DECIDER: A Decision Diagram Based Hierarchical Test Generation System

Gert Jervan, Antti Markus, Jaan Raik and Raimund Ubar
Department of Computer Engineering, Tallinn Technical University
Raja 15, EE0026 Tallinn, Estonia
jaan@pld.ttu.ee

Abstract

Current paper presents a hierarchical test pattern generation system that uses register-transfer level VHDL and gate-level EDIF netlist descriptions as inputs. The system includes appropriate interfaces to synthesize Decision Diagram (DD) models, a DD based test pattern generator and a fault simulator to evaluate the quality of the generated tests. In the paper, the structure of the system is presented. Additionally, representation of different design abstraction levels using decision diagrams is explained. The performance of the system is compared to other state-of-the-art tools for sequential circuit test generation.

1. Introduction

As the degree of integration in VLSI designs has been growing over the years, so has the need for automation of different design tasks. Design automation helps to shorten the time-to-market cycle and increases significantly designer's productivity. Automation was first introduced on the lower levels of design tasks, such as placement and routing, and together with the growth of design complexities, moved gradually to higher levels, e.g. logic synthesis, high-level synthesis (HLS) and hardware/software co-design. Nowadays the goal is clearly to automate the entire design cycle from conceptualization to generation of silicon layout [1].

During recent years, more-and-more high-level synthesis tools have become available. These tools are used by designers to automatically generate Register-Transfer Level (RTL) descriptions from design's behavioral description. In the RTL descriptions the design is usually partitioned into a control part, i.e. a finite state machine, and a datapath part containing a network of interconnected functional units (FU). The HLS tools take into account several constraints, as speed, area, or testability, and allow the designer to quickly compare the trade-offs between alternative RTL implementations.

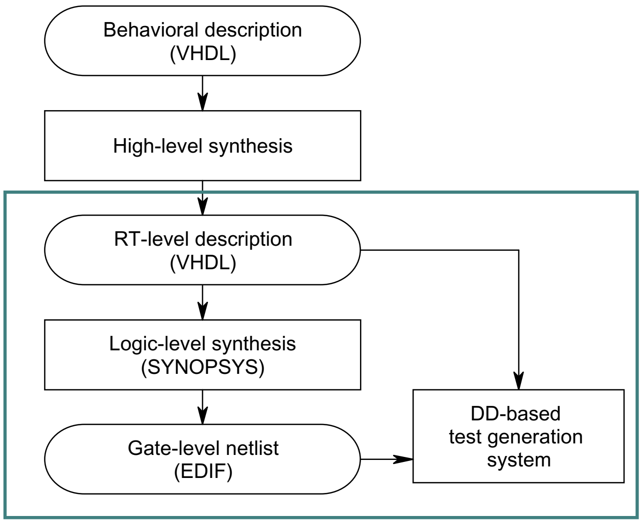
Behavioral description (VHDL)
High-level synthesis
RT-level description (VHDL)
Logic-level synthesis (SYNOPSYS)
Gate-level netlist (EDIF)
DD-based test generation system

Figure 1. The Design Cycle

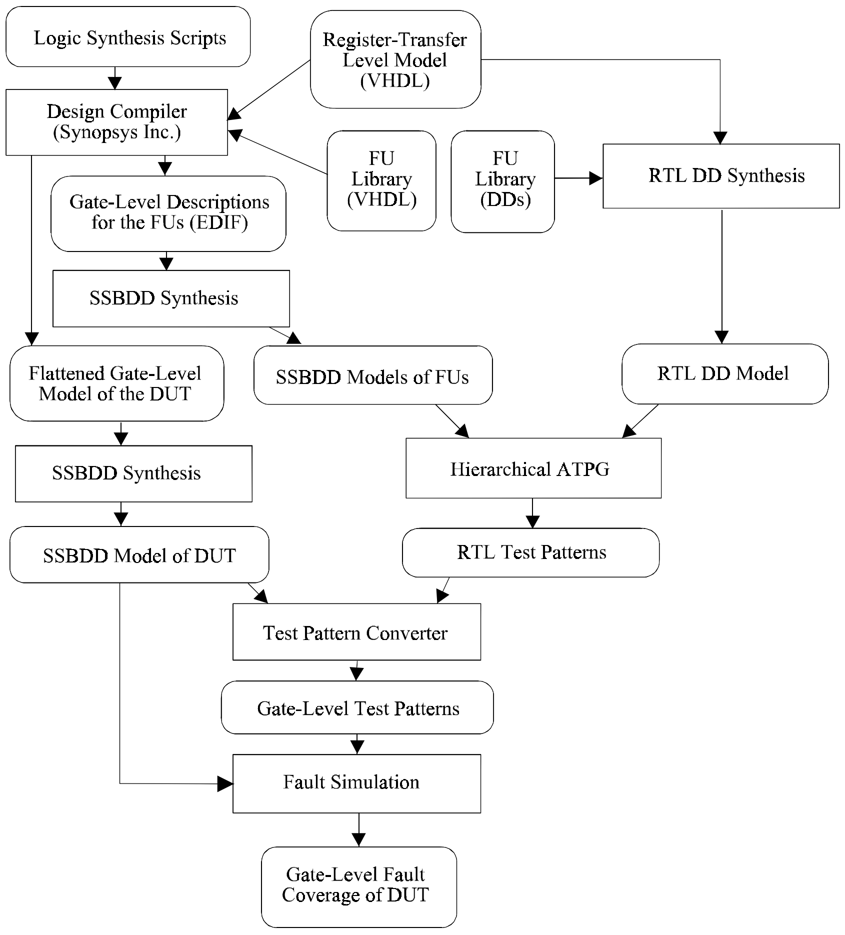

Figure 2. Data Flow of the DECIDER System

With the appearance of high-level synthesis, a number of automated test generation approaches [2,3] were introduced that took advantage of register-transfer level information while generating tests for gate-level faults. Current paper presents a hierarchical test generation system that operates on RTL and gate-level model descriptions. At present, the system utilizes Design Compiler [4] from Synopsys Inc. for logic-level synthesis. In Figure 1, the basic design flow and the place of test generation in it is shown.

The paper is organized as follows. Section 2 describes the structure of a novel DECIsion Diagram based test genERation system, called DECIDER. Section 3 explains how hierarchical design descriptions can be represented by Decision Diagram (DD) models. More detailed information about the approach can be found in [5]. In Section 4, experimental results and comparison with other tools for sequential circuit test generation are given. Finally, conclusive remarks are presented.

2. System Implementation

Figure 2 presents general structure of the DD-based test generation environment. It consists of DD interfaces from VHDL and EDIF formats, a hierarchical test pattern generator, a fault simulator and a number of libraries and scripts to support the DD synthesis. As an input for the system is design representation in RTL VHDL, where the circuit is partitioned into control and datapath parts. Similar representations can be synthesized by a HLS tool.

Additionally, two libraries of Functional Units (FU) are provided: one in VHDL and the other in DD format. The VHDL FU library describes the behavior of FUs that is necessary for the high-level synthesis tool to map the behavior of the circuit into a netlist of FUs specified in the library. The library is also used, along with the RTL VHDL model, during logic synthesis by Design Compiler. The DD FU library is required by RTL decision diagram synthesis tool to substitute the modules in the RTL model with corresponding decision diagrams.

Design Compiler performs logic synthesis and writes out gate-level EDIF netlists of the whole design and of each FU separately. Dedicated scripts for automating these tasks have been implemented. Subsequently, the netlists are converted into Structurally Synthesized BDD (SSBDD) models. (The concept of SSBDD model representations is briefly discussed in Section 3).

The hierarchical Automatic Test Pattern Generator (ATPG) operates on the SSBDD models of the FUs and on the RTL DD model of the circuit. As an output the ATPG generates test patterns. However, the patterns do not offer precise information about achieved fault coverage. In order to measure the actual gate-level fault coverage of the generated tests, the test patterns have to be fault simulated on the structural level description of the whole device.

3. Decision Diagram Representations

A Decision Diagram (DD) is defined as a non-cyclic directed graph whose nodes are labeled by variables (constants or algebraic expressions). For each value from a set of predefined values of a non-terminal node variable, there exists a corresponding output branch from the node. Consider a situation where all variables are fixed to some value. By these values, for each non-terminal node a certain output branch is chosen entering into a successor node. Let us call these connections between nodes - *activated branches* and the chains of them - *activated paths.* For each combination of values of variables, there exists a *main activated path* from the root node to some terminal node. This relation describes a mapping from a Cartesian product of the sets of values for variables in all nodes to the joint set of values for variables in terminal nodes. Therefore, by DDs it is possible to represent arbitrary functions $Y=F(x)$, where Y is the variable whose value will be calculated on the DD and x is the vector of all variables in nodes of the DD.

As a hierarchical input to the test generator are descriptions where the architecture of the circuit is described at the RT-level and the low-level structure is given at the gate level. Both these levels can be described by DD models. In the RTL descriptions, designs are partitioned into datapath and control parts, where datapath is represented structurally by a netlist of interconnected blocks. The building blocks of datapath are registers, multiplexers and functional units (FU), where functions can be arbitrary arithmetic or logic operations.

Datapath can be represented by a system of DDs, where for each primary output and for each register, a DD corresponds. In the DD models, the non-terminal nodes correspond to control signals and terminal nodes represent operations. Register transfers and constant assignments are treated as special cases of operations. Activated branches between the nodes determine, which operation is assigned to the variable represented by DD with each value combination of the control signals. Figure 3 shows an example of a DD representation for a datapath register.

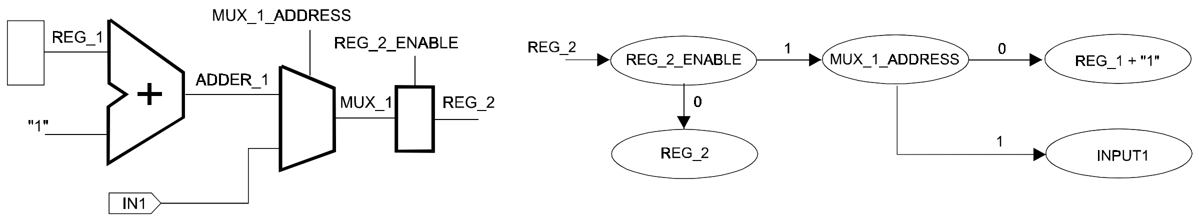

Figure 3. DD representation for a Datapath Fragment

The control part of an RTL description is described as a Finite State Machine (FSM) state table. Similar to datapath, the state table can be represented by a DD model. In that case, the non-terminal nodes correspond to current state and conditions (FSM inputs) and terminal nodes hold vectors with the values of next state and control signals (FSM outputs). Figure 4 shows an example of a fragment of an FSM state table and the corresponding DD representation. In the DD, *q* denotes the next state and *q'* denotes the current state value. Variables out1, out2, out3 and out4 are output signals of the FSM. The DD in Figure 4 describes the behavior of the FSM at the current state being equal to s5.

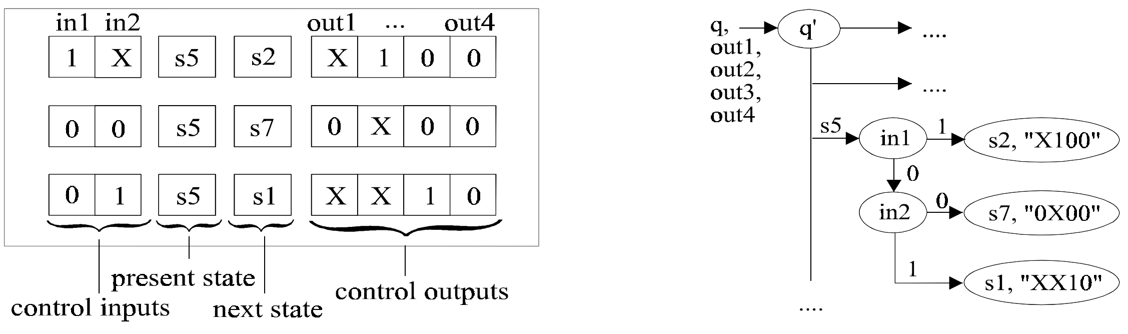

Figure 4. Representing an FSM State Table by a DD

In current hierarchical test generation approach, gate-level descriptions of the datapath modules are transformed into Structurally Synthesized BDD (SSBDD) models. Differently from BDDs, which represent function only, SSBDDs support test generation for gate-level structural faults without representing these faults explicitly. Furthermore, the worst case complexity for generating SSBDDs is linear in respect to the number of logic gates, while it is exponential for BDDs

SSBDD models for combinational circuits can be synthesized by a simple superposition procedure. We generate an SSBDD for a circuit output by starting from the output, substituting recursively all the gates by their respective elementary BDDs until primary inputs are reached. In order to avoid repetitive occurrences of subdiagrams in the model, the recursion can be terminated in fanout branches and SSBDDs can be synthesized for each primary output and fanout point separately. In that case the circuit will be described as a system where for each fanout-free region an SSBDD corresponds.

4. Experimental Results

Experiments were carried out on two highly sequential circuits, a Greatest Common Divisor (GCD), which belongs to the HLSynth92 benchmark suite, and an 8-bit multiplier example. The synthesized RTL version of the GCD circuit contains a datapath with 5 registers and an FSM with 12 states. The circuit was chosen as it reveals a number of difficult test generation problems. It contains a global data dependent loop and only one of the registers is directly observable. The multiplier mult8x8 has a complex 16-bit datapath containing several feedback loops.

Actual quality of the generated test sequences was measured by applying gate-level fault simulation to the circuits and by neglecting a set of obviously untestable faults (e.g. lines tied to constants). The achieved fault coverages for datapath parts were 95.1 % for the GCD circuit and 95.9 % for mult8x8, respectively. Although control part faults were not explicitly targeted, fault coverages measured for control parts were high. Achieved test generation times were short. Both of the circuits were tested in less than 20 seconds. The number of test sequences was less than 100. Due to the fact that sequential circuits were considered, each test sequence consisted of multiple clock cycles. Table 1 presents the experimental results, which were run on a 233 MHz Pentium II computer with 64 MB RAM under Windows 95 operating system.

Circuit	gcd	mult8x8
Number of gate-level faults	1066	4432
Gate-level fault coverage DP (%)	95.1	95.9
Fault coverage CP (%)	89.4	92.1
Total gate-level fault coverage (%)	91.8	94.4
Test generation time (s)	14.6	17.7
Number of generated test sequences	53	93
Total test length (number of clock cycles)	627	2797

Table 1. Test Generation Results

In Table 2, comparative results with a gate-level sequential test pattern generator HITEC [6], a genetic test pattern generator GATEST [7] and a novel hierarchical test pattern generation approach published in [8] are given. (Note that in [8], the high level test frames were generated manually). The comparison is carried out on the example of the GCD circuit, which is the only circuit common with the experiments in [8]. As we can see from the table, the proposed DD-based technique outperforms the other test generation tools in all categories. It achieves a higher fault coverage in a much shorter time and generates less test sequences than [8]. The number of test sequences for [6] and [7] is not known.

	DECIDER	Hier. [8]	GATEST [7]	HITEC [5]
fault coverage, %	**91.8**	90.4	62.6	74.4
time, s	**14.6**	1068	636	49320
test sequences	**53**	60	N.A.	N.A.

Table 2. Comparative Results

5. Conclusions

In current paper, the role of hierarchical test pattern generation in an automated design cycle is explained. The design of a hierarchical test generation system, called DECIDER, is presented. In order to provide automated test pattern generation for hierarchical systems, a number of diagnostic tools and appropriate interfaces have been implemented.

A novel hierarchical test generation approach based on decision diagram models is introduced. Differently from known methods, both, higher and lower design abstraction levels, and both, control and data paths are handled here by a uniform approach. In addition, different types of datapath components (registers, FUs, multiplexers) are handled in a uniform manner.

Comparison with the known state-of-the-art test generators for sequential circuits shows the advantages of the DD-based system. Experiments on the Greatest Common Divisor circuit indicates that the proposed tool achieves a significantly higher fault coverage with a speed 44 – 3400 times higher than in [6, 7, and 8].

References

1. D. Gajski, N. Dutt, A. Wu, S. Lin. *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.
2. J. Lee and J. H. Patel. Architectural level test generation for microprocessors. *IEEE Trans. Computer-Aided Design*, vol. 13, no. 10, pp. 1288-1300, Oct. 1994.
3. J. Lee, J. H. Patel. Hierarchical test generation under intensive global functional constraints. *Proc.29th ACM/IEEE Design Automation Conf.,* pp. 261-266, June 1992.
4. *Design Compiler Reference Manual Version 3.0*, Synopsys Inc., Dec. 1992.
5. G. Jervan, A. Markus, J. Raik, R. Ubar. Hierarchical Test Generation with Multi-Level Decision Diagram Models. *Proc. of the 7th IEEE North Atlantic Workshop*, West Greenwich, RI, USA, May 28-29, 1998.
6. T. M. Niermann, J. H. Patel. HITEC: A test generation package for sequential circuits. *Proc. of the European Conf. Design Automation (EDAC)*, pp.214-218, 1991.
7. E. M. Rudnick, J. H. Patel, G. S. Greenstein, T. M. Niermann. Sequential circuit test generation in a genetic algorithm framework. *Proc. of the Design Automation Conf.*, pp. 698-704, 1994.
8. E. M. Rudnick, R. Vietti, A. Ellis, F. Corno, P.Prinetto, M. Sonza Reorda. Fast sequential circuit test generation using high-level and gate-level techniques. *Proc. of the DATE Conf.*, 1998.
9. R. Ubar. Test Generation for Digital Circuits Using Alternative Graphs. *Proc. of Tallinn Technical University, Estonia*, No. 409, pp. 75-81 (in Russian), 1976.
10. R. Ubar. Test Synthesis with Alternative Graphs. *IEEE Design & Test of Computers*, pp. 48-57, Spring 1996.
11. IEEE Standard VHDL Language Reference Manual. *IEEE.* 1988.