# Book Reviews

## BIST the hard way

**Scott Davidson**
Sun Microsystems

> **Reviewed in this issue**
> *A Designer's Guide to Built-in Self-Test*, by Charles E. Stroud (Springer, 2002, ISBN 1-402-07050-0, 344 pp., $125).

**I CAN HARDLY THINK** of a person better qualified to write a book titled *A Designer's Guide to Built-in Self-Test* than Chuck Stroud. As he recounts in his must-read preface, he implemented BIST in ASICs while at Bell Labs in the 1980s. To do this, he had to find BIST algorithms in the literature or develop his own, build his own tools, solve the many problems that don't turn up until you actually implement a technique, and convince his management to use the results. He was so successful that his colleagues also implemented BIST in their designs.

BIST is conceptually very simple: You move test generation and the collection of test results onto the chip. If the logic is close to combinational, or the memory is testable with an algorithm that's relatively easy to implement, you get good results and high fault coverage. But real life is not so simple, and one of the strengths of this book is its attention to the traps that a BIST designer will run into. Stroud well illustrates these traps in his many footnotes, describing cases where he ran into these problems.

Stroud begins the book with a short overview of BIST, starting with the most basic principles. But even here he takes care to mention when a BIST architecture is not usually used in practice. Chapter 2 covers fault models, simulation, and detection. These descriptions are up to date, covering the relatively recent concept of N-detection. ATPG, usually covered in testing books, is not covered here, which keeps the focus on BIST as an alternate strategy. This chapter is quite comprehensive, including things like fault sampling and fault collapsing.

Chapter 3 covers DFT concepts such as scan and boundary scan, which are some of the prerequisites for practical logic-BIST design. I doubt that you could learn DFT from this chapter, but it is an excellent refresher.

The next seven chapters cover BIST for logic. Chapter 4 is one of the most readable descriptions of test pattern generators that I have ever seen. It covers counters, linear feedback shift registers, and cellular automata, all at a reasonably high level. The chapter is practical, giving good reasons for all choices made in actual implementations. Readers will not get enough information to become an expert on test pattern generators from this chapter alone, but they will be able to understand which type of generator fits which application. The next chapter covers output response analyzers with an emphasis on aliasing and how to prevent it.

Chapter 6 describes the use of BIST in the factory and field. Stroud provides a list of requirements for system-level BIST use, which should prove valuable. I have one addition for this list: The IC running BIST should be resettable without power cycling. Designers do forget things like this and the items on Stroud's list. One important use of BIST is absent from this chapter, though. BIST is an excellent technique for testing hard IP cores without having to reveal the core's details to the customer.

The following chapters cover four logic-BIST techniques: built-in logic block observers, and pseudo-exhaustive, circular, and scan BIST. These chapters provide an excellent overview of BIST techniques used in real designs. I must admit that I am not impartial, having been somewhat involved in the work on pseudo-exhaustive and circular BIST at Bell Labs, work which few people today know about. The chapter on scan BIST, the most commonly used, is good at laying out the challenges of making this technique provide high coverage for real designs.

The next chapter, on non-intrusive BIST architec-


0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

tures, is a mixture of device-level techniques that don't affect the core logic, and system-level techniques. Although interesting, I am afraid this chapter might be confusing; it's less coherent than the other chapters.

The most widely adopted form of BIST is not used to test logic—the subject of the book so far—but to test memories. Although Chapter 12 covers the basics of this topic, I don't think this important subject gets its due. This chapter is shorter than the subsequent one on BIST for ROMs and programmable logic arrays. It lacks discussion of the sharing of BIST engines between memories, of programmability, and of bit mapping for diagnosability. Nor does it discuss external memory BIST, in which a BIST engine on an IC tests stand-alone memories elsewhere on the board. The reader is more likely to have come across this type of BIST than any other, and it's a shame that the chapter does not elaborate on it.

Chapter 14 covers mixed-signal BIST. Mixed-signal BIST is hardly a mainstream area yet, but this chapter is a good beginning for those who are interested, and I am glad to see it in this book. The final chapter deals with concurrent fault detection. It's a quick overview of the subject, with a welcome focus on what is practical.

**STROUD STAYS TRUE** to the title of his book throughout, never forgetting to provide hints to a designer interested in implementing BIST. Each chapter on a BIST technique ends with a table comparing the approaches, an excellent idea that all books reviewing DFT methodologies should adopt. This book stays at just the right level. Most designers today will implement BIST using commercial tools, and they can learn what the tool is doing to their design here. In fact, I wish the book had given some mention of BIST tools commercially available today. Although details on companies and products would be inappropriate, a naïve reader might think that he must implement BIST by hand or with homegrown tools—just like Stroud did in his pioneering work 20 years ago. This book will give the reader the benefit of his experience. ■

■ Direct questions, comments, and contributions for this department to Scott Davidson, Sun Microsystems, 910 Hermosa Court, M/S USUN05-217, Sunnyvale, CA 94085; scott.davidson@sun.com.

SET INDUSTRY STANDARDS

wireless networks
gigabit Ethernet
enhanced parallel ports
802.11 token rings
FireWire

Computer Society members work together to define standards like IEEE 1003, 1394, 802, 1284, and many more.

HELP SHAPE FUTURE TECHNOLOGIES • JOIN A COMPUTER SOCIETY STANDARDS WORKING GROUP AT

computer.org/standards/