High-Level Synthesis and Implementation of Built-In Self-Testable Data Path Intensive Circuit

Han Bin Kim

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Dong S. Ha, Chair
J. R. Armstrong
F. G. Gray
S. M. Henry
S. F. Midkiff

December 15, 1999
Blacksburg, Virginia

High-Level Synthesis and Implementation of Built-In Self-Testable Data Path Intensive Circuit

Han Bin Kim

ABSTRACT

A high-level built-in self-test (BIST) synthesis is a process of transforming a behavioral description into a register-transfer level structural description while minimizing BIST overhead. Existing high-level BIST synthesis methods focus on one objective, minimizing either area overhead or test time. Hence, those methods do not render exploration of a large design space, which may result in a local optimum.

In this thesis, we present three methods, which aim to address the problem. The first method tries to find a register assignment for each *k*-test session in a heuristic manner, where $k=1,2,\ldots,N$ and N is the number of modules of the circuit. Therefore, it offers a range of designs with different figures of merit in area and test time. The second method is based on integer linear programming (ILP). The proposed ILP based method performs the three tasks, assignments of registers, interconnections, and BIST registers, concurrently to yield optimal or near-optimal designs. We describe a complete set of ILP formulations for the three tasks. The ILP based method achieves optimal solutions for most circuits in hardware overhead, but it takes long processing time. The third method, the region-wise heuristic method. It partitions a given data flow graph into smaller regions based on control steps and applies the ILP to each region successively to reduce the processing time.

To measure the performance of BIST accurately and to demonstrate the practicality of our BIST synthesis method, we implemented a DSP circuit; an 8×8 two-dimensional discrete cosine transform (DCT) processor. We implemented two versions of the algorithm, one with incorporation of our BIST method and the other without BIST, to verify the validity of our simplified cost model to estimate BIST area overhead. The two major parts of the circuit, data

path and controller, were synthesized using our high-level BIST synthesis tool. All the circuits are implemented and laid out using an ASIC design flow developed at Virginia Tech.

Experimental results show that the three proposed high-level BIST synthesis methods perform better than or comparable to existing BIST synthesis systems. They indeed yield various designs that enable users to trade between area overhead and test time. The region-wise heuristic method reduces the processing time by several orders of magnitude, while the quality of the solution is slightly compromised compared with the ILP-based optimal method. The implementation of DCT circuits demonstrate that our method is applicable to industry size circuits, and the BIST area overhead measured at the layout is close to the estimated one.

Acknowledgements

I would first like to thank to Dr. Dong S. Ha, my advisor, for incessant support and guidance toward excel in this research and writing a paper. The achievements described in the dissertation were possible by numerous people's help.

I would like to give thanks to people in Advantest, Mr. Yamaguchi Takahiro, Mr. Takeshi Takahashi, and Mr. Mashiro ishida who involved with initiating and taking care of this High-Level BIST Synthesis project. Special thanks are directed to Mr. Takeshi Takahashi who was willing to help me in programming and setup the fundamental structure of the project.

One of the most valuable supports to solve a high-level synthesis problem comes from Dr. Tae Hyung Park who provides me a lot of guidance and insight involved with integer linear programming.

The library development project, although not research-oriented one, was one of the toughest tasks, accomplished with the help of Mr. Tae Hoon Kim and Mr. Jun Suk Kim at Cadence and Dr. Kwang Hyun Kim at Samsung. I also appreciate help from our group members, Carrie Aust, Jia Fei, Meenatchi Jagasivamani, and Jos Sulistyo. I'm also thankful to Andrew Bouldey in helping documentation of User's Manual. I have enjoyed discussions and spent a lot of good time with our group members, Byung Ha Joo, Mohammed Osman, Prashanth Aprameyan, Kumaravel Jagasivamani, Ranpara Samirkumar, and Suk Won Kim. I also would like to thanks to my committee members for their valuable comments and feedbacks.

Finally, I would like to appreciate my family: Myung Joo Lee, my wife, Barron, my son and Ko Un, my daughter who spend their time not much with me but always welcome and support me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The design process of a digital system, especially of a digital signal processing system, usually starts at a behavior level and descends to a structural level. Test synthesis is to incorporate design-for-testability features in the design process. Until the end of 80's, testability was usually inserted into a structural level (mostly at the gate level) as a post processing of the logic design. However, the approach often fails to yield a good solution in terms of hardware overhead, fault coverage and testing time due to an inappropriate choice of hardware structure made in an earlier design stage.

In order to address the problem, researchers investigated the incorporation of testability into the front-end of the design process called high-level test synthesis [1]-[26]. High-level synthesis is to transform a behavioral description of a design into a structural implementation comprised of data path logic and control logic [27]. High-level test synthesis incorporates some testability feature(s) during the high-level synthesis, and the resultant circuit is easier to test than the original circuit in which testability is not considered. An excellent survey for high-level test synthesis is available in [1].

Design-for-testability (DFT) technique [28] deals with improving testability of a circuit under design. While improving the testability, it degrades some of design characteristics such as area, performance, and power consumption. However, DFT reduces test cost, improves product quality, and reduces time-to-market. DFT technique has, in generally, three different approaches: ad-hoc design-for-testability, scan-based design, and BIST-based design.

Most common ad-hoc DFT technique is a test-point insertion in which a hard-to-test node is identified and a dedicated access port is added to the node. Other ad-hoc technique includes abiding rules such as avoidance of asynchronous circuit and redundant logic. Scan-based DFT technique is most widely used. Some or all system registers are modified to be *scannable* by

which internal nodes can be controllable and observable from the outside. It results in better testability while sacrificing area and performance. Scan design is primarily for improving accessing of internal nodes. However, it is often hard to access I/O pins when a chip is embedded in a multiple-layered board. Scan design can be employed along I/O pins of a chip, which is referred to as *boundary scan*, or *external scan.* BIST-based DFT techniques becomes popular not only for embedded memory testing, but also logic circuit testing. Built-in self-test (BIST) is one of design-for-test (DFT) techniques in which a test pattern generator (TPG) and a signature analyzer (SA) are embedded on the same chip [28]. A BIST is necessary to test hard-to-access internal nodes such as embedded memory. Also it has other advantages such as i) at-speed testing is possible, ii) the need for test pattern generation and test data evaluation is eliminated, iii) expensive automatic test equipment (ATE) can be avoided, and iv) field testing is possible. Parallel BIST, which is based on random pattern testing, employs test pattern generators and test data evaluators for every module under test (which is usually a combinational circuit). Parallel BIST often achieves relatively high fault coverage compared with other BIST methods such as circular BIST [29].

High-level synthesis is a process of transforming a behavioral description into a structural description comprising data path logic and control logic [27]. First, a behavioral description is converted into a control data flow graph (CDFG). Then operations are scheduled in clock cycles (scheduling), a hardware module is assigned to each operation (module assignment), and registers are assigned to input and output variables (register assignment). The basic tasks in high-level synthesis area such as scheduling, transformation, module assignment, register assignment, and interconnection assignment are briefly described in the following.

Scheduling determines the precise *start time* of each operation for a given data flow graph [27],[30]. The start times must satisfy the original dependencies of the graph, which limit the amount of parallelism of the operations. Transformation such as pipelining and retiming is to modify a given DFG, while preserving the functionality of the original DFG. A transformed DFG usually yields a different hardware structure, which leads to a different structure for testability and power consumption. Assignment tasks in the high-level synthesis can be solved

by the graph coloring algorithm. There have been a number of algorithms for graph coloring, but, three most popular methods used in high-level synthesis are left edge algorithm [31], clique partitioning [32], and perfect vertex elimination scheme (PVES) [33]. All the methods mentioned above are based on heuristic to shorten the processing, thus, they do not guarantee an optimal solution.

Integer linear programming (ILP) has been used to perform specific tasks in high-level synthesis. Hafer and Parker pioneered in formulating high-level synthesis problems into an ILP model in 1983 [35]. Since then, many researchers investigated ILP models to address synthesis problems [36]-[39]. Some recent works include Gebotys and Elmasry [38] and Rim et al. [39]. Gebotys and Elmasry [38] applied an ILP to architectural synthesis where a scheduling and a module/register allocation are performed concurrently. Rim et al. presented an ILP model to solve the binding problem with focus on minimizing hardware resources (such as modules, registers, and wires) [39]. Various ILP formulations for scheduling and binding problems are available in literature [30]. A major advantage of an ILP based approach is that the obtained solution is optimal though computationally intensive due to the inherent nature of ILP, which involves an exhaustive search.

High-level test synthesis generates register-transfer level data paths with enhanced testability. Depending on the testability schemes incorporated, high-level test synthesis systems can be classified into three groups. The first group of high-level test synthesis systems aims to improve the controllability and/or the observability of the circuit [3]-[8]. The methods are ones such as insertion of test points, minimization of cycles and/or sequential depth. The second group of high-level test synthesis systems incorporates the scan technique [9]-[16]. While incorporating the scan technique, the systems try to minimize sequential depth and/or cycles, and maximize the number of input and output registers. The third group of high-level test synthesis systems employs built-in self-test (BIST), specifically parallel BIST [17]-[26]. Parallel BIST, which is based on random pattern testing, employ test pattern generators and test data evaluators for every module under test (which is usually a combinational circuit). Parallel BIST often achieves relatively high fault coverage compared with other BIST methods such as circular BIST [29]. In

this thesis, we present a high-level BIST synthesis method that employs the parallel BIST structure.

In this thesis, we describe our three different methods for a high-level BIST synthesis system; heuristic based method, ILP based optimal method, and region-wise ILP based method. We describe our methods briefly in the following.

The existing high-level BIST synthesis methods, described in the section 1.3, focus on one objective, minimizing either the area overhead [17]-[24] or the test time [25], [26]. Hence, those methods do not render exploration of large design space, which may result in a local optimum. Another aspect that was overlooked in those methods is that area overhead and test time are often traded in BIST (as well as other design-for-testability methods). Therefore, it is more desirable to offer various design alternatives (with different area overhead and test time) to the designer, and let the designer choose a proper design for his/her needs. Our method intends to address those two problems.

For a scheduled and module-assigned data flow graph, our heuristic based method allocates signature registers which guarantee the circuit be tested in *k*-test session, where *k* is 1, 2, ..., N, and N is the number of modules. Our method [40] tries to find an optimal design (which incurs the smallest area overhead) for each k-test session. Hence, it explores a far larger design space compared with other methods. It also allows designers to trade area and test time. A designer whose concern is only area overhead can choose the most area efficient design among N designs, while one concerned with only test time chooses the design for *k*=1.

Another proposed method is based on ILP, which generates an optimal solution. The focus of high-level BIST synthesis is register assignment, which involves three subtasks, system register assignment, BIST register assignment and their interconnections (called interconnection assignment). To reduce the complexity involved in the assignment process, existing high-level BIST synthesis methods decouple the three subtasks and perform the subtasks sequentially at the cost of global optimality. We present ILP formulations for high-level BIST synthesis with an objective of minimizing the area for each *k*-test session where *k* is 1, 2, ...N and N is the number of modules. Hence, our ILP based method tries to find N optimal (in area) BIST circuits of

which a BIST circuit for a *k*-test session tests the entire modules in exactly *k* sub-test sessions. Hence, like our heuristic method presented in [40], our ILP-based method [41] offers a range of designs with different figures of merit in area and test time. Our experimental results show that the proposed approach successfully synthesizes BIST circuits for each test session for all six circuits experimented. All the BIST circuits are better in area overhead than those generated by existing BIST synthesis methods.

The last proposed method is a region-wise ILP method [42], which intends to reduce the processing time for our ILP, based method. The heuristic divides a given data flow graph into sub-regions and applies an ILP for each region successively. Our experimental results show that the proposed heuristic reduces the processing time by several orders of magnitude, while the quality of BIST designs is compromised moderately.

To evaluate our proposed high-level synthesis system, we implemented one of widely used DSP circuits; an 8×8 two-dimensional discrete cosine transform (DCT) processor. A brief overview of implementation of DCT circuit and related topics such as design flow and library development are presented in Chapter 4.

The thesis is organized in the following manner. We briefly explain high-level BIST synthesis, an ILP model, and describe necessary terms in Chapter 2. We present our three methods for high-level BIST synthesis, the heuristic based method, the ILP based method, and the region-wise ILP method in Chapter 3. In Chapter 4, we describe implementation of an 8×8 two-dimensional discrete cosine transform (DCT) processor, and an ASIC design flow. Chapter 5 contains experimental results for the three methods. The performance of our method is compared with other BIST synthesis methods. We also present results on a large BIST circuit synthesized by the proposed method and layout through a standard cell design approach. Chapter 6 concludes the thesis.