CHAPTER 1

INTRODUCTION

In a positive sense, this book is about gaining a competitive edge in the integrated circuit marketplace. What it will suggest to the reader is that there is unrecognized value hidden in the safety-net margins that exist in the various descriptive views of any piece of intellectual property. This hidden value, which might be useful on any given integrated circuit design, can normally be left "on the table." However, this hidden value can and should be used by the aggressive design engineer (or manager) to beat market competitors. The pages ahead will reveal how the typical design house can enhance the performance, reduce the power, and improve the density of standard-cell logic. It will show how to add value to the generic, foundry-provided standard-cell library that many companies use "out of the box." It will identify low-risk opportunities where aggressive designers and managers can shave off margin from overdesigned standard cells.

However, the other side of the preceding is also true. That is to say, this is a dangerous book. The reason it is dangerous is that no engineer or manager has ever been fired because of not following the herd and either accomplished or failed to accomplish exactly what any other engineer or manager accomplished or failed to accomplish. The not-so-hidden message of this book is that by breaking from the herd and attempting to use that safety net margin, wherever it is found, the results, *when successful*, will reap market benefit. The dangerous downside of this activity, however, is the risk of taking a little too much of the perceived margin and actually forcing a failure on silicon. The cost of such aggressiveness could easily be career limitations. Now, before you put this book back down, let me point out that taking advantage of such margin is not a new idea. In fact, it is rather old and it used to be rather common for the design engineer to take advantage of

Engineering the CMOS Library: Enhancing Digital Design Kits for Competitive Silicon, First Edition. David Doman.

it. It used to be called "adding value" by *not* doing it the same way as everybody else. However, over time, as the cost of development of integrated circuits and the market cost of failure of those integrated circuits have increased, this concept has been replaced by "how not to mess up" by doing it the same way as everybody else. The result is that most integrated circuits are now designed in an extremely conservative manner.

About twenty years ago, as of the writing of this book, I was working for a microprocessor design company in the data communications integrated circuit design center. At that time, the design center was working on some Fiber Distributed Data Interface (FDDI) devices. During the development of some of those integrated circuits, I was asked if it was possible to decrease the cost by reducing the number of layers that the router that was being used on that particular design could use. Specifically, the design center was using a near-first-generation sea-of-gates router, and I was asked if it were possible to route the design as a channeled route using the sea-of-gates router that was available. I told the design manager that I could do channeled routing with the sea-of-gates router but that the sea-of-gates router was not a channeled router. My answer perplexed the design manager until I told him that you can use a wrench to hammer a nail into a wall, but that does not make a wrench a hammer.

The preceding illustration is what this book is basically about. Too often in the current world of integrated circuit design, we look for the perfect hammer in order to drive the perfect nail into the perfect wall exactly as the hammer and nail and wall manufacturer has instructed us. This is the previously mentioned safe "I won't mess up" follow-the-herd method of integrated circuit design. However, doing that will mean your results will be exactly like the results of the competing design center that set up shop down the block. This book is about looking for the ways of using the tools that we have in creative ways. It is about using the wrench because the hammer is too expensive. It is also about using the hammer—but in ways that the design center down the block will not.

What this book is not about is device physics. You will not find any physics-based or electronics-based equation anywhere in it. There is no consideration herein about ion currents or oxide thickness. There already are more than enough books available on all of those subjects (and others as well). Rather, this book explores the everyday considerations of the library design and support engineer. That library design and support engineer may be in a specific library function in a separate organization or be a design engineer who is integrated within the design center that is using the library that is under consideration and is doing exploration of the library as an additional function to his or her regular design activities. This book is meant to be used as an instigator of thought. As mentioned, there are no physics or electronics equations in this book (although there are a few Boolean logic equations). Indeed, I have taken the tack of explaining in either words or illustrations as opposed to hard physics. An outcome of this book should be that design centers might push technologies and libraries in ways that the fabrication houses might be nervous about but that would (or should or could) be covered sufficiently through the validation and verification efforts of the design centers to be rendered viable.

A digital design kit, sometimes referred to as a DDK, is a collection of small-scale functions together with the various views that describe these functions for various process, design, and test (and other) tools. They are used by design-center engineers to design modern application-specific integrated circuits (ASICs) that can be processed successfully in the external fabrication house that supplied the original digital design kit.

In a sense, a DDK can be thought of as a collection of various bricks, each brick different, together with an assortment of descriptions of each type of brick that allows them to be placed together in such a manner as to further allow somebody to build a structure. Another way of looking at a digital design kit is as an alphabet, together with

the rules of usage of this alphabet for word and sentence construction that allow somebody to write a book. In either of these two cases, the bricks or the alphabets, if you can keep your supply separate from those available to anybody else, then you can build either a unique structure or better book. Otherwise, if you can somehow adjust your bricks or alphabet in a manner not available to others, you can still build the better structure or write the better book.

For some design centers—those that belong to companies that have their own fabrication facilities—these DDKs can be viewed as private in-house tools that are not available to design centers for competing companies. This is good because it allows the company to adjust the fabrication and hence the views of the digital design kit in ways that generate a competitive advantage for their designs over those of competing companies. For many or even most companies around today, however, this is not the case. These "fabrication-less" companies subcontract the processing of their designs to external fabrication houses and have no real control of the processing at the external fabrication houses and must accept the process "as is"—unless they are willing to pay significant pricing and significant time and engineering resources to convince the external fabrication house that it is worth their time and effort and cost to allow special handling of the design centers designs. As a result, few such design centers are willing to expend the energy to do so. Hence, the DDK they use to do their designs is the same DDK that the next design center down the street is using. These design centers end up using the digital design kit "straight out of the cellophane" with no adjustments.

There are advantages for having common views and tools, such as the ability to hire workers already familiar with such tools and views. However, the real result of this is that if two or more such companies decide to do the same sort of design for the same market while using the same digital design kit, they tend to produce nearly the same design. This is further forced because most design centers use the same Engineering Design Automation (EDA) tools or at least very similar ones. Yes, there may be Input-Output Structure (IO) order differences, perhaps scan-chain differences, and the embedded software might have been written somewhat differently, but these are all secondary to the fact that the two or more competing designs will be all roughly the same size, have the same cost to produce, and have the same power, all assuming they have the same features. The companies are then forced to compete in the market on price margin alone.

Now from the external fabrication house's point of view, it is advantageous to force common design practices across the ASIC design industry. These houses have developed business models that allow them to operate profitably by accepting ASIC designs that are produced by design centers that used their DDKs. These business models basically say that if the design customer used the digital design kit "as is" and the EDA tools in conjunction with these digital design kit files (assuming the design works), then the design will come out of the external fabrication house working or the external fabrication house will pay for it. Conversely, if the design customer decides to *not* use the digital design kit "as prescribed by the external fabrication house" but instead uses it with some changes and the resulting design does not work, then it is the design customer's fault and the external fabrication house has no fiduciary responsibility. Clearly, it is the external fabrication house's best interest to have solid DDKs. How can they do this? The easiest way is to allow margin in the various views of the kit, in effect making those views "bulletproof." Worst-case timing views are slightly slower, and best-case timing views are slightly better than expected. Worst-case power and best-case power views are guard banded. Similarly, for logical views, the external fabrication houses do not allow proper simulation of some actual although nonstandard functionality. In the end, view after

view is a little worse performance or a little less functionality per model than would otherwise be expected. They are all "margined" to one degree or another.

However, this added margin is precisely what the design-center customer wants to have removed or, at a minimum, transformed. This removal would allow the design-center customer to compete in the market not on price margin alone, but on power, performance, or size, the first two of which can be viewed as feature additions and the last of which enables profitability even in a price-margin-alone environment. The transformation of the margin from something that is decreed by the fabrication house into something that can be controlled and potentially proportioned by the design-center manager allows judgment calls to be made. If there is no performance or area penalty, for instance, then such margin additions could be added back into the calculation without harming its competitive position in the marketplace. Contrarily, if the penalty is too large, it can be at least partially minimized to the extent required to allow sufficient cost-margin savings, again allowing the resulting finished device to be cost competitive in that same marketplace.

This book describes areas where the margins in the views can be found and how to determine their extent and how extensively these margins can be safely removed.

Wait, you might be thinking. I just said that actually doing this "margin removal" (or any "margin transformation") could cause the fiduciary burden to be shifted to the design-center customer. Yes, that is true. However, in the perspective of the business model of the design-center customer, the extent of the risk may be worth it. This is not a technical decision, but rather a financial decision. Control of that financial decision, as determined by the amount of margin removal that is being considered, is thus moved to the realm of the design-center manager. This book should give the designer-center manager that knowledge needed to make those decisions.

One additional comment: just because a little medicine might be good for you, more is not necessarily better. That dictum is also true in engineering. Use this book at your own risk. Many of the techniques on margin reduction listed herein are as dangerous as they may sound, and proper caution needs to be taken when partaking in many of them, otherwise the design center that you support may start to produce nothing more than expensive sand.

Now, before going to much further, I do have one apology to the reader. As I review this book, I find that I often use "negative language" (for instance, "if one doesn't do something, and as a result, something doesn't happen, then ...."). I apologize. Nearly three decades in developing CMOS logic libraries, where the common function always involved an inversion (or NOT function) has trained my mind to think in that manner. I hope, however, that the average reader involved in the industry does not find this "negative" language any more awkward than I do.

1.1 ADDING PROJECT-SPECIFIC FUNCTIONS, DRIVE STRENGTHS, VIEWS, AND CORNERS

Beyond what the previous section says about margin removal (or margin adjustment), this book has a second goal: to allow adding items to a digital design kit in order to personalize it.

A specific digital design kit probably comes from the external fabrication house with three sets of timing and power views (abbreviated as PVT for "process corner, voltage corner, temperature corner"). These three PVT probably correspond to one worst-case

processing corner, one best-case processing corner, and one typical processing corner. If the design center has a different market niche that requires a different set of voltages of operation or environmental temperatures, or if it is willing to accept slightly less or slightly more processing variation, then it has to pay the external fabrication house perhaps several hundred thousand dollars to generate and support such new PVT timing or power views. This book should allow the design center the knowledge of what it takes to make such a PVT characterization on its own. With that knowledge, the design center should be better equipped to make the decision whether to buy a new PVT from the external fabrication house, "make" it internally (with the business benefits but support cost of doing so), or "reengineer" the market decision such that it becomes unnecessary.

A specific digital design kit will have a limited number of logical functions, although that number might be very extensive. If the design center has identified some special function that is not easily built by the logical functions present, it might choose to add that function by means of a custom design. The external fabrication house may have the ability, assuming that the design center goes forward with the design, to say that the fiduciary responsibility for the design moves to the design center if this new function does not work but remains if the fault is the result of some other part not being modified by the design center. This may seem bad for the design center, but it is the model of many of them. These centers have specific "special functions or logic" that they have used repeatedly in the past. These special functions or logic are considered to be a competitive advantage for those design centers with such logic. They are in business specifically because they have such functional or logic blocks. This book tells the design center how they can better add such functionality within the external fabrication house's DDK framework.

A design center may determine that it can build a smaller, faster, or less-power-consuming design if it had access to the functions supported by the digital design kit, but it requires a slightly larger or slightly smaller version of a function (in terms of area of performance or power). If this slightly adjusted cell is a larger version of one that is already present, then the design center might be tempted to parallel up multiple copies of the function. If the slight adjustment is to make it smaller, actual trimming of transistor widths might become needed. This book will discuss the possible means of doing so safely.

A design center might use an EDA tool that is not supported by views delivered from the external fabrication house. Although this book will not explain how to develop such views, it will give the design center the knowledge of how to ensure that design-center–generated views are consistent with the views that are delivered by the external fabrication house. It will also describe the efforts needed to deliver and support such views across multiple designs and view and tool updates.

The bottom line is that even though some people see the DDK as cast in stone, the design center that is willing to extend or adjust such kits can deliver better designs (i.e., cheaper to produce, test, or support). These designs will allow those design centers to compete in their chosen markets on more than just commodity (i.e., price-margin) techniques. This book will allow the engineers and managers of those design centers additional degrees of freedom in making the decisions to extend and adjust such digital design kits.

1.2 WHAT IS A DDK?

A digital design kit is a collection of physical views of certain functions that a fabrication house is capable of processing at a given technology node. Coupled to this are the other physical (read place and route) views as well as the logical, temporal, power, noise, and

test views that represent these various functions for (usually some, maybe most) of the various engineering design automation tools available on the market. Moreover, the typical DDK release will be for what has become known as a "Frankenstein" flow, with Cadence logical and LEF (library exchange format) views and Synopsys timing, power, and noise views and supported in the physical design kit (PDK), by Mentor verification decks, although there are exceptions to each instance, with Cadence, Synopsys, and Mentor being "the big three" EDA companies. These views are supplied by a fabrication house where any resultant design that uses these supplied cells and views will be processed. Some of these views can be adjusted, assuming knowledgeable guidance, with a fair amount of impunity from the fabrication house, while other views are touched with less impunity, even by the design-center experts. These views differ from and need to remain consistent with the decks and views in the PDK that also comes from the fabrication house.

More specifically, at the very minimum, a DDK will contain the following.

- Graphical Database System Stream-file format (GDS) (physical), which are views of one or more standard cell (stdcell) libraries, usually including IO-specific libraries, which can be used on designs that are destined to be processed in a particular fabrication house. These views are a binary format, viewable in any one of several GDS viewers. ASCII based human readable formats of GDS can be created through various EDA tools.
- Gate-level SPICE (simulation program with integrated circuit emphasis) netlist of each cell in the various libraries, SPICE being a cross between a physical view (which it represents) and a simulatable view, which is one of its two major purposes, and a verification source, which is its other major purpose. SPICE is ASCII based and easily human readable.
- Some further physical representation (possibly a physical LEF, an ASCII-based and human-readable description of the place and router pertinent parts of the GDS). This further physical representation is used by automatic place and route tools, in conjunction with a logical LEF, usually part of the PDK that contains information on layer definitions that the router can use during design place and route.
- Timing, power, and noise views, which are typically but not necessarily referred to as *liberty files*, a name that comes from the extension that is most often used ("lib") on versions of these ASCII files that were originally purely for Synopsys tools. Being ASCII, they are easily readable. More so, most liberty writers tend to output highly structured versions of the format, which further aids human understanding on reading. The Backus-Naur Format (BNF) for these files have since been "open sourced"—that is, they can usually be read and understood by most EDA timing tools. One aside: Synopsys has not frozen the BNF for its liberty files and thus care must be taken with every new Synopsys release in order to keep previous versions of liberty files from becoming antiquated.
- Logical and test views, typically being slightly adjusted versions of the same logical format, but with one geared toward aiding logical simulations and the other geared to representing actual internal nodes of a stdcell (or IO) function that allows for accurate tracking of "reachability and observability" for internal fault tracking. These views may come in one or both of two major logical tool formats (Cadence's open-source VERILOG file or IEEE's open-source VHDL format). They are both ASCII based and human readable.

- There may be other views, especially white papers, verification or validation documentation, and release-tracking documents, all in some form of human-readable format.
- Sometimes other, more EDA-tool-specific views can also be included.

Along with the preceding major views, support for special tools may be added to a DDK release by a fabrication house. These tools can include RAM/ROM memory compilers. Once installed on a system, RAM/ROM memory compilers, at the press of a few buttons, automatically generate all of the preceding formats for a specific RAM/ROM configuration supported by the fabrication house. The timing, power, and noise view is initially a machine-estimated version. Typical fabrication-house vendors, however, offer the option of characterizing the specific RAM/ROM in order to allow cleaner design closure of a design that uses that specific RAM/ROM, and these fabrication house usually offer this either free or for a small fee.

The reason that RAM/ROM compilers exist as opposed to a long list of usable RAM and ROM instances is that there is such a larger list of possible combined number of words, bits, and column-multiplexer (MUX) instances that could be chosen by design teams, all of which would therefore otherwise need to be supported.

Finally, there may be certain specific circuits (or analog subcircuit pieces, such as various layer resistors of multilayer capacitors or diode or PNP or NPN structures that can be used to build analog function in the particular technology node and in the specific fabrication house). Some of these, which are dependent on the technology node, will be custom and uneditable or only marginally editable structures (the more complex and deep the technology node, the more likely that this is so). Some will be just Boolean layer definitions. Most will have rather pared-down subsets of the previously mentioned views, just enough to allow some amount of simulation (usually in SPICE) and design-level verification (including netlist capability).

Now how extensive can any of the preceding be edited? It depends on the technology node and the desire of the design center. Deeper technology nodes will tend to have stricter requirements for the physical views than higher technology nodes. The real reason for this is that as geometries get smaller, they tend to go from drawn silicon shapes being transferred to the mask through lithographic treatments such as serifs at corners of polygons through optical-proximity correct techniques, through phase shifting, and to exact set widths and spaces. For the deepest technologies, what is drawn on the mask can bear little visible relation with what is desired on silicon. In those instances, the operative order is "Don't touch." Trust that the fabrication house understands better than the design team what certain polygons need to be, where they need to be, and why they need to be. However, for the higher nodes, even those in the 90-nanometer range, which includes some phase-shifted mask and some strong optical-proximity correction, it is worth pursuing with the fabrication house when a question on a physical view arises. For higher technology nodes than that, there is a good chance that you can find optimizations that the fabrication house has left "on the table." In addition, it is highly advantageous at these higher technology nodes to attempt to do just that. The reasoning is that these nodes have been "in the marketplace" for such a long time that every design house that wants to compete in a market will have access to the same generic package. Assuming that all design centers make the same decision on technology node (which is a valid assumption given that they would be privilege to the same marketing information), they will produce roughly the same size product in

the same rough timeline. This means that the design centers will be competing on price alone. Having a "pushed" technology might just be the edge needed to undercut the opposition in such an environment. As far as the logical and timing views are concerned, we will discuss them later in the book, but even at the deepest technology nodes the fabrication house has generated "loose" views that a design team can beat and thereby find additional margin over the competition.