Guest Editor's Introduction:

Test Synthesis—Many Things to Many People

Butterflies undergo tremendous change, quietly assuming different forms. Test synthesis behaves similarly, and has become many things to many people—what the end user really wants, what electronic design automation vendors currently supply, and what researchers see on the horizon. This issue of IEEE Design & Test reflects all these forms, drawing on the expertise demonstrated at the first IEEE International Test Synthesis Workshop (Santa Barbara, May 1994) and the Test Synthesis Seminar held as part of the 25th IEEE International Test Conference (Washington, D.C., October 1994).

Robert C. Aitken (Hewlett-Packard) sets the scene with an end-user perspective of test synthesis. He points out that “Everyone seems to know what test synthesis is, but few people agree on any given definition.” Is it something that takes place at the gate level? Or does it occur at a higher level, such as the register-transfer level? Aitken discusses both gate- and register-transfer level test synthesis and generally surveys the approaches taken by EDA vendors, commenting on each method’s capabilities and limitations.

He sees gate-level tools as implementing a two-pass process. First, they synthesize from a hardware description language, such as VHDL or Verilog, down to the gate level. Then, they resynthesize to build scan-path structures and generate patterns.

RTL approaches lie more in the single-pass domain and synthesize testability structures in the main design synthesis process. EDA vendors develop their current tools from another synthesis tool’s gate-level netlist, but a clear trend for the major vendors is to upgrade their products to a single-pass system.

Aitken concludes with advice on choosing a test synthesis tool set. He points out that first-time users should carefully think through their requirements from the perspectives of methodology and final test. What counts is not any single tool’s speed, but the overall tool flow’s ability to create a testable device complete with a working test program. (In other words, go for the big picture before evaluating detailed features.)

R.G. (Ben) Bennetts
Synopsys, Inc.

Continuing with the pragmatic viewpoint, Henry Cox (Symplify Corporation) presents how one EDA vendor carries out the front-end rule-checking phase of a two-pass test synthesis tool set. Rule checking is critical to the downstream operations of scan-path insertion and pattern generation. For a given DFT methodology, such as full scan with multiplexed D flip-flops, rule checking identifies potential problems, such as race conditions during scan. It also assesses the controllability and observability of the state flip-flops and carries out several checks based on the limitations of tools downstream. Cox presents detailed examples of rule violations and describes how to overcome them.

The next two articles give us a glimpse of the test synthesis horizon. Kwang-Ting (Tim) Cheng (University of California, Santa Barbara) addresses pattern generation for partial-scan sequential circuits, a practical problem at the back end of a test synthesis flow. A problem occurs in circuits based on multiplexed D flip-flops if we use the system clock as the scan clock during the scan operation: Nonscan flip-flops change their state during scan-in and scan-out. This state change invalidates the usual
assumption of the reverse-time sequential pattern generation procedure—that is, that any state can be scanned into the scan register without affecting the state of nonscan flip-flops. We can solve this state retention problem by

1. using the scan-enable control signal to gate the clock between the scan and nonscan elements such that, during scan mode, the nonscan elements are not clocked
2. converting the scan elements into clocked-scan flip-flops that use a separate scan clock for scan shifting and disabling the system clock during this operation
3. extending sequential pattern generation capability to accommodate the state change of the nonscan elements

Method 1 carries a clock-skew penalty and method 2 requires greater silicon area. Cheng addresses method 3 and describes an extension to the Back algorithm accounting for the behavior of all flip-flops during a single (reverse-time) clock pulse. Results based on the ISCAS89 benchmark circuits (from the 1989 International Symposium on Circuits and Systems) indicate a potential loss of coverage (up to 4.2%) compared to gated-clock solutions. However, this method has the advantage of applying tests in real time, thereby potentially detecting propagation-delay faults.

The last article is the collaborative effort of Carnegie Mellon and McGill Universities. Thomas Marchok et al. explore the implications of retiming on pattern-generation runtime and fault coverage. Retiming occurs in sequential circuit synthesis and involves replacing state flip-flops across sections of the combinational logic to improve a circuit’s time-related behavior, usually at the expense of area. At the same time, retiming can increase the runtime of a sequential pattern generation tool and decrease fault coverage compared to the original circuit. This occurs because the retimed and original circuits’ state transition graphs differ, but their I/O behavior remains the same. In particular, retiming can introduce equivalent (redundant) and invalid states that are unreachable from a reset state. Both affect the initialization of a circuit to a unique reset state via a synchronizing sequence and further complicate the pattern generation procedure.

Experiment results indicate retiming’s impact on runtime and fault coverage. In one circuit, coverage dropped from 99.3% to 54.6% for a set of patterns that took 11.6 times longer to generate. The authors explore possible causes of these results and finally conclude that since retiming does not introduce logically redundant faults, the test sets generated for the original circuit are also valid for the retimed circuits. They come to this conclusion by observing that retiming alters the clock cycle, but not the sequence, in which logic values move through a circuit; further experiments confirm this. The authors recommend a methodology in which we design the circuit and tests, implement retiming, and then confirm fault coverage with a fault simulator.

This issue attempts to portray the many faces of test synthesis, and overall the authors present a balanced view. I believe test synthesis is about to undergo a formidable metamorphosis—from the current 2-pass, low-level approach to the higher level, more integrated 1-pass approach. And when this happens, the synthesis of hardware testability structures will become so fully integrated into the design process that “design for test” will cease to be a visible and painful overhead. Instead, it will assume its rightful place as one of the many constraints at the start of the design process, challenging in its implementation, but rewarding in its value to the final product.

My thanks go to the authors and referees for their efforts in bringing these articles to publication.

K.G. (Ben) Bennetts is a senior test consultant with Synopsys, Inc. Prior to joining Synopsys, Bennetts was a freelance consultant and lecturer on DPT topics with Bennetts Associates, Southampton, UK. His consultancy has included technical and marketing activities in test technology, test strategies, and DFT techniques, working with companies such as Philips and Alcatel. He has given tutorials at the IEEE International Test Conference and the European Test Conference. Bennetts was program chair for the first three ETCs and has published over 70 papers and two books on test subjects. He was an invited speaker at ITC 92 and is currently general chair of the IEEE International Test Synthesis Workshop.

Address questions or comments about this special issue to Ben Bennetts at Synopsys (Northern Europe) Ltd., Imperium, Imperial Way, Worton Grange, Reading RG2 0TD, UK; benb@synopsys.com.