Chip Design for Non-Designers:
An Introduction

Juan-Antonio Carballo

PennWell®

_Carballo_Book.indb 3 2/20/08 4:53:47 PM

CONTENTS

_Carballo_Book.indb 5 2/20/08 4:53:47 PM

_Carballo_Book.indb 6 2/20/08 4:53:47 PM

_Carballo_Book.indb 7 2/20/08 4:53:47 PM

INTRODUCTION

Who should read this book—and why

This book provides a practical introduction to modern chip design methodologies. It is intended for manufacturing-oriented and other non-design professionals with an interest in the pre–tape-out (pre-manufacturing) side of design. This book concentrates on functional, logic, circuit, and layout design using state-of-the-art methods and tools. More focus is given to the most popular design styles, including semicustom design. Many practical and useful examples are included throughout.

Readers who complete this book can expect to acquire a solid working knowledge of modern chip design methodologies. The examples, exercises, and bibliographic references provide an excellent supplement for application of the concepts learned. Readers will become truly fluent with how to design modern chips for varied applications.

Several types of readers will be interested in this book. First, it is intended for chip industry professionals and academics who need a practical introduction to modern chip design, including manufacturing-oriented and other non-design professionals, plus business practitioners who need technical background in design. Second, it can be leveraged for improved performance in many semiconductor-related jobs—from those held by research and development professionals to technical marketing managers, technology strategists, business development professionals, and technical sales professionals.

Focus areas

The focus is on the fundamentals of chip design, which should survive the evolution in design technology. As a result, the reader should be able to fully benefit from this book for at least five years. There is a lack of books providing similar value that are currently in publication. While there are good books on chip design, they tend to have different objectives and are detailed and lengthy. This book does not try to compete with or outdo those books. It is intended for non-design professionals who need a practical and useful introduction.

This book may also be used as an academic text, for a course that could be called Introduction to Chip Design or Introduction to Design Methodologies, especially in a manufacturing-oriented degree program. It is also a suitable source of material for internal seminars, and indeed, when appropriate, I use it for my own seminars, with audiences ranging from 30 to 200 people.

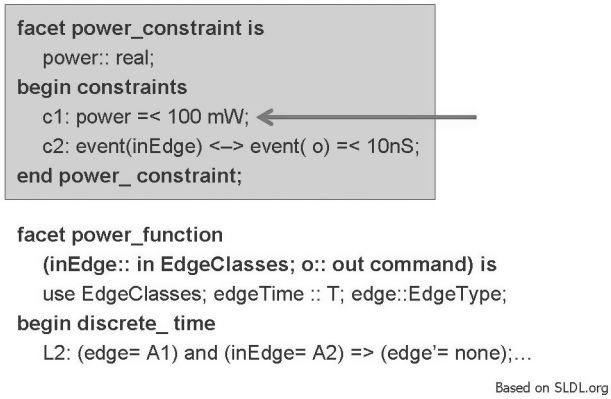

Fig. 2–1. A portion of a formal design specification in SLDL

Figure 2–1 depicts a portion of a *formal* design specification in SLDL. As can be seen in figure 2–1, the chip or circuit under design cannot have a power consumption over 100 milliwatts. This is expressed as a *power constraint.* In addition, there is another constraint that is of a timing nature: a particular signal event, *inEdge,* at an input pin triggers an output event, *o*, 10 nanoseconds later. Although for simplicity, the description in figure 2–1 is not complete, the bottom line is that the output needs to appear no later than 10 nanoseconds after the input event happens. Thus, both constraints are essentially performance requirements.

Question. Why don't hardware and chip designers use the same languages as software developers to create a chip specification? What is the key difference between hardware (chips) and software (code that runs on processors)? Does it make sense to start from an English-language description in both cases?

Exercise. Describe a specification for a chip that does two sums (of four variables) in parallel, in a software development language that you know, such as C, C++, Java, or Perl. Explain the issues that arise during development and why a chip design language, as described in this section, might help.

HOW ARE SPECIFICATIONS DEVELOPED?

The chicken-and-egg question that remains to be answered is, How exactly do we come up with the requirements that make up a design specification? This is a question that matters to the success of any chip project; without an accurate and clear specification, the success of a project is in jeopardy.

_Carballo_Book.indb 16 2/20/08 4:53:55 PM

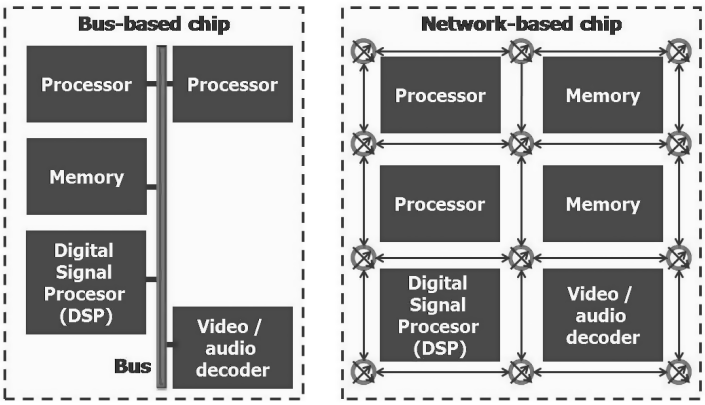

Fig. 3–5. Comparison of chips based on bus interconnections with networks on chip

Most modern chips connect large cores via an interconnection structure called a *bus*. Buses are the shared highways for on-chip (or off-chip, which is beyond the scope of this book) communications signals, with limited capacity. Because of its limited capacity and the target core's limited attention span, when a core (e.g., a microprocessor core) wants to talk to another core (e.g., a memory core, for writing on it or reading from it), it may need to apply for access to that bus beforehand. If the bus is not being accessed by another core to use that memory, then the bus can be used to make the connection and transfer the data.

Buses can be categorized into *blocking* and *nonblocking*. A blocking bus is used in complete bursts of data. Once the burst of data is transmitted, the bus is released and can be used by a different core or set of cores. Nonblocking buses are used on a cycle-by-cycle basis—that is, for every few clock ticks, some data may be transferred, but others may be able to access the bus in between. Here is a simplified example of blocking bus operation in SystemC:

```
Status read_in_bursts (start_address A, data *D,
length = 6, lock = FALSE, priority = 1)
{...wait(transmission_complete); ...}
```

In this example, the operation performed is a blocking *read* operation from memory, starting at memory address *A*, then reading six blocks of data, with top priority. Until this operation is finished, we cannot directly access this bus to perform other operations. *Status* is the value returned and indicates when the operation is complete. The *wait* statement indicates this blocking characteristic.

A similar function can be written for a nonblocking read that returns immediately. Thus, it does not block access to the bus and lets the simulation continue for the designer.

_Carballo_Book.indb 29 2/20/08 4:53:59 PM

Question. Why not create a prototype of the design on a board, so it can be simulated faster?

A prototype of almost any digital chip can be built by wiring a set of FPGAs on a board, thereby resulting in a fast and cost-effective solution. Building such a board and making changes on it (possibly rewiring the board and changing the FPGA's configuration), however, may take a long time, is subject to errors, and may not be easy to debug (because not all signals are accessible from outside). Making prototypes is clearly very useful as the design nears its end and is in a more stable state, after fixing most bugs. At this point, the focus is on simulating the chip very rapidly to uncover remaining bugs and possibly to ensure that the software runs well on the chip's processor.

Although emulation is typically focused on function and not so much on power, speed, or area, it can be used to create such estimates as well. After emulation, the chip can be designed following the aforementioned method. Function is not to be discarded, though; functional bugs found may account for up to 70% of the time and people required in order to complete a chip and serve it during its life cycle.

_Carballo_Book.indb 44 2/20/08 4:54:03 PM

The basic mapping to a technology library is the "dumb" (although not uncomplicated) part. Mapping needs to be done in an extremely efficient manner, because today's circuits need to be very fast and have a very low power consumption. For each logic operation, the best combination of logic library cells needs to be selected.

Logic synthesis tools are semiautomated tools that perform a set of operations, mostly sequential, including the following:

- *Simplify logic (no redundancy).* Logic descriptions typically start out as redundant. Signals and variables that could be shared are often not identified. Logic synthesis identifies signals and logic gates that can be shared among various functions, so that the area and power consumption can be improved.

- *Reorganize logic (efficiency).* Logic synthesis also reorganizes logic so that it can be more effectively mapped to the gates in the library.

- *Decompose logic (so that it maps to existing library gates).* As mentioned earlier, logic descriptions are usually not expressed, so this mapping needs to be done anyway.

There are many ways in which logic synthesis could map a logic description into library gates. Some criteria need to be applied for what amounts to be an optimization process. The most popular criteria include the following:

- *Timing.* The netlist is generated such that it is either the fastest possible or guaranteed to meet a certain speed requirement. This tactic usually implies that the resulting circuit consumes more power and takes more area than it would otherwise.

- *Area.* Gates are optimized for area; thus, the smallest possible size for each individual gate that satisfies all other constraints is chosen. As a result, resulting chips coming off the fabrication line may be slower on average yet consume less power than they would using a different tactic.

- *Power.* Gates are optimized for power consumption. This usually implies an area-efficient approach, as smaller gates tend to consume less power. However, it is typically at the expense of speed. This approach is more commonly used for low-power devices, especially in portable consumer applications such as cell phones.

Figure 4–7 depicts graphically the logic synthesis process, including its various inputs and outputs. As figure 4–7 indicates, one key input of the logic synthesis

_Carballo_Book.indb 58 2/20/08 4:54:07 PM

or more wires or device pins) that gets stuck at 1 or 0 permanently, regardless of what the circuit is actually doing. The fault model refers to a certain type of manufacturing defect, but theory has shown it to be a very effective way to provide test inputs for combinational logic. If a design team can provide tests that check for every node at which the circuit might get stuck, then the circuit is said to have 100% stuck-at fault coverage, a widely used benchmark in chip testing.

Other fault models exist. For example, *shorts* tests for unwanted short circuits between two or more wires, and *opens* tests for unwanted separation between two nodes that are supposed to be connected. DFT is most often focused on producing the right logic and test inputs/outputs, to ensure that these faults are tested for when the chip comes off the manufacturing line.

As with most other design tasks, DFT can be divided into at least two subtasks: test synthesis and test verification.

Test synthesis

The goal of test synthesis is apparently simple: to make the chip (or circuit) more testable. What exactly does this mean? Testing the chip is a complex process that can vary widely, depending on the type of chip and its market application.

There are two key goals of test synthesis that will benefit any specific chip or circuit block: test logic generation and test input generation.

Test logic generation. In this step, the designer, with the help of DFT tools, generates extra logic that helps identify and/or diagnose faults. This task is generally pursued in the early design stages. Key goals are to help feed test vectors (data inputs) easily into the circuit and to help take out the results for evaluation once the chip has run on these vectors.

The most common approach to test logic generation is *scan-based DFT*. Scan-based DFT converts a certain number of the latches in a design into scannable latches (defined and discussed earlier). By choosing scannable latches, a sequential logic design is converted, for testing purposes, into something close to a combinational circuit. With scannable latches, one can insert any input vector desired into any of the latches in a very quick and efficient manner (i.e., by scanning in the input data). Without scannable latches, one would have to insert some data at the pins of the circuit, which would then propagate naturally, after a number of clock cycles, to the latch where we want those data to be. The number of cycles necessary for just that one test case could be thousands or millions, which drives testing cost and time up significantly—potentially to thousands of dollars per chip. Therefore, scan logic test generation makes a circuit more testable. In addition to scan latches, other logic may need to be added to improve the testability of the circuit.

_Carballo_Book.indb 86 2/20/08 4:54:14 PM

Design entry

Analog circuit design is the oldest type of circuit design. Circuit design entry is, in this case, done using a schematic editor. For many years, schematics have been the most common format used to describe, explain, enter, annotate, and evaluate analog circuits. It just makes sense. The process is as close as one can get to drawing the circuit on a piece of paper or a white board.

Schematic entry is therefore a very important piece of analog circuit design, as it is the main description of the circuit itself! Accordingly, there are several goals of schematic entry, as outlined using examples in the following paragraphs.

The first goal of schematic entry is to describe and document the goals of this circuit (including its requirements) and the approach that will be followed in designing it.

Example. Consider having an annotation in the schematic that says "This is analog amplifier version 1.0, in 0.13 μm CMOS technology, and expected gain of 40 decibels." The gain of an amplifier is the amplification multiple that is applied to its input, to obtain its output—that is, a ratio between the output voltage amplitude (voltage gain) or power consumption (power gain) and the input voltage amplitude or power consumption, respectively. A decibel is a logarithmic metric without units, with a base of 10, that expresses an amount relative to a certain reference, typically power consumption. Because it is logarithmic, very large or very small amounts can be represented. For metrics such as amplification gain, it can be expressed as

$$\text{Gain} = 20 \log \frac{V_{\text{out}}}{V_{\text{in}}} \tag{5.3}$$

based on a voltage ratio, or

$$\text{Gain} = 10 \log \frac{P_{\text{out}}}{P_{\text{in}}} \tag{5.4}$$

based on a power consumption ratio.

Since power consumption is roughly proportional to the square of the voltage, six decibels is equivalent to a twofold gain in voltage, and three decibels is equivalent to a twofold gain in power consumption.

The second goal of schematic entry is to simulate the circuit. An extensive discussion of how simulation occurs in digital circuits has already been undertaken. In the case of analog circuits, a few aspects stand out as different:

- Devices need to be simulated that do not show up in most digital circuit schematics (at least not before extraction from layout has been executed).

_Carballo_Book.indb 114 2/20/08 4:54:25 PM