# Automatic Constraint Based Test Generation for Behavioral HDL Models

*Siva Kumar Sastry Hari, Vishnu Vardhan Reddy Konda, Kamakoti V, V. M. Vedula and M. Kailashnath*
***Reconfigurable and Intelligent Systems Engineering Group (RISE)***
*Department of Computer Sc and Engineering, Indian Institute of Technology Madras,, India*

*http://www.cs.iitm.ernet.in*

## Abstract

*The proposed work involves conversion of a given circuit model into a set of constraints and employing constraint solvers to generate tests for it. The method is demonstrated for the 16-bit DLX-architecture.*

## 1. Introduction

The ever-growing demand for greater performance, complex functionality and faster time to market, coupled with the exponential growth in hardware size has resulted in the Functional Test Generation (FTG) being widely acknowledged as the bottleneck of the hardware design cycle. Random test generation does not guarantee the coverage of all the functionalities, especially for complex designs. This necessitates directed tests that can cover the corner cases not covered by random tests. This paper proposes a constraint-based Directed Behavioral Level Functional Test Generation (DBFTG) tool that is capable of generating test vectors given a behavioural level description of the Design Under Test (DUT) and a higher level specification.

The objectives of the proposed tool are as follows:

1. A fully-automated framework to generate directed tests for functional verification of any digital system, specifically of microprocessors, is proposed.
2. The proposed methodology accepts as input a behavioral level Verilog model and converts it into an Assignment Decision Diagram (ADD) based data structure called the Assign-Always-Module graph. The graph is further *optimized* and converted into a set of integer constraints.

Some of the features achieved in the proposed Directed Behavioral Level Functional Test Generation (DBFTG) tool are:

1. The input to the proposed technique is a behavioral level HDL model
2. Automatic generation of the constraint model from the behavioral model
3. Word-level constraints in contrast to bit-level constraints
4. A single constraint model with unified control and data paths
5. Handles sequential designs
6. Scalability with increasing design size
7. Verifying complex scenarios

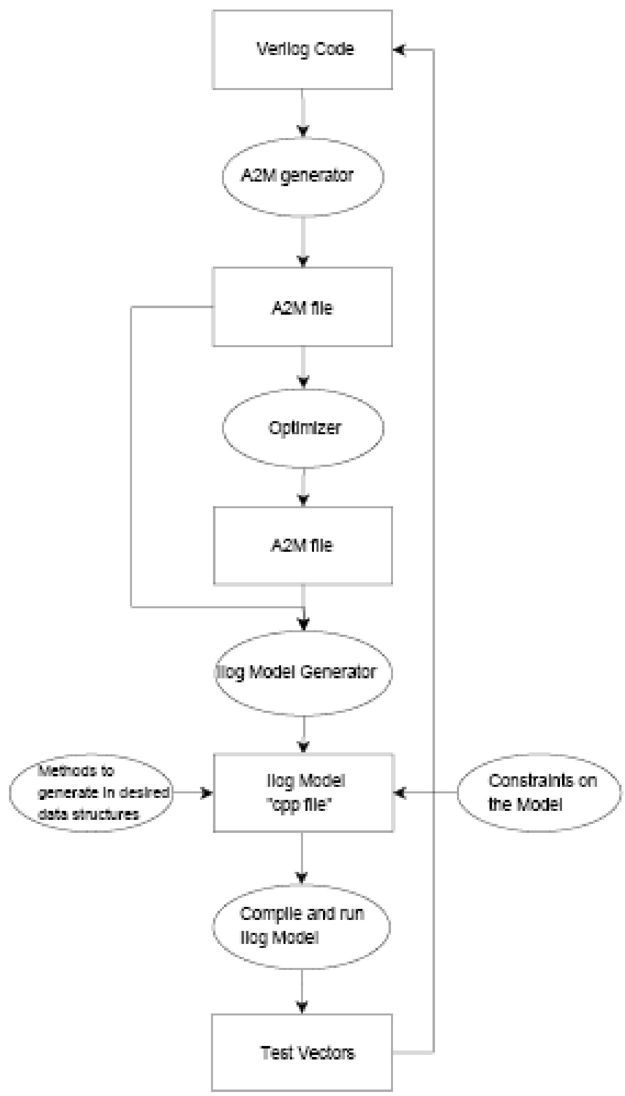


**Figure 1:** Tool Flow

## 2. The DBFTG Approach

Fig 1 shows the complete design flow for the proposed approach. The approach has three major phases:

1. Automatic Generation of the $A^2M$ graph from the given behavioral HDL model and its optimization;
2) Automatic Generation of the model constraints from the Optimized $A^2M$ graph; and,
3) Modeling of scenario constraints and generation of the functional test.

## 3. Experimental Results

Experiments were performed by employing the present method on a 16-bit DLX processor model. All the results reported are on a HP workstation xw4200. The time and memory utilized are as reported by the ILOG constraint solver. Table 1 shows that for a full adder circuit, the time and memory to generate the constraint model increases linearly with the number of time-frames unrolled. Table 2 shows the time required by various phases in generating the ILOG model for the 16-bit DLX processor model. The total time consumed for the entire three phases was around 0.35s. This illustrates the efficiency of the proposed methodology in handling large and complex designs.

| No of Time frames Unrolled | Time (in sec) | Memory (in MB) |
|---|---|---|
| 10 | 0.02 | 1.01 |
| 20 | 0.04 | 1.95 |
| 50 | 0.1 | 4.72 |
| 100 | 0.21 | 9.33 |
| 500 | 1.65 | 46.6 |
| 1000 | 4.48 | 92.13 |
| 10000 | 260.94 | 922.98 |

**Table 1 :** Results on Full Adder

| Phase | Timing |
|---|---|
| $A^2M$ generation | $0.022s$ |
| $A^2M$ optimization | $0.145s$ |
| Constraint Model Generation | $0.198s$ |

**Table 2 :** Timing Analysis

Table 3 shows the time taken for unrolling the DLX model to 10 time-frames and generating test vectors to access two specified memory addresses under three different scenarios. In the first scenario the time-frame numbers for the memory access and the memory addresses were different and independent of each other, while in the other two scenarios, the two memory addresses were dependent, leading to constraints that were dependent on each other. From table 3 it is seen that the constraint solver consumes less time and memory to solve *independent constraints* in comparison to *dependent constraints*. From table 4, it is seen that the constraint solver consumes more time and memory with increasing number of time frames for which the model is unrolled.

| Scenario | No of Time Frames | Time (in sec) | Memory (in MB) |
|---|---|---|---|
| Access two different Memory Addresses in two different Time frames | 10 | 14.8 | 134.53 |
| Access same Memory Address in time-frames 7 & 8 | 10 | 66.74 | 380.66 |
| Access consecutive Memory Addresses in time frames 7 & 8 | 10 | 68.3 | 380.69 |

**Table 3 :** Instruction Generation for Memory Access

| Scenario | No of Time Frames | Time (in sec) | Memory (in MB) |
|---|---|---|---|
| No RAW No WAW for 5 time-frames | 6 | 32.96 | 182.42 |
| No RAW No WAW for 5 time-frames | 10 | 84.98 | 379.92 |
| No RAW No WAW for 3 time-frames | 6 | 31.86 | 182.16 |
| No RAW No WAW for 3 time-frames | 10 | 81.97 | 379.31 |

**Table 4 :** Instruction Generation without Hazard

## 5. References


1. I. Ghosh, L. Zhang, and M. Hsiao, “Automatic design validation framework for HDL descriptions via RTL atpg,” in *Proceedings of the Asian Test Symposium*, 2003, pp. 148–153.
2. J. Lee and J. H. Patel, “Architectural level test generation for microprocessors, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13(10), October 1994, pp. 1288–1300.
3. V. M. Vedula, J. A. Abraham, and T. Larrabee, “Program slicing for hierarchical test generation,” in *Proceedings of the VLSI Test Symposium(VTS)*, Monterey, CA, USA, 2002, pp. 237–243.
4. L. Chen, S. Ravi, A. Raghunathan, and S. Dey, “A scalable software based self-test methodology for programmable processors,” in *Proceedings of the Design Automation Conference (DAC)*, 2003, pp. 548–553.
5. L. Chen and S. Dey, “Software-based self testing methodology for processor cores,” in *IEEE Transaction on Computer Aided Design and Integrated Circuits Systems*, vol. 20(3), March 2001, pp. 369–380.
6. S. Ravi and N. K. Jha, “Fast test generation for circuits with rtl and gate-level views,” in *Proceedings of the International Test Conference(ITC)*, 2001, pp. 1068–1077.
7. F. Fallah, S. Devadas, and K. Keutzer, “Functional vector generation for hdl models using linear programming and boolean satisfiability,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20(8), August 2001, pp. 994–1002.
8. J. Yuan, C. Pixley, A. Aziz, and K. Albin, “A framework for constrained functional verification,” in *ICCAD ’03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, Washington, DC, USA: IEEE Computer Society, 2003, p. 142.
9. D. Prasad, R. Archana, V. Karthik, Senthilkumar, V. Kamakoti, S. M. Kailasnath, and V. M. Vedula, “A novel unified framework for functional verification of processors using constraint solvers,” in *Proceedings of the VLSI Design and Test symposium (VDAT)*, 2006, pp. 418–426.


## Acknowledgement


The work was supported under the IIT Madras-Intel joint research initiative functional test generation for processor architectures.