# Cost/Quality Trade-off in Synthesis for BIST*

Peter Bukovjan[1] Laurent Ducerf-Bourbon[2] Meryem Marzouki[2†]

[1]ON Semiconductor, B. Němcové 1720, 756 61 Rožnov p/R, Czech Republics
Tel. (+420)651 667 147 - Fax. (+420)651 652 099
S20232@onsemi.com

[2]LIP6/ASIM Laboratory, Couloir 55-65, 4 Place Jussieu, 75252 Paris Cedex 05, France
Tel. (+33)1 44 27 39 67 - Fax. (+33)1 44 27 72 80
Laurent.Ducerf@lip6.fr Meryem.Marzouki@lip6.fr

## Abstract

*This paper details our allocation for Built-in Self Test (BIST) technique used in the Interactive Design for Test reuse in Allocation for Testability (IDAT) tool. IDAT tool objective is to fulfill the designer requirements regarding selected design and testability attributes of a circuit data-path to be synthesized. A related tool is used to synthesize a test controller for the final testable circuit. The allocation process of BIST resources in the datapath is driven by two trade-off techniques performed in order to: (1) at the local level, select the optimal set of Functional Units (FUs) to be BISTed, using a new testability analysis method and (2) at the global level, for each selected FU of this set, choose either to allocate its BIST version (when available in a library) or to connect it to an internal Test Pattern Generator (TPG) and Test Results Checker (TRC). When necessary, a last step of the process is the allocation of scan chains used to test the remaining untested interconnections. Experiments show the results of our allocation for BIST technique on two benchmarks.*

**Keywords:** BIST, synthesis for testability, DFT reuse, testability analysis

*This work has been partly supported by MEDEA AT403-SMT Project, by a grant from the Grant Agency of Czech Republic, contract No. 102/98/1463 and by ON Semiconductor Czech Design Center in Rožnov p/R.

†Corresponding author and presenter

## 1 Introduction

Built-In Self Test (BIST) techniques are based on the inclusion of test structures into a device so that it becomes able to test itself. This is achieved by the addition and corresponding connection of one or more internal Test Pattern Generators (TPGs), usually in the form of Linear Feedback Shift Registers (LFSRs) and one or more internal Test Result Checkers (TRCs) often in the form of Multiple-Input Signature Registers (MISRs). TPG and TRC can also be implemented in the form of Built-In Logic-Block Observers (BILBO) or CBILBO registers, which can work as both TPGs and TRCs. Historically, BIST techniques have concentrated on Functional Units (FUs) test, but some recent approaches also allow interconnection testing. Detailed background on BIST techniques can be found in [1].

Like all techniques attempting to improve the circuit testability, BIST techniques are used under the key constraint of reaching a required testability while keeping reasonable the area overhead and the test time. Approaches based on the use of BIST registers (usually BILBOs) typically have the main objective of maximizing the sharing of these registers among tested blocks and minimizing their number (as for example in [8] or in [9]).

In [2] selected existing registers and FUs are used during the test phase as various TPGs and TRCs (LFSRs, arithmetic generators, etc.) to test remaining FUs. The extension of this approach to interconnection testing is achieved in [3]. This test is ensured by test point allocation. The disadvantage of this technique is the necessity of fault simulation to identify

unobservable faults at each iteration step.

In this paper we propose a new allocation for BIST technique which optimizes, from the testability point of view, cost/quality trade-off relation during circuit synthesis. The described technique, which is already implemented in the data-path allocation for testability tool IDAT, takes into account the availability of a test controller, synthesized using a complementary tool. The methodology is based on two trade-off processes techniques performed in order to: (1) select the optimal set of Functional Units (FUs) to be BISTed, using a new testability analysis method and (2) for each selected FU of this set, choose either to allocate its BIST version (when available in a library) or to connect it to a unique internal Test Pattern Generator (TPG) and a unique Test Results Checker (TRC). The proposed allocation for BIST process is iterative and interactive (driven by a set of users requirements). The TPG and TRC are shared by the FUs.

## 2 Overview of IDAT tool

The Interactive Design for test reuse in Allocation for Testability (IDAT) tool is used with a classical High-Level Synthesis (HLS) system to create an RTL circuit architecture optimized not only according to area and performance criteria, but also according to testability criteria. The whole allocation for testability process is viewed as the automatic generation of testable architecture starting from a draft RTL architecture enriched by selected design and testability attributes (this enriched architecture is called *Rich Virtual Architecture* (RVA)), and making profit of the *Design for Test (DFT) reuse* concept. IDAT tool is able to modify existing data-path architecture in order to insert either test points, scan or BIST structures into the circuit. IDAT is complemented by a tool synthesizing the on-chip test controller which implements the test plan (under development).

IDAT allocation for testability process consists in five steps: (1) extraction of the RVA architecture from the allocation step result of HLS system, (2) extraction of testable versions of allocated components (FUs) from a library, (3) calculation of design and testability attributes (detailed in [4]) for original circuit, where the data-path is already separated from the original circuit controller by a block of multiplexers, (4) cost/quality trade-off process itself, according to the testability methodology chosen by the designer: available methodologies in IDAT are test point insertion, scan test and BIST. This process relies at each iteration step on the calculation of circuit attributes, (5) Modified (testable) RTL data-path return to High-Level Synthesis system, in order to have precise scheduling be completed.

## 3 BIST architecture

An example of the draft RTL architecture taken as input by IDAT is depicted in figure 1. IDAT result, i.e. the resulting BIST data-path architecture is shown in figure 2. Main features of our BIST architecture are as follows:

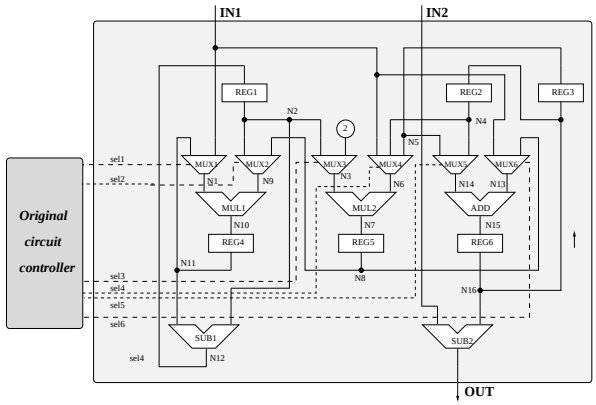


Figure 1: Original architecture of circuit Diffeq.

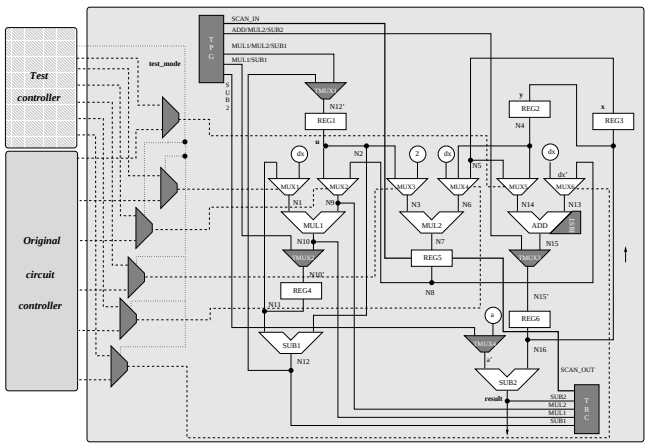


Figure 2: BIST architecture created by IDAT.

- All *select* inputs of the allocated multiplexers, all *RESET/SET* inputs of sequential components and all registers *write enable control* inputs are connected through the multiplexers to the added test controller.
- Already existing BIST versions of selected FUs are used to replace corresponding non testable

FUs. This replacement in one hand improves test time and decreases complexity of main TPG and TRC if allocated, but in the other hand the resulting area overhead of BIST version is generally not negligible. When a BIST FU is allocated, all its test control signals are directly connected to the test controller.

- If a BIST version of a selected FU does not exist or if the area overhead resulting from the replacement of an original component by its BIST equivalent is too big, a unique TPG and a unique TRC are added to the data-path in order to test the selected components. All FU inputs (including control inputs) are directly or through one or more components connected to the TPG and to the TRC. The TPG and TRC are shared by all the non BIST FUs, thus they are parameterized and dynamically reconfigurable: the bit-width of TPG outputs and the bit-width of TRC inputs are modifiable. Up to now, we have designed a test structure generator, which allows to synthesize such structures as LFSR, MISR, and BILBO. This generator takes as input the test structure parameters given by IDAT. Other kind of test structures, e.g. based on cellular automata [7], can also be taken into account by IDAT.
- Remaining untested interconnections can be tested by scan chains, starting from the TPG and ending at the TRC. In this case the *scan enable* inputs are connected together to the test controller.

# 4 Testability analysis for BIST

In case of BIST methology, the testability analysis process is used to guide the selection of the set of FUs which should be BISTed. The general testability analysis approach has already been published in [6]. More details can also be found in [4].

During a test session, the circuit data-path is controlled by the added test controller. The purpose of the testability analysis algorithm - called propagation algorithm - is the calculation of the *controllability* ($C$) and *observability* ($O$) of each node. The controllability (resp. observability) calculation for BIST circuit starts both from TPG outputs (resp. TRC inputs) and scan components outputs (resp. inputs) and ends both at TRC inputs (resp. TPG outputs) and scan components inputs (resp. outputs). In current IDAT version, the controllability and observability metrics reflect test problems caused by non-initializable feedback loops and transparency loose caused by traversing different FUs.

Obtained controllability and observability values for each node are then used to calculate the *controllability and observability exigencies*. These new metrics reflect the actual circuit test plan, where more than 90% of the test time is consumed by testing complex components, the FUs. The *controllability exigency* of a node corresponds to the necessity to control this node with respect to the test complexity of the components driven by this node. In the same way, the *observability exigency* of a node refers to the necessity to observe this node with respect to the test complexity of the component driving this node. These metrics, in our BIST technique, guide the selection of FUs which need to be tested.

# 5 BIST cost/quality trade-off

The cost/quality trade-off process is the core of the allocation for testability process. The trade-off process has the purpose of making the best choice between various versions of a given component to be allocated, and to suggest inclusion of additional testability structures, in order to satisfy user requirements. In addition to user requirements, the trade-off algorithm takes also into account other criteria, including the mandatory requirements imposed by the chosen testability method, in our case BIST. Details on the case of Scan-based methodology can be found in [5].

The trade-off process searches first of all to satisfy the user requirements. If this is impossible with the existing library then a dialogue starts between IDAT and the user, who is asked to modify step by step his requirements, being guided by the system.

Testability problems are global in nature. A 100% testable component may show testability problems when it is embedded into a whole data-path. To avoid this situation, the trade-off algorithm is decomposed into two main modules: the *local trade-off* module, which acts at the component (FU) level, and the *global trade-off*, which operates at the data-path level.

During the local trade-off, the attributes of original component and the attributes of each of its BIST versions are compared with corresponding user requirements from the user requirements vector. This results in a list of all versions ordered according to the level of satisfaction of user requirements. The global trade-off for BIST is made at two levels:

- Among all FUs in order to select the set of FUs for which the test will have maximal effect to overall circuit testability: the selection of a FU to test is guided by the testability analysis results and also, when more than one candidate with the

same complexity exists, by the number of interconnections which can be tested together with this FU. A specific algorithm has been developed to select the longest possible test path and to reuse already allocated connections from TPG and to TRC, avoiding possible reconvergences (see figure 3).

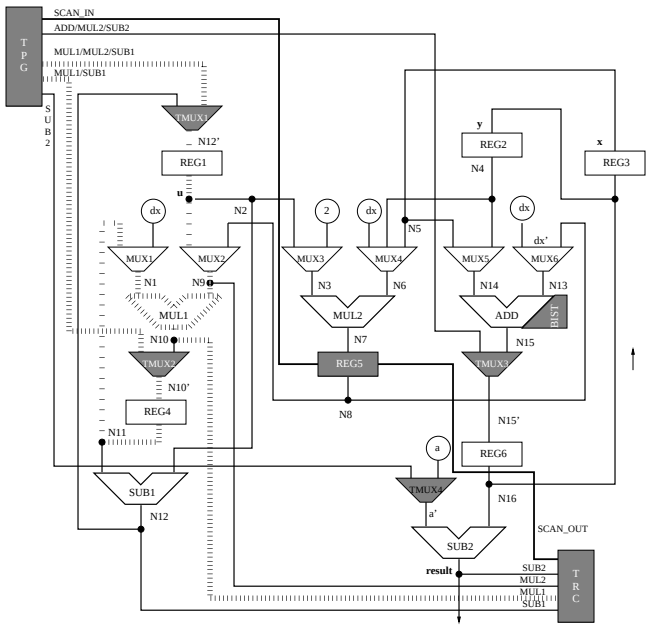


Figure 3: Data-path of the circuit from figure 2 with emphasized test paths to test MUL1.

In figure 3, the selected FU to test is MUL1, and together with the test of MUL1 also the following interconnections are tested: TMUX1 → REG1, REG1 → MUX2, MUX2 → MUL1, TMUX2 → REG4, REG4 → MUX1 and MUX1 → MUL1. In IDAT current version the path reusing already existing connection has higher priority than the longest possible path.

- Between possibilities to either transform an original FU into its BIST version, when such a version exists, or to test the original FU using the reconfigurable TPG and TRC: the choice of a BIST version of an FU, compared to the connection of the original FU to a shared TPG and TRC usually decreases the test time of the circuit, reduces the TPG/TRC complexity and results in bigger area overhead. From the last reason, it is sometimes more advantageous to connect an FU to the TPG and TRC although its BIST version exists.

The trade-off is implemented with a branch-and-bound algorithm, using the previously described criteria. More details on the overall cost/quality trade-off during the allocation for testability process can be found in[4] and in [5].

# 6 Results

## 6.1 Branch-and-bound example

The branch-and-bound algorithm used is demonstrated on the following simple example. Let us have a circuit with four FUs which characteristics are summarized in table 1. None of these FUs has a BIST version, so each of them, when selected to be tested, must be connected to the TPG and TRC. User requirements are as follows: testability 80%, area overhead 20%. The branch-and-bound algorithm steps are depicted in table 2.

| FU name | % of global area | input # | output # |
|---|---|---|---|
| FU1 | 40% | 16 | 8 |
| FU2 | 20% | 256 | 128 |
| FU3 | 15% | 16 | 8 |
| FU4 | 15% | 16 | 8 |

Table 1: Example of four FUs to demonstrate BIST branch-and-bound algorithm.

| BBA step | executed operation | % of T | % of AO | Connected FU 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|
| 1 | separate data-path | 0 | 5 | | | | |
| 2 | connect FU1 | 41 | 8 | • | | | |
| 3 | connect FU2 | 65 | 19 | • | • | | |
| 4 | connect FU3 | 82 | 21 | • | • | • | |
| 5 | disconnect FU3 | 65 | 19 | • | • | | |
| 6 | connect FU4 | 84 | 21 | • | • | | • |
| 7 | disconnect FU4 | 65 | 19 | • | • | | |
| 8 | disconnect FU2 | 41 | 8 | • | | | |
| 9 | connect FU3 | 58 | 10 | • | | • | |
| 10 | connect FU4 | 81 | 12 | • | | • | • |

Table 2: Steps of the BIST branch-and-bound algorithm.

The result of the branch-and-bound algorithm execution are: FU1, FU3 and FU4 connected to the TPG and TRC, which are generated and added to the data-path when the first FU connection is made. This configuration satisfies user requirements of maximal circuit testability and minimal area overhead. Even though FU2 was first selected to be tested by the testability analysis process, this FU has finally not been connected to the TPG and TRC, because its high number of inputs and outputs results in too big area for the TPG and TRC (since both depend on the con-

nected FU bit-widths), resulting in an unsatisfactory total area overhead.

## 6.2 Tseng benchmark

The first benchmark on which we will demonstrate our allocation for BIST algorithm is benchmark Tseng. Tseng is originally described at the behavioral level, while the input of IDAT is an RVA architecture. We have extracted data-path architecture from a simplified RTL one from [10] (see figure 4) and modified it by adding the necessary labels to components and nodes. The necessary design attributes were obtained by synthesis.

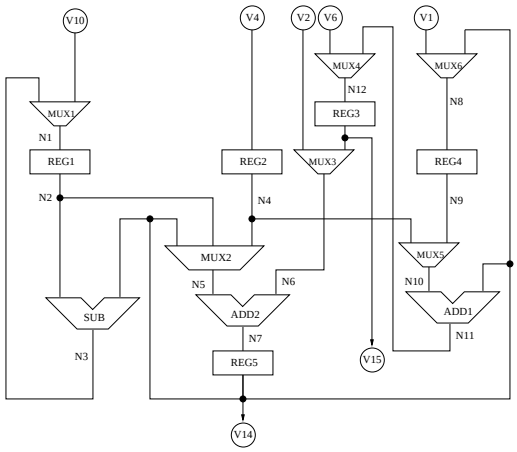


Figure 4: Data-path architecture of Tseng benchmark.

The evolution of the global attributes after each modification performed by IDAT is depicted in the diagram of figure 5. Each column with a different pattern, labeled from 1 to 4, refers to different circuit configuration number.

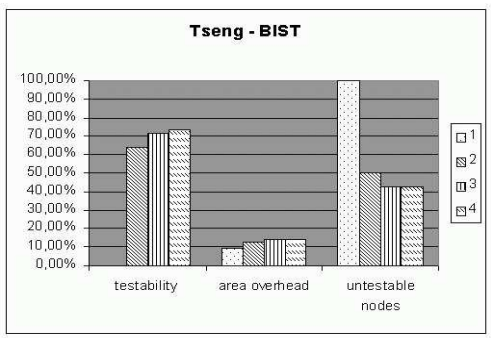


Figure 5: Selected resulting attributes (measurable in percents) for different circuit configurations of benchmark Tseng allocated for BIST.

Configuration 1 corresponds to the original circuit with allocated block of multiplexers, separating the data-path from the original circuit controller. Other configurations correspond to the following step:

- (2) SUB selected: for SUB input N2, primary input V10 is connected to TPG, for SUB input, V14 signal N7 is connected to TPG, and SUB output N3 is directly connected to TRC,
- (3) ADD2 selected: for ADD2 input N5, signal N7 is already connected to TPG, for ADD2 input N6, primary input V6 is connected to TPG, and ADD2 output N7 is directly connected to TRC,
- (4) ADD1 selected: for ADD1 input N10 and ADD1 input V14, signal N7 is already connected to TPG, there is no direct reconvergence because signal N7 is connected through one register (REG5) to ADD1 input V14 and through two registers (REG5 and REG4) to ADD1 input N10 (with a different sequential depth), for ADD1 output N11, signal N6 is connected to TRC.

## 6.3 Diffeq benchmark

The second benchmark on which we will demonstrate our allocation for BIST algorithm is Diffeq. Like Tseng benchmark, Diffeq benchmark is originally described at the behavioral level. Its RTL data-path architecture is reprinted from [10] into figure 1. We have modified it by adding the necessary labels to components and signals, then we have converted the output comparator to subtractor SUB2 and finally we have connected this data-path to the circuit controller. The selected global attributes for each modification step performed by IDAT are depicted in the diagram of figure 6.

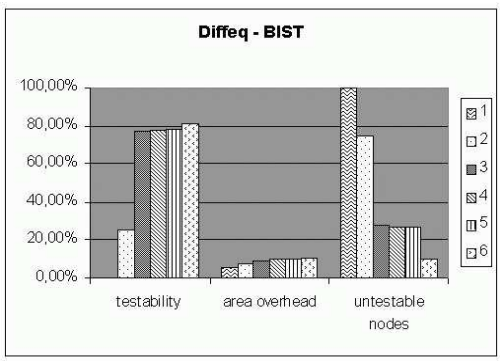


Figure 6: Selected resulting attributes (measurable in percents) for different circuit configurations of benchmark Diffeq allocated for BIST.

Configuration 1 corresponds to the original circuit with allocated block of multiplexers. Subsequent modifications are as follows:

- (2) MUL1 selected: for MUL1 input N1, signal N10 is connected to TPG, for MUL1 input N9, signal N12 is connected to TPG, and MUL1 output N10 is directly connected to TRC,
- (3) MUL2 selected: for MUL2 input N3, signal N12 is already connected to TPG, for MUL2 input N6, signal N15 is connected to TPG, for MUL2 output, and signal N13 is connected to TRC,
- (4) SUB2 selected: primary input IN2 is directly connected to TPG, for SUB2 input N16, signal N15 is already connected to TPG, and primary output OUT is directly connected to TRC,
- (5) SUB1 selected: for SUB1 input N11 signal, N10 is already connected to TPG, for SUB1 input N2, signal N12 is already connected to TPG, and SUB1 output N12 is directly connected to TRC,
- (6) ADD selected: for ADD input N14, signal N15 is already connected to TPG, for ADD input N13, primary input IN1 is connected to TPG, and ADD output N15 is directly connected to TRC.

# 7 Conclusion

This paper has described the allocation for BIST technique included in the allocation for testability tool IDAT based on cost/quality trade-off iteration algorithm, driven by a set of user requirements. Main advantages of the proposed allocation for BIST technique, besides the foreseen short test application time, are:

- (1) the capability to test not only FUs, but also the interconnections,
- (2) the DFT reuse feature,
- (3) the minimization of added test structures, since a unique TPG and a unique TRC are shared by many FUs (to this purpose, the TPG and TRC are parameterized and dynamically reconfigurable),
- (4) the avoidance of fault simulation necessity, and
- (5) the consolidation, in a single tool, IDAT, of the synthesis of different testability methods (test point insertion, Scan, BIST).

# References


[1] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital System Testing and Testable Design, Revised Printing.* IEEE Press, NJ, USA, 1990. ISBN 0-7803-1062-4.

[2] D. Berthelot, M.L. Flottes, and B. Rouzeyre. Optibist: A tool for bisting datapaths. In *Proceedings of IEEE European Test Workshop*, pages 123–127, Barcelona, Spain, May 1998.

[3] D. Berthelot, M.L. Flottes, and B. Rouzeyre. Datapath bist scheme for full testing. In *Compendium of Papers from IEEE European Test Workshop*, Constance, Germany, May 1999.

[4] P. Bukovjan. Allocation for testability in high-level synthesis. Technical report, LIP6 Laboratory, Paris, December 1999.

[5] P. Bukovjan, M. Marzouki, and W. Maroufi. Design for testability reuse in synthesis for testability. In *Proceedings of XII Symposium on Integrated Circuits and System Design*, pages 206–209, Natal, Brazil, September 1999.

[6] P. Bukovjan, M. Marzouki, and W. Maroufi. Testability analysis in high-level synthesis for testability. In *Proceedings from 10th European Workshop on Dependable Computing*, pages 71–75, Wien, Austria, May 1999.

[7] P. Sérgio Cardoso, M. Strum, J.R. de A. Amazonas, and W.J. Chau. Comparison between quasi-uniform linear cellular automata and linear feedback shift registers as test pattern generators for built-in self-test applications. In *Proceedings of XII Symposium on Integrated Circuits and System Design*, pages 198–200, Natal, Brazil, September 1999.

[8] H. Harmanani and C.A. Papachristou. An improved method for RTL synthesis with testability tradeoffs. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 30–35, November 1993.

[9] I. Parulkar, S. Gupta, and M.A. Breuer. Data path allocation for synthesizing rtl design with low bist area overhead. In *Proceedings of Design Automation Conference*, pages 395–401, San Francisco, June 1995.

[10] P.G. Paulin, J.P. Knight, and E.F. Girczyc. Hal: A multi-paradigm approach to automatic data path synthesis. In *Proceedings of the Design Automation Conference*, pages 263–270, June 1986.