Testability Analysis and Cost/Quality Trade-off in Synthesis for Testability*

Peter Bukovjan Meryem Marzouki† Walid Maroufi

LIP6 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
Peter.Bukovjan@lip6.fr, Meryem.Marzouki@lip6.fr, Walid.Maroufi@lip6.fr

Abstract

This paper gives an overview of the IDAT system, based on Interactive Design for Testability reuse in Allocation for Testability. In the context of High-Level Synthesis for Testability, the allocation for testability process mainly consists in searching for the best cost/quality trade-off between the designer requirements and testability means which can be proposed by the system, considering the available components in a library and the possibility to generate additional testability structures. The cost/quality trade-off is also based on the result of a testability analysis process. The paper describes the trade-off and testability analysis processes, and then gives experimentation results.

Keywords: synthesis for testability, DFT reuse, testability analysis

1 Introduction

1.1 State of the Art

Most of the existing High-Level Synthesis (HLS) tools are designed with the only purpose of design criteria optimization, without considering testability issues at the high-level. The resulting circuit architecture will then give, after lower level synthesis, either a very hard to test circuit if testability is not included later, or a testable circuit with very bad performances and high area overhead, since test issues have been considered too late in the whole synthesis process.

To avoid these drawbacks, some authors have then considered High-Level Synthesis for Testability (HL-SFT) solutions, where testability is taken into account before RT-Level generation. Comprehensive overviews of such solutions can be found in [1], [6], [12], [13], and [8].

A popular targeted HLS phase to introduce testability is the allocation and binding step, where variables and functions from the behavioral description are mapped to registers and operators. The large number of possible solutions and the strong impact of these solutions to the final circuit architecture offers many possibilities to improve the overall circuit testability. However, these approaches suffer from two main disadvantages, the first one being that they consider the behavioral description of the circuit, which means that the resulting architecture will not allow an accurate mapping of the testability resources, and the second one being that they do not make profit of already existing testable components or testability structures. In addition, the solutions proposed in the literature are loosely interactive.

1.2 The IDAT System Approach

The HL-SFT approach that we have defined has the purpose of overcoming these drawbacks. The whole process is viewed as the automatic generation of testable architectures, starting from a *Rich Virtual Architecture* (RVA), and making profit of the *Test Reuse* concept [9], [10]. This approach then considers as input a structural information (the virtual architecture of the circuit to be synthesized), which is neither the behavioral description, nor the RTL description: it then avoids both an inaccurate and unpredictable mapping of the testability resources, and a loss of performances in the resulting testable architecture. Another important aspect of the approach presented here is the test reuse and resource sharing concepts, since the cost/quality trade-off is made relying on existing testable structures. These structures are then allocated according to a target test methodology and to a set of constraints dealing with cost and performances.

The allocation for testability process presented in this paper is complemented with a global test scheduling process [10]. This article is a brief summary of a detailed research report [2]. It is organized as follows: section 2 presents the overall IDAT system for allocation for testability, then section 3 details the testability analysis module, and section 4 explains

*This work has been partly supported by MEDEA-SMT Project and by a Grant from the Grant Agency of Czech Republic, contract No. 102/98/1463.

†Corresponding author

the cost/quality trade-off process. The implementation of the IDAT system is described in section 5, and finally experimental results are provided in section 6.

2 Allocation for testability

The IDAT (*Interactive DfT reuse in Allocation for Testability*) system starts from the RVA *Rich Virtual Architecture* description of the circuit to be synthesized. This RVA description is a draft description of the circuit architecture, where the operators have already been allocated, and the variables have been assigned to registers. Contrarily to the RTL architecture, fine synchronization information are not yet taken into account. The RVA description is then obtained after the allocation and binding steps of the HLS process. This draft architecture is enriched with design attributes, like area, bit-width, maximal delay, etc. quantified using generic units.
The virtual architecture is after that modified by the IDAT system, taking into account the designer requirements in terms of testability, and availability of testable components and testability structures. The best test cost/quality ratio is searched.

The allocation for testability process consists in three steps: (1) extraction of the virtual architecture from the result of a HLS system, (2) extraction of testable versions of allocated components from a library, (3) a first testability analysis of the virtual architecture and (4) the test cost/quality trade-off itself, according to the testability methodology chosen by the designer: scan-based or BIST-based methodology. This process can itself call the testability analysis module, if needed.

3 Testability analysis

3.1 Testability analysis definition

Our testability analysis process is based on the calculation of the combinational and sequential controllability and observability factors. This is the first step required to evaluate the testability inclusion to be made. Although some concepts are common to other methods presented in the literature ([5], [4] and [3]), our testability analysis process differs from them by the fact that it operates on the virtual architecture and not on the true RTL one or the CDFG, in addition to using different calculation formulas.
The controllability and observability of a given node is defined using both a combinational and sequential factor. The combinational factor ranges from 0 to 1, and measures the test generation cost. The sequential factor ranges from 0 to infinity, and measures the sequential depth of the node, i.e. the estimated number of transition steps in our testability analysis algorithm to calculate the corresponding combinational value. After that, the testability of the node is calculated, so as to undertake subsequent actions (allocation for testability) on hard-to-test nodes.

3.2 Propagation algorithm

A *propagation algorithm* has been developed in order to calculate the controllability and observability of the virtual architecture nodes. This algorithm is based on progressive calculation of the controllability and observability of the circuit component ports.

The controllability evaluation starts from the primary inputs and the scan components outputs (if any). Their combinational controllability (CC) is set to 1 and their sequential controllability (SC) is set to 0 (best possible values). All other nodes have their CC set to 0 and their SC set to infinity (worst possible values).
The controllability calculation then progresses towards the primary outputs and the scan components inputs, such that the worst values are modified only if the controllability of all the component inputs is greater than 0. However, this condition doesn't apply if the component is a multiplexer. For multiplexers, if the *select* input is not shared with other multiplexers, then the controllability of the multiplexer output is set to its best input controllability, otherwise, a *multiplexer output controllability trade-off algorithm* is executed, in order to find the higher average output controllability value of all the multiplexers sharing the *select* pin, and affect it to this input.

The observability calculation symmetrically starts from the primary outputs and the scan components inputs (if any). Their combinational observability (CO) is set to 1 and their sequential observability (SO) is set to 0 (best possible values). All other nodes have their CO set to 0 and their SO set to infinity (worst possible values).
The observability calculation then progresses backwards towards the primary inputs and the scan components outputs, such that the worst values are modified only when the inputs of the components on the path, other than the currently processed input, have a CC greater than 0. This condition doesn't apply when the component is a multiplexer.

The *controllability transfer factor* (resp. the *observability transfer factor*) defined in [4] is used to reflect the difficulty to control the output of a component from its inputs (resp. to observe the inputs of a component from its output).

3.3 Global testability indicator

The CC, SC, CO, SO values are used to calculate the testability of each node, and at the end the *global testability indicator* (T_{global}) for the whole datapath.

This indicator is one of the criteria of the cost/quality trade-off process, and guides the user in his acceptance of the choices provided by the IDAT system. T_{global} is defined as the average of all the node testabilities T_i:

$$T_{global} = \frac{1}{n} * \sum_{i=1}^{n} T_i,$$

where n is the number of nodes in the circuit.
The testability of each node is calculated as the average of the controllability and observability indicators of this node:

$$T_i = \frac{1}{2} * (C_i + O_i)$$

and the controllability and observability indicators C_i and O_i are defined as

$C_i = \frac{1}{2} * (CC_i + \frac{1}{SC_i+1})$ and $O_i = \frac{1}{2} * (CO_i + \frac{1}{SO_i+1})$.

The node testability value as well as the global testability indicator range from 0 to 1.

4 Trade-off process

The trade-off process is the core of the allocation for testability process. Its criteria include the user requirements and their corresponding weights. The trade-off process has the purpose of making the best choice between various versions of a given component to be allocated, or to suggest inclusion of additional testability structures, in order to satisfy the user requirements. In addition, the trade-off algorithm takes into account the mandatory requirements imposed by the chosen testability method (BIST or scan).
The trade-off process searches first of all to satisfy the user's requirements. If it is impossible with the existing library then a dialogue starts between our system 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 datapath. To avoid this situation, the trade-off algorithm is decomposed into two main modules: the *local trade-off* module, which acts at the component level, and the *global trade-off*, which operates at the datapath level.

4.1 Trade-off criteria

The trade-off process is a part of the allocation for testability process. This means that a library of components to allocate is available, each component being described by its testability attributes, in addition to its design attributes.

Just like a cost/quality trade-off between the *design attributes* of related components (area, maximal delay and power consumption) is made during the allocation step of a classical HLS process, the new cost/quality trade-off has to be made between the *testability attributes* of related components (area overhead, extra delay, extra power consumption, testability, etc.) during the allocation for testability process, considering also other parameters which are the *user's requirements* and some *weighing factors* to help the system balancing the various requirements. The design and testability attributes of each component are summarized in a *characteristic vector*. Actual circuit components configurations are stored in the *allocation vector*. The user's requirements saved in the *user requirements vector* are classified into three groups according to their relationship in a component or circuit testability description, and this also corresponds to the user expertise in the field of HL-SFT and DFT (beginner, advanced, expert).

As already stated, the trade-off process is made at two levels: at the local level (among different component versions) and at the global level (circuit architecture). From this point of view the attributes can also be distinguished as the *local attributes* represented by the characteristic vector and the *global attributes*, which are calculated from local attributes, using equations given in [2], and then saved to the *global attributes vector*. Most of the attributes exist at both local and global levels. In the sequel, only the global trade-off process will be detailed. The local trade-off has the purpose of ordering different testable versions of a given components, taking only into account local attributes of this component (e.g. area overhead, extra I/O, etc.).

4.2 Global trade-off

The goal of the global trade-off is depending upon the choice of the global testability methodology made by the designer: scan or BIST methodology.
In case of scan methodology, the goal of the global trade-off is to find the optimal set of library components to be replaced by their scan version, when needed, after testability analysis. To reach this goal, the global trade-off algorithm first evaluates all the global attributes for the *allocation vector*, which contains all the components allocated in the datapath (except multiplexers). Then, the algorithm searches in this vector all the failing components with respect to the user requirement vector, using the testability analysis process, until the best solution is found, or until ending with no solution found.

In case of BIST methodology, the global trade-off algorithm attempts to replace all complex components by their BISTable versions if they exist, or proposes to add a Test Pattern Generator (TPG) and a Test Result Checker (TRC) to the datapath, to be shared by all the non-testable components. Registers are replaced with a scan chain, which input (resp. output) is connected to the TPG (resp. TRC).

4.3 Trade-off functions

The trade-off algorithm uses as computing functions the following ones: (1) a *distance function* to estimate the difference between the value of a given attributes in the allocation vector and its corresponding value in the user requirement vector, (2) a *penalty-credit function* (PCF) which normalizes the resulting value of the distance function, and takes into account the weight of the requirements, and (3) a *score function* which attributes a score to a component (local trade-off) or to the datapath (global trade-off), in order to classify the different proposed alternatives. These functions are very similar to those used in [14] and [15], since we are facing the same kind of problems, at a different level of abstraction.

5 Implementation

The IDAT system has been implemented using the C++ language. Datapath components are represented as objects with related attributes and related methods to compute these attributes.
Input to the IDAT system is a VHDL description of the datapath virtual architecture, which is automatically translated into the Rich Virtual Architecture (RVA) format, or directly an RVA description. An RVA description contains, in addition to the virtual architecture description itself, information on the type of the signals (DATA or CONTROL), information on the component function, and information on its design attributes. Other inputs to IDAT are the user requirements with regards to testability.
Output of the IDAT system is the modified RVA description of the datapath (in VHDL), together with the list of all the architectural modifications made during the allocation for testability process. This output is provided back to the High-Level Synthesis system, which then completes its tasks.

6 Results

The obtained results with the IDAT system are shown in the following, using the Greatest Common Divider (GCD) example for the trade-off process in view of BIST methodology and the Diffeq benchmark for testability analysis and the trade-off processes in view of scan methodology. In addition, the testability analysis process is compared with another approach of the literature.

6.1 GCD example

A behavioral description of the Greatest Common Divider (GCD) was first synthesized by HLS system AMICAL [7], which generated the bus-based non-testable datapath virtual architecture (figure 1). The allocation for testability process starts after the classical allocation and binding steps of AMICAL and considers all components as virtual ones. Component AS is an adder/substractor functional unit.

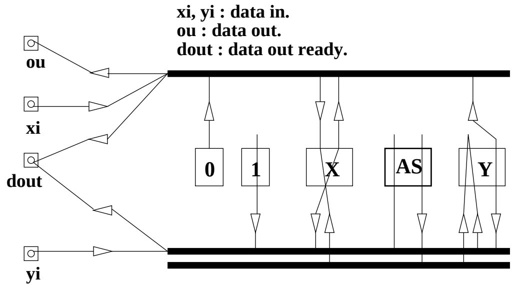

Figure 1: GCDbus datapath generated by AMICAL

During processing, user requirements are introduced according to table 1. The original, non-testable version of component AS is named *AS-D* (see table 2). Let's assume that there exist four different BISTable versions of AS in the library, named *AS-A*, *AS-B*, *AS-C*, and *AS-E* (see table 3).

Attribute	Requirement value	Weight
method	BIST	-
area_over	10	10
testability	99	10
test_time	50	0

Table 1: User's requirements and weights to GCDbus

Comp	*area*	*max_del*	*power*	*CTF*	*OTF*
AS-D	450	100	700	0.817	0.868

Table 2: Attributes of original adder/substractor

These versions are ordered, using the trade-off functions defined in section 4, as follows: A (best), E, C, B.
Results of the trade-off processing are presented in figure 2. Two major conflicts are detected:

- For the area overhead attribute, the user requires 10% but the circuit with smallest AS version only offers 15% area overhead.
- For the testability attribute, the user requires 99% but the circuit with best testable AS version allocated only offers 95% testability.

At this stage, the system invites the user to modify his requirements in order to allow allocation. Three cases are distinguished:

- **case 1:** The user modifies the two requirements (*area_over* to 15 and *testability* to 95). A solution (version A of AS) is proposed (figure 3).
- **case 2:** The user only accepts to modify the area overhead requirement. In this case it is

Attribute	AS-A	AS-B	AS-C	AS-E
$area$	517	550	531	517
max_del	130	125	160	140
$power$	1000	1200	900	850
$area\_over$	15	22	18	15
$extra\_delay$	30	25	60	40
$extra\_power$	43	71	28	21
$extra\_I/O$	4	4	4	4
$reset$	NO	NO	NO	NO

Table 3: Adder/substractor design and testability attributes

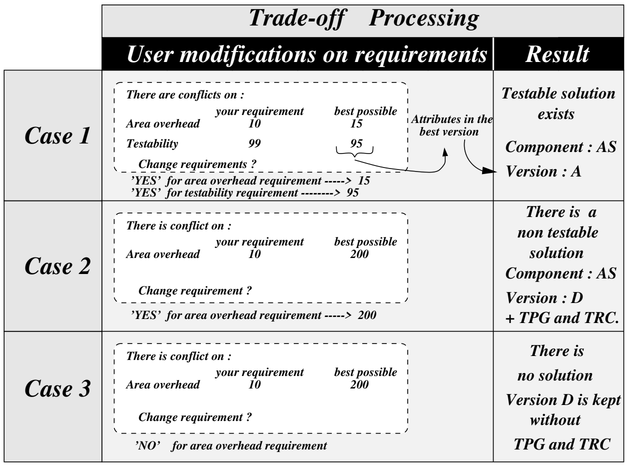

Figure 2: Processing results for circuit GCDbus

proposed to keep AS-D version (original, non-testable), and to add a TPG and a TRC (figure 4). This solution gives 100% testability, but area overhead increases to 200% (in the GCD example, there is only one functional component).

- **case 3:** The user does not accept to modify the area overhead requirement. The IDAT system stops at this point, with no accepted solution found.

6.2 Diffeq benchmark

The Diffeq benchmark [11] has been used to demonstrate our testability analysis process. Diffeq is originally described at the behavioral level, while the input of our process is a virtual architecture (see section 2). We have then used as virtual architecture a simplified version of its RTL one, which was generated by the HLS system HAL presented in [11].
Although this circuit is often used for benchmarking, it has been difficult to compare our results with those reported in [1], [6], [12], [13] and [8] in terms of fault coverage and test generation time. One reason is that we don't have a powerful sequential ATPG available at our site, but the main, intrinsic, reason is that the results of our system are strongly related to the provided virtual architecture, which depends upon the HLS system used.
Nevertheless, the IDAT system was applied to this

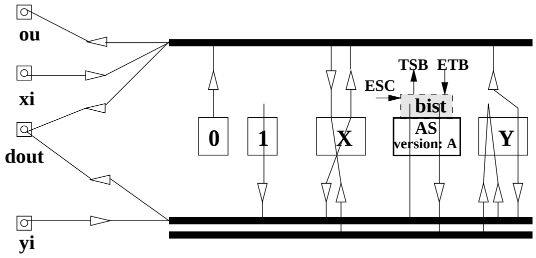

Figure 3: GCDbus architecture with AS BIST

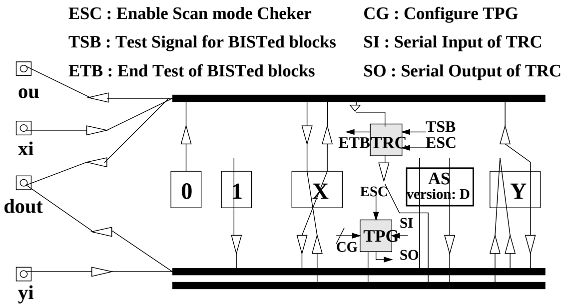

Figure 4: GCDbus architecture with TPG and TRC

benchmark, and it has been checked that the resulting datapath, where components are re-allocated under global testability constraints, is actually more testable that the original datapath provided by the HLS system without considering testability issues.
The Diffeq scheme [11] has been reproduced in this paper, adding necessary labels to components and signals (figure 5), in order to demonstrate our testability analysis results.

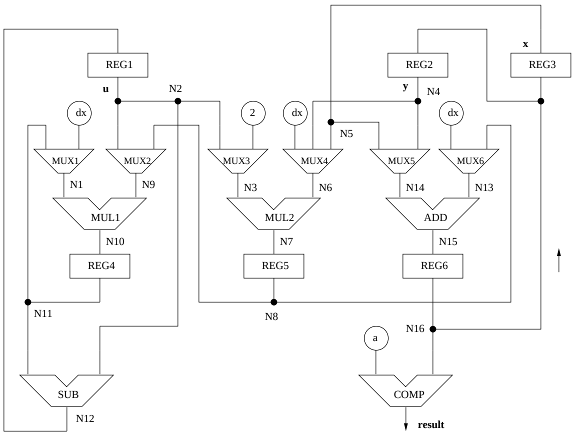

Figure 5: Diffeq datapath generated by HAL

Attributes extracted from original components, together with controllability and observability transfer factors (calculated using equations from [4]) are summarized in table 4. The user (beginner in this case) requirements for Diffeq testability are summarized in table 5.
The allocation for testability algorithm then extracts from a library of testable components all possible testable equivalents of circuit components (registers

Comp	*area*	*max_del*	*power*	*CTF*	*OTF*
ADD-NT	430	100	700	0.817	0.868
SUB-NT	470	790	900	0.817	0.868
MUL-NT	4138	1000	900	0.769	0.802
CMP-NT	360	110	650	0.860	0.642
MUX2-NT	146	22	200	1	1
MUX3-NT	247	22	220	1	1
REG-NT	282	2	500	1	1

Table 4: Original extracted components attributes for circuit Diffeq

Attribute	Requirement value	Weight
method	SCAN	-
area_over	5%	8
testability	60%	9
test_time	400	1

Table 5: User's requirements and weights to Diffeq

and functional units).
Testability analysis results for the first allocation vector (av_1) corresponding to the original (non-testable) components are summarized in table 6, which gives the values obtained for the quantities defined in section 3. The testability analysis process identifies as main uncontrollable nodes: N2, N14, and N16 (see figure 5). To improve testability of these nodes, the trade-off process is started, and, after calculation of the global attributes of the original allocation vector (table 4), some of the following components were offered to be replaced by their existing testable equivalent. These components are ordered according to their testability influence: REG1, REG2, REG3, REG6.

Node	*CC*	*SC*	*CO*	*SO*	*C*	*O*	*T*
a	1	0	0	∞	1	0	0.50
dx	1	0	0	∞	1	0	0.50
2	0	∞	0	∞	0	0	0
N1	1	1	0	∞	0.75	0	0.38
N2	0	∞	0	∞	0	0	0
N3	0	∞	0	∞	0	0	0
N4	0	∞	0.75	4	0	0.48	0.24
N5	0	∞	0.75	4	0	0.48	0.24
N6	1	1	0	∞	0.75	0	0.38
N7	0.77	2	0	∞	0.55	0	0.28
N8	0.77	3	0	∞	0.51	0	0.25
N9	0.77	4	0	∞	0.48	0	0.24
N10	0.59	5	0	∞	0.38	0	0.19
N11	0.59	6	0	∞	0.37	0	0.18
N12	0	∞	0	∞	0	0	0
N13	1	1	0	∞	0.75	0	0.38
N14	0	∞	0.75	3	0	0.50	0.25
N15	0	∞	0.87	2	0	0.60	0.30
N16	0	∞	0.87	1	0	0.69	0.34
result	0	∞	1	0	0	1	0.50

Table 6: Testability analysis results for av_1

The second iteration starts with a second allocation vector, where register REG1 has been replaced by its scan version. Testability analysis results for allocation vector 2 (av_2) are summarized in table 7. The testability of many nodes has been increased (N1, N2, N3, N6, N7, N10, N11, N12), while the testability of other nodes remains the same.

Node	*CC*	*SC*	*CO*	*SO*	*C*	*O*	*T*
a	1	0	0	∞	1	0	0.50
dx	1	0	0.7	4	1	0.45	0.73
2	0	∞	0.56	7	0	0.34	0.17
N1	1	1	0.7	3	0.75	0.47	0.61
N2	1	0	0.62	4	1.00	0.41	0.70
N3	1	1	0.56	6	0.75	0.35	0.55
N4	0	∞	0.76	∞	0	0.46	0.23
N5	0	∞	0.76	5	0	0.46	0.23
N6	1	1	0.56	6	0.75	0.35	0.55
N7	0.77	2	0.7	5	0.55	0.43	0.49
N8	0.77	4	0	∞	0.48	0	0.24
N9	1	1	0.7	3	0.75	0.47	0.61
N10	0.82	2	0.87	2	0.58	0.60	0.59
N11	0.82	3	0.87	1	0.53	0.69	0.61
N12	0.67	5	1	0	0.42	1.00	0.71
N13	1	1	0	∞	0.75	0	0.38
N14	0	∞	0.75	3	0	0.50	0.25
N15	0	∞	0.87	2	0	0.60	0.30
N16	0	∞	0.87	1	0	0.69	0.34
result	0	∞	1	0	0	1.00	0.50

Table 7: Testability analysis results for av_2

Remaining main uncontrollable nodes are N4, N5, N14, N15, N16. To improve their testability, IDAT suggests to replace register REG2 or REG6 by testable ones. REG2 has a higher priority from the testability point of view (sequential observability depth SO of N4 being higher than that of N16), thus IDAT first replaces it by its scan equivalent, obtaining a third allocation vector (av_3). The resulting testability values are summarized in table 8. It shows that the testability of all nodes has been increased, or have remained the same, and that there is no remaining uncontrollable node. Moreover, all the user requirements have been satisfied, as shown in table 9, and the allocation for testability process ends with allocation vector av_3, which is then translated into modifications of the virtual architecture.

6.3 Testability analysis comparison

In order to compare our testability analysis process with another approach presented in the literature and developed at Linköping University [4], we have used the example depicted in figure 6, from [4]. Table 10 summarizes the comparison results. The correlation between results obtained with both approaches is very high, while indicating a slightly more preciseness of Gu's method, as shown in table 11. This slight difference is explained by the fact that Gu's method requires more knowledge on the synthesized circuit, namely information on the control part. One can note that our testability analysis process nearly reaches the same performances, by only working on the virtual architecture.

Node	CC	SC	CO	SO	C	O	T
a	1	0	0.67	1	1.00	0.59	0.79
dx	1	0	0.87	6	1.00	0.51	0.75
2	0	∞	0.56	11	0	0.32	0.16
N1	1	1	0.7	5	0.75	0.43	0.59
N2	1	0	0.67	1	1.00	0.59	0.79
N3	1	1	0.56	10	0.75	0.33	0.54
N4	1	0	0.87	6	1.00	0.51	0.75
N5	0.817	4	0.87	6	0.51	0.51	0.51
N6	1	1	0.56	10	0.75	0.33	0.54
N7	0.769	2	0.7	9	0.55	0.40	0.48
N8	0.769	3	0.7	7	0.51	0.41	0.46
N9	1	1	0.7	5	0.75	0.43	0.59
N10	0.769	2	0.87	3	0.55	0.56	0.56
N11	0.769	3	0.87	1	0.51	0.69	0.60
N12	0.628	4	1	0	0.44	1.00	0.72
N13	1	1	0.87	4	0.75	0.54	0.64
N14	1	1	0.87	5	0.75	0.52	0.63
N15	0.817	2	1	1	0.58	0.75	0.66
N16	0.817	3	1	0	0.53	1.00	0.77
result	0.702	5	1	0	0.43	1.00	0.72

Table 8: Testability analysis results for av_3

Global attribute	av_1		av_2		av_3	
	value	OK	value	OK	value	OK
area_over	0%	yes	1.9%	yes	3.8%	yes
testability	25.73%	no	46.21%	no	62.06%	yes
test_time	274	yes	380	yes	372	yes

Table 9: Calculated global attributes for Diffeq

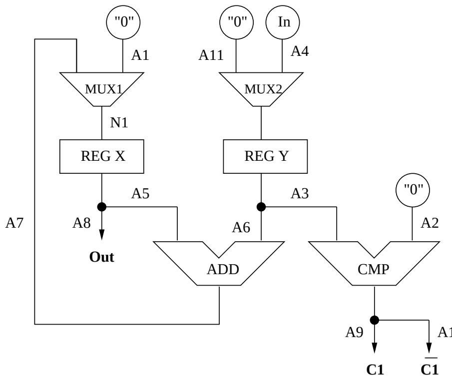

Figure 6: Example from Gu thesis [4]

Node	Gu's				IDAT			
	CC	SC	CO	SO	CC	SC	CO	SO
A1	-	-	0.86	2	0	∞	1.00	3
A2	-	-	0.55	1	0	∞	0.64	2
A3	1.00	2	0.55	1	1.00	1	0.87	2
A4	1.00	0	0.60	3	1.00	0	0.87	5
A5	0.62	6	0.86	2	0.82	7	0.87	4
A6	1.00	1	0.60	2	1.00	1	0.71	4
A7	0.72	3	0.86	2	0.82	4	1.00	3
A8	0.62	5	1.00	0	0.82	7	1.00	0
A9	0.86	2	0.86	1	0.86	2	1.00	0
A10	0.86	2	0.86	1	0.86	2	1.00	0
A11	-	-	0.52	5	0	∞	0.87	5

Table 10: Comparison of Gu and IDAT testability analysis results for the *example*

7 Conclusion

This paper has described the IDAT system, designed for allocation for testability of a datapath. Two processes have been presented, the testability analysis process and the test cost/quality trade-off process. Provided results show the capabilities of the system. IDAT is working on a virtual architecture of the datapath, enriched with design attributes of each components. Next step is to integrate the IDAT system in a HLS system designed at our laboratory, named UGH (User-Guided High-Level Synthesis System). UGH is being developed in the framework of the European ESPRIT project COSY/OMI.

8 Acknowledgment

The authors are grateful to Laurent Ducerf-Bourbon (LIP6) for his VHDL to RVA interface in the ALLIANCE environment, and to Vladimír Drábek and Zdeněk Kotásek (DCSE, Technical University of Brno) for their help.

References

[1] L.J. Avra and E.J. McCluskey. High-level synthesis of testable design: An overview of university systems. In *IEEE International Test Conference - Test Synthesis Seminar*, Washington D.C., October 1994.

[2] P. Bukovjan, M. Marzouki, and W. Maroufi. Allocation for testability by searching cost/quality trade-off in high-level synthesis for testability. Technical report, LIP6 Laboratory, Paris, February 1999.

[3] M.L. Flottes, R. Pires, and B. Rouzeyre. Analyzing testability from behavioral to rt level. In *Proceedings of European Design and Test Conference*, pages 158–165, Paris, France, March 1997.

[4] X. Gu. *RT Level Testability Improvement by Testability Analysis and Transformations.* PhD thesis, Linkoping University, Sweden, 1996. LiU-Tek-Ph.D 414, ISBN 91-7871-654-3.

[5] X. Gu, K. Kuchcinski, and Z. Peng. Testability analysis and improvement from vhdl behavioral specification. In *Proceesings of the European Design Automation Conferenece with EURO-VHDL*, pages 644–649, 1994.

[6] S.K. Gupta. Synthesis of testable combinational circuits: An overview of university activities. In *IEEE International Test Conference - Test Synthesis Seminar*, Washington D.C., 1994.

[7] A. Jerraya, I. Park, and K. O'Brien. Amical: An iterative high-level synthesis environment. In *European Design and Test Conference*, Paris, France, 1993.

[8] M.T.-C. Lee. *High-Level Test Synthesis of Digital VLSI Circuits.* Artech House Boston - London, 1997. ISBN 0-89006-907-7.

[9] T.E. Marchok, A. El-Maleh, J. Rajski, and W. Maly. Testability implication of performance-driven logic synthesis. *IEEE Design and Test of Computers*, 12(2):32–38, Summer 1995.

CC	0.996259
SC	0.920580
CO	0.943943
SO	0.895200

Table 11: Correlation coefficients for the *example*

[10] M. Marzouki, V.C. Alves, A.R. Antunes, and W. Maroufi. Global test scheduling and control in a hl-sft environment. In *1st IEEE High-Level Design Validation and Test Workshop*, pages 14–17, November 1996.

[11] 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.

[12] I. Pomeranz and S.M. Reddy. Synthesis of testable sequential circuits: An overview of university activities. In *IEEE International Test Conference - Test Synthesis Seminar*, Washington D.C. - USA, 1994.

[13] K.D. Wagner and S. Dey. High-level synthesis for testability: a survey and perspectives. In *Proceedings of the Design Automation Conference*, pages 131–136, 1996.

[14] X.A. Zhu. *A Knowledge-Based System for Testable Design Methodology Selection.* PhD thesis, University of Southern California, Los Angeles, August 1986. CRI-86-23.

[15] X.A. Zhu and M.A. Breuer. A knowledge-based system for selecting test methodologies. In *IEEE Design and Test of Computers*, October 1988.