundred microns thick), circular silicon wafer , with each wafer holding hundreds of die (sometimes people use dies or dice for the plural of die). The transistors and wiring are made from many layers (usually between 10 and 15 distinct layers) built on top of one another. Each successive mask layer has a pattern that is defined using a mask similar to a glass photographic slide. The first half-dozen or so layers define the transistors. The last half-dozen or so layers define the metal wires between the transistors (the interconnect).

A full-custom IC includes some (possibly all) logic cells that are customized and all mask layers that are customized. A microprocessor is an example of a full-custom IC-designers spend many hours squeezing the most out of every last square micron of microprocessor chip space by hand. Customizing all of the IC features in this way allows designers to include analog circuits, optimized memory cells, or mechanical structures on an IC, for example. Full-custom ICs are the most expensive to manufacture and to design. The manufacturing lead time (the time it takes just to make an IC-not including design time) is typically eight weeks for a full-custom IC. These specialized full-custom ICs are often intended for a specific application, so we might call some of them full-custom ASICs.

We shall discuss full-custom ASICs briefly next, but the members of the IC family that we are more interested in are semicustom ASICs , for which all of the logic cells are predesigned and some (possibly all) of the mask layers are customized. Using predesigned cells from a cell library makes our lives as designers much, much easier. There are two types of semicustom ASICs that we shall cover: standard-cell-based ASICs and gate-array-based ASICs. Following this we shall describe the programmable ASICs , for which all of the logic cells are predesigned and none of the mask layers are

customized. There are two types of programmable ASICs: the programmable logic device and, the newest member of the ASIC family, the field-programmable gate array.

1.1.1 Full-Custom ASICs

In a full-custom ASIC an engineer designs some or all of the logic cells, circuits, or layout specifically for one ASIC. This means the designer abandons the approach of using pretested and precharacterized cells for all or part of that design. It makes sense to take this approach only if there are no suitable existing cell libraries available that can be used for the entire design. This might be because existing cell libraries are not fast enough, or the logic cells are not small enough or consume too much power. You may need to use full-custom design if the ASIC technology is new or so specialized that there are no existing cell libraries or because the ASIC is so specialized that some circuits must be custom designed. Fewer and fewer full-custom ICs are being designed because of the problems with these special parts of the ASIC. There is one growing member of this family, though, the mixed analog/digital ASIC, which we shall discuss next.

Bipolar technology has historically been used for precision analog functions. There are some fundamental reasons for this. In all integrated circuits the matching of component characteristics between chips is very poor, while the matching of characteristics between components on the same chip is excellent. Suppose we have transistors T1, T2, and T3 on an analog/digital ASIC. The three transistors are all the same size and are constructed in an identical fashion. Transistors T1 and T2 are located adjacent to each other and have the same orientation. Transistor T3 is the same size as T1 and T2 but is located on the other side of the chip from T1 and T2 and has a different orientation. ICs are made in batches called wafer lots. A wafer lot is a group of silicon wafers that are all processed together. Usually there are between 5 and 30 wafers in a lot. Each wafer can contain tens or hundreds of chips depending on the size of the IC and the wafer.

If we were to make measurements of the characteristics of transistors T1, T2, and T3 we would find the following:

- Transistors T1 will have virtually identical characteristics to T2 on the same IC. We say that the transistors match well or the tracking between devices is excellent.
- Transistor T3 will match transistors T1 and T2 on the same IC very well, but not as closely as T1 matches T2 on the same IC.
- Transistor T1, T2, and T3 will match fairly well with transistors T1, T2, and T3 on a different IC on the same wafer. The matching will depend on how far apart the two ICs are on the wafer.
- Transistors on ICs from different wafers in the same wafer lot will not match very well.
- Transistors on ICs from different wafer lots will match very poorly.

For many analog designs the close matching of transistors is crucial to circuit operation. For these circuit designs pairs of transistors are used, located adjacent to each other. Device physics dictates that a pair of bipolar transistors will always match more precisely than CMOS transistors of a comparable size. Bipolar technology has historically been more widely used for full-custom analog design because of its improved precision. Despite its poorer analog properties, the use of CMOS technology for analog functions is increasing. There are two reasons for this. The first reason is that CMOS is now by far the most widely available IC technology. Many more CMOS ASICs and CMOS standard products are now being manufactured than bipolar ICs. The second reason is that increased levels of integration require mixing analog and digital functions on the same IC: this has forced designers to find ways to use CMOS

technology to implement analog functions. Circuit designers, using clever new techniques, have been very successful in finding new ways to design analog CMOS circuits that can approach the accuracy of bipolar analog designs.

1.1.2 Standard-Cell-Based ASICs

A cell-based ASIC (cell-based IC, or CBIC -a common term in Japan, pronounced "sea-bick") uses predesigned logic cells (AND gates, OR gates, multiplexers, and flip-flops, for example) known as standard cells . We could apply the term CBIC to any IC that uses cells, but it is generally accepted that a cell-based ASIC or CBIC means a standard-cell-based ASIC.

The standard-cell areas (also called flexible blocks) in a CBIC are built of rows of standard cells-like a wall built of bricks. The standard-cell areas may be used in combination with larger predesigned cells, perhaps microcontrollers or even microprocessors, known as megacells . Megacells are also called megafunctions, full-custom blocks, system-level macros (SLMs), fixed blocks, cores, or Functional Standard Blocks (FSBs).

The ASIC designer defines only the placement of the standard cells and the interconnect in a CBIC. However, the standard cells can be placed anywhere on the silicon; this means that all the mask layers of a CBIC are customized and are unique to a particular customer. The advantage of CBICs is that designers save time, money, and reduce risk by using a predesigned, pretested, and precharacterized standard-cell library . In addition each standard cell can be optimized individually. During the design of the cell library each and every transistor in every standard cell can be chosen to maximize speed or minimize area, for example. The disadvantages are the time or expense of designing or buying the standard-cell library and the time needed to fabricate all layers of the ASIC for each new design.

Figure 1.2 shows a CBIC (looking down on the die shown in Figure 1.1b, for example). The important features of this type of ASIC are as follows:

- All mask layers are customized-transistors and interconnect.
- Custom blocks can be embedded.
- Manufacturing lead time is about eight weeks.

FIGURE 1.2 A cell-based ASIC (CBIC) die with a single standard-cell area (a flexible block) together with four fixed blocks. The flexible block contains rows of standard cells. This is what you might see through a low-powered microscope looking down on the die of Figure 1.1(b). The small squares around the edge of the die are bonding pads that are connected to the pins of the ASIC package.

Each standard cell in the library is constructed using full-custom design methods, but you can use these predesigned and precharacterized circuits without having to do any full-custom design yourself. This design style gives you the same performance and flexibility advantages of a full-custom ASIC but

reduces design time and reduces risk.

Standard cells are designed to fit together like bricks in a wall. Figure 1.3 shows an example of a simple standard cell (it is simple in the sense it is not maximized for density-but ideal for showing you its internal construction). Power and ground buses (VDD and GND or VSS) run horizontally on metal lines inside the cells.

FIGURE 1.3 Looking down on the layout of a standard cell. This cell would be approximately 25 microns wide on an ASIC with l (lambda) = 0.25 microns (a micron is 10^{-6} m). Standard cells are stacked like bricks in a wall; the abutment box (AB) defines the "edges" of the brick. The difference between the bounding box (BB) and the AB is the area of overlap between the bricks. Power supplies (labeled VDD and GND) run horizontally inside a standard cell on a metal layer that lies above the transistor layers. Each different shaded and labeled pattern represents a different layer. This standard cell has center connectors (the three squares, labeled A1, B1, and Z) that allow the cell to connect to others. The layout was drawn using ROSE, a symbolic layout editor developed by Rockwell and Compass, and then imported into Tanner Research's L-Edit.

Standard-cell design allows the automation of the process of assembling an ASIC. Groups of standard cells fit horizontally together to form rows. The rows stack vertically to form flexible rectangular blocks (which you can reshape during design). You may then connect a flexible block built from several rows of standard cells to other standard-cell blocks or other full-custom logic blocks. For example, you might want to include a custom interface to a standard, predesigned microcontroller together with some memory. The microcontroller block may be a fixed-size megacell, you might generate the memory using a memory compiler, and the custom logic and memory controller will be built from flexible standard-cell blocks, shaped to fit in the empty spaces on the chip.

Both cell-based and gate-array ASICs use predefined cells, but there is a difference-we can change the transistor sizes in a standard cell to optimize speed and performance, but the device sizes in a gate array are fixed. This results in a trade-off in performance and area in a gate array at the silicon level. The trade-off between area and performance is made at the library level for a standard-cell ASIC.

Modern CMOS ASICs use two, three, or more levels (or layers) of metal for interconnect. This allows wires to cross over different layers in the same way that we use copper traces on different layers on a printed-circuit board. In a two-level metal CMOS technology, connections to the standard-cell inputs and outputs are usually made using the second level of metal (metal2 , the upper level of metal) at the tops and bottoms of the cells. In a three-level metal technology, connections may be internal to the logic cell (as they are in Figure 1.3). This allows for more sophisticated routing programs to take advantage of the extra metal layer to route interconnect over the top of the logic cells. We shall cover the details of routing ASICs in Chapter 17.

A connection that needs to cross over a row of standard cells uses a feedthrough. The term feedthrough can refer either to the piece of metal that is used to pass a signal through a cell or to a space in a cell waiting to be used as a feedthrough-very confusing. Figure 1.4 shows two feedthroughs: one in cell A.14 and one in cell A.23.

In both two-level and three-level metal technology, the power buses (VDD and GND) inside the standard cells normally use the lowest (closest to the transistors) layer of metal (metal1). The width of each row of standard cells is adjusted so that they may be aligned using spacer cells . The power buses, or rails, are then connected to additional vertical power rails using row-end cells at the aligned ends of each standard-cell block. If the rows of standard cells are long, then vertical power rails can also be run in metal2 through the cell rows using special power cells that just connect to VDD and GND. Usually the designer manually controls the number and width of the vertical power rails connected to the standard-cell blocks during physical design. A diagram of the power distribution scheme for a CBIC is shown in Figure 1.4.

FIGURE 1.4 Routing the CBIC (cell-based IC) shown in Figure 1.2. The use of regularly shaped standard cells, such as the one in Figure 1.3, from a library allows ASICs like this to be designed automatically. This ASIC uses two separate layers of metal interconnect (metal1 and metal2) running at right angles to each other (like traces on a printed-circuit board). Interconnections between logic cells uses spaces (called channels) between the rows of cells. ASICs may have three (or more) layers of metal allowing the cell rows to touch with the interconnect running over the top of the cells.

All the mask layers of a CBIC are customized. This allows megacells (SRAM, a SCSI controller, or an MPEG decoder, for example) to be placed on the same IC with standard cells. Megacells are usually

supplied by an ASIC or library company complete with behavioral models and some way to test them (a test strategy). ASIC library companies also supply compilers to generate flexible DRAM, SRAM, and ROM blocks. Since all mask layers on a standard-cell design are customized, memory design is more efficient and denser than for gate arrays.

For logic that operates on multiple signals across a data bus-a datapath (DP)-the use of standard cells may not be the most efficient ASIC design style. Some ASIC library companies provide a datapath compiler that automatically generates datapath logic . A datapath library typically contains cells such as adders, subtracters, multipliers, and simple arithmetic and logical units (ALUs). The connectors of datapath library cells are pitch-matched to each other so that they fit together. Connecting datapath cells to form a datapath usually, but not always, results in faster and denser layout than using standard cells or a gate array.

Standard-cell and gate-array libraries may contain hundreds of different logic cells, including combinational functions (NAND, NOR, AND, OR gates) with multiple inputs, as well as latches and flip-flops with different combinations of reset, preset and clocking options. The ASIC library company provides designers with a data book in paper or electronic form with all of the functional descriptions and timing information for each library element.

1.1.3 Gate-Array-Based ASICs

In a gate array (sometimes abbreviated to GA) or gate-array-based ASIC the transistors are predefined on the silicon wafer. The predefined pattern of transistors on a gate array is the base array , and the smallest element that is replicated to make the base array (like an M. C. Escher drawing, or tiles on a floor) is the base cell (sometimes called a primitive cell). Only the top few layers of metal, which define the interconnect between transistors, are defined by the designer using custom masks. To distinguish this type of gate array from other types of gate array, it is often called a masked gate array (MGA). The designer chooses from a gate-array library of predesigned and precharacterized logic cells. The logic cells in a gate-array library are often called macros . The reason for this is that the base-cell layout is the same for each logic cell, and only the interconnect (inside cells and between cells) is customized, so that there is a similarity between gate-array macros and a software macro. Inside IBM, gate-array macros are known as books (so that books are part of a library), but unfortunately this descriptive term is not very widely used outside IBM.

We can complete the diffusion steps that form the transistors and then stockpile wafers (sometimes we call a gate array a prediffused array for this reason). Since only the metal interconnections are unique to an MGA, we can use the stockpiled wafers for different customers as needed. Using wafers prefabricated up to the metallization steps reduces the time needed to make an MGA, the turnaround time , to a few days or at most a couple of weeks. The costs for all the initial fabrication steps for an MGA are shared for each customer and this reduces the cost of an MGA compared to a full-custom or standard-cell ASIC design.

There are the following different types of MGA or gate-array-based ASICs:

- Channeled gate arrays.
- Channelless gate arrays.
- Structured gate arrays.

The hyphenation of these terms when they are used as adjectives explains their construction. For example, in the term "channeled gate-array architecture," the gate array is channeled , as will be explained. There are two common ways of arranging (or arraying) the transistors on a MGA: in a channeled gate array we leave space between the rows of transistors for wiring; the routing on a channelless gate array uses rows of unused transistors. The channeled gate array was the first to be developed, but the channelless gate-array architecture is now more widely used. A structured (or embedded) gate array can be either channeled or channelless but it includes (or embeds) a custom block.

1.1.4 Channeled Gate Array

Figure 1.5 shows a channeled gate array . The important features of this type of MGA are:

- Only the interconnect is customized.
- The interconnect uses predefined spaces between rows of base cells.
- Manufacturing lead time is between two days and two weeks.

FIGURE 1.5 A channeled gate-array die. The spaces between rows of the base cells are set aside for interconnect.

A channeled gate array is similar to a CBIC-both use rows of cells separated by channels used for interconnect. One difference is that the space for interconnect between rows of cells are fixed in height in a channeled gate array, whereas the space between rows of cells may be adjusted in a CBIC.

1.1.5 Channelless Gate Array

Figure 1.6 shows a channelless gate array (also known as a channel-free gate array , sea-of-gates array , or SOG array). The important features of this type of MGA are as follows:

- Only some (the top few) mask layers are customized-the interconnect.
- Manufacturing lead time is between two days and two weeks.

FIGURE 1.6 A channelless gate-array or sea-of-gates (SOG) array die. The core area of the die is completely filled with an array of base cells (the base array).

The key difference between a channelless gate array and channeled gate array is that there are no predefined areas set aside for routing between cells on a channelless gate array. Instead we route over the top of the gate-array devices. We can do this because we customize the contact layer that defines the connections between metal1, the first layer of metal, and the transistors. When we use an area of transistors for routing in a channelless array, we do not make any contacts to the devices lying underneath; we simply leave the transistors unused.

The logic density-the amount of logic that can be implemented in a given silicon area-is higher for channelless gate arrays than for channeled gate arrays. This is usually attributed to the difference in structure between the two types of array. In fact, the difference occurs because the contact mask is customized in a channelless gate array, but is not usually customized in a channeled gate array. This leads to denser cells in the channelless architectures. Customizing the contact layer in a channelless gate array allows us to increase the density of gate-array cells because we can route over the top of unused contact sites.

1.1.6 Structured Gate Array

An embedded gate array or structured gate array (also known as masterslice or masterimage) combines some of the features of CBICs and MGAs. One of the disadvantages of the MGA is the fixed gate-array base cell. This makes the implementation of memory, for example, difficult and inefficient. In an embedded gate array we set aside some of the IC area and dedicate it to a specific function. This embedded area either can contain a different base cell that is more suitable for building memory cells, or it can contain a complete circuit block, such as a microcontroller.

Figure 1.7 shows an embedded gate array. The important features of this type of MGA are the following:

- Only the interconnect is customized.
- Custom blocks (the same for each design) can be embedded.
- Manufacturing lead time is between two days and two weeks.

FIGURE 1.7 A structured or embedded gate-array die showing an embedded block in the upper left corner (a static random-access memory, for example). The rest of the die is filled with an array of base cells.

An embedded gate array gives the improved area efficiency and increased performance of a CBIC but with the lower cost and faster turnaround of an MGA. One disadvantage of an embedded gate array is that the embedded function is fixed. For example, if an embedded gate array contains an area set aside for a 32 k-bit memory, but we only need a 16 k-bit memory, then we may have to waste half of the embedded memory function. However, this may still be more efficient and cheaper than implementing a 32 k-bit memory using macros on a SOG array.

ASIC vendors may offer several embedded gate array structures containing different memory types and sizes as well as a variety of embedded functions. ASIC companies wishing to offer a wide range of embedded functions must ensure that enough customers use each different embedded gate array to give the cost advantages over a custom gate array or CBIC (the Sun Microsystems SPARCstation 1 described in Section 1.3 made use of LSI Logic embedded gate arrays-and the 10K and 100K series of embedded gate arrays were two of LSI Logic's most successful products).

1.1.7 Programmable Logic Devices

Programmable logic devices (PLDs) are standard ICs that are available in standard configurations from a catalog of parts and are sold in very high volume to many different customers. However, PLDs may be configured or programmed to create a part customized to a specific application, and so they also belong to the family of ASICs. PLDs use different technologies to allow programming of the device. Figure 1.8 shows a PLD and the following important features that all PLDs have in common:

- No customized mask layers or logic cells
- Fast design turnaround
- A single large block of programmable interconnect
- A matrix of logic macrocells that usually consist of programmable array logic followed by a flip-flop or latch

FIGURE 1.8 A programmable logic device (PLD) die. The macrocells typically consist of programmable array logic followed by a flip-flop or latch. The macrocells are connected using a large programmable interconnect block.

The simplest type of programmable IC is a read-only memory (ROM). The most common types of ROM use a metal fuse that can be blown permanently (a programmable ROM or PROM). An electrically programmable ROM , or EPROM , uses programmable MOS transistors whose characteristics are altered by applying a high voltage. You can erase an EPROM either by using another high voltage (an electrically erasable PROM , or EEPROM) or by exposing the device to ultraviolet light (UV-erasable PROM , or UVPROM).

There is another type of ROM that can be placed on any ASIC-a mask-programmable ROM (mask-programmed ROM or masked ROM). A masked ROM is a regular array of transistors permanently programmed using custom mask patterns. An embedded masked ROM is thus a large, specialized, logic cell.

The same programmable technologies used to make ROMs can be applied to more flexible logic structures. By using the programmable devices in a large array of AND gates and an array of OR gates, we create a family of flexible and programmable logic devices called logic arrays . The company Monolithic Memories (bought by AMD) was the first to produce Programmable Array Logic (PAL ®, a registered trademark of AMD) devices that you can use, for example, as transition decoders for state

machines. A PAL can also include registers (flip-flops) to store the current state information so that you can use a PAL to make a complete state machine.

Just as we have a mask-programmable ROM, we could place a logic array as a cell on a custom ASIC. This type of logic array is called a programmable logic array (PLA). There is a difference between a PAL and a PLA: a PLA has a programmable AND logic array, or AND plane , followed by a programmable OR logic array, or OR plane ; a PAL has a programmable AND plane and, in contrast to a PLA, a fixed OR plane.

Depending on how the PLD is programmed, we can have an erasable PLD (EPLD), or mask-programmed PLD (sometimes called a masked PLD but usually just PLD). The first PALs, PLAs, and PLDs were based on bipolar technology and used programmable fuses or links. CMOS PLDs usually employ floating-gate transistors (see Section 4.3, "EPROM and EEPROM Technology").

1.1.8 Field-Programmable Gate Arrays

A step above the PLD in complexity is the field-programmable gate array (FPGA). There is very little difference between an FPGA and a PLD-an FPGA is usually just larger and more complex than a PLD. In fact, some companies that manufacture programmable ASICs call their products FPGAs and some call them complex PLDs . FPGAs are the newest member of the ASIC family and are rapidly growing in importance, replacing TTL in microelectronic systems. Even though an FPGA is a type of gate array, we do not consider the term gate-array-based ASICs to include FPGAs. This may change as FPGAs and MGAs start to look more alike.

Figure 1.9 illustrates the essential characteristics of an FPGA:

- None of the mask layers are customized.
- A method for programming the basic logic cells and the interconnect.
- The core is a regular array of programmable basic logic cells that can implement combinational as well as sequential logic (flip-flops).
- A matrix of programmable interconnect surrounds the basic logic cells.
- Programmable I/O cells surround the core.
- Design turnaround is a few hours.

We shall examine these features in detail in Chapters 4-8.

1.2 Design Flow

Figure 1.10 shows the sequence of steps to design an ASIC; we call this a design flow . The steps are listed below (numbered to correspond to the labels in Figure 1.10) with a brief description of the function of each step.

FIGURE 1.10 ASIC design flow.

1. Design entry. Enter the design into an ASIC design system, either using a hardware description language (HDL) or schematic entry .
2. Logic synthesis. Use an HDL (VHDL or Verilog) and a logic synthesis tool to produce a netlist -a description of the logic cells and their connections.
3. System partitioning. Divide a large system into ASIC-sized pieces.
4. Prelayout simulation. Check to see if the design functions correctly.
5. Floorplanning. Arrange the blocks of the netlist on the chip.
6. Placement. Decide the locations of cells in a block.
7. Routing. Make the connections between cells and blocks.
8. Extraction. Determine the resistance and capacitance of the interconnect.
9. Postlayout simulation. Check to see the design still works with the added loads of the interconnect.

Steps 1-4 are part of logical design , and steps 5-9 are part of physical design

1.3 Case Study

Sun Microsystems released the SPARCstation 1 in April 1989. It is now an old design but a very important example because it was one of the first workstations to make extensive use of ASICs to achieve the following:

- Better performance at lower cost
- Compact size, reduced power, and quiet operation
- Reduced number of parts, easier assembly, and improved reliability

The SPARCstation 1 contains about 50 ICs on the system motherboard-excluding the DRAM used for the system memory (standard parts). The SPARCstation 1 designers partitioned the system into the nine

ASlCs shown in Table 1.1 and wrote specifications for each ASIC-this took about three months 1 . LSI Logic and Fujitsu designed the SPARC integer unit (IU) and floating-point unit (FPU) to these specifications. The clock ASIC is a fairly straightforward design and, of the six remaining ASICs, the video controller/data buffer, the RAM controller, and the direct memory access (DMA) controller are defined by the 32-bit system bus (SBus) and the other ASICs that they connect to. The rest of the system is partitioned into three more ASICs: the cache controller , memory-management unit (MMU), and the data buffer. These three ASICs, with the IU and FPU, have the most critical timing paths and determine the system partitioning. The design of ASICs 3-8 in Table 1.1 took five Sun engineers six months after the specifications were complete. During the design process, the Sun engineers simulated the entire SPARCstation 1-including execution of the Sun operating system (SunOS).

TABLE 1.1 The ASICs in the Sun Microsystems SPARCstation 1.

SPARCstation 1 ASIC	Gates (k-gates)
1 SPARC integer unit (IU)	20
2 SPARC floating-point unit (FPU)	50
3 Cache controller	9
4 Memory-management unit (MMU)	5
5 Data buffer	3
6 Direct memory access (DMA) controller	9
7 Video controller/data buffer	4
8 RAM controller	1
9 Clock generator	1

Table 1.2 shows the software tools used to design the SPARCstation 1, many of which are now obsolete. The important point to notice, though, is that there is a lot more to microelectronic system design than designing the ASICs-less than one-third of the tools listed in Table 1.2 were ASIC design tools.

TABLE 1.2 The CAD tools used in the design of the Sun Microsystems SPARCstation 1.

Design level	Function	Tool 2
ASIC design	ASIC physical design	LSI Logic
	ASIC logic synthesis	Internal tools and UC Berkeley tools
	ASIC simulation	LSI Logic
Board design	Schematic capture	Valid Logic
	PCB layout	Valid Logic Allegro
	Timing verification	Quad Design Motive and internal tools
Mechanical design	Case and enclosure	Autocad
	Thermal analysis	Pacific Numerix
	Structural analysis	Cosmos
Management	Scheduling	Suntrac
	Documentation	Interleaf and FrameMaker

The SPARCstation 1 cost about $9000 in 1989 or, since it has an execution rate of approximately 12 million instructions per second (MIPS), $750/MIPS. Using ASIC technology reduces the motherboard to about the size of a piece of paper-8.5 inches by 11 inches-with a power consumption of about 12 W. The SPARCstation 1 "pizza box" is 16 inches across and 3 inches high-smaller than a typical IBM-compatible personal computer in 1989. This speed, power, and size performance is (there are still SPARCstation 1s in use) made possible by using ASICs. We shall return to the SPARCstation 1, to look more closely at the partitioning step, in Section 15.3, "System Partitioning."

---

1. Some information in Section 1.3 and Section 15.3 is from the SPARCstation 10 Architecture Guide-May 1992, p. 2 and pp. 27-28 and from two publicity brochures (known as "sparkle sheets"). The first is "Concept to System: How Sun Microsystems Created SPARCstation 1 Using LSI Logic's ASIC System Technology," A. Bechtolsheim, T. Westberg, M. Insley, and J. Ludemann of Sun Microsystems; J-H. Huang and D. Boyle of LSI Logic. This is an LSI Logic publication. The second paper is "SPARCstation 1: Beyond the 3M Horizon," A. Bechtolsheim and E. Frank, a Sun Microsystems publication. I did not include these as references since they are impossible to obtain now, but I would like to give credit to Andy Bechtolsheim and the Sun Microsystems and LSI Logic engineers.

2. Names are trademarks of their respective companies.

1.4 Economics of ASICs

In this section we shall discuss the economics of using ASICs in a product and compare the most popular types of ASICs: an FPGA, an MGA, and a CBIC. To make an economic comparison between these alternatives, we consider the ASIC itself as a product and examine the components of product cost: fixed costs and variable costs. Making cost comparisons is dangerous-costs change rapidly and the semiconductor industry is notorious for keeping its costs, prices, and pricing strategy closely guarded secrets. The figures in the following sections are approximate and used to illustrate the different components of cost.

1.4.1 Comparison Between ASIC Technologies

The most obvious economic factor in making a choice between the different ASIC types is the part cost . Part costs vary enormously-you can pay anywhere from a few dollars to several hundreds of dollars for an ASIC. In general, however, FPGAs are more expensive per gate than MGAs, which are, in turn, more expensive than CBICs. For example, a 0.5 m m, 20 k-gate array might cost 0.01-0.02 cents/gate (for more than 10,000 parts) or $2-$4 per part, but an equivalent FPGA might be $20. The price per gate for an FPGA to implement the same function is typically 2-5 times the cost of an MGA or CBIC.

Given that an FPGA is more expensive than an MGA, which is more expensive than a CBIC, when and why does it make sense to choose a more expensive part? Is the increased flexibility of an FPGA worth the extra cost per part? Given that an MGA or CBIC is specially tailored for each customer, there are extra hidden costs associated with this step that we should consider. To make a true comparison between the different ASIC technologies, we shall quantify some of these costs.

1.4.2 Product Cost

The total cost of any product can be separated into fixed costs and variable costs :

total product cost = fixed product cost + variable product cost ¥ products sold (1.1)

Fixed costs are independent of sales volume -the number of products sold. However, the fixed costs amortized per product sold (fixed costs divided by products sold) decrease as sales volume increases. Variable costs include the cost of the parts used in the product, assembly costs, and other manufacturing costs.

Let us look more closely at the parts in a product. If we want to buy ASICs to assemble our product, the total part cost is

total part cost = fixed part cost + variable cost per part ¥ volume of parts. (1.2)

Our fixed cost when we use an FPGA is low-we just have to buy the software and any programming equipment. The fixed part costs for an MGA or CBIC are higher and include the costs of the masks, simulation, and test program development. We shall discuss these extra costs in more detail in Sections 1.4.3 and 1.4.4. Figure 1.11 shows a break-even graph that compares the total part cost for an FPGA, MGA, and a CBIC with the following assumptions:

- FPGA fixed cost is $21,800, part cost is $39.
- MGA fixed cost is $86,000, part cost is $10.
- CBIC fixed cost is $146,000, part cost is $8.

At low volumes, the MGA and the CBIC are more expensive because of their higher fixed costs. The total part costs of two alternative types of ASIC are equal at the break-even volume . In Figure 1.11 the break-even volume for the FPGA and the MGA is about 2000 parts. The break-even volume between the FPGA and the CBIC is about 4000 parts. The break-even volume between the MGA and the CBIC is higher-at about 20,000 parts.

FIGURE 1.11 A break-even analysis for an FPGA, a masked gate array (MGA) and a custom cell-based ASIC (CBIC). The break-even volume between two technologies is the point at which the

total cost of parts are equal. These numbers are very approximate.

We shall describe how to calculate the fixed part costs next. Following that we shall discuss how we came up with cost per part of $39, $10, and $8 for the FPGA, MGA, and CBIC.

1.4.3 ASIC Fixed Costs

Figure 1.12 shows a spreadsheet, "Fixed Costs," that calculates the fixed part costs associated with ASIC design.

	FPGA		MGA		CBIC	
Training:	$800		$2,000		$2,000	
Days		2		5		5
Cost/day		$400		$400		$400
Hardware	$10,000		$10,000		$10,000	
Software	$1,000		$20,000		$40,000	
Design:	$8,000		$20,000		$20,000	
Size (gates)		10,000		10,000		10,000
Gates/day		500		200		200
Days		20		50		50
Cost/day		$400		$400		$400
Design for test:			$2,000		$2,000	
Days				5		5
Cost/day				$400		$400
NRE:			$30,000		$70,000	
Masks				$10,000		$50,000
Simulation				$10,000		$10,000
Test program				$10,000		$10,000
Second source:	$2,000		$2,000		$2,000	
Days		5		5		5
Cost/day		$400		$400		$400
Total fixed costs	$21,800		$86,000		$146,000	

FIGURE 1.12 A spreadsheet, "Fixed Costs," for a field-programmable gate array (FPGA), a masked gate array (MGA), and a cell-based ASIC (CBIC). These costs can vary wildly.

The training cost includes the cost of the time to learn any new electronic design automation (EDA) system. For example, a new FPGA design system might require a few days to learn; a new gate-array or cell-based design system might require taking a course. Figure 1.12 assumes that the cost of an engineer (including overhead, benefits, infrastructure, and so on) is between $100,000 and $200,000 per year or $2000 to $4000 per week (in the United States in 1990s dollars).

Next we consider the hardware and software cost for ASIC design. Figure 1.12 shows some typical figures, but you can spend anywhere from $1000 to $1 million (and more) on ASIC design software and the necessary infrastructure.

We try to measure productivity of an ASIC designer in gates (or transistors) per day. This is like trying to predict how long it takes to dig a hole, and the number of gates per day an engineer averages varies wildly. ASIC design productivity must increase as ASIC sizes increase and will depend on experience, design tools, and the ASIC complexity. If we are using similar design methods, design productivity ought to be independent of the type of ASIC, but FPGA design software is usually available as a complete bundle on a PC. This means that it is often easier to learn and use than semicustom ASIC design tools.

Every ASIC has to pass a production test to make sure that it works. With modern test tools the generation of any test circuits on each ASIC that are needed for production testing can be automatic, but it still involves a cost for design for test . An FPGA is tested by the manufacturer before it is sold to you and before you program it. You are still paying for testing an FPGA, but it is a hidden cost folded into the part cost of the FPGA. You do have to pay for any programming costs for an FPGA, but we can include these in the hardware and software cost.

The nonrecurring-engineering (NRE) charge includes the cost of work done by the ASIC vendor and the cost of the masks. The production test uses sets of test inputs called test vectors , often many thousands of them. Most ASIC vendors require simulation to generate test vectors and test programs for production testing, and will charge for a test-program development cost . The number of masks required by an ASIC during fabrication can range from three or four (for a gate array) to 15 or more (for a CBIC). Total mask costs can range from $5000 to $50,000 or more. The total NRE charge can range from $10,000 to $300,000 or more and will vary with volume and the size of the ASIC. If you commit to high volumes (above 100,000 parts), the vendor may waive the NRE charge. The NRE charge may also include the costs of software tools, design verification, and prototype samples.

If your design does not work the first time, you have to complete a further design pass (turn or spin) that requires additional NRE charges. Normally you sign a contract (sign off a design) with an ASIC vendor that guarantees first-pass success-this means that if you designed your ASIC according to rules specified by the vendor, then the vendor guarantees that the silicon will perform according to the simulation or you get your money back. This is why the difference between semicustom and full-custom design styles is so important-the ASIC vendor will not (and cannot) guarantee your design will work if you use any full-custom design techniques.

Nowadays it is almost routine to have an ASIC work on the first pass. However, if your design does fail, it is little consolation to have a second pass for free if your company goes bankrupt in the meantime. Figure 1.13 shows a profit model that represents the profit flow during the product lifetime . Using this model, we can estimate the lost profit due to any delay.

FIGURE 1.13 A profit model. If a product is introduced on time, the total sales are $60 million (the area of the higher triangle). With a three-month (one fiscal quarter) delay the sales decline to $25 million. The difference is shown as the shaded area between the two triangles and amounts to a lost revenue of $35 million.

Suppose we have the following situation:

- The product lifetime is 18 months (6 fiscal quarters).
- The product sales increase (linearly) at $10 million per quarter independently of when the product is introduced (we suppose this is because we can increase production and sales only at a fixed rate).
- The product reaches its peak sales at a point in time that is independent of when we introduce a product (because of external market factors that we cannot control).
- The product declines in sales (linearly) to the end of its life-a point in time that is also independent of when we introduce the product (again due to external market forces).

The simple profit and revenue model of Figure 1.13 shows us that we would lose $35 million in sales in this situation due to a 3-month delay. Despite the obvious problems with such a simple model (how can we introduce the same product twice to compare the performance?), it is widely used in marketing. In the electronics industry product lifetimes continue to shrink. In the PC industry it is not unusual to have a product lifetime of 18 months or less. This means that it is critical to achieve a rapid design time (or high product velocity) with no delays.

The last fixed cost shown in Figure 1.12 corresponds to an "insurance policy." When a company buys an ASIC part, it needs to be assured that it will always have a back-up source, or second source , in case something happens to its first or primary source. Established FPGA companies have a second source that produces equivalent parts. With a custom ASIC you may have to do some redesign to transfer your ASIC to the second source. However, for all ASIC types, switching production to a second source will involve some cost. Figure 1.12 assumes a second-source cost of $2000 for all types of ASIC (the amount may be substantially more than this).

1.4.4 ASIC Variable Costs

Figure 1.14 shows a spreadsheet, "Variable Costs," that calculates some example part costs. This spreadsheet uses the terms and parameters defined below the figure.

	FPGA	MGA	CBIC	Units
Wafer size	6	6	6	inches
Wafer cost	1,400	1,300	1,500	$
Design	10,000	10,000	10,000	gates
Density	10,000	20,000	25,000	gates/sq.cm
Utilization	60	85	100	%
Die size	1.67	0.59	0.40	sq.cm
Die/wafer	88	248	365	
Defect density	1.10	0.90	1.00	defects/sq.cm
Yield	65	72	80	%
Die cost	25	7	5	$
Profit margin	60	45	50	%
Price/gate	0.39	0.10	0.08	cents
Part cost	$39	$10	$8	

FIGURE 1.14 A spreadsheet, "Variable Costs," to calculate the part cost (that is the variable cost for a product using ASICs) for different ASIC technologies.

- The wafer size increases every few years. From 1985 to 1990, 4-inch to 6-inch diameter wafers

were common; equipment using 6-inch to 8-inch wafers was introduced between 1990 and 1995; the next step is the 300 cm or 12-inch wafer. The 12-inch wafer will probably take us to 2005.

- The wafer cost depends on the equipment costs, process costs, and overhead in the fabrication line. A typical wafer cost is between $1000 and $5000, with $2000 being average; the cost declines slightly during the life of a process and increases only slightly from one process generation to the next.
- Moore's Law (after Gordon Moore of Intel) models the observation that the number of transistors on a chip roughly doubles every 18 months. Not all designs follow this law, but a "large" ASIC design seems to grow by a factor of 10 every 5 years (close to Moore's Law). In 1990 a large ASIC design size was 10 k-gate, in 1995 a large design was about 100 k-gate, in 2000 it will be 1 M-gate, in 2005 it will be 10 M-gate.
- The gate density is the number of gate equivalents per unit area (remember: a gate equivalent, or gate, corresponds to a two-input NAND gate).
- The gate utilization is the percentage of gates that are on a die that we can use (on a gate array we waste some gate space for interconnect).
- The die size is determined by the design size (in gates), the gate density, and the utilization of the die.
- The number of die per wafer depends on the die size and the wafer size (we have to pack rectangular or square die, together with some test chips, on to a circular wafer so some space is wasted).
- The defect density is a measure of the quality of the fabrication process. The smaller the defect density the less likely there is to be a flaw on any one die. A single defect on a die is almost always fatal for that die. Defect density usually increases with the number of steps in a process. A defect density of less than 1 cm^{-2} is typical and required for a submicron CMOS process.
- The yield of a process is the key to a profitable ASIC company. The yield is the fraction of die on a wafer that are good (expressed as a percentage). Yield depends on the complexity and maturity of a process. A process may start out with a yield of close to zero for complex chips, which then climbs to above 50 percent within the first few months of production. Within a year the yield has to be brought to around 80 percent for the average complexity ASIC for the process to be profitable. Yields of 90 percent or more are not uncommon.
- The die cost is determined by wafer cost, number of die per wafer, and the yield. Of these parameters, the most variable and the most critical to control is the yield.
- The profit margin (what you sell a product for, less what it costs you to make it, divided by the cost) is determined by the ASIC company's fixed and variable costs. ASIC vendors that make and sell custom ASICs have huge fixed and variable costs associated with building and running fabrication facilities (a fabrication plant is a fab). FPGA companies are typically fabless -they do not own a fab-they must pass on the costs of the chip manufacture (plus the profit margin of the chip manufacturer) and the development cost of the FPGA structure in the FPGA part cost. The profitability of any company in the ASIC business varies greatly.
- The price per gate (usually measured in cents per gate) is determined by die costs and design size. It varies with design size and declines over time.
- The part cost is determined by all of the preceding factors. As such it will vary widely with time, process, yield, economic climate, ASIC size and complexity, and many other factors.

As an estimate you can assume that the price per gate for any process technology falls at about 20 % per year during its life (the average life of a CMOS process is 2-4 years, and can vary widely). Beyond the life of a process, prices can increase as demand falls and the fabrication equipment becomes harder to maintain. Figure 1.15 shows the price per gate for the different ASICs and process technologies using

the following assumptions:

- For any new process technology the price per gate decreases by 40 % in the first year, 30 % in the second year, and then remains constant.
- A new process technology is introduced approximately every 2 years, with feature size decreasing by a factor of two every 5 years as follows: 2 m m in 1985, 1.5 m m in 1987, 1 m m in 1989, 0.8-0.6 m m in 1991-1993, 0.5-0.35 m m in 1996-1997, 0.25-0.18 m m in 1998-2000.
- CBICs and MGAs are introduced at approximately the same time and price.
- The price of a new process technology is initially 10 % above the process that it replaces.
- FPGAs are introduced one year after CBICs that use the same process technology.
- The initial FPGA price (per gate) is 10 percent higher than the initial price for CBICs or MGAs using the same process technology.

From Figure 1.15 you can see that the successive introduction of new process technologies every 2 years drives the price per gate down at a rate close to 30 percent per year. The cost figures that we have used in this section are very approximate and can vary widely (this means they may be off by a factor of 2 but probably are correct within a factor of 10). ASIC companies do use spreadsheet models like these to calculate their costs.

FIGURE 1.15 Example price per gate figures.

Having decided if, and then which, ASIC technology is appropriate, you need to choose the appropriate cell library. Next we shall discuss the issues surrounding ASIC cell libraries: the different types, their sources, and their contents.

1.5 ASIC Cell Libraries

The cell library is the key part of ASIC design. For a programmable ASIC the FPGA company supplies you with a library of logic cells in the form of a design kit , you normally do not have a choice, and the cost is usually a few thousand dollars. For MGAs and CBICs you have three choices: the ASIC vendor (the company that will build your ASIC) will supply a cell library, or you can buy a cell library from a third-party library vendor , or you can build your own cell library.

The first choice, using an ASIC-vendor library , requires you to use a set of design tools approved by the

ASIC vendor to enter and simulate your design. You have to buy the tools, and the cost of the cell library is folded into the NRE. Some ASIC vendors (especially for MGAs) supply tools that they have developed in-house. For some reason the more common model in Japan is to use tools supplied by the ASIC vendor, but in the United States, Europe, and elsewhere designers want to choose their own tools. Perhaps this has to do with the relationship between customer and supplier being a lot closer in Japan than it is elsewhere.

An ASIC vendor library is normally a phantom library -the cells are empty boxes, or phantoms , but contain enough information for layout (for example, you would only see the bounding box or abutment box in a phantom version of the cell in Figure 1.3). After you complete layout you hand off a netlist to the ASIC vendor, who fills in the empty boxes (phantom instantiation) before manufacturing your chip.

The second and third choices require you to make a buy-or-build decision . If you complete an ASIC design using a cell library that you bought, you also own the masks (the tooling) that are used to manufacture your ASIC. This is called customer-owned tooling (COT , pronounced "see-oh-tee"). A library vendor normally develops a cell library using information about a process supplied by an ASIC foundry . An ASIC foundry (in contrast to an ASIC vendor) only provides manufacturing, with no design help. If the cell library meets the foundry specifications, we call this a qualified cell library . These cell libraries are normally expensive (possibly several hundred thousand dollars), but if a library is qualified at several foundries this allows you to shop around for the most attractive terms. This means that buying an expensive library can be cheaper in the long run than the other solutions for high-volume production.

The third choice is to develop a cell library in-house. Many large computer and electronics companies make this choice. Most of the cell libraries designed today are still developed in-house despite the fact that the process of library development is complex and very expensive.

However created, each cell in an ASIC cell library must contain the following:

- A physical layout
- A behavioral model
- A Verilog/VHDL model
- A detailed timing model
- A test strategy
- A circuit schematic
- A cell icon
- A wire-load model
- A routing model

For MGA and CBIC cell libraries we need to complete cell design and cell layout and shall discuss this in Chapter 2. The ASIC designer may not actually see the layout if it is hidden inside a phantom, but the layout will be needed eventually. In a programmable ASIC the cell layout is part of the programmable ASIC design (see Chapter 4).

The ASIC designer needs a high-level, behavioral model for each cell because simulation at the detailed timing level takes too long for a complete ASIC design. For a NAND gate a behavioral model is simple. A multiport RAM model can be very complex. We shall discuss behavioral models when we describe Verilog and VHDL in Chapter 10 and Chapter 11. The designer may require Verilog and VHDL models

in addition to the models for a particular logic simulator.

ASIC designers also need a detailed timing model for each cell to determine the performance of the critical pieces of an ASIC. It is too difficult, too time-consuming, and too expensive to build every cell in silicon and measure the cell delays. Instead library engineers simulate the delay of each cell, a process known as characterization . Characterizing a standard-cell or gate-array library involves circuit extraction from the full-custom cell layout for each cell. The extracted schematic includes all the parasitic resistance and capacitance elements. Then library engineers perform a simulation of each cell including the parasitic elements to determine the switching delays. The simulation models for the transistors are derived from measurements on special chips included on a wafer called process control monitors (PCMs) or drop-ins . Library engineers then use the results of the circuit simulation to generate detailed timing models for logic simulation. We shall cover timing models in Chapter 13.

All ASICs need to be production tested (programmable ASICs may be tested by the manufacturer before they are customized, but they still need to be tested). Simple cells in small or medium-size blocks can be tested using automated techniques, but large blocks such as RAM or multipliers need a planned strategy. We shall discuss test in Chapter 14.

The cell schematic (a netlist description) describes each cell so that the cell designer can perform simulation for complex cells. You may not need the detailed cell schematic for all cells, but you need enough information to compare what you think is on the silicon (the schematic) with what is actually on the silicon (the layout)-this is a layout versus schematic (LVS) check.

If the ASIC designer uses schematic entry, each cell needs a cell icon together with connector and naming information that can be used by design tools from different vendors. We shall cover ASIC design using schematic entry in Chapter 9. One of the advantages of using logic synthesis (Chapter 12) rather than schematic design entry is eliminating the problems with icons, connectors, and cell names. Logic synthesis also makes moving an ASIC between different cell libraries, or retargeting , much easier.

In order to estimate the parasitic capacitance of wires before we actually complete any routing, we need a statistical estimate of the capacitance for a net in a given size circuit block. This usually takes the form of a look-up table known as a wire-load model . We also need a routing model for each cell. Large cells are too complex for the physical design or layout tools to handle directly and we need a simpler representation-a phantom

1.6 Summary

In this chapter we have looked at the difference between full-custom ASICs, semi-custom ASICs, and programmable ASICs. Table 1.3 summarizes their different features. ASICs use a library of predesigned and precharacterized logic cells. In fact, we could define an ASIC as a design style that uses a cell library rather than in terms of what an ASIC is or what an ASIC does.

TABLE 1.3 Types of ASIC.

ASIC type	Family member	Custom mask layers	Custom logic cells
Full-custom	Analog/digital	All	Some

Semicustom	Cell-based (CBIC)	All	None
	Masked gate array (MGA)	Some	None
Programmable	Field-programmable gate array (FPGA)	None	None
	Programmable logic device (PLD)	None	None

You can think of ICs like pizza. A full-custom pizza is built from scratch. You can customize all the layers of a CBIC pizza, but from a predefined selection, and it takes a while to cook. An MGA pizza uses precooked crusts with fixed sizes and you choose only from a few different standard types on a menu. This makes MGA pizza a little faster to cook and a little cheaper. An FPGA is rather like a frozen pizza-you buy it at the supermarket in a limited selection of sizes and types, but you can put it in the microwave at home and it will be ready in a few minutes.

In each chapter we shall indicate the key concepts. In this chapter they are

- The difference between full-custom and semicustom ASICs
- The difference between standard-cell, gate-array, and programmable ASICs
- The ASIC design flow
- Design economics including part cost, NRE, and breakeven volume
- The contents and use of an ASIC cell library

1.7 Problems

1.1 (Break-even volumes, 60 min.) You need a spreadsheet program (such as Microsoft Excel) for this problem.

- a. Build a spreadsheet, "Break-even Analysis," to generate Figure 1.11.
- b. Derive equations for the break-even volumes (there are three: FPGA/MGA, FPGA/CBIC, and MGA/CBIC) and calculate their values.
- c. Increase the FPGA part cost by $10 and use your spreadsheet to produce the new break-even graph. Hint: (For users of Excel-like spreadsheets) use the XY scatter plot option. Use the first column for the x -axis data.
- d. Find the new break-even volumes (change the volume until the cost becomes the same for two technologies).
- e. Program your spreadsheet to automatically find the break-even volumes. Now graph the break-even volume (for a choice between FPGA and CBIC) for values of FPGA part costs ranging from $10-$50 and CBIC costs ranging from $2-$10 (do not change the fixed costs from Figure 1.12).
- f. Calculate the sensitivity of the break-even volumes to changes in the part costs and fixed costs. There are three break-even volumes and each of these is sensitive to two part costs and two fixed costs. Express your answers in two ways: in equation form and as numbers (for the values in Section 1.4.2 and Figure 1.11).
- g. The costs in Figure 1.11 are not unrealistic. What can you say from your answers if you are a defense contractor, primarily selling products in volumes of less than 1000 parts? What if you are a PC board vendor selling between 10,000 and 100,000 parts?

1.2 (Design productivity, 10 min.) Given the figures for the SPARCstation 1 ASICs described in

Section 1.3 what was the productivity measured in transistors/day? and measured in gates/day? Compare your answers with the figures for productivity in Section 1.4.3 and explain any differences. How accurate do you think productivity estimates are?

1.3 (ASIC package size, 30 min.) Assuming, for this problem, a gate density of 1.0 gate/mil ^{2} (see Section 15.4, "Estimating ASIC Size," for a detailed explanation of this figure), the maximum number of gates you can put in a package is determined by the maximum die size for each of the packages shown in Table 1.4. The maximum die size is determined by the package cavity size; these are package-limited ASICs. Calculate the maximum number of I/O pads that can be placed on a die for each package if the pad spacing is: (i) 5 mil, and (ii) 10 mil. Compare your answers with the maximum numbers of pins (or leads) on each package and comment. Now calculate the minimum number of gates that you can put in each package determined by the minimum die size.

TABLE 1.4 Die size limits for ASIC packages.

Package 1	Number of pins or leads	Maximum die size 2 (mil ^{2})	Minimum die size 3 (mil ^{2})
PLCC	44	320 ¥ 320	94 ¥ 94
PLCC	68	420 ¥ 420	154 ¥ 154
PLCC	84	395 ¥ 395	171 ¥ 171
PQFP	100	338 ¥ 338	124 ¥ 124
PQFP	144	350 ¥ 350	266 ¥ 266
PQFP	160	429 ¥ 429	248 ¥ 248
PQFP	208	501 ¥ 501	427 ¥ 427
CPGA	68	480 ¥ 480	200 ¥ 200
CPGA	84	370 ¥ 370	200 ¥ 200
CPGA	120	480 ¥ 480	175 ¥ 175
CPGA	144	470 ¥ 470	250 ¥ 250
CPGA	223	590 ¥ 590	290 ¥ 290
CPGA	299	590 ¥ 590	470 ¥ 470
PPGA	64	230 ¥ 230	120 ¥ 120
PPGA	84	380 ¥ 380	150 ¥ 150
PPGA	100	395 ¥ 395	150 ¥ 150
PPGA	120	395 ¥ 395	190 ¥ 190
PPGA	144	660 ¥ 655	230 ¥ 230
PPGA	180	540 ¥ 540	330 ¥ 330
PPGA	208	500 ¥ 500	395 ¥ 395

1.4 (ASIC vendor costs, 30 min.) There is a well-known saying in the ASIC business: "We lose money on every part-but we make it up in volume." This has a serious side. Suppose Sumo Silicon currently has two customers: Mr. Big, who currently buys 10,000 parts per week, and Ms. Smart, who currently buys 4800 parts per week. A new customer, Ms. Teeny (who is growing fast), wants to buy 1200 parts per week. Sumo’s costs are

wafer cost = $500 + ($250,000/ W),

where W is the number of wafer starts per week. Assume each wafer carries 200 chips (parts), all parts are identical, and the yield is

yield = 70 + 0.2 ¥ (W - 80) % (1.3)

Currently Sumo has a profit margin of 35 percent. Sumo is currently running at 100 wafer starts per week for Mr. Big and Ms. Smart. Sumo thinks they can get 50 cents more out of Mr. Big for his chips, but Ms. Smart won't pay any more. We can calculate how much Sumo can afford to lose per chip if they want Ms. Teeny's business really badly.

- a. What is Sumo's current yield?
- b. How many good parts is Sumo currently producing per week? (Hint: Is this enough to supply Mr. Big and Ms. Smart?)
- c. Calculate how many extra wafer starts per week we need to supply Ms. Teeny (the yield will change-what is the new yield?). Think when you give this answer.
- d. What is Sumo's increase in costs to supply Ms. Teeny?
- e. Multiply your answer to part d by 1.35 (to account for Sumo's profit). This is the increase in revenue we need to cover our increased costs to supply Ms. Teeny.
- f. Now suppose we charge Mr. Big 50 cents more per part. How much extra revenue does that generate?
- g. How much does Ms. Teeny's extra business reduce the wafer cost?
- h. How much can Sumo Silicon afford to lose on each of Ms. Teeny's parts, cover its costs, and still make a 35 percent profit?

1.5 (Silicon, 20 min.) How much does a 6-inch silicon wafer weigh? a 12-inch wafer? How much does a carrier (called a boat) that holds twenty 12-inch wafers weigh? What implications does this have for manufacturing?

- a. How many die that are 1-inch on a side does a 12-inch wafer hold? If each die is worth $100, how much is a 20-wafer boat worth? If a factory is processing 10 of these boats in different furnaces when the power is interrupted and those wafers have to be scrapped, how much money is lost?
- b. The size of silicon factories (fabs or foundries) is measured in wafer starts per week. If a factory is capable of 5000 12-inch wafer starts per week, with an average die of 500 mil on a side that sells for $20 and 90 percent yield, what is the value in dollars/year of the factory production? What fraction of the current gross national (or domestic) product (GNP/GDP) of your country is that? If the yield suddenly drops from 90 percent to 40 percent (a yield bust) how much revenue is the company losing per day? If the company has a cash reserve of $100 million and this revenue loss drops "straight to the bottom line," how long does it take for the company to go out of business?
- c. TSMC produced 2 million 6-inch wafers in 1996, how many 500 mil die is that? TSMC's $500 million Camas fab in Washington is scheduled to produce 30,000 8-inch wafers per month by the year 2000 using a 0.35 mm process. If a 1 Mb SRAM yields 1500 good die per 8-inch wafer and there are 1700 gross die per wafer, what is the yield? What is the die size? If the SRAM cell size is 7 mm^2 , what fraction of the die is used by the cells? What is TSMC's cost per bit for SRAM if

the wafer cost is $2000? If a 16Mb DRAM on the same fab line uses a 16 mm^2 die, what is the cost per bit for DRAM assuming the same yield?

1.6 (Simulation time, 30 min.) ". . . The system-level simulation used approximately 4000 lines of SPARC assembly language . . . each simulation clock was simulated in three real time seconds" (Sun Technology article).

- a. With a 20 MHz clock how much slower is simulated time than real time?
- b. How long would it take to simulate all 4000 lines of test code? (Assume one line of assembly code per cycle-a good approximation compared to the others we are making.)

The article continues: "the entire system was simulated, running actual code, including several milliseconds of SunOS execution. Four days after power-up, SPARCstation 1 booted SunOS and announced: ’hello world’ ."

- c. How long would it take to simulate 5 ms of code?
- d. Find out how long it takes to boot a UNIX workstation in real time. How many clock cycles is this?
- e. The machine is not executing boot code all this time; you have to wait for disk drives to spin-up, file systems checks to complete, and so on. Make some estimates as to how much code is required to boot an operating system (OS) and how many clock cycles this would take to execute.

The number of clock cycles you need to simulate to boot a system is somewhere between your answers to parts d and e.

- f. From your answers make an estimate of how long it takes to simulate booting the OS. Does this seem reasonable?
- g. Could the engineers have simulated a complete boot sequence?
- h. Do you think the engineers expected the system to boot on first silicon, given the complexity of the system and how long they would have to wait to simulate a complete boot sequence? Explain.

1.7 (Price per gate, 5 min.) Given the assumptions of Section 1.4.4 on the price per gate of different ASIC technologies, what has to change for the price per gate for an FPGA to be less than that for an MGA or CBIC-if all three use the same process?

1.8 (Pentiums, 20 min.) Read the online tour of the Pentium Pro at http://www.intel.com (adapted from a paper presented at the 1995 International Solid-State Circuits Conference). This is not an ASIC design; notice the section on full-custom circuit design. Notice also the comments on the use of ’assert’ statements in the HDL code that described the circuits. Find out the approximate cost of the Intel Pentium (3.3 million transistors) and Pentium Pro (5.5 million transistors) microprocessors.

- a. Assuming there a four transistors per gate equivalent, what is the price per gate?
- b. Find out the cost of a 1 Mb, 4 Mb, 8 Mb, or 16 Mb DRAM. Assuming one transistor per memory bit, what is the price per gate of DRAM?
- c. Considering that both have roughly the same die size, are just as complex to design and to manufacture, why is there such a huge difference in price per gate between microprocessors and DRAM?

1.9 (Inverse embedded arrays, 10 min.) A relatively new cousin of the embedded gate array, the inverse-embedded gate array , is a cell-based ASIC that contains an embedded gate-array megacell. List the features as well as the advantages and disadvantages of this type of ASIC in the same way as for the other members of the ASIC family in Section 1.1.

1.10 (0.5-gate design, 60 min.) It is a good idea to complete a 0.5-gate ASIC design (an inverter connected between an input pad and an output pad) in the first week (day) of class. Capture the commands in a report that shows all the steps taken to create your chip starting from an empty directory-halfgate .

1.11 (Filenames, 30 min.) Start a list of filename extensions used in ASIC design. Table 1.5 shows an example. Expand this list as you use more tools.

TABLE 1.5 CAD tool filename extensions.

Extension	Description	From	To
.ini	Viewlogic startup file, library search paths, etc.	Viewlogic/Viewdraw	Internal tools use
.wir	Schematic file		other Viewlogic tools

1. PLCC = plastic leaded chip carrier, PQFP = plastic quad flat pack, CPGA = ceramic pin-grid array, PPGA = plastic pin-grid array.

2. Maximum die size is not standard and varies between manufacturers.

3. Minimum die size is an estimate based on bond length restrictions.

1.8 Bibliography

The Addison-Wesley VLSI Design Series covers all aspects of VLSI design. Mead and Conway [1980] is an introduction to VLSI design. Glasser and Dobberpuhl [1985] deal primarily with NMOS technology, but their book is still a valuable circuit design reference. Bakoglu's book [1990] concentrates on system interconnect issues. Both editions of Weste and Eshraghian [1993] describe full-custom VLSI design.

Other books on CMOS design include books by Kang and Leblebici [1996], Wolf [1994], Price [1994], Hurst [1992], and Shoji [1988]. Alvarez [1993] covers BiCMOS, but concentrates more on technology than design. Embabi, Bellaouar, and Elmasry [1993] also cover BiCMOS design from a similar perspective. Elmasry's book [1994] contains a collection of papers on BiCMOS design. Einspruch and Hilbert [1991]; Huber and Rosneck [1991]; and Veendrick [1992] are introductions to ASIC design for nontechnical readers. Long and Butner [1990] cover gallium arsenide (GaAs) IC design. Most books on CMOS and ASIC design are classified in the TK7874 section of the Library of Congress catalog (T is for technology).

Several journals and magazines publish articles on ASICs and ASIC design. The IEEE Transactions on Very Large Scale Integration (VLSI) Systems (ISSN 1063-8210, TK7874.I3273, 1993-) is dedicated to

VLSI design. The IEEE Custom Integrated Circuits Conference (ISSN 0886-5930, TK7874.C865, 1979-) and the IEEE International ASIC Conference (TK7874.6.I34a, 1988-1991; TK7874.6.I35, ISSN 1063-0988, 1991-) both cover the design and use of ASICs. EE Times (ISSN 0192-1541, http://techweb.cmp.com/eet) is a newsletter that includes a wide-ranging coverage of system design, ASICs, and ASIC design. Integrated System Design (ISSN 1080-2797), formerly ASIC & EDA) is a monthly publication that includes ASIC design topics. High Performance Systems (ISSN 0887-9664), formerly VLSI Design (ISSN 0279-2834), deals with system design including the use of ASICs. EDN (ISSN 0012-7515, http://www.ednmag.com) has broader coverage of the electronics industry, including articles on VLSI and systems design. Computer Design (ISSN 0010-4566) is targeted at systems-level design but includes coverage of ASICs (for example, a special issue in August 1996 was devoted to ASIC design).

The Electronic Industries Association (EIA) has produced a standard, JESD12-1B, "Terms and definitions for gate arrays and cell-based digital integrated circuits," to define terms and definitions.

University Video Communication (http://www.uvc.com) produces several videotapes on computer science and engineering topics including ASIC design. Maly’s book [1987] is a picture book containing drawings and cross-sections of devices, and shows how a transistor is fabricated.

It is difficult to obtain detailed technical information from ASIC companies and vendors apart from the glossy brochures (sparkle sheets). It used to be possible to obtain data books on cell libraries (now these are large and difficult to produce, and are often only available in electronic form) as well as design guidelines and handbooks. Fortunately there are now many resources available on the World Wide Web, which are, of course, constantly changing. EDAC (Electronic Design Automation Companies) has a Web page (http://www.edac.org) with links to most of the EDA companies. The Electrical Engineering page on the World Wide Web (E2W3) (http://www.e2w3.com) contains links to many ASIC related areas, including distributors, ASIC companies, and semiconductor companies. SEMATECH (Semiconductor Manufacturing Technology) is a nonprofit consortium of U.S. semiconductor companies and has a Web page (http://www.sematech.org) that includes links to major semiconductor manufacturers. The MIT Semiconductor Subway (http://www-mtl.mit.edu) is more oriented toward devices, processes, and materials but contains links to other VLSI industrial and academic areas. There is a list of EDA companies at http://www.yahoo.com under Business_and_Economy in Companies/Computers/Software/Graphics/CAD/IC_Design .

The MOS Implementation Service (MOSIS), located at the Information Sciences Institute (ISI) at the University of Southern California (USC), is a "silicon broker" for universities in the United States and also provides commercial access to fabrication facilities (http://www.isi.edu). Professor Don Bouldin maintains The Microelectronic Systems Newsletter, formerly the MOSIS Users Group (MUG) Newsletter, at http://www-ece.engr.utk.edu/ece .

NASA (http://nppp.jpl.nasa.gov/dmg/jpl/loc/asic

1.9 References

Alvarez, A. R. (Ed.). 1993. BiCMOS Technology and Applications. Norwell, MA: Kluwer. ISBN 0-7923-9384-8. TK7871.99.M44.

Bakoglu, H. B. 1990. Circuits, Interconnections, and Packaging for VLSI. Reading, MA: Addison-Wesley, 527 p. ISBN 0-86341-165-7. TK7874.B345. Based on a Stanford Ph.D. thesis and contains chapters on: devices and interconnections, packaging, transmission lines, cross talk, clocking of high-speed systems, system level performance.

Einspruch N. G., and J. L. Hilbert (Eds.). 1991. Application Specific Integrated Circuit (ASIC) Technology. San Diego, CA: Academic Press. ISBN 0122341236. TK7874.V56 vol. 23. Includes: "Introduction to ASIC technology," Hilbert; "Market dynamics of the ASIC revolution," Collett; "Marketing ASICs," Chakraverty; "Design and architecture of ASIC products," Hickman et al.; "Model and library development," Lubhan; "Computer-aided design tools and systems," Rowson; "ASIC manufacturing," Montalbo; "Test and testability of ASICs," Rosqvist; "Electronic packaging for ASICs," Herrell and Prokop; "Application and selection of ASICs," Mitchell; "Designing with ASICs," Wilkerson; "Quality and reliability," Young.

Elmasry, M. I. 1994. BiCMOS Integrated Circuit Design: with Analog, Digital, and Smart Power Applications. New York: IEEE Press, ISBN 0780304306. TK7871.99.M44.B53.

Embabi, S. H. K., A. Bellaouar, and M. I. Elmasry. 1993. Digital BiCMOS Integrated Circuit Design. Norwell: MA: Kluwer, 398 p. ISBN 0-7923-9276-0. TK7874.E52.

Glasser, L. A., and D. W. Dobberpuhl. 1985. The Design and Analysis of VLSI Circuits. Reading, MA: Addison-Wesley, 473 p. ISBN 0-201-12580-3. TK7874.G573. Detailed analysis of circuits, but largely nMOS.

Huber, J. P., and M. W. Rosneck. 1991. Successful ASIC Design the First Time Through. New York: Van Nostrand Reinhold, 200 p. ISBN 0-442-00312-9. TK7874.H83.

Hurst, S. L. 1992. Custom VLSI Microelectronics. Englewood Cliffs, NJ: Prentice-Hall, 466 p. ISBN 0-13-194416-9. TK7874.H883.

Kang, S-M, and Y. Leblebici. 1996. CMOS Digital Integrated Circuits: Analysis and Design. New York: McGraw-Hill, 614 p. ISBN 0070380465.

Long, S. I., and S. E. Butner. 1990. Gallium Arsenide Digital Integrated Circuit Design. New York: McGraw-Hill, 486 p. ISBN 0-07-038687-0. TK7874.L66.

Maly, W. 1987. Atlas of IC Technologies: An Introduction to VLSI Processes. Menlo Park, CA: Benjamin-Cummings, 340 p. ISBN 0-8053-6850-7. TK7874.M254. Cross-sectional drawings showing construction of nMOS and CMOS processes.

Mead, C. A., and L. A. Conway. 1980. Introduction to VLSI Systems. Reading, MA: Addison-Wesley, 396 p. ISBN 0-201-04358-0. TK7874.M37.

Price, T. E. 1994. Introduction to VLSI Technology. Englewood Cliffs, NJ: Prentice-Hall, 280 p. ISBN 0-13-500422-5. TK7874.P736.

Shoji, M. 1988. CMOS Digital Circuit Technology. Englewood Cliffs, NJ: Prentice-Hall, 434 p. ISBN 0131388436. TK7871.99.M44. See also Shoji, M., High Speed Digital Circuits, Reading, MA:

Addison-Wesley, 1996, 360 p., ISBN 0-201-63483-X, TK7874.65.S56

Weste, N. H. E., and K. Eshraghian. 1993. Principles of CMOS VLSI Design: A Systems Perspective. 2nd ed. Reading, MA: Addison-Wesley, 713 p. ISBN 0-201-53376-6. TK7874.W46. Concentrates on full-custom design.

Wolf, W. H. 1994. Modern VLSI Design: A Systems Approach. Englewood Cliffs, NJ: Prentice-Hall, 468 p. ISBN 0-13-588377-6. TK7874.65.W65.

Veendrick, H. J. M. 1992. MOS ICs from Basics to ASICs.