INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

TECHNICAL UNIVERSITY OF BRNO
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
DEPARTMENT OF MICROELECTRONICS

PH.D. THESIS
to obtain the title of
DOCTOR OF THE INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
AND
DOCTOR OF TECHNICAL UNIVERSITY IN BRNO
FIELD: MICROELECTRONICS

Presented and defended to open public by
Peter BUKOVJAN

Allocation for Testability in High-Level Synthesis

COMMITTEE

Mr	Pierre	GENTIL	President
Mr	Jan	HLAVICKA	Reviewer
Mr	Christian	LANDRAULT	Reviewer
Mr	Jaromír	BRZOBOHATY	Thesis Director
Mrs	Meryem	MARZOUKI	Thesis Director
Mr	Zdenek	KOTASEK	Examiner

Thesis prepared partly in TIMA/INPG Laboratory, partly in LIP6/UPMC Laboratory
and partly at Technical University of Brno

Avertissement

Ce document présente mes travaux effectués dans le cadre d'une thèse en co-tutelle entre l'Institut National Polytechnique de Grenoble (France) et l'Université Technique de Brno (République tchèque). Dans son ensemble, ce document est rédigé en anglais. Toutefois le chapitre 1 du document est en français ; il constitue un résumé étendu des chapitres 2 à 11.

Abstract

This thesis belongs to the domain of hardware synthesis for testability. The objective of our work was to develop and implement a flexible allocation for testability tool, which is able to operate in conjunction with most of already existing High-Level Synthesis (HLS) systems in order to reach testable circuit data-path with minimal area overhead and performance degradation. The proposed allocation for testability tool performs necessary testability improving modifications of the data-path for external test (insertion of test points or scan) or for internal test - Built-In Self Test (BIST) at circuit Register-Transfer Level (RTL). The modified RTL design can be then returned to connected HLS system to perform final optimizations.

This thesis is mainly focused on RTL data-path modifications of a design, describing also the concept of the whole testable circuit architecture, composed from modified data-path, original circuit controller and newly created test controller.

In order to minimize performance degradation and maximize testability of final circuit, the cost/quality trade-off algorithm - with its key part RTL data-path testability analysis method - was developed to guide the allocation of testability resources. Proposed testability analysis technique calculates for each circuit node its controllability, observability and testability values. These values together with the values of test complexity of each component compose then a part of node controllability, observability and testability exigency metrics used to detect hard-to-test nodes with the largest influence to overall circuit testability.

Presented allocation for testability method was implemented into *Interactive Design for Test reuse in Allocation for Testability* - IDAT - tool. IDAT encapsulates test point insertion, scan approach and BIST into one tool giving the user more freedom in test methodology selection. Other feature of IDAT, which makes it different from already existing allocation for testability methods is, as the name implies, the user interactive approach and Design for Test (DFT) reuse concept. Our cost/quality trade-off algorithm takes into account the user experience in DFT area, his requirements to certain design and testability attributes of final circuit, available means from the library of components with implemented testability structure and imposed test method constraints. The cost/quality trade-off algorithm for scan and BIST test methodology is implemented in the form of the branch-and-bound optimization algorithm.

To demonstrate IDAT performance, the RTL architectures of two well known HLS benchmarks Diffeq and Tseng together with circuit Bert were taken. In the case of hard-to-test circuit Diffeq, very optimistic cost/quality relations were obtained, showing that important testability improvement was reached with minimal area overhead. Calculated testability metrics for Tseng circuit allocated for scan are compared with the real values of gate level fault coverage obtained by using of Mentor Graphics tools resulting in nearly 80% correlation between our defined metrics testability and corresponding gate level fault coverage.

Résumé

Cette thèse traite de la synthèse matérielle de haut niveau en vue de la testabilité. Notre travail a permis le développement d'une méthodologie originale pour ce faire, et sa mise en oeuvre dans un outil flexible d'allocation qui soit capable de fonctionner en conjonction avec la plupart des systèmes de synthèse de haut niveau existants, avec pour objectif l'obtention de chemins de données testables pour une augmentation minimale de la surface et une faible dégradation des performances. L'outil d'allocation pour la testabilité proposé réalise les modifications nécessaires du chemin de données pour le test externe (insertion de points de test ou *scan*) ou pour le test interne (BIST: *Built-In Self Test*), pour des descriptions au niveau transfert de registre (RTL). Le résultat est retourné au système de synthèse de haut niveau pour les optimisations finales.

Cette thèse est principalement axée sur la modification d'un chemin de données décrit au niveau RTL, prenant en compte également l'architecture testable dans son ensemble, composée du chemin de données modifié, du contrôleur original et du contrôleur de test généré.

Pour minimiser la dégradation des performances et optimiser la testabilité du circuit final, un algorithme réalisant le compromis coût/qualité - incluant une méthode d'analyse de la testabilité au niveau RTL - a été développé pour guider l'allocation des ressources de test. L'analyse de testabilité proposée calcule pour chaque nœud du circuit les valeurs de contrôlabilité, d'observabilité et de testabilité. Associées à la complexité de test de chacun des composants, ces valeurs permettent de définir l'exigence de contrôlabilité, d'observabilité et de testabilité, mesures utilisées pour détecter les nœuds difficiles à tester et leur influence sur le reste du circuit.

La méthode d'allocation en vue de la testabilité présentée est implémentée dans un outil nommé IDAT (*Interactive Design for test in Allocation for Testability*). IDAT intègre les approches par insertion de points de test, *scan* et BIST en un outil unique, permettant à l'utilisateur une totale liberté de sélection de la méthodologie. L'originalité d'IDAT réside également dans l'interactivité avec l'utilisateur, ainsi que dans le concept de réutilisation de blocs testables existants (*DFT reuse*). Notre algorithme de compromis coût/qualité respecte l'expérience du concepteur en matière de conception en vue du test, ses attentes sur les attributs de conception et de testabilité du circuit final, tout en exploitant les ressources de la bibliothèques de composants testables. Cet algorithme est implémenté sous la forme d'un parcours *branch and bound* de l'espace des solutions.

Afin de démontrer les performances d'IDAT, des circuits de référence en synthèse de haut niveau ont été considérés. Dans le cas de circuits difficiles à tester, un compromis coût/qualité très satisfaisant a été obtenu, montrant une augmentation importante de la testabilité pour un faible coût en surface. Les estimations de testabité obtenues par IDAT ont été comparées à celles indiquées par d'autres outils universitaires, ainsi qu'au taux de couverture obtenu avec un outil commercial. Les résultats de ces comparaisons témoignent d'une bonne corrélation.

Acknowledgments

First of all I would like to express my gratitude to all members of the committee of defense of my thesis for their acceptance to be in this committee. I would like to thank especially Prof. Jan Hlavička from Czech Technical University in Prague and Dr. Christian Landrault from LIRMM-CNRS in Montpellier for their effort with reviewing of my thesis.

I would like to thank my advisor Meryem Marzouki for her guidance and her help during my stay at Institute National Polytechnique de Grenoble and later at Laboratory of Informatics at University of Paris 6 (LIP6).

I greatly appreciate the help of my colleagues from the Integrated systems architecture and micro-electronics department (ASIM) of LIP6 under direction of Prof. Alain Greiner, especially Laurent Ducerf-Bourbon and Walid Maroufi. Their contributions have greatly improved the quality of this work. I would like also to thank my colleagues from Technical University of Brno Zdeněk Kotásek and Vladimír Drábek. Their expertise in the area of gate level testing provided a good background of my research. Many thanks to Jan Velich, Jan Vrbík, Luboš Hradecký and Robert Kvaček from ASI-Centrum in Prague who helped me with Mentor Graphics tools. On the administrative side, I would like to thank especially to Nicole Fouque from ASIM for her readiness to help me to solve many administrative problems concerning my stay at LIP6.

Finally I would like to thank my parents for their support. Last but not least I would like to thank Iva for bringing me happiness.

This work has been mainly supported by the grant of French Government, then by MEDEA-SMT project, by the grant from Grant Agency of Czech Republic, contract No. 102/98/1463 and also by ON Semiconductor Czech Design Center in Rožnov p/R.

Peter Bukovjan

Contents

List of Figures

List of Tables

Chapter 1

Allocation en vue de la testabilité dans le cadre de la synthèse de haut niveau (Résumé étendu)

Résumé :
Nous présentons dans ce chapitre une synthèse en langue française de nos travaux, détaillés dans les chapitres 2 à 11 qui suivent. Notre travail est consacré au développement d'une approche pour la synthèse en vue de la testabilité (*Synthesis for Testability* ou SFT). Cette approche est implémentée dans l'outil pour l'allocation en vue de la testabilité IDAT. Dans le contexte de la synthèse de haut niveau, ce processus repose sur la recherche du meilleur compromis coût/qualité entre les demandes de l'utilisateur et les moyens pour améliorer la testabilité. Notre outil considère les composants accessibles dans la bibliothèque et la possibilité de générer des structures de test supplémentaires. La décision dans le processus de compromis est basée sur les résultats de l'analyse de la testabilité.

Mots clés :
synthèse en vue de la testabilité, réutilisation, DFT, analyse de testabilité, scan, BIST

1.1 Introduction

La plupart des outils de synthèse de haut niveau (*High-Level Synthesis* ou HLS) sont conçus pour optimiser un ensemble de critères de conception, sans considération pour les conséquences sur la testabilité du circuit obtenu. Celui-ci est soit très difficile à tester, soit testable avec de mauvaises performances et une grand surface additionnelle si la testabilité est considérée trop tard dans le processus de synthèse.

Pour éviter ces conséquences, plusieurs auteurs considèrent la solution de la synthèse de haut niveau en vue de la testabilité (*High-Level Synthesis for Testability* ou HL-SFT). Dans cette solution, la testabilité est incluse comme critère d'optimisation avant la génération d'architecture au niveau transfert de registres (*Register-Transfer Level* ou RTL). Une présentation détaillée de ces solutions peut être trouvée par exemple dans [Lee97].

L'étape habituellement exploitée par les systèmes qui introduisent la testabilité dans le processus de synthèse de haut niveau est celle de l'allocation de ressources. Dans cette étape, les variables et les fonctions de description comportementale du circuit sont assignées aux registres et aux opérateurs. Le grand nombre de solutions possibles et l'influence importante de ces solutions sur l'architecture finale du circuit offrent beaucoup de possibilités pour améliorer la testabilité du circuit global. Mais les approches existantes souffrent de trois inconvénients importants :

- elles ne tirent pas profit des composants testables ou des structures testables existants
- elles sont difficiles à guider pour obtenir le meilleur compromis coût/qualité entre les désirs de l'utilisateur et les moyens accessibles
- elles sont orientées soit vers la synthèse pour le test interne soit vers la synthèse pour le test externe[1].

L'approche HL-SFT que nous avons définie et implémentée dans notre outil IDAT (*Interactive Design for Test Reuse in Allocation for Testability*), vise à surmonter ces inconvénients. Le processus d'allocation en vue de la testabilité est considéré comme la génération automatique d'architectures testables à partir de l'architecture virtuelle enrichie (*Rich Virtual Architecture* ou RVA), en exploitant le concept de réutilisation du DFT [MAAM96].

Ce document est organisé de la façon suivante : la section 1.2 présente un état de l'art des travaux dans le domaine considéré. La section 1.3 décrit l'outil pour l'allocation en vue de la testabilité IDAT. La section 1.4 résume les architectures créées par IDAT. La section 1.5 présente ensuite le module d'analyse de la testabilité et la section 1.6 expose le processus de compromis coût/qualité du circuit. L'implémentation de l'outil IDAT est décrite dans la section 1.7. Enfin, la section 1.8 présente les résultats expérimentaux obtenus.

1.2 État de l'art

1.2.1 Conception en vue du scan

Le scan partiel est une technique de DFT ordinairement utilisée pour améliorer la contrôlabilité et l'observabilité des noeuds internes d'un circuit. Le principe du scan est

[1] À notre connaissance seuls le système UST TEST [Bre95] de l'Université de Californie du Sud et l'outil IBM TestBench [Ibm00] offrent ces deux techniques au niveau RTL. Les outils de Mentor Graphics sont aussi capables de gérer l'insertion des deux structures scan et BIST, mais l'introduction du chemin du scan n'est réalisée qu'au niveau porte. L'inconvénient principal de ces outils est la non réutilisation du DFT et le manque de contrôlabilité du processus de synthèse.

montré en figure 1.1. Plus de détails concernant les techniques de scan peuvent être trouvés dans [ABF95].

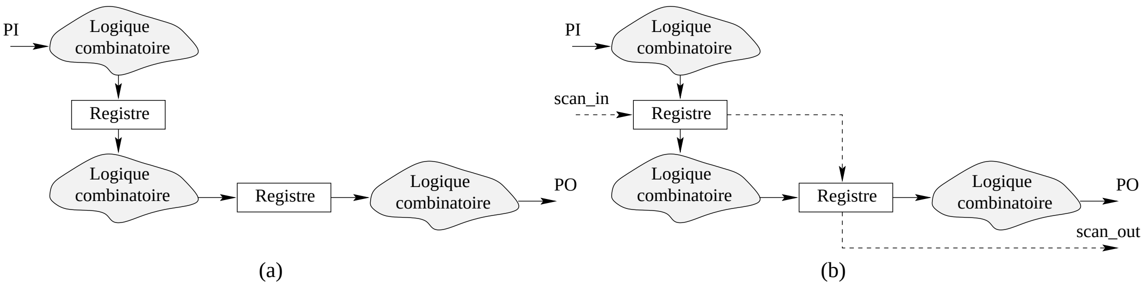


Figure 1.1: Principe du scan : (a) système générique ; (b) système avec chemin de scan

1.2.2 Conception en vue du BIST

La technique du BIST (*Built-In Self Test*) est basée sur l'inclusion de structures de test dans un circuit afin de lui permettre de se tester lui-même. On obtient de tels systèmes par l'ajout et la connexion d'un ou plusieurs générateurs de vecteurs de test (*Test Pattern Generators* ou TPGs) ordinairement sous la forme de LFSR (*Linear Feedback Shift Registers*) et d'un ou plusieurs analyseurs de réponses (*Test Result Checkers* ou TRC), souvent des MISRs (*Multiple-Input Signature Registers*). TPG et TRC peuvent être aussi implémentés sous la forme de registres BILBO (*Built-In Logic-Block Observer*) ou CBILBO qui peuvent fonctionner simultanément comme TPG et TRC. Historiquement, les techniques de BIST se sont concentrées sur le test d'Unités Fonctionnelles (UFs), mais des approches plus récentes permettent aussi le test des interconnexions. Plus de détails concernant les techniques de BIST peuvent être trouvés dans [ABF95].

Comme toutes les techniques qui s'efforcent d'améliorer la testabilité du circuit, la technique du BIST vise l'obtention de la testabilité demandée en maintenant la surface additionnelle et le temps de test à des valeurs acceptables. Les approches basées sur l'utilisation des registres BIST (ordinairement BILBO) ont typiquement comme objectif principal la maximisation du partage de ces registres entre les blocs à tester et donc la minimisation du nombre total de ces registres (comme par exemple dans [HP93] ou dans [PGB95]).

Dans [BFR98], les registres et les UFs choisis sont utilisés pendant la phase de test en tant que TPGs et TRCs (LFSRs, générateurs arithmétiques, etc.) pour le test des autres UFs. L'extension de cette approche au test d'interconnexions est présentée dans [BFR99]. Ce test est assuré par l'allocation de points de test. L'inconvénient de cette technique est la nécessité de simulation de fautes à chaque étape de l'itération pour identifier les fautes non observables.

1.3 Allocation en vue de la testabilité

Le processus d'allocation en vue de la testabilité est implémenté dans l'outil IDAT. En fonction de la méthode de test choisie, ce processus réalise un compromis coût/qualité entre les demandes de l'utilisateur et la disponibilité des composants testables. Ce processus est complété par l'étape de génération du plan de test [MAAM96].

L'outil IDAT est fait pour être utilisé avec un système classique de synthèse de haut niveau qui produit une architecture RTL. L'environnement global de synthèse obtenu n'optimise pas le circuit simplement selon les critères de surface et de performance, mais aussi selon la testabilité. Le processus d'allocation en vue de la testabilité peut être vu dans son ensemble comme la génération automatique d'architectures testables, à partir d'une architecture RTL enrichie par les attributs de conception (*Rich Virtual Architecture* RVA), en tirant profit du concept de réutilisation du DFT. L'outil IDAT est capable de modifier l'architecture du chemin de données existant par l'insertion de points de test, de scan ou de BIST dans un circuit. L'outil IDAT est complété par le générateur de plan de test (*Test Scheduler* ou TesS) et la bibliothèque dynamique d'UFs comprenant le générateur de structures pour le BIST (*BIST Structure Generator* ou BSG) [DBBM00].

Le processus d'allocation en vue du test est constitué de cinq étapes :

1. L'extraction de l'architecture RVA issue de l'étape d'allocation des ressources du système de synthèse de haut niveau.

2. L'extraction des versions testables des UFs du circuit, situées dans la bibliothèque de composants testables.

3. Le calcul des attributs de conception et de test du circuit original où le chemin de données est déjà isolé du contrôleur du circuit original par un bloc de multiplexeurs.

4. Le compromis coût/qualité selon la méthode de test choisie par le concepteur. Les méthodes possibles dans IDAT sont l'insertion de points du test, le scan et le BIST. Ce processus se fie à chaque itération au calcul des attributs du circuit.

5. Le chemin de donnees RTL modifié (testable) est retourné au système de synthèse de haut niveau pour compléter l'ordonnancement.

1.4 Architectures de test

Un exemple d'architecture RTL schématique pris comme entrée d'IDAT est présenté en figure 1.2.

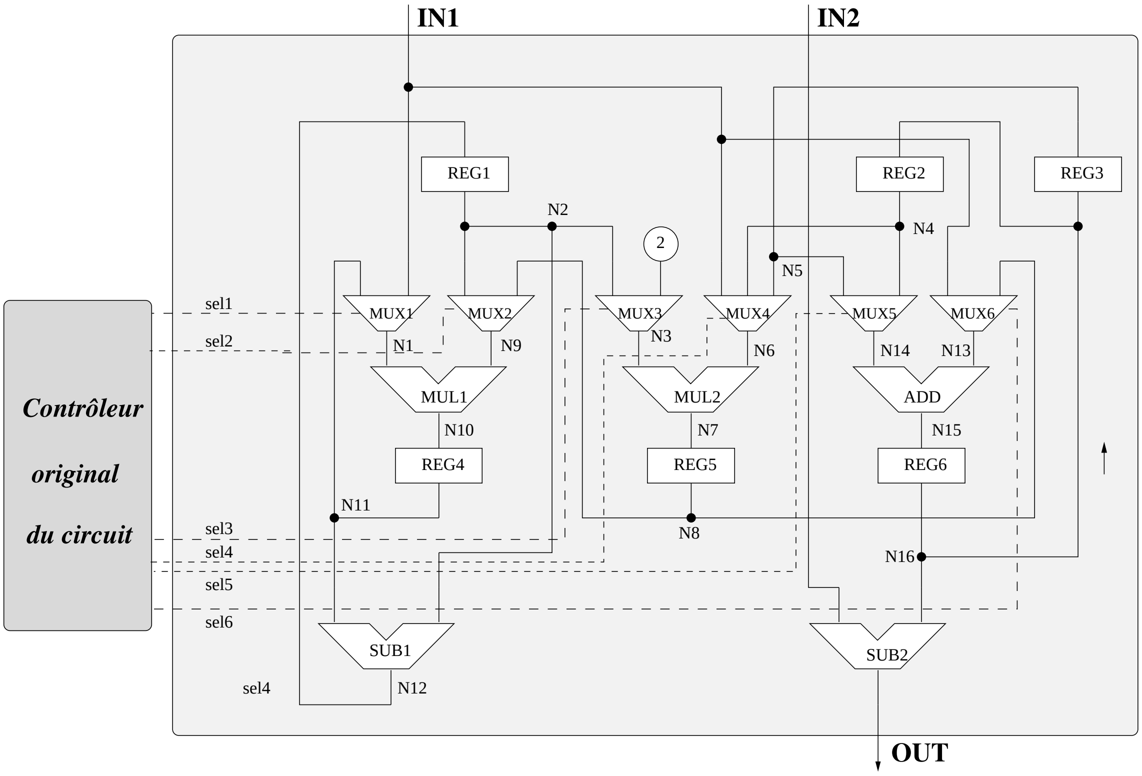


Figure 1.2: Architecture du circuit Diffeq

1.4.1 Architecture avec insertion de points de test

L'insertion de points de test est une technique simple et immédiate de DFT pour améliorer la contrôlabilité et l'observabilité des signaux sélectionnés. Les points de test sont insérés de la façon suivante :

- Pour améliorer la contrôlabilité d'un noeud, on insère un multiplexeur dont une entrée est une entrée primaire nouvellement créée.
- Pour améliorer l'observabilité du noeud, celui-ci devient une nouvelle sortie primaire. Un exemple d'architecture avec des points de test insérés est présenté en figure 1.3.

1.4.2 Architecture scan

L'architecture de chemin de données de type scan path est illustré par la figure 1.4. Les caractéristiques principales de notre architecture scan sont les suivantes :

- Les entrées *select* des multiplexeurs ajoutés, les entrées *RESET/SET* des composants séquentiels et les entrées des signaux de contrôle *write enable* des registres sont connectés par des multiplexeurs au contrôleur de test.

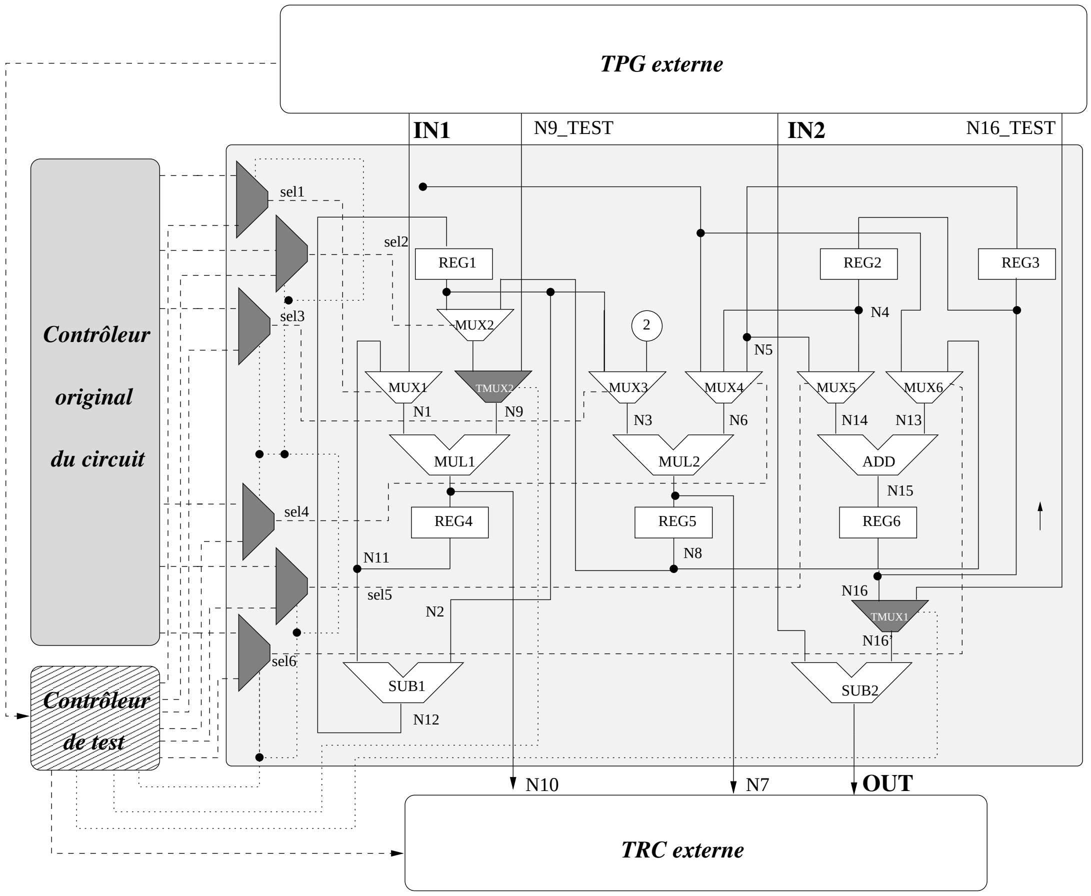


Figure 1.3: Architecture du circuit Diffeq avec insertion de points de test

- Le processus de substitution de composants par leur équivalent scan n'est pas limité aux registres, mais généralisé à toutes les UFs séquentielles qui existent dans la bibliothèque en version scan.
- Les chemins de scan multiple sont réalisables.

1.4.3 Architecture BIST

L'architecture du chemin de données de type BIST est illustrée par la figure 1.5. Les caractéristiques principales de notre architecture BIST sont les suivantes :

- Les entrés *select* des multiplexeurs ajoutés, les entrées *RESET/SET* des composants séquentiels et les entrées des signaux de contrôle *write enable* des registres sont connectés par des multiplexeurs au contrôleur de test.
- Les versions BIST existantes des UFs sélectionnées sont utilisées pour remplacer les UFs équivalentes non testables. Ce remplacement améliore le temps de test, diminue la complexité des TPG et TRC (dans le cas où le TPG et le TRC sont placés

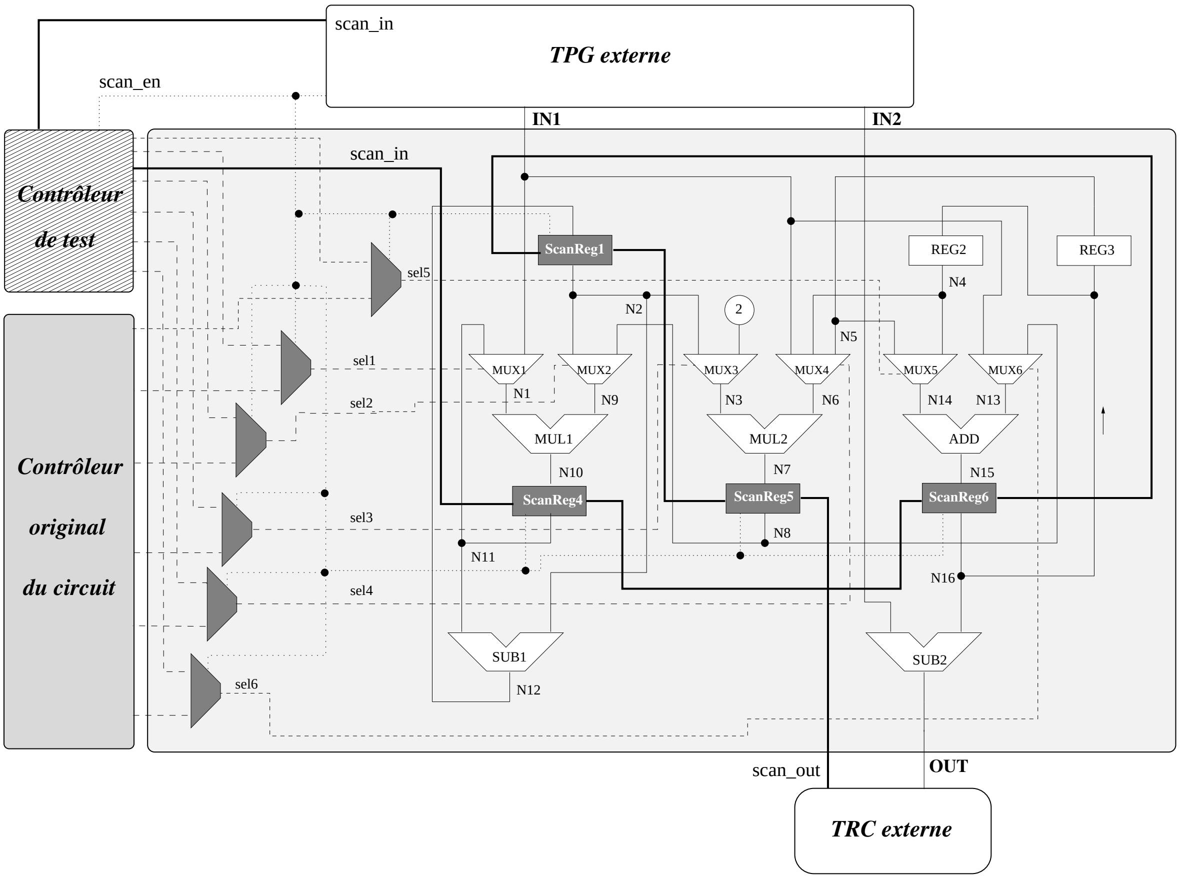


Figure 1.4: Architecture du circuit Diffeq avec chemin de scan

dans le circuit), mais la surface additionnelle de la version BIST n'est généralement pas négligeable. Quand la substitution est effectuée, les signaux de contrôle sont directement connectés au contrôleur de test.

- Si la version BIST d'une UF sélectionnée n'existe pas ou si la surface additionnelle est trop importante, un TPG et un TRC uniques sont ajoutés dans le chemin de données pour tester les composants sélectionnés. Toutes les entrées des UFs (entrées de contrôle incluses) sont connectées au TPG et au TRC, directement ou à travers un ou plusieurs composants. Les TPG et TRC sont partagés par les UFs qui ne sont pas BISTées. Pour cette raison, ils sont paramétrables et dynamiquement reconfigurables : la largeur de sortie du TPG et la largeur d'entrée du TRC sont modifiables. D'ores et déjà, nous avons implémenté le générateur de structures de test ; il permet de synthétiser des structures à base de LFSR, MISR et BILBO. Ce générateur prend comme entrée les paramètres des structures de test fournis par IDAT. D'autre types de structures de test, basées par exemple sur les automates cellulaires [CSdAAC99], peuvent être aussi pris en considération par IDAT.

- Les interconnexions qui demeurent non testées peuvent l'être par un chemin de

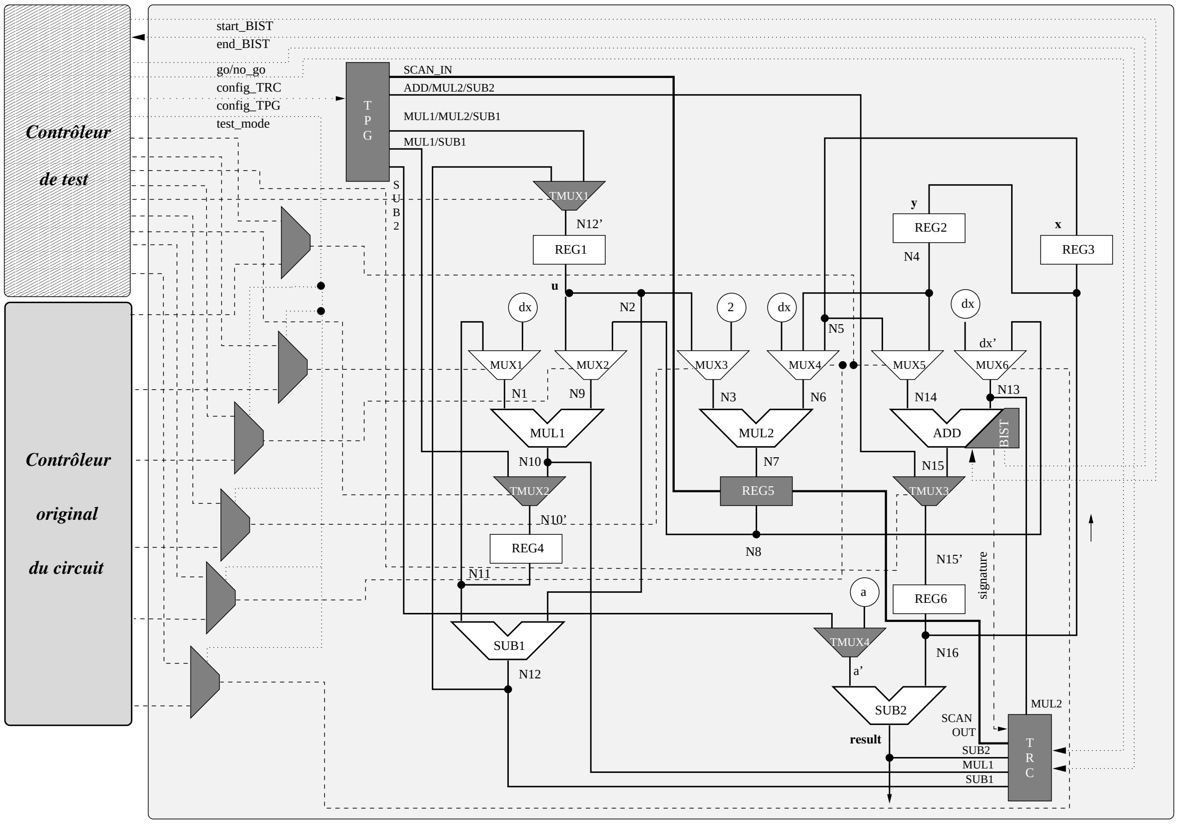


Figure 1.5: Architecture du circuit Diffeq avec BIST

scan prenant sa source au TPG et finissant au TRC. Dans ce cas, les entrées *scan enable* sont connectées au contrôleur de test.

1.5 Analyse de testabilité

Notre analyse de testabilité est basée sur le calcul de la contrôlabilité, de l'observabilité et de la testabilité de chaque noeud dans le circuit. Malgré la présence dans ce document de concepts communs avec d'autres approches connues de la littérature (en particulier avec l'approche de Gu [GKP94]), notre processus d'analyse de testabilité diverge de ces approches par le fait que nous supposons la présence d'un contrôleur de test quel que soit la méthode de test : insertion de points de test, scan ou BIST. Notre approche générale d'analyse de la testabilité du circuit a été déjà publiée dans [BMM99].

L'algorithme de propagation a été développé pour calculer la contrôlabilité (C) et l'observabilité (O) de chaque noeud dans le circuit. Dans la version actuelle d'IDAT, la contrôlabilité et l'observabilité reflètent les problèmes de test causés par les boucles fermées (non initialisables) et le problème de la perte de transparence causée par le passage à travers les différentes UFs.

Les valeurs obtenues de contrôlabilité et d'observabilité pour chaque noeud sont ensuite utilisées dans le calcul des exigences de contrôlabilité et d'observabilité (CE et OE). Ces nouvelles métriques reflètent le plan du test effectif du circuit où plus de 90% du temps de test est consommé par le test des composants complexes (les UFs). L'exigence de contrôlabilité du noeud correspond à la nécessité de contrôler ce noeud considérant la complexité de test des composants qui dépendent de ce noeud. De même, l'exigence d'observabilité du noeud correspond à la nécessité d'observer ce noeud tenant compte de la complexité de test des composants en amont.

1.5.1 Analyse de testabilité de l'architecture avec insertion de points de test

Dans le cas de l'insertion de point de test, le processus d'analyse de la testabilité est utilisé pour guider cette insertion. Le calcul de la contrôlabilité (resp. de l'observabilité) de chaque noeud part des entrées (resp. des sorties) primaires et s'achève aux sorties (resp. aux entrées) primaires.

1.5.2 Analyse de testabilité de l'architecture scan

Dans le cas de la méthodologie scan, le processus d'analyse de la testabilité est utilisé pour guider la sélection d'un ensemble de composants séquentiels (registres, UFs), qui doivent être transformés en version scan. Le calcul de la contrôlabilité (resp. de l'observabilité) de chaque noeud part des entrées (resp. des sorties) primaires et des sorties (resp. des entrées) des composants scan et s'achève aux sorties (resp. aux entrées) primaires et aux entrées (resp. aux sorties) des composants scan.

1.5.3 Analyse de testabilité de l'architecture BIST

Dans le cas de la méthodologie BIST, le processus d'analyse de la testabilité est utilisé pour guider la sélection d'un ensemble d'UFs qui doivent être BISTées. Le calcul de la contrôlabilité (resp. de l'observabilité) de chaque noeud part des sorties du TPG (resp. des entrées du TRC) et des sorties (resp. des entrées) des composants scan et s'achève aux entrées TRC (resp. aux sorties du TPG) et aux entrées (resp. aux sorties) des composants scan.

1.6 Processus de compromis coût/qualité

Le processus de compromis coût/qualité du test est le coeur du processus d'allocation pour la testabilité. Le processus de compromis a pour but de faire le meilleur choix entre différentes versions d'un composant et propose l'inclusion de structures additionnelles de test pour satisfaire les spécifications de l'utilisateur. En plus des demandes de l'utilisateur, l'algorithme de compromis prend en considération d'autres critères, comme les demandes imposées par la méthode de test sélectionnée. Le compromis est implémenté à l'aide d'un algorithme *branch-and-bound* en utilisant les critères décrits.

Le processus de compromis cherche d'abord à satisfaire les demandes de l'utilisateur. Si c'est impossible avec la bibliothèque existante, le dialogue commence entre IDAT et l'utilisateur à qui il est demandé de modifier pas à pas ses demandes. L'utilisateur est guidé dans ses décisions par l'outil.

Le problème de la testabilité d'un circuit est par nature global. Un composant 100% testable peut montrer des problèmes de testabilité quand il est immergé dans le chemin de données. Pour éviter cette situation, l'algorithme de compromis est décomposé en deux modules principaux : le module de compromis local, qui agit au niveau des composants, et le module de compromis global, qui fonctionne au niveau du chemin de données.

1.6.1 Critères de compromis

Les critères du compromis coût/qualité sont les suivants :

- Les *attributs locaux* sont les attributs des composants alloués originellement par le système de synthèse de haut niveau et ceux des composants extraits de la bibliothèque de composants testables. Tous ces attributs constituent le vecteur caractéristique de chaque composant.
- Les *attributs globaux* sont les attributs du circuit constitué des composants alloués et interconnectés, y compris les structures additionnelles de test. Ces attributs sont liés à chaque configuration du circuit et forment le vecteur des attributs globaux.
- Les *spécifications de l'utilisateur relativement à la qualité du circuit* sont les déclarations explicites des objectifs de testabilité ainsi que certaines caractéristiques de conception du circuit. L'utilisateur doit spécifier toutes ses demandes afin de guider le processus d'allocation. Elles sont classées en trois groupes selon leur importance dans le rapport coût/qualité. Cette classification correspond aussi à l'expérience de l'utilisateur dans le domaine de la synthèse de haut niveau en vue de la testabilité et dans le domaine de la conception en vue du test.
- Les *coefficients de pondération* permettent à l'utilisateur d'exprimer ses préférences pour l'ordonnancement de ses demandes.
- Les *contraintes imposées par la méthode de test* influent sur le processus de sélection de la version optimale d'un composant et d'incorporation des structures du test.

1.6.2 Fonctions de compromis

L'algorithme de compromis utilise les fonctions de calcul suivantes :

- Une *fonction de distance* pour estimer la différence entre la valeur d'un attribut du vecteur d'allocation et sa valeur correspondante dans le vecteur des spécifications de l'utilisateur.
- Une *fonction de crédits/pénalités* qui normalise le résultat de la fonction de distance, afin d'effectuer les comparaisons.
- Une *fonction de score* qui attribue une valeur à chaque version du composant pour classer les différentes alternatives proposées.

Ces fonctions sont équivalentes aux fonctions utilisées dans [Zhu86] et [ZB88], puisqu'on résout le même type de problème, à un niveau d'abstraction plus élevé.

1.6.3 Compromis local

Lorsque le chemin de données du circuit est extrait du système de synthèse de haut niveau, IDAT entame la recherche dans la bibliothèque des composants testables. Les composants testables, dont la fonction correspond aux composants originels (non testables) et dont la testabilité est conforme aux spécifications de l'utilisateur, sont identifiés.

Lors du compromis local, les attributs du composant originel et les attributs de chaque version testable de ce composant sont comparés avec les spécifications de l'utilisateur. Le résultat de cette comparaison est la liste de toutes les versions, ordonnée selon le niveau de satisfaction des spécifications.

1.6.4 Compromis global

Compromis coût/qualité en mode scan

Certains des concepts du scan partiel sont basés sur la recherche des noeuds les moins observables et contrôlables dans le circuit, pour améliorer sa contrôlabilité et son observabilité globales par remplacement des registres sélectionnés au moyen de leurs équivalents scan. Dans notre approche, nous prenons en considération l'existence d'UFs scannables dans la bibliothèque de composants testables. La recherche des noeuds les moins testables dans notre méthode d'allocation en vue de la testabilité est basée sur la connaissance de l'exigence de contrôlabilité et de l'exigence d'observabilité pour chaque noeud du circuit. Ces exigences sont fortement dépendantes de la complexité du test des composants auxquels le noeud sélectionné est connecté. Autrement dit, une UF complexe, dont les entrées ne sont pas suffisamment contrôlables et/ou dont les sorties ne sont pas suffisamment observables en fonction de cette métrique, est soit directement transformée en son équivalent scan, s'il existe, soit sélectionnée pour être connectée à des ressources de scan.

L'algorithme d'allocation des ressources de scan commence par l'identification du noeud le moins testable dans le circuit. Il compare ensuite ses exigences de contrôlabilité (CE) et d'observabilité (OE) :

- $CE \geq OE$: Si le noeud identifié est l'entrée d'une UF qui peut être transformée en version scan, cette UF est sélectionnée. Si ce noeud est connecté aux entrées de plusieurs UFs qui peuvent être transformées en versions scan, l'UF la plus complexe est sélectionnée. Dans les autres cas, on cherche un registre ou une UF connecté en aval du noeud identifié et pouvant être transformé en version scan.

- $CE < OE$: Si le noeud identifié est la sortie d'une UF qui peut être transformée en version scan, cette UF est sélectionnée. Dans les autres cas, on cherche un registre ou une UF connecté en amont du noeud identifié et pouvant être transformé en version scan.

Le composant sélectionné est alors transformé en son équivalent scan et correctement connecté dans le chemin de données. Le nombre de chemins de scan à créer dépend du compromis entre le nombre maximal de cycles de test (temps de test) spécifié et la surface additionnelle maximale tolérée.

Compromis coût/qualité en mode BIST

Le compromis global pour le BIST est effectué à deux niveaux :

- Entre toutes les UFs pour sélectionner l'ensemble des UFs dont la testabilité a l'influence la plus forte sur la testabilité globale du circuit. La sélection d'une UF à tester est guidée par les résultats d'analyse de la testabilité et, si plusieurs candidats de même complexité de test existent, par le nombre d'interconnexions qui peuvent être testées simultanément au test de cette UF. Un algorithme spécifique a été développé pour sélectionner le chemin de test le plus long et pour réutiliser les connections au TPG et au TRC déjà existants en empêchant les reconvergences possibles (voir figure 1.6). Sur la figure 1.6, l'UF sélectionnée pour le test est MUL1. On teste simultanément MUL1 et les interconnexions suivantes : TMUX1 → REG1, REG1 → MUX2, MUX2 → MUL1, TMUX2 → REG4, REG4 → MUX1 et MUX1 → MUL1. Dans la version actuelle d'IDAT, le chemin permettant la réutilisation des connexions déjà existantes est prioritaire sur le plus long chemin possible.

- Entre les possibilités de transformer une UF originelle en sa version BIST, si la version BIST existe, ou de tester l'UF originelle en utilisant les TPG et TRC reconfigurables. Si on compare le choix de la version BIST d'une UF avec le branchement de cette UF aux TPG et TRC partagés par plusieurs UFs, la solution BIST réduit généralement le temps de test du circuit (du fait du parallélisme possible), réduit la complexité du TPG et du TRC, mais produit une surface additionnelle généralement plus importante. Pour cette dernière raison, il est parfois plus avantageux de connecter les TPG et TRC aux UFs, même si elles existent également en version BIST.

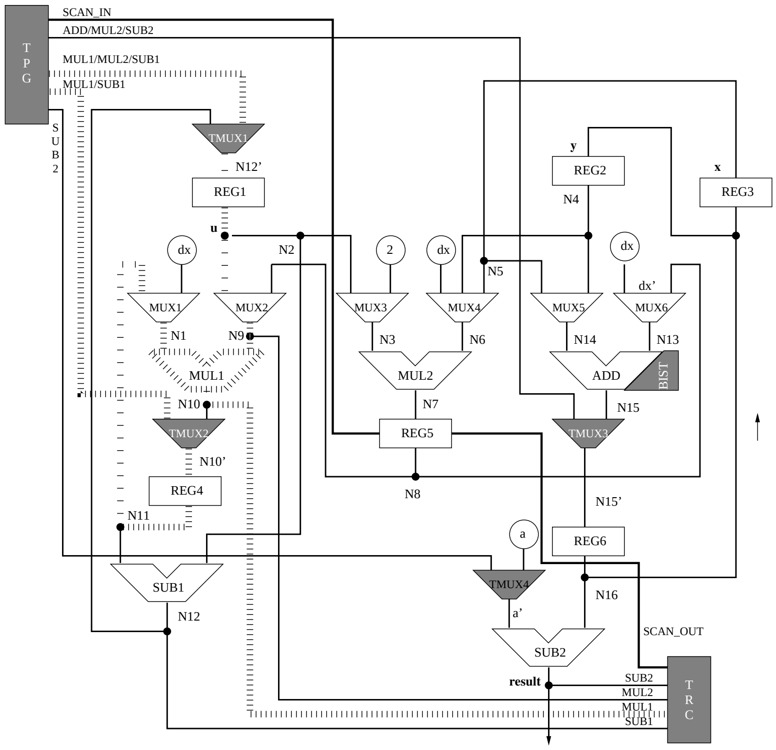


Figure 1.6: Circuit de la figure 1.5 montrant le chemin de test de MUL1

1.7 Implémentation

Dans le cadre de l'environnement de la suite de CAO ALLIANCE [Ali99] développée au laboratoire, l'entrée de l'outil IDAT est la description VHDL-RTL du chemin du données qui est automatiquement traduite au format RVA (*Rich Virtual Architecture*). Dans le cadre d'autres environnements, il est nécessaire de créer le traducteur produisant le RVA. Il est aussi possible d'utiliser une description RVA créée manuellement. Une description RVA contient, en plus de la description de l'architecture RTL, l'information concernant le type des noeuds (DATA, CONTROL et CLK), l'information concernant la transparence des composants pour les vecteurs de test et l'information sur les attributs de conception comme par exemple la surface.

La sortie de l'outil IDAT est, entre autres, la description (en VHDL) du chemin de données modifié (testable), avec la liste de toutes les modifications architecturales effectuées pendant le processus d'allocation en vue de la testabilité. Cette sortie est retournée au système de synthèse de haut niveau, pour poursuivre ses phases subséquentes.

1.8 Résultats

Cette section présente les résultats de nos algorithmes d'insertion de points de test et d'allocation en mode scan et en mode BIST. Ces résultats sont fournis pour deux benchmarks, Diffeq et Tseng, décrits originellement au niveau comportemental. Les attributs de conception ont été obtenus par synthèse.

Le chemin de données du circuit Diffeq au niveau RTL est repris de [PKG86] (figure 1.2). Afin de pouvoir utiliser notre bibliothèque de composants, nous avons ajouté à ce schéma des étiquettes pour les noms des composants et des noeuds, et nous avons remplacé le comparateur en sortie du circuit par le soustracteur SUB2. Enfin, nous avons connecté le chemin de données de Diffeq au contrôleur du circuit.

Le chemin de données du circuit Tseng au niveau RTL est également repris de [PKG86] (figure 1.7). Comme pour Diffeq, nous avons ajouté à ce schéma les étiquettes avec les noms des composants et des noeuds.

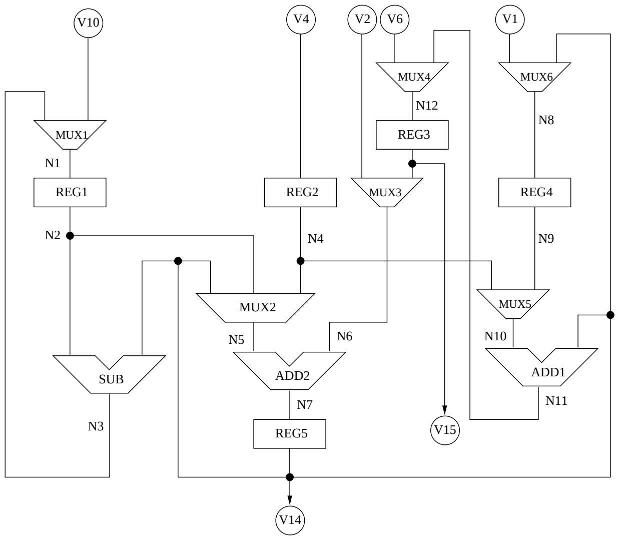
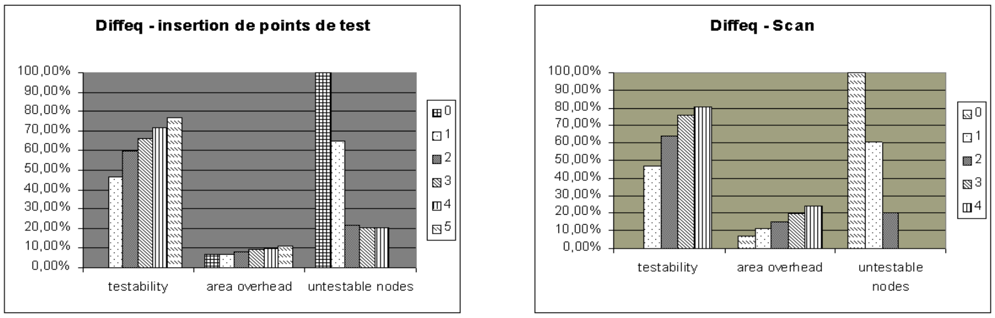


Figure 1.7: Architecture du circuit Tseng

1.8.1 Résultats en mode insertion de points du test

Benchmark Diffeq

Les spécifications de l'utilisateur sont les suivantes : testabilité supérieure à 75%, surface additionnelle inférieure à 12% et aucun noeud non testable. Les attributs globaux sélectionnés sont illustrés à chaque étape de la modification du chemin de données par le diagramme de la figure 1.8(a). Les modifications réalisées sont les suivantes : (1) N10 devient entrée primaire, (2) N16 devient entrée primaire, (3) N9 devient entrée primaire, (4) N7 devient entrée primaire et (5) N2 devient entrée primaire.

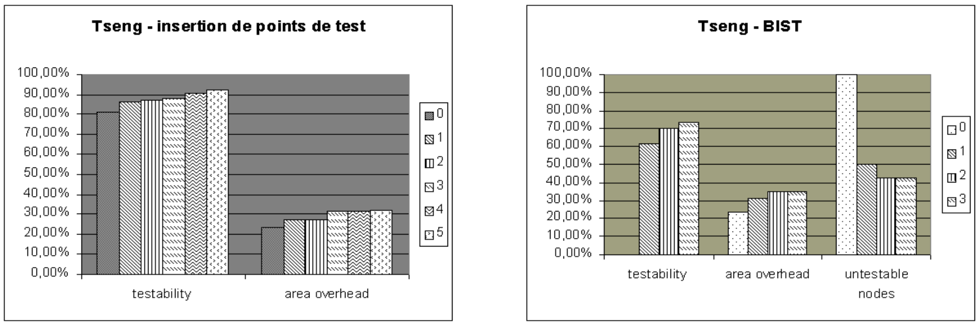


(a) (b)

Figure 1.8: Sélection d'attributs pour différentes configurations de Diffeq, (a) avec insertion de points de test, (b) avec chemin de scan

Benchmark Tseng

Les spécifications de l'utilisateur sont les suivantes : testabilité supérieure à 90% et surface additionnelle inférieure à 40%. Les attributs globaux sélectionnés sont illustrés à chaque étape de la modification du chemin de données par le diagramme de la figure 1.9(a). Les modifications réalisées sont les suivantes : (1) V14 devient entrée primaire, (2) N3 devient entrée primaire, (3) N7 devient entrée primaire, (4) N2 devient entrée primaire et (5) N4 devient entrée primaire.



(a) (b)

Figure 1.9: Sélection d'attributs pour différentes configurations du circuit Tseng, (a) avec insertion de points de test, (b) avec BIST

1.8.2 Résultats en mode scan

Exemple branch-and-bound

Le déroulement de l'algorithme *branch-and-bound* utilisé est montré sur l'exemple illustré en figure 1.10. Des versions scan existent pour les deux UFs *FU1* et *FU2*. La surface additionnelle de *FU1* est supérieure à la surface additionnelle de *FU2* dans ses versions scan. Les spécifications de l'utilisateur sont les suivantes : testabilité du circuit supérieure à 90% et et surface additionnelle inférieure à 11%. Les étapes de l'algorithme *branch-and-bound* sont présentées en table 1.1.

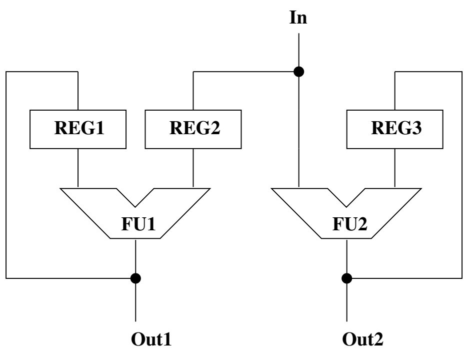


Figure 1.10: Exemple de circuit pour session branch-and-bound en mode scan

Cet exemple montre la situation ou, bien que la version scan de *FU1* existe, il est plus avantageux, du point de vue de la surface additionnelle totale, de tester cette UF par d'autres moyens. Le test est assuré par la transformation de *REG1* en registre scan et par l'utilisation simultanée de l'entrée primaire *In*. Cette solution induit peu de surface additionnelle. Le problème est que parfois, lorsque l'UF testée contient plusieurs boucles fermées, un tel test externe (par rapport à cette UF) n'est pas très efficace et il est nécessaire, pour satisfaire la demande de testabilité, de transformer cette UF en sa version scan.

Étape	Opération effectuée	T	AO	Étape	Opération effectuée	T	AO
1	remplacement de FU1	80%	10%	5	retour à l'original de REG3	80%	10%
2	remplacement de FU2	99%	20%	6	retour à l'original de FU1	50%	0%
3	retour à l'original de FU2	80%	10%	7	remplacement de REG1	60%	2%
4	remplacement de REG3	85%	12%	8	remplacement de FU2	90%	10%

Table 1.1: Exemple utilisé pour la session branch-and-bound en mode scan. T : testabilité, AO : surface additionnelle. Les valeurs soulignées ne satisfont pas les demandes de l'utilisateur

Benchmark Diffeq

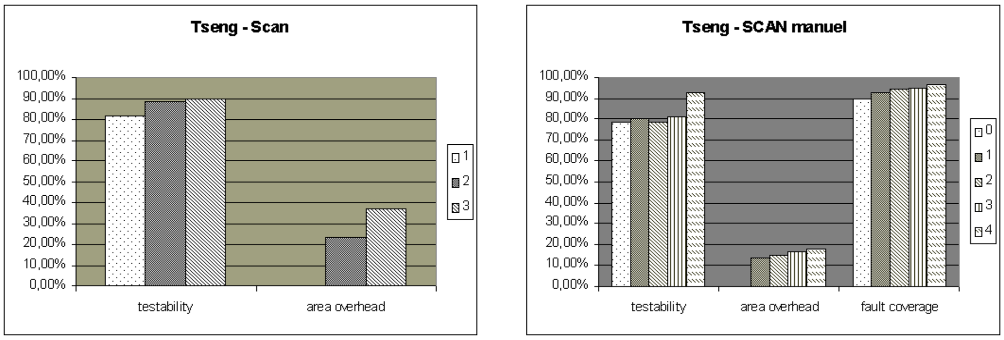
Les spécifications de l'utilisateur sont les suivantes : testabilité supérieure à 77%, surface additionnelle inférieure à 25% et quantité de noeuds non testables inférieure à 3%. Les attributs globaux sélectionnés sont illustrés à chaque étape de la modification du chemin de données par le diagramme de la figure 1.8(b).

La configuration 0 correspond au circuit originel avec le bloc de multiplexeurs (voir section 1.4). Dans la bibliothèque des composants testables, seuls les registres existent en version scan. IDAT effectue alors ces transformations dans l'ordre suivant : *REG4*, *REG6*, *REG1* et *REG5*. Puisque l'utilisateur ne demande pas un temps de test spécifique, tous les registres susmentionnés sont connectés en un seul chemin de scan.

Kotásek et al. dans [KRH00] ont obtenu des résultats similaires par application d'une méthode formelle. Leurs résultats sélectionnent soit l'ensemble {REG1, REG4, REG6} soit l'ensemble {REG1, REG3, REG4} comme les meilleurs candidats pour la transformation en version scan.

Benchmark Tseng

Les attributs globaux sélectionnés à chaque étape de la modification du chemin de données du circuit Tseng sont illustrés par le diagramme de la figure 1.11(a). Les spécifications de l'utilisateur sont les suivantes : testabilité supérieure à 89% et surface additionnelle inférieure à 60%. Les modifications réalisées sont les suivantes : (1) *REG5* est transformé en son équivalent scan et connecté au chemin de scan et (2) *REG1* est transformé en son équivalent scan et également connecté au chemin de scan.



(a) (b)

Figure 1.11: Sélection d'attributs pour différentes configurations du circuit Tseng en mode scan, (a) obtenues automatiquement avec IDAT, (b) obtenues manuellement

Pour comparer l'attribut testabilité calculé par IDAT avec le taux réel de couverture de fautes, nous avons décidé de choisir manuellement certains registres qui seront transformés en leurs équivalents scan. Les configurations obtenues ont été évaluées par IDAT et par les outils *DFTadvisor* et *FlexTest* de la société Mentor Graphics. La corrélation entre les valeurs de testabilité calculées et les valeurs des taux de couverture de fautes mesurées a été observée. L'évolution des attributs globaux après chaque étape de la modification du chemin de données est illustrée par le diagramme de la figure 1.11(b). La configuration 0 correspond au circuit originel avec le bloc de multiplexeurs. Les configurations de 1 à 4 correspondent aux chemins de données où les registres *REG5*, *REG2*, *REG4* et *REG1* sont successivement connectés à un chemin de scan.

Conformément à notre attente, les valeurs de taux de couverture de fautes sont améliorées avec un nombre élevé de registres connectés au chemin du scan élevé. La corrélation entre les valeurs de testabilité et les valeurs du taux de couverture de fautes est de 96,23%.

1.8.3 Résultats en mode BIST

Exemple branch-and-bound

Le déroulement de l'algorithme *branch-and-bound* utilisé est montré sur un exemple d'école comprenant quatre UFs. Les caractéristiques de cet exemple sont résumées en table 1.2.

UF	surface totale	nombre de bits d'entrée	nombre de bits de sortie
UF1	40%	16	8
UF2	20%	256	128
UF3	15%	16	8
UF4	15%	16	8

Table 1.2: Exemple utilisé pour la session branch-and-bound en mode BIST

Les résultats de l'algorithme *branch-and-bound* sont les suivants (table 1.3) : *UF1*, *UF3* et *UF4* sont connectées au TPG et au TRC. Le TPG et le TRC sont générés et ajoutés au chemin de données au moment du premier branchement demandé de l'UF. La configuration finale satisfait les demandes de l'utilisateur en testabilité et en surface additionnelle. Bien que *UF2* ait été premièrement sélectionnée par le processus d'analyse de la testabilité pour être testable, cette UF n'a pas été connectée au TPG et au TRC, parce que son grand nombre d'entrées et de sorties implique une surface trop importante du TPG et du TRC (la surface du TPG et du TRC dépend de la largeur de bits des UFs connectées). Une surface trop importante du TPG et du TRC induit une surface additionnelle non satisfaisante pour le circuit global.

Benchmark Diffeq

Les spécifications de l'utilisateur sont les suivantes : testabilité supérieure à 79%, surface additionnelle inférieure à 24% et quantité de noeuds non testables inférieure à 15%. Les attributs globaux sélectionnés sont illustrés à chaque étape de la modification du chemin de données par le diagramme de la figure 1.12.

Étape	Opération d'IDAT	T	AO	UF1	UF2	UF3	UF4
1	séparation du chemin de données	0%	5%				
2	UF1 connectée	41%	8%	•			
6	UF2 connectée	65%	19%	•	•		
4	UF3 connectée	82%	21%	•	•	•	
5	UF3 déconnectée	65%	19%	•	•		
6	UF4 connectée	84%	21%	•	•		•
7	UF4 déconnectée	65%	19%	•	•		
8	UF2 déconnectée	41%	8%	•			
9	UF3 connectée	58%	10%	•		•	
10	UF4 connectée	81%	12%	•		•	•

Table 1.3: Étapes de l'algorithme branch-and-bound en mode BIST (section 1.8.3)

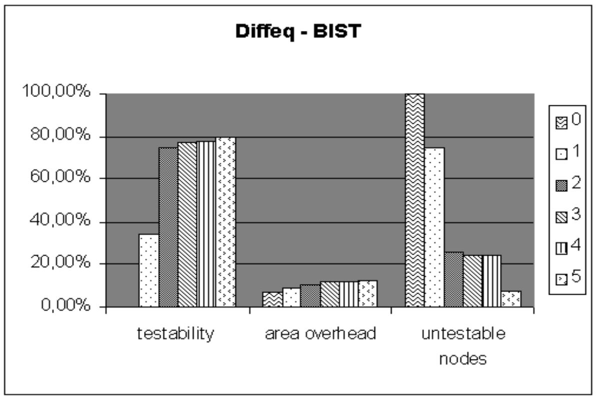


Figure 1.12: Sélection d'attributs pour différentes configurations du circuit Diffeq en mode BIST

Les modifications réalisées sont les suivantes :

1. *MUL1* sélectionné : l'entrée *N1* est connectée par le noeud *N10* au TPG, l'entrée *N9* est connectée par le noeud *N12* au TPG, la sortie *N10* est connectée directement au TRC.

2. *MUL2* sélectionné : l'entrée *N3* est déjà connectée par le noeud *N12* au TPG, l'entrée *N6* est connectée par le noeud *N15* au TPG, la sortie primaire *OUT* est connectée directement au TRC.

3. *SUB2* sélectionné : l'entrée primaire *IN2* est connectée directement au TPG, l'entrée *N16* est déjà connectée par le noeud *N15* au TPG, la sortie *N10* est connectée directement au TRC.

4. *SUB1* sélectionné : l'entrée *N11* est déjà connectée par le noeud *N10* au TPG, l'entrée *N2* est déjà connectée par le noeud *N12* au TPG, la sortie *N12* est connectée directement au TRC.

5. *ADD* sélectionné : l'entrée *N14* est déjà connectée par le noeud *N15* au TPG, l'entrée *N13* est connectée par l'entrée primaire *IN1* au TPG, la sortie *N15* est connectée directement au TRC.

Benchmark Tseng

Les spécifications de l'utilisateur sont les suivantes : testabilité supérieure à 73%, surface additionnelle inférieure à 35% et quantité de noeuds non testables inférieure à 45%. Les attributs globaux sélectionnés sont illustrés à chaque étape de la modification du chemin de données par le diagramme de la figure 1.9(b). Les modifications réalisées sont les suivantes :

1. *SUB* sélectionné : l'entrée *N2* est connectée par l'entrée primaire *V10* au TPG, l'entrée *V14* est déjà connectée par le noeud *N7* au TPG, la sortie *N3* est connectée directement au TRC.

2. *ADD2* sélectionné : l'entrée *N5* est déjà connectée par le noeud *N7* au TPG, l'entrée *N6* est connectée par l'entrée primaire *V6* au TPG, la sortie *N7* est connectée directement au TRC.

3. *ADD1* sélectionné : l'entrée *N10* et l'entrée *V14* sont déjà connectés par le noeud *N7* au TPG (il n'y pas de reconvergence directe, parce que le noeud *N7* est connecté par *REG5* à *V14* et par *REG5* et *REG4* à l'entrée *N10* avec une profondeur séquentielle différente), l'entrée *N6* est connectée par l'entrée primaire *V6* au TPG, la sortie *N7* est connectée directement au TRC.

1.8.4 Résultats récents avec la nouvelle version d'IDAT

Les dernières modifications d'IDAT, réalisées très récemment, nous ont permis de calculer l'évolution générale de l'attribut testabilité par rapport à l'attribut surface additionnelle. Les résultats présentés dans cette section montrent, pour les deux benchmarks Diffeq et Tseng, et trois méthodes de test, la meilleure testabilité qui peut être obtenue en l'absence de restriction sur la surface additionnelle.

Bien qu'il soit généralement impossible, quelle que soit la méthodologie du test, d'obtenir une testabilité de 100%, les résultats obtenus montrent comment la testabilité peut être améliorée avec une surface additionnelle relativement réduite, en utilisant un outil d'allocation automatique en vue de la testabilité comme IDAT.

Mode insertion de points de test

L'évolution de la testabilité par rapport à la surface additionnelle pour les benchmarks Tseng et Diffeq sont respectivement illustrées en figure 10.9 et 10.10. La particularité du mode d'insertion des points de test est la surface additionnelle totale très importante (78% pour Tseng et 152% pour Diffeq). Cela est dû à la surface additionnelle nécessaire pour les plots. En effet, en mode insertion de points de test, on ajoute plusieurs entrées/sorties primaires au circuit, or on sait que ces ajouts peuvent être rédhibitoires, plus en raison du coût du boîtier encapsulant le circuit que du coût en surface additionnelle.

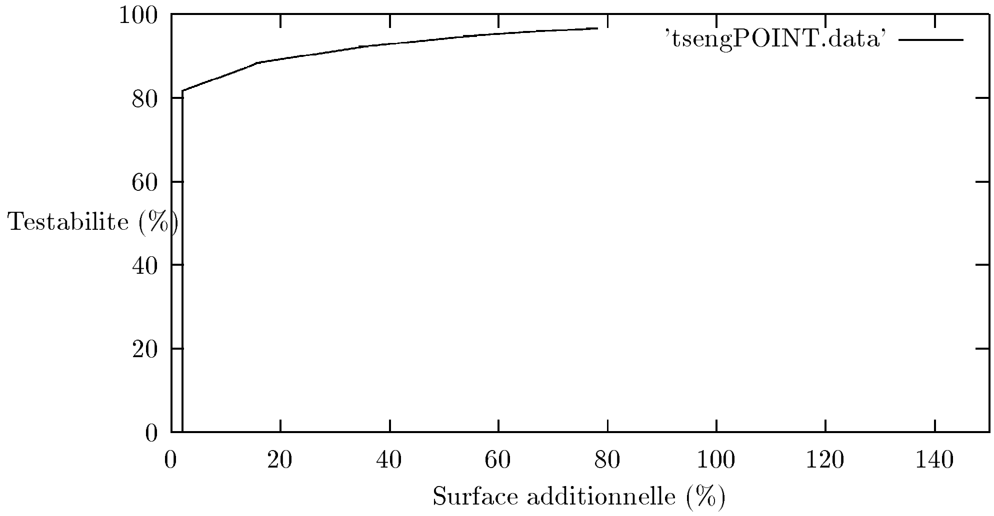


Figure 1.13: Tseng : évolution de la testabilité en fonction de la surface additionelle (mode insertion de points de test)

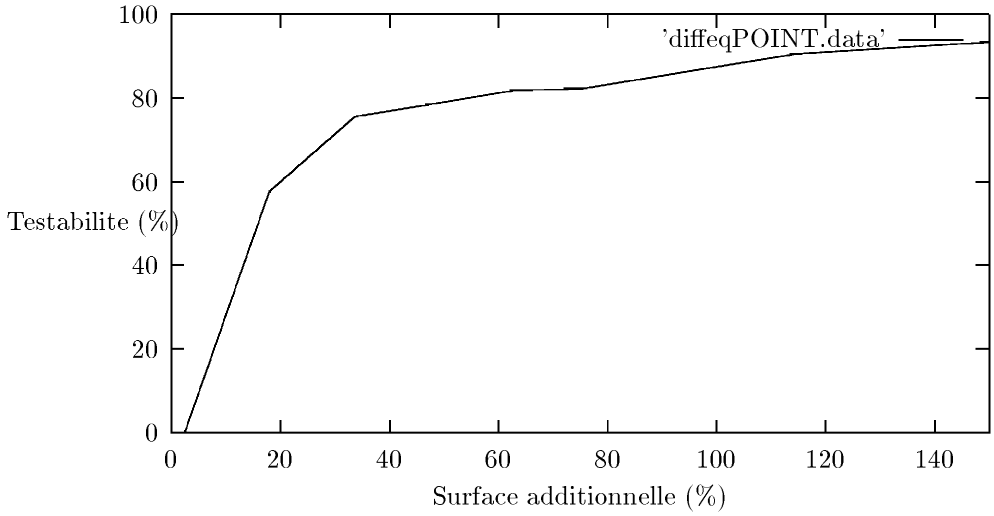


Figure 1.14: Diffeq : évolution de la testabilité en fonction de la surface additionelle (mode insertion de points de test)

Pour le circuit Tseng, la testabilité maximale est de 96.71%, pour une surface additionnelle de 78.24% plots inclus et de 5% si l'on exclut la surface des plots.

Pour le circuit Diffeq, la testabilité maximale, obtenue par transformation de presque chaque noeud en point de contrôle ou en point d'observation, atteint 93.71%. Dans ce cas, la surface additionnelle est de 152.68%, alors que la surface additionnelle du coeur du circuit (sans plots) est seulement de 2%.

Mode scan

L'évolution de la testabilité par rapport à la surface additionnelle en mode scan est illustrée pour Tseng et pour Diffeq respectivement en figure 10.11 et 10.12. Pour Tseng la testabilité maximale atteint 89.98%, pour une surface additionnelle maximale de 9.66%. Pour le circuit Diffeq, la testabilité maximale atteint 81.82%, pour une surface additionnelle maximale de 11.74%.

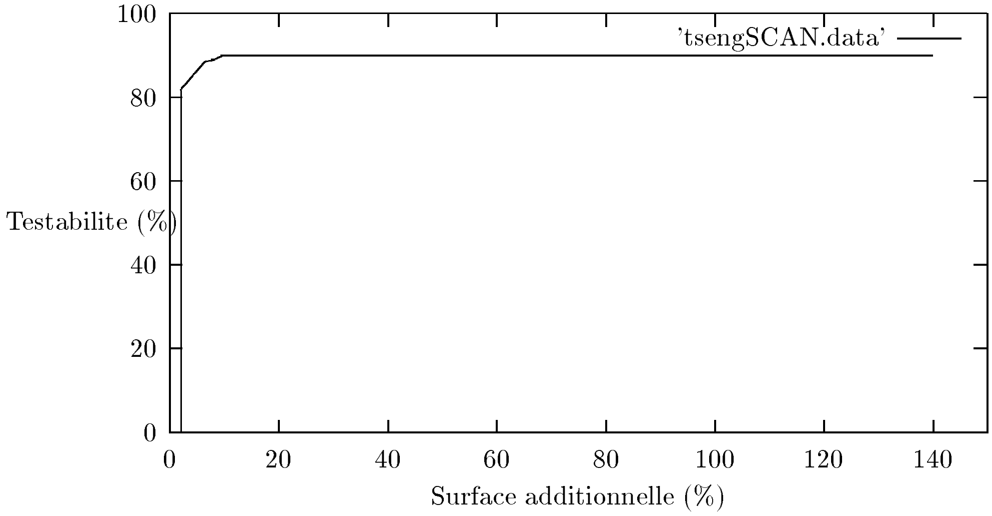


Figure 1.15: Tseng : évolution de la testabilité en fonction de la surface additionelle (mode scan)

Mode BIST

L'évolution de la testabilité par rapport à la surface additionnelle en mode BIST est illustrée pour Tseng et pour Diffeq respectivement en figure 10.13 et 10.14.

Pour Tseng la testabilité maximale atteint 73.35%, pour une surface additionnelle maximale de 9.36%. Pour le circuit Diffeq la testabilité maximale atteint 80.59%, pour une surface additionnelle maximale de 17.33%.

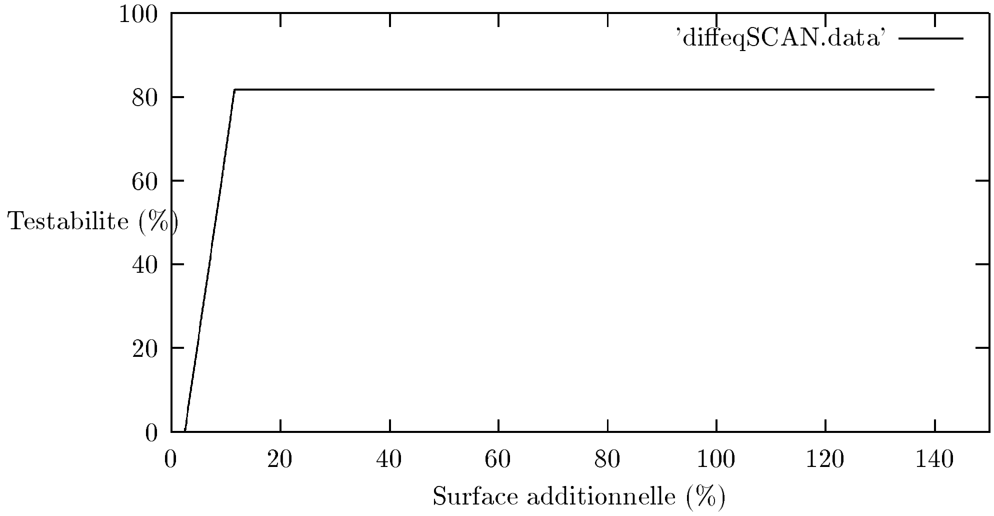


Figure 1.16: Diffeq : évolution de la testabilité en fonction de la surface additionelle (mode scan)

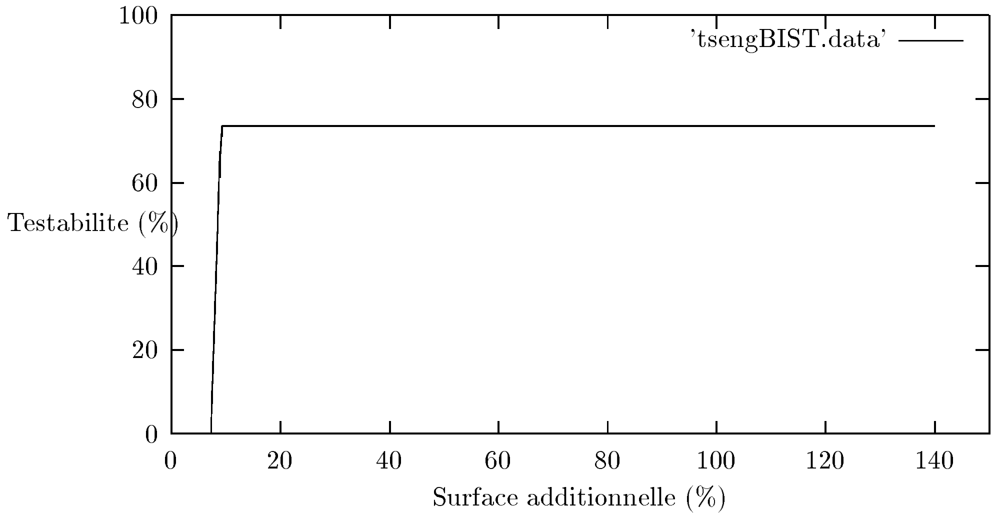


Figure 1.17: Tseng : évolution de la testabilité en fonction de la surface additionelle (mode BIST)

1.8.5 Remarques sur les résultats présentés

Bien que les circuits Diffeq et Tseng soient souvent utilisés comme benchmarks par différents auteurs, il était difficile de comparer nos résultats avec les résultats annoncés par exemple dans [AM94], [Gup94], [PR94], [WD96] et [Lee97], du point de vue du taux de couverture et du temps de génération de vecteurs de test. L'une des raisons

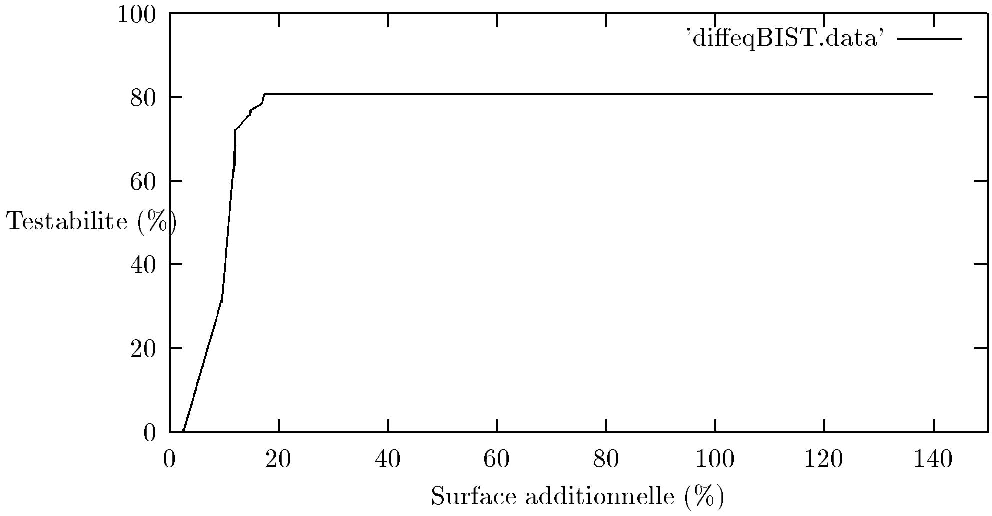


Figure 1.18: Diffeq : évolution de la testabilité en fonction de la surface additionelle (mode BIST)

est que nous ne disposons pas d'ATPG séquentiel sur notre site, mais la raison principale est que les résultats de notre système sont forcément dépendants d'une part de l'architecture RTL initialement générée par le système de synthèse et d'autre part de la bibliothèque de composants testables disponibles.

1.9 Conclusion

Ce document a décrit notre approche de l'allocation de ressources en vue de la testabilité, implémentée dans l'outil IDAT. L'idée principale d'IDAT est la réutilisation des blocs du circuit (composants) avec des structures du test déjà implémentées. L'allocation est guidée par un processus de compromis coût/qualité entre les spécifications de l'utilisateur et les ressources de testabilité disponibles. La réutilisation est réalisée en fin de processus de placement des composants dans le chemin de données, car, à cette étape, l'architecture du circuit est déjà connue et on peut mieux identifier les parties difficiles à tester qu'au niveau d'abstraction comportemental du circuit, tout en conservant tous les avantages du travail à un haut niveau d'abstraction.

Les résultats présentés montrent les capacités de l'outil IDAT. La prochaine étape est l'intégration d'IDAT dans le système de haut niveau UGH (*User-Guided High-Level Synthesis System*), développé au laboratoire LIP6.

Chapter 2

Introduction

This thesis deals with the problem of testable hardware synthesis, namely the cost/quality trade-off in the allocation for testability process of a circuit data-path considering the parallel creation of a circuit test controller.

The objective of hardware testing is to detect physical faults introduced during or after production of digital integrated circuit in order to ensure fault-free products. This objective can be easily achieved, when the circuit testability was taken into account already during the design phase rather than after the design. This is done by the transformation of the classical design process into *Design for Testability (DFT)* process, where the creation of hard-to-test architecture should be avoided. In order to increase the testability of a complex circuit, additional testability structures often need to be used. From this point of view, the drawback of most DFT methods is generally a considerable amount of area and delay overhead.

In this thesis a *new allocation for testability methodology* is defined. This methodology is based on the cost/quality trade-off between user requirements, selected testability methodology constraints and available resources in a library of components with implemented testability structures (called in the following *testable components*). The technique of reusing of investment for the testable components, which can significantly save time-to market, is called *DFT reuse*. The allocation for testability process is driven by defined cost/quality trade-off, where important source of data are the results of our *Register Transfer-Level (RTL) data-path testability analysis*.

A synthesis flow for Very Large Scale Integrated (VLSI) design is depicted in section 2.1 showing the position, where proposed allocation for testability methodology is applied. The drawback of existing high level synthesis for testability systems are shortly summarized in section 2.2, followed by objectives of presented approach in section 2.3. The contributions of this thesis are summarized in section 2.4. The organization of this thesis is presented in section 2.5.

2.1 Motivation

Nowadays a complex digital VLSI circuit is usually not designed manually, but a special set of design tools providing synthesis of such a circuit from its behavioral description to the final layout are used for this purpose (see figure 2.1).

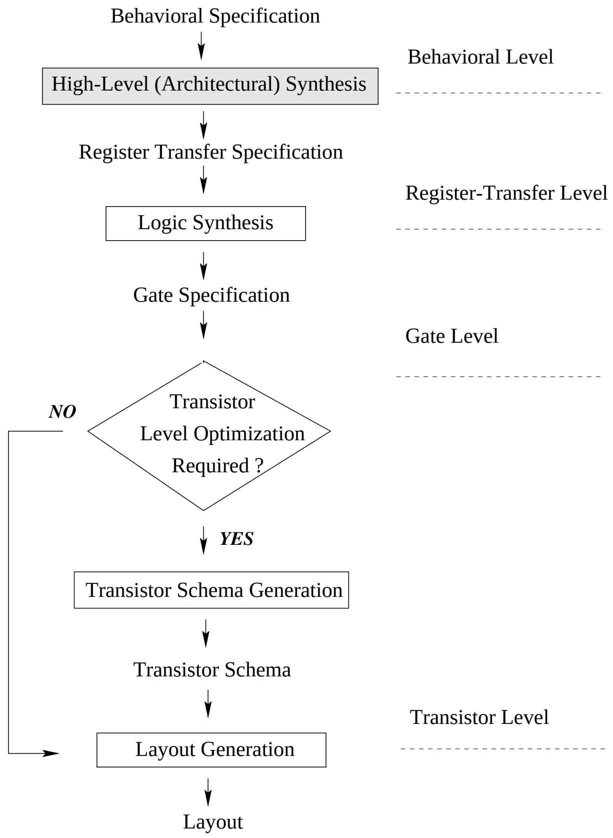


Figure 2.1: Synthesis flow for VLSI design.

Tools providing the synthesis from behavioral level of a circuit to its RTL are called *High-Level Synthesis (HLS)* tools. Most of the existing HLS tools are developed with the only purpose of design criteria (area overhead, maximal delay, etc.) optimization, without considering testability issues at the high-level. The resulting circuit architecture will then probably 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 already taken into account before RTL generation.

A popular targeted HLS phase to introduce testability is the allocation and binding step (see section 3.2), 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 on the final circuit architecture offer many possibilities to improve the overall circuit testability. Some of these approaches are overviewed in chapter 4.

2.2 Problem formulation

However, most of existing HL-SFT approaches suffer from the following main disadvantages:

- They do not make profit of already existing testable components apart from scan registers.
- They cannot be easily controlled by a set of user requirements in order to obtain the best cost/quality trade-off for designed testable circuit between these requirements and available means.
- They are focused either on synthesis for internal or on synthesis for external test. According to our knowledge only USC TEST system [Bre95] from the University of Southern California and IBM TestBench [Ibm00] tools offers both techniques at RTL.[1] The main disadvantage of all of these tools is that they do not support DFT reuse and the DFT synthesis process can not be easily controlled by user requirements.

2.3 Objectives of presented approach

The aim of HL-SFT approach, that we have defined and implemented in our *Interactive Design for Test reuse in Allocation for Testability* (IDAT) system, was to overcome the above mentioned drawbacks. The whole allocation for testability process is viewed as the automatic generation of testable architectures, starting from a *Rich Virtual Architecture* (RVA) (see chapter 5), and making profit of the *DFT reuse* concept presented in [MAAM96].

IDAT system is supposed to be used in conjunction with some of already existing HLS tool, with the program generating a TPG and a TRC called BIST Structure Generator (BSG)[2], and with the system creating the global scheduling and implementing the test controller for a final testable circuit [MAAM96].[3] These tools together forms

[1] Mentor Graphics tools are also able to cope with the insertion of both scan and BIST structure, but the introduction of scan chains is made as far as at the gate level.

[2] This program has been created by Assan Saar in the framework of his stay at LIP6.

[3] Such an interface, which connects IDAT with a HLS tool and with the TPG and TRC generator, and the system called Test Scheduler (TesS) creating the test controller is being developed by Laurent Ducerf-Bourbon from LIP6 in the framework of his Ph.D. project. More details can be found in [DBBM00].

our HL-SFT system. Our system attempts to obey the three main principles defined in [MAA95] for an efficient Synthesis For Testability (SFT) tool. It should be:

- *pragmatic in its objectives* - It mainly means that it fits a particular class of HLS environment, where the target architecture (bus-based or multiplexor-based) and the target test methodology are the constraints of the allocation for testability process[4]. The IDAT allocation for testability approach is based on the DFT reuse (reusing of existing components or circuit blocks with implemented testability structures - testable components). In the case of lack of some testable version in corresponding library an alternative testability improvement algorithm is applied according to the targeted test architecture.
- *systematic in its approaches* - The whole SFT process is not focused only to some part of the considered circuit namely circuit data-path or circuit test controller, but both of these parts are synthesized in parallel creating one *global testable architecture*. IDAT system, which modifies the circuit data-path, transfers all necessary information to the TesS tool for global scheduling, which creates corresponding test controller. As was already mentioned, the whole HL-SFT approach is restricted by constraints imposed by chosen testability methodology. For each testability methodology IDAT system allows the testability evaluation with respect to quality and cost requirements.
- *flexible in its use* - IDAT remains open to the user interventions. Of course user-friendliness was also the important criterion during IDAT development.

2.4 Contributions

The main contributions of this thesis are as follows:

- *Interactive Design for Test reuse concept* - Reusing of existing components or circuit blocks with implemented testability structures can save time-to-market of designed circuit with inserted test structures. Presented approach is, according to our knowledge, one of the first interactive algorithms based on DFT reuse ever presented. The other approach based on previous hardware investment is Beenker et al. approach [BvEG95]. The latest product of Synopsys - *DC Expert Plus* [Syn00] can reuse blocks with already implemented scan.
- *Consolidation of test point insertion, scan and BIST approaches to one system* - This contribution brings the user more freedom in the selection of test method, which improves cost/quality trade-off for designed circuit. Key circuit attributes like area overhead, number of test cycles and testability can be compared for a circuit architecture modified by any of above approaches.
- *Trade-off criteria* - The whole allocation for testability process is driven by the *cost/quality trade-off* between three groups of criteria:

[4]The support of bus-based architectures was implemented into our system by Benoît Miramond from LIP6 in the framework of his DEA project.

  - user requirements to the final circuit
  - limitations related to a target test methodology
  - existing testable components in the library.

- *Testability evaluation* - To determine the overall circuit testability and to guide the allocation of testability resources, the RTL data-path testability analysis algorithm was developed. This algorithm calculates the controllability and the observability metrics for each node, which complemented by the test complexity of circuit components connected to related nodes, compose controllability exigency and observability exigency metrics.

- *Trade-off algorithm* - The core of this thesis and the core of IDAT system, the trade-off algorithm, was defined. This algorithm, making profit from DFT reuse, is driven by the design and testability circuit attributes evaluated after each data-path modification. In order to obtain the best cost/quality relation for final synthesized circuit, the trade-off is consequently made at two levels: component level and circuit level.

2.5 Thesis organization

This thesis is organized as follows:

- In chapter 3 the basic definitions from the domain of testing and DFT are provided. The principles of the best known DFT techniques are reminded together with the brief summary of HLS phases. The basic rules of HL-SFT are also presented.

- The related works from the area of testability analysis are summarized in chapter 4 followed by the brief summary of already existing SFT approaches at all circuit abstraction levels.

- An overview of our allocation for testability concept based on the trade-off between user requirements, test methodology constraints and available testability resources is given in chapter 5. Used design representation is described, followed by the description of our principles to allocate related data-path for selected test methodology. A short overview of the cost/quality trade-off process is also made.

- The used testability analysis is detailed in chapter 6. New testability metrics called controllability exigency and observability exigency are defined based on controllability and observability of each internal node and on the influence of these metrics to overall circuit testability. Then our testability analysis algorithm, called propagation algorithm, is explained, followed by the comparison of our testability analysis results with the results obtained by the Linköping University testability analysis method.

- Chapter 7 is focused on the description of component and circuit attributes. Some of these attributes like area, delays, testability, etc. are used to describe both component and circuit characteristics while other ones are used at component (local) or at circuit (global) level only.

- Chapter 8 details the proposed trade-off algorithm for test point insertion, scan and BIST including definitions of the trade-off criteria and used trade-off functions.
- The complete user manual of IDAT tool is provided in chapter 9 including two typical system session examples.
- Obtained resulting global attributes for different circuits configurations of benchmark Tseng and Diffeq together with circuit Bert are summarized and analyzed in chapter 10. In the case of benchmark Tseng the values of defined testability attributes are compared with real values of fault coverage and number of test cycles obtained using Mentor Graphics tools.
- Conclusions and future directions of research are presented in chapter 11.
- Appendix A lists the abbreviations used in this thesis.
- Appendix B shows benchmark Diffeq in its input RVA format.

Chapter 3

Background

Nowadays, the VLSI technology allows companies to build large complex systems containing millions of transistors on a single chip. Such a complexity is often not limited only by used technology but also, during the design phase, by the availability of adequate synthesis automation tools - Computer-Aided Design (CAD) tools. A set of VLSI circuit synthesis CAD tools, allowing to achieve short time-to-market and low cost of designed circuit, is depicted in figure 2.1 in the previous chapter. In this thesis we will mainly focus on the most recent part of this design flow, a design technique allowing architectural synthesis from a behavioral description of a circuit - High-Level Synthesis (HLS).

The consequence of increasing circuits complexity is the decrease of the accessibility of internal circuit nodes, which brings problems when testing such a circuit. If a circuit is hardly testable, some defects can remain undetected. Consequences can be easily recognized: poor product quality, losses of customers and finally losses of profit. So, the question is not "Why do we need test ?", but rather "Which test methodology is suitable for our circuit to obtain the best cost/quality ratio ?".

This chapter provides the background to understand the allocation for testability method described in this thesis. The basic terminology concerning testing and existing Design for Testability (DFT) concepts are overviewed in section 3.1. Section 3.2 gives an introduction to HLS phases. Since HLS enables the designer to handle complex designs, testing needs to be considered with design needs already in this phase. Section 3.3 presents four HL-SFT design rules which, complemented with any DFT technique and applied during the HLS process, lead to low cost highly testable circuit.

3.1 Design for testability

The goal of DFT is to develop easily testable designs. As the name implies, DFT does this by uniting the disciplines of design (where the goal is to get working product to market as quickly as possible) and test (where the goal is to create high-quality, low cost test sets).

The testing process itself can be separated into two phases: *test generation phase* (preparation of test vectors) and *test application phase*. The cost of testing, which includes both of these phases, can be reduced by applying some DFT technique.

In this section the basic terminology from the area of testing is presented followed by the brief overview of existing DFT techniques. More comprehensive overviews of such techniques can be found for example in [AS88], in [RS89] or in [Wil90].

3.1.1 Basic definitions

There are many reasons why a circuit can function incorrectly. These reasons can be classified into three types: design faults, manufacturing defects and "wear-out" due to normal circuit operation. Moreover, a circuit may also become defective due to some external reasons (physical or thermal stress, radiation).

To identify the faulty circuit three type of test can be used:

- *Functional test* - functionality of the circuit is verified. This test is based on deterministically derived test sets.
- *Parametric test* - voltage levels, supply current (IDDQ test), output delay and other circuit parameters are verified.
- *Structural test* - circuit structure is verified. This is widely used for testing complex digital circuits.

The *fault model* is a hypothesis on how the circuit fails. The most commonly used fault model for digital circuits is the single stuck-at fault model. This model assumes that the design contains only one fault, which is either permanent 1 or permanent 0 on some circuit node. A test detects a fault in the circuit if the obtained value on the observed output port is different from the expected one. The main advantage of the single stuck-at fault model is that such a model can be easily implemented and it is technology independent. Other fault models are: stuck-open, stuck-short, bridging faults, floating gates, threshold drift, etc.

The *testability* is the property of a circuit that makes it easy (and sometimes possible) to test. To quantify the testability, the metrics *fault coverage* and *test time* are generally used. The fault coverage is defined as the percentage of all existing faults detected by the test. The fault coverage is dependent on the test time - see figure 3.1.

Usually when increasing the test time the fault coverage also increases. The test time is generally expressed as the number of necessary test vectors to obtain certain fault coverage. Supposing application of one test vector in each test cycle, the number of test vectors refers also to the number of test cycles (test time). The fault coverage not

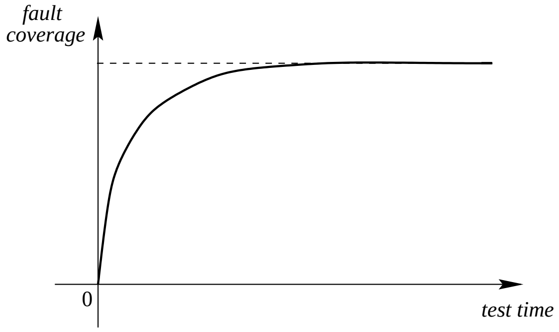


Figure 3.1: Typical dependence of fault coverage to test time.

only depends on the test time but also on used fault model and on the level of circuit abstraction. As stated in [GE92], 90% of RTL fault coverage may actually corresponds to 60% of gate level fault coverage.

The accurate evaluation of fault coverage requires the use of a *fault simulator*. A fault simulator checks the efficiency of applied test vectors to detect possible circuit structural faults - see figure 3.2. The metrics of these test vectors efficiency is the obtained fault coverage. Test vectors are generated using an *Automatic Test Patterns Generator (ATPG)*.

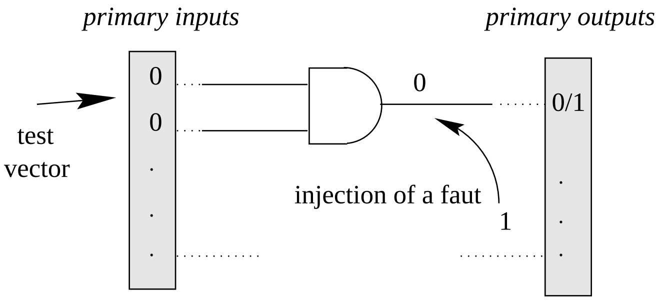


Figure 3.2: Idea of fault simulator: Test vector applied to primary inputs is able to detect injected fault. Fault is observable at primary outputs.

Analytic metrics representing the circuit testability are the controllability and observability of its nodes. The *controllability* of a circuit node is traditionally defined as the ability to control a logic value at this node from primary inputs, while the *observability* of a node is traditionally defined as the ability to observe a desired value at this node (set there by primary inputs) on primary outputs.

A circuit can be tested:

- *on-line* - Circuit test is performed in parallel with the normal circuit operation (concurrent testing). Methods for on-line testing are based both on replication of selected circuit components and on different types of data encoding.

- *off-line* - When an off-line test technique is supposed, there are two working modes for a circuit. Normal mode and test mode.

Because this thesis belongs to the domain of off-line test we will focus only to this area. The overview of existing off-line DFT techniques is given in the following subsection.

3.1.2 DFT techniques

Several off-line DFT techniques are used to improve the testability of a digital circuit. These techniques, which can be theoretically implemented at any structural level of circuit abstraction, are generally divided into three categories: ad-hoc techniques, structural techniques and built-in self test techniques.

Ad-hoc techniques

An ad-hoc technique employs additional hardware to improve controllability and observability of circuit nodes or divides a circuit into simpler sub-blocks (divide and conquer approach). Such a technique usually solves the testing problem for a particular circuit only. Because it is more or less heuristic, an ad-hoc technique is very dependent on the designed circuit and on the designer.

Test point insertion is the typical representative of an ad-hoc technique. It is a simple and straightforward approach to improve the controllability and observability of circuit hard-to-test nodes:

- To improve the controllability of one bit node a simple AND gate (to provide 0-controllability) resp. an OR gate (to provide 1-controllability) is added to this node. For multi-bit node, for the same purpose, a simple two inputs multiplexer can be used (see figure 3.3).
- To improve the observability of a node this node is directly made primary output.

Structural techniques

In contrast to the ad-hoc methods, the structured approaches aim to solve general testing problems with a systematic design methodology.

When an external ATPG to detect faults in the combinational blocks of a circuit is used, it is possible to reach almost 100% fault coverage. But this does not happen for sequential blocks, where it is sometimes very difficult to determine their internal states. The *scan path* is the structural technique which attempts to solve this problem. A sequential block transformed by a scan path technique can then be regarded as a collection of separated combinational blocks for which the test can be generated with an external ATPG tool.

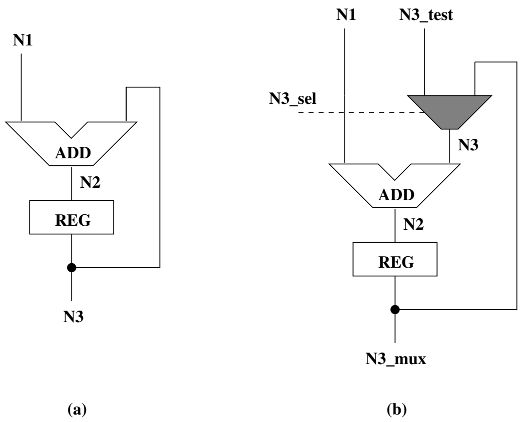


Figure 3.3: Idea of test point insertion: (a) a part of original circuit (node N3 uncontrollable); (b) the same circuit with inserted test (control) point to N3 node.

The principles of *scan design* are in the conversion of circuit registers into scan registers. In the circuit test mode the registers are interconnected into the scan chain, through which test data (test vectors and responses to them) are shifted (see figure 3.4). This modification creates in effect new control and observation points.

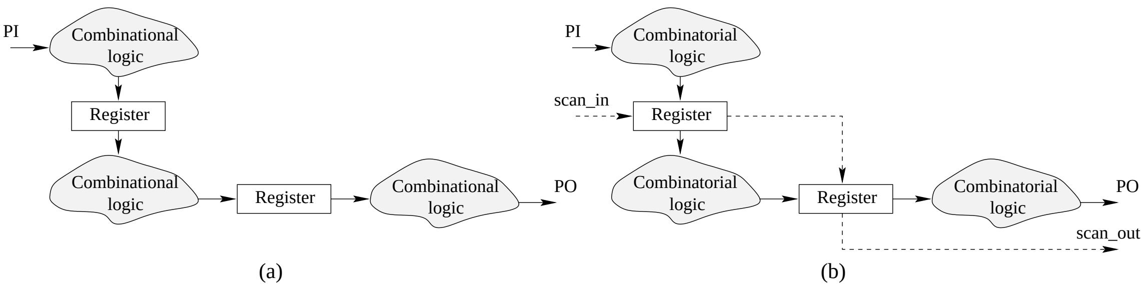


Figure 3.4: Idea of scan design: (a) generic system; (b) this system with implemented scan path.

In *full scan* design all the registers are converted to scan ones. The advantage of such a design is that a combinational ATPG providing high fault coverage is sufficient to generate the test. The main disadvantage is relatively high area and timing overhead. The compromise between these overheads and test quality can be the *partial scan* approach. In this approach, only the selected registers are converted to scan ones. However, in order to obtain high fault coverage, the remaining sequential parts impose the using of more complex sequential ATPG tool.

A number of scan techniques (LSSD, scan/set, random access scan, etc.) exists. Unfortunately, all of them have one common disadvantage - inability of final scan design to be tested at-speed. At-speed test often identifies more defective chips than the same test applied at a lower speed.

Built-in self test

Although structured DFT techniques reduce test generation costs they do not reduce the volume of test data or allow the circuit to be tested at-speed. Built-in Self Test (BIST) approach overcomes these drawbacks by embedding test circuitry into the design so that the design can test itself. The response of such a circuit to command "start test" is compacted only to "good" or "faulty" circuit and/or to a small signature.

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), arithmetic generator or ROM for deterministic test, and one or more internal *Test Result Checkers* (TRCs) often in the form of Multiple-Input Signature Registers (MISRs) sometimes also with the ROM, which store compacted correct responses. Such a BIST configuration is able to produce as responses "good" or "faulty" circuit. 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.

The *on-chip BIST controller* (test controller) determines the module test concurrency by controlling the data-path multiplexers (resp. buses) during BIST operation.

3.2 High-level synthesis

The basic task of HLS is the mapping of the behavioral description of a digital system into an RTL design consisting of a data-path and a control unit [Gaj94]. An example of such a circuit can be seen in figure 5.1 in chapter 5. The whole HLS process is driven by some user defined design constraints and generally consists of highly dependent, but usually separate, tasks: scheduling, allocation and binding. These steps are iterative. When the data-path synthesis is accomplished the circuit controller is generated.

This section presents used description formats and then gives a very brief overview of each HLS phase. For further studies in this area we recommend [Gaj94], [WC95] and [AM94].

3.2.1 Description formats

The HLS system's input description is usually an *algorithmic description* of design behavior not yet containing structural information such as component types and their interconnection. Such a description is close to human comprehension of a problem. Designer usually writes the input description in a special language, similar to classical programming languages, providing a set of variables and a set of operations for computing the variables current values. Conditional constructs and repeated iterations are also supported by such a language.

This behavioral description is then converted to the internal representation of the future data-path (*Data-Flow Graph* - DFG) and the representation of the future controller (*Finite-State Machine* - FSM), ore more often in a *Control and Data Flow Graph* (CDFG).

Final *structural description* (at RTL) of the data-path consists of functional units (adder, multipliers, etc.), storage units (registers and memories) and interconnection units (multiplexers and buses).

Both algorithmic and structural description formats are supported by two standard *hardware-description languages*: Verilog and VHDL.

An example of a simple data-path representations at different levels of abstraction is depicted in figure 3.5. The behavioral representation is first converted to DFG representation. HLS operations are then applied to the DFG transformed later into an RTL implementation.

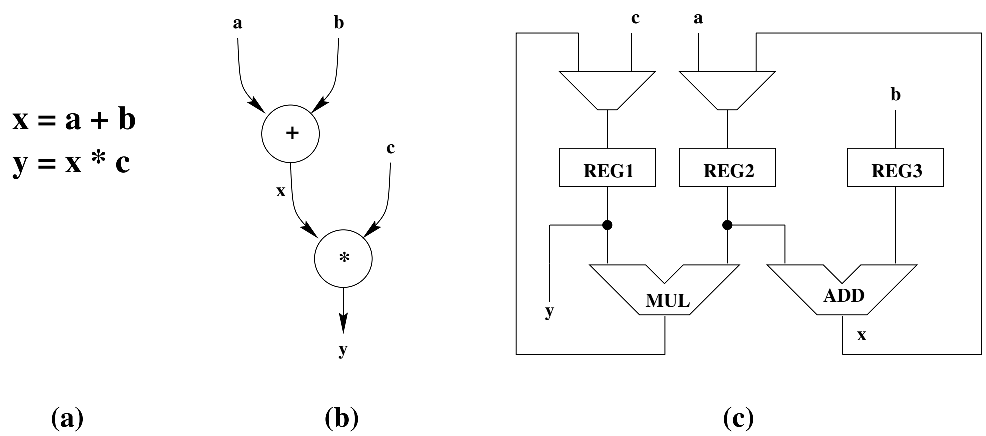


Figure 3.5: High-level synthesis example from [Lee97]: (a) behavioral description; (b) data-flow graph, (c) one possible structural implementation.

3.2.2 Scheduling

The scheduling determines the execution order of operations required in a behavioral description. Two main types of scheduling algorithm exist:

- *Resource-constrained scheduling* - allocation step is performed before scheduling. User has completely specified all the available resources and clock cycle length. The goal is to produce a design with the best possible performance under resource constraints.

- *Time-constrained scheduling* - allocation is performed after scheduling. List of resources is not yet available, but the performance is already specified. The goal is to produce a design with the lowest possible cost under time constraints.

3.2.3 Allocation and binding

The *allocation* determines the type and the quantity of resources used in chip architecture. The goal of allocation is to make the trade-off between the design cost and performance.

The *binding* consists of assigning the operation and memory accesses in each clock cycle to available hardware resources extracted from used library. A resource can be shared by different operations, which are never executed in the same clock cycle (they are time disjoint). The binding consists of three phases:

- storage binding - variables to registers and memory units are assigned
- Functional Units (FUs) binding - each operation in a control step is assigned to a FU.
- interconnection binding - multiplexors or busses for each data transfer among primary ports, storage units and functional units are assigned.

3.2.4 Synthesis of controller

Once the schedule and the data-path is made, it is necessary to synthesize a controller that will drive the data-path as required by the schedule. Generally, a controller is synthesized as an FSM. Such a controller is composed of two parts:

- sequencer - drives the sequence of states
- decoder - decodes the instructions related to requested state (activates corresponding control signals).

3.3 High-level synthesis for testability

As was already explained, the RTL architecture of a circuit is synthesized by one or more passing through allocation, binding and scheduling phase. During the classical HLS process the final RTL architecture is optimized according to the design attributes, usually circuit area and operating frequency. The HL-SFT adds to this optimization criteria also the circuit testability. A HL-SFT system aims to derive an inherently testable architecture at minimal area/delay overhead.

In order to improve the testability of a complex circuit, it is often necessary to add to this circuit some test structures. These test structures are implemented in the allocation phase of HL-SFT process. Classical allocation phase is in such a way converted into the *allocation for testability* phase. Because the allocation and scheduling processes are highly dependent, the allocation for testability should be complemented by the *scheduling for testability*. Created test schedule can then be implemented into the controller performing the schedule in the normal operating mode or a separated test controller can be created for this purpose.

Lee defines in [Lee97] four *HL-SFT rules*:

1. Wherever possible, allocate a register to at least one primary input or primary output.
2. Reduce the sequential depth from an input register to an output register.
3. Reduce the number of sequential loops.
4. Schedule the operations in order to allow previous three rules.

Sometimes the realization of above rules is not sufficient to obtain an easily testable circuit. In such a case classical DFT concepts, known from test synthesis at lower levels, are very useful. According to the selected test methodology such a DFT concept applied during HL-SFT together with above HL-SFT rules can produce very good testable circuit with only small area overhead.

Existing HL-SFT approaches are overviewed in chapter 4.

Chapter 4

Related Work

The core of our work is an allocation for testability algorithm driven among others by an RTL data-path testability analysis method. From this reason it is necessary to give the reader a comprehensive overview of already existing approaches in both of these areas.

This chapter is organized as follows: section 4.1 reviews the previous research on testability analysis methods and section 4.2 summarizes the previous research on synthesis for testability approaches. Both sections include a brief summary of the best known commercially available tools.

4.1 Testability analysis

Testability analysis tools have been developed as another mean for evaluating the testability of a design without doing time consuming fault simulation. These tools usually calculate some testability metrics for each circuit node or register. Obtained values are used to guide the testability improvement modifications of the circuit. The typical analytic testability metrics for a node are controllability and observability. The enhancement of nodes controllability and observability improves overall circuit testability and generally leads to a reduction of the number of test vectors.

The research in the area of testability analysis started in early seventies. Ten years later the first approaches of RTL testability analysis were introduced followed by methods of behavioral testability analysis.

4.1.1 Gate level

The earliest work in the area of testability analysis is probably Rutman method [Rut72] based on the calculation of 1(0) - controllability and observability metrics. Rutman's work was later extended by Breuer [Bre78].

These approaches were the sources of inspiration for the most popular gate-level testability analysis algorithm SCOAP [Gol80] developed by Goldstein. SCOAP computes six metrics for every node in the circuit:

- *0 - combinational controllability*, *1 - combinational controllability* and *combinational observability*, reflecting the number of other nodes that must be set to either control or observe a particular node
- *0 - sequential controllability*, *1 - sequential controllability* and *sequential observability*, reflecting the number of clock cycles required to control or observe a node.

These six metrics, can be found also in SCOAP successor CAMELOT [BMR81].

More references from the area of testability analysis research at the gate level can be found for example in [AS88] or in [ABF90].

4.1.2 Register-transfer level

However, it is quite difficult to use SCOAP testability metrics concept at higher level of abstraction, namely RTL. At RTL the *multi-bit signals* interconnect relatively complex components with generally unknown gate level implementation.

Linköping university

Gu et al. methodology [GKP94] of controllability and observability calculation for circuit nodes at RTL is used to guide DFT techniques based on partial scan and test point insertion. Their definition of testability metrics, which was probably inspired by Chen et al. approach [CM89], influences also testability definition used by our allocation for testability method described in this thesis. Like Chen et al., Gu et al. have defined both controllability and observability by two factors:

- *Combinational factors:* CC (combinational controllability) and CO (combinational observability), which are ranked from 0 to 1, where 1 represents the best controllability resp. observability, reflecting transparency problems.
- *Sequential factors:* SC (sequential controllability) and SO (sequential observability), measuring the sequential complexity of a circuit, are natural numbers representing the estimated number of steps that a combinational test generation algorithm needs to be repeatedly applied to a sequential circuit. The SC of a node represents also the number of clock cycles to control this node. The SCs of all primary inputs are equal to 0. In the same way the SO of a node represents also the number of clock cycles to observe this node. The SOs of all primary outputs are equal to 0.

Detailed calculation of these factors for combinational and sequential circuits together with *controllability and observability transfer factors* can be found in [Gu96]. Gu et al. apply their testability analysis to the Extended Timed Petri Net (ETPN) representation of a design, which is an intermediate design description in the form of graph consisting of the data-path and the control part. Gu et al. work was later extended in [YP98] by introducing the *global testability indicator* denoting the global testability of entire circuit.

Other approaches

The program for RTL testability analysis TMEAS is presented in [Gra79]. TMEAS calculates controllability and observability metrics for all nodes and *testability transfer coefficients* for all components at RTL. To calculate controllability and observability of internal nodes, the system of linear equations is used. Results from TMEAS analysis are used to guide test point insertion.

RTL testability analysis based on the theory of information is proposed in [TA89]. *Information transition coefficient* (representing controllability metrics) and *mutual information transfer coefficient* (core of the observability metrics) are assigned to each circuit component and node using either sampling technique based on circuit functional simulation or composition technique.

In [CP95] the testability analysis technique for BIST at RTL is presented based on the metrics of *register randomness* (related to register output controllability) and *register transparency* (corresponding to register input observability). This technique is used to guide the process of partitioning the circuit to combinational sub-blocks driven by corresponding registers transformed either into TPG or into TRC.

In [KRH00] they propose the formal approach of the RTL testability analysis based on the identity transfer path approach and the classification of registers in the circuit according to their possible role during the test application. This testability analysis is consequently used to select the set of registers to be scanned.

4.1.3 Behavioral level

The approach to the hierarchical testability analysis of the data-path from behavioral level to RTL was published by Pires et al. in [FPR97] and in [Pir98]. At RTL this approach focuses on reconvergence and transparency problems. It is, according to our knowledge, after Steensma et al. [SGCM93] testability analysis (based on symbolic controllability and observability descriptions), the second existing approach taking into account the reconvergence of RTL component inputs. The testability analysis results are used successively to drive registers allocation, interconnections generation and finally to perform the test plan generation.

RTL testability analysis described in [CP95] was extended at behavioral level in [LPB97]. Described tool, based on BIST methodology, is able to operate in tandem with a classical synthesis tool (Synopsys Behavioral Compiler and structural Design Compiler) through common VHDL interface. The entropy based metrics *variable randomness*, *variable transparency* and *module sensitivity* are then used to guide BIST insertion process.

BETA tool [CWS91] uses quantitative approach, which classify variables to completely controllable, non-completely controllable, observable and non-completely observable according to the ability to control resp. observe bits in related variable. This classification is used to guide the test point insertion process.

In [LP97] the behavioral testability metrics *variable range* and *statement hardness* are introduced. These metrics then compose *testability* metrics for each variable (including state variables). The variable range analysis is based on the assumption that the reduced range of a variable at a line in the behavioral description tends to form hard-to-test part. The statement hardness captures the testability hardness, which different statements lead to when implemented in hardware. The statement hardness (SH) at line l is defined like:

$$SH(l) = \sum_{op \in OP} area(op, w) \times control(l),$$

where OP is the set of operations at line l in the behavioral specification. Function $control(l)$ denotes the number of statements controlled by line l and function $area(op, w)$ gives the area estimation for the operation op with word length w based on available library. This testability analysis technique can be applied also to a scan design. The disadvantage of this technique can be seen in the assumption that each variable is implemented as a dedicated register. This assumption disables future optimization of the registers number.

4.2 Synthesis for testability

The goal of synthesis for testability is to create easily testable circuit while meeting area and delay constraints. To achieve this goal for a more complex circuit it is often necessary to include certain test structures into the synthesized circuit. This process is called *test synthesis*. Obeying certain test synthesis constraints - DFT rules, summarized in [Cox95] and [BCM97], it is possible to obtain fully testable circuit with minimal area and delay overhead.

Generally, the test synthesis consists of four steps defined in [Ait95] and valid at each level of circuit abstraction:

- test methodology selection
- test structure insertion
- verification of circuit functionality
- test program preparation.

4.2.1 Gate level

During a long time only gate level test synthesis tools were available. In addition to the number of test point insertion ad-hoc methodologies, a huge number of university approaches concerning full or partial scan at gate level exists. The scan approaches differ in the way that shift registers are designed, organized and accessed. Because gate level DFT techniques are already well established in the designers community, because of many commercially available tools, in the following paragraphs we refer only to the best known gate level DFT commercially available tools. A comprehensive overview of others gate level DFT approaches can be found for example in [AS88], [RS89] or in [Wil90].

Synopsys *DC Expert Plus* [Syn00] supports both full and partial hierarchical scan methodologies including fault coverage evaluation obtained by using of either combinational or sequential ATPG (provided by *TestGen*). The choice of optimal set of scan registers in partial scan is based on performance, testability and area constraints. The insertion of a scan chain is possible also at RTL.

DFT Advisor of Mentor Graphics [Men99], allows in the same way as Synopsys *DC Expert Plus*, both full and partial scan. The modified design can be evaluated by corresponding ATPG provided for partial scan by *FlexTest* tool and for full scan by *FastScan* tool.

TestBench's DFT Synthesis tool of IBM [Ibm00] is able to work not only at gate level but also at RTL. This tool provides the implementation of both full and partial scan. The evaluation is made by *TestBench's Test Generation and Simulation* tool.

4.2.2 Register-transfer level

RTL test synthesis tools produce test structures to merge with the rest of the design before logic synthesis. Typically RTL test synthesis tool produces BIST circuit, but of course scan based techniques or test point insertion techniques are still applicable at this level of circuit abstraction.

University approaches

Most of partial scan approaches exist for data-paths at gate level. More recent approaches attempts to introduce the selection of scan chains at RTL. In [FPRV98] the partial scan methodology consists in preparing a set of registers from which scan flip-flops will be selected at gate level by a commercial scan path selection tool. A similar approach, based on pre-selection of registers at RTL, was published also in [FS96]. In this paper, in contrast to [FPRV98], the testability of the whole circuit (data-path and controller) is taken into account.

A method for RTL synthesis with testability trade-offs for BIST was presented in

[CP91]. In this approach, they interactively remove or inject BIST registers (in the form of LFSR) to best satisfy user requirements to circuit testability minimizing area and performance degradations.

In [BFR98] 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 was published in [BFR99]. 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.

ADEPT system [CLP94] takes as input VHDL-RTL description and transforms it into *execution flow graph* model to calculate testability measures called *Testability Sequence Range* (TSR). The values of TSR for each node are used to guide partial scan registers selection.

In [GRJ95] a test point insertion methodology for hierarchical testability of an RTL design is presented. A small number of multiplexers are selectively inserted to make the circuit hierarchically testable. Since no scan is used, the test set can be applied at-speed.

Commercially available tools

Synopsys *DC Expert Plus* [Syn00] supposing both full and partial scan and described in the previous subsection can be also used at RTL.

LBIST Architect of Mentor Graphics [Men99] automatically inserts scan-based BIST structures into circuit making the trade-off between design requirements and resulting fault coverage. Related *MBIST Architect* is specialized to memory BIST.

LogicVision [Log99] proposes similar tools to *LBIST Architect* and *MBIST Architect* called *logicBIST* and *memBIST*. *LogicBIST* tool supports not only internal test but also external one (insertion of multiple scan chains). The BIST capability is based on standard scan design techniques. Conversion to BIST is based on adding a pseudo-random TPG to the scan chain inputs and a MISR to the scan chain outputs. In spite of using scan technique, this approach allows to test a circuit at-speed.

TestBench's Design-for-Test Synthesis tool of IBM [Ibm00] performs insertion of both BIST and scan structures. The evaluation of modified design is made by *TestBench's Test Generation* tool.

Disadvantages of already existing RTL SFT tools

The main drawbacks of already existing SFT tools working at RTL are:

- They do not make profit of already existing testable components apart from scan registers, while there exists now tools to generate scanable and BISTable versions of FUs, including memory blocks.

- They cannot be easily controlled by a set of user requirements in order to obtain the best cost/quality trade-off for designed testable circuit between these requirements and available means.
- They are generally focused either on synthesis for internal or on synthesis for external test.
- As is also stated in [BFR00], most of existing BIST solutions suppose that all functional units are tested by the same type of TPG and TRC.

4.2.3 Behavioral level

A popular targeted HLS phase in many HL-SFT systems is the *register binding phase*. In this phase each variable from the behavioral description is mapped to certain register. The large number of possible solutions and the strong impact of these solutions on the final structure offer many possibilities to improve final circuit testability. An exhaustive overview of HL-SFT systems can be found in [Lee97], in [AM94], in [Gup94], in [PR94] and in [WD96]. According to our knowledge there is not yet a commercially available HL-SFT tool.

Synthesis for external testing

PHITS-PS [LCP93] generates the implementation with either no scan or partial scan, where the number of allowed scan registers is an input parameter. Testability issues are addressed mainly during the register binding phase. This register binding algorithm attempts to reduce sequential depth between input and output registers, reduce the number of cycles and increase the number of input and output registers.

BETS [DPR93] system assumes *dedicated register file hardware model*. It breaks Control-Data Flow Graph (CDFG) loops with minimal number of scan registers. Then it performs simultaneous scheduling and allocation driven by a cost function.

TBINET [MJS94] performs simultaneous module and register allocation by the mapping of the allocation process onto a sequence of *minimum cost network flow problems*. The network simplex method finds the optimal integer flow solution.

The system proposed by Pires et al. [FPRV98] [Pir98] uses a HLS methodology for partial scan modification process driven by a hierarchical testability analysis starting from scheduled DFG.

The Kim et al. system [KCL94] performs a simultaneous scheduling and binding/allocation for testability algorithm without assuming particular test strategy. Defined testability measures are based on the controllability and the observability of registers, the *sequential depth* and the *number of sequential loops*. The goal of this algorithm is to maximize the number of input and output registers, minimize the sequential depth between input and output registers and minimize the number of self-adjacent registers.

The AMBIANT system [VTAA93] starts with testability improving modifications of the input behavioral description followed by HLS, where the system attempts to share functional and test logic to obtain minimal area overhead.

The Chen-Karnik-Saab system [CKS94] uses a testability analyzer to analyze the circuit description in the form of a CDFG. Then the system classifies all variables into groups with different testability, selects test points and inserts test structures.

Scan insertion at behavioral level is offered also in [NAR00]. The selection of memory elements to be included into scan chain is performed by testability analysis, which combines structural based, FSM based and textual (presented in [LP97]) based approaches.

Synthesis for internal testing

SYNTEST ([PCH91] and [HPCN92]) is a user interactive system with a graphical interface. SYNTEST consists of several algorithmic synthesis tools for scheduling, testable allocation, and optimum test points selection (to insert BIST registers). In the binding phase of proposed HL-SFT algorithm each operation is first mapped to the testable functional block and then all compatible blocks are merged to minimize area and delay overhead with the consideration of testing cost. The allocation process is guided by the testability metrics called *randomness* and *transparency*. Cost functions are used to measure area gains of blocks sharing. Finally the post-synthesis trade-off and successive modifications at RTL are performed.

In RALLOC [Avr91] the registers binding procedure consists of analyzing scheduled DFG to determine, which variable must be stored in BILBO register. The number of self-adjacent registers is minimized by imposing testability constraints during allocation in the form of *register conflict graph*. The edges in this graph are: schedule conflict edges, testability conflict edges and cost constraint edges. A *graph coloring heuristic* is applied to achieve lowest-cost testable implementation of designed circuit. To evaluate the resulting data-path, a cost function is used.

The objective of SYNCBIST [HO94] is to synthesize a BIST testable data-path with high test concurrency. The system includes three major synthesis algorithms: behavioral restructuring, HLS for BIST data-path and finally structural synthesis determining the subset of registers to be configured as TPGs and TRCs. To achieve high test concurrency, the test paths in the circuit are identified and scheduled. Possible data-path conflicts are classified into hard and soft conflicts. During HLS, the scheduling and binding algorithms attempt to minimize the number of conflicts in the generated data-path.

A method for trade-off between the area overhead and test time for BIST was presented in [BFR00]. They take into the account the existence of a test library storing for each test resource its characteristics (functional model, number of test vectors, corresponding fault coverage and area) and define some rules to allocate these resources. The initial behavioral description is first scheduled using timing and area constraints creating new test behavioral description composed of so called behavioral patterns. Then actual components (including test resources) are determined from behavioral patterns and allocated. Selected behavioral patterns serves in test mode as test resources.

Chapter 5

Allocation for Testability - Overview

When the High-Level Synthesis for Testability (HL-SFT) process is viewed as the automatic generation of testable architectures ([MAA95] and [MAAM96]), it involves on the one hand the choice of the suitable test structures to be allocated according to a target test methodology and to a set of constraints dealing with cost and performances; on the other hand, the HL-SFT process involves a global test scheduling.

This thesis deals with a new allocation for testability techniques which optimize, from the testability point of view, cost/quality trade-off ratio during circuit synthesis. The described techniques to modify existing data-path architecture in order to insert test points, scan chains or BIST structure, are already implemented in the data-path allocation for testability tool. This tool takes into account the availability of a test controller, synthesized using a complementary tool. The whole system is intended to work in conjunction with most of already existing HLS tools, performing the following main steps:

1. Extraction of RTL data-path together with components' design and test attributes from the allocation step result of connected HLS system.

2. Extraction of testable versions of allocated components from a library.

3. Calculation of design and testability attributes for original circuit, where the data-path is already separated from the original circuit controller by a block of multiplexers. Some of the global testability attributes, like the number of test cycles, are estimated by the tool, preparing the global schedule and creating the test controller. This tool runs in parallel with our data-path allocation for testability tool.

4. Cost/quality trade-off process among user requirements, test methodology constrains and available means in the library of testable components. This process performs modification of the circuit data-path 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. Returning of the modified RTL data-path to the connected HLS system in order to perform fine scheduling at subsequent synthesis step.

These phases are the backbone of our *Interactive DFT reuse in Allocation for Testability* system IDAT.

This chapter is organized as follows: The used design representations are described in section 5.1. Section 5.2 focuses on the implementation of one important step of our allocation for testability method - allocation of new multiplexers in order to make testing algorithm independent from the original circuit controller. Then design architectures for all supported test methodologies are presented in section 5.3 starting by the description of test point insertion approach and followed by the description of allocation for scan approach and the description of allocation for BIST approach. Finally, in section 5.4, the presented trade-off process is overviewed. More details of this process for each above mentioned test methodologies are provided in chapter 8.

5.1 Design representation

The first step of the proposed allocation for testability process is the extraction of RTL architecture from a connected HLS system. This RTL architecture is, after the extraction from the HLS system, enriched by selected components' design and testability attributes quantified using generic units. These attributes are described in details in chapter 7. Obtained *Rich Virtual Architecture* (RVA) description of synthesized circuit enters to IDAT system.

RVA description is an internal language of IDAT tool. The structure of this language is like VHDL with the syntax resembling to LISP or SKILL language. Such a format is then easily transformable to any standard hardware description language format like VHDL or Verilog. A translator from and to VHDL is implemented in IDAT[1]. The RVA description corresponding to the RTL architecture depicted in figure 5.1 can be found in appendix B.

5.2 Disabling of original controller

In our allocation for testability method, in order to not to care about the state dependencies in original circuit controller, the testing algorithm of the data-path is made independent from the original circuit controller. This is made by the connection of each:

- *select* input of all allocated multiplexers
- *RESET/SET* input of all sequential components
- *write-enable control* input of all registers

[1]The interface from Alliance VLSI system [Ali99] subset of VHDL to RVA was made by Laurent Ducerf-Bourbon from LIP6 in the framework of his Ph.D. thesis.

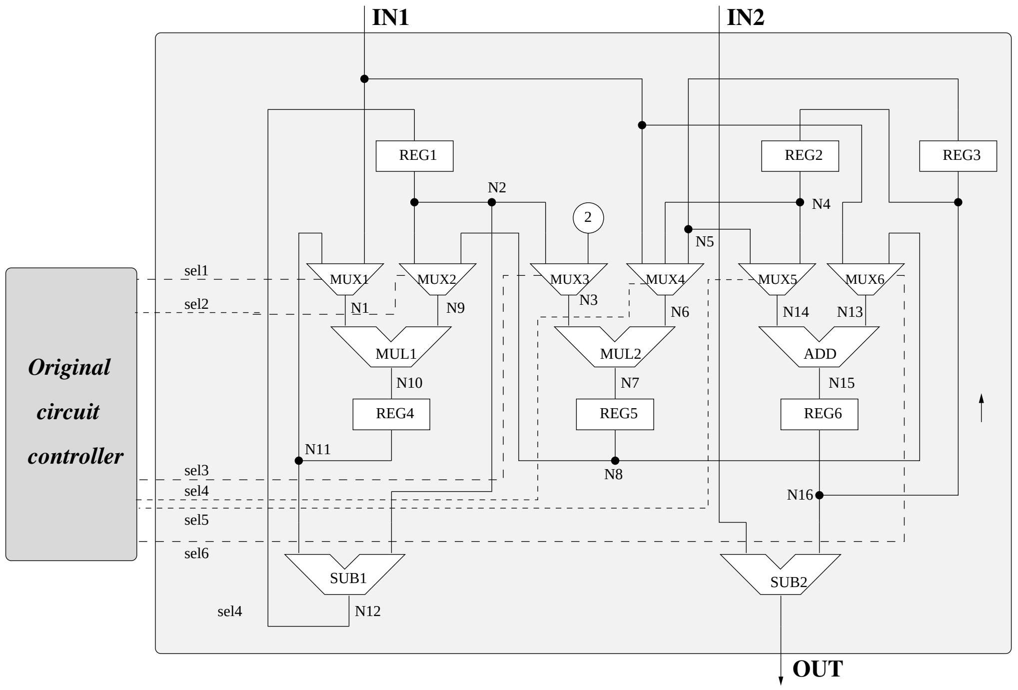


Figure 5.1: RTL architecture of circuit Diffeq.

through the new multiplexers to the added test controller (see figure 5.2). These multiplexers provide switching between normal and test mode.

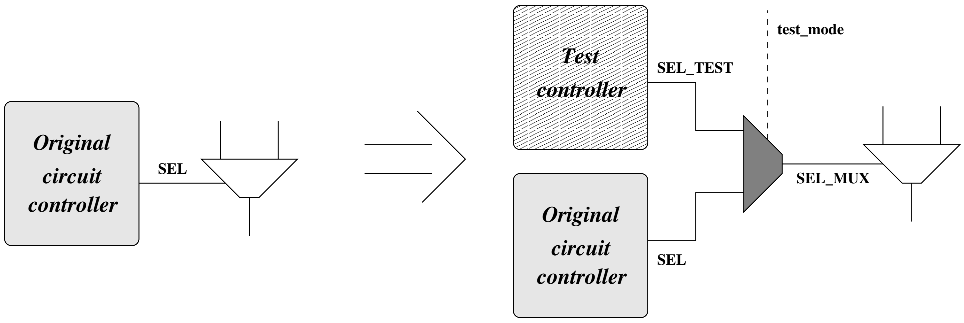


Figure 5.2: Disabling of original controller and connection of the new test controller.

The select inputs and all other control signals of FUs are in internal test mode connected to TPG and in the external test mode to the test controller. The BIST control signals of a BIST FU are connected also to the test controller.

Beyond simple and quick testability analysis algorithm, which is already implemented in IDAT tool, the expected advantage of such concept should be both saving of external sequential Automated TPG (ATPG) algorithm complexity and quick and straightforward test reaching high fault coverage.

The disadvantage of this approach can be seen in the fact that many new multiplexers are automatically added to the circuit without any initial trade-off. Such a trade-off, called *shared selection inputs mode* and described in more details in chapter 6, is planned for the future version of IDAT.

5.3 Testable architectures

5.3.1 Architecture with inserted test points

Test point insertion is a simple and straightforward DFT technique to improve the controllability and observability of selected signals. Test points are inserted in the following way:

- To improve the controllability of a node, it is connected as one of the two inputs of a newly allocated multiplexer. The second multiplexer input is made new primary (test) input (see figure 3.3).
- To improve the observability of a node, this node is directly made new primary (test) output.

The data-path is separated from the original circuit controller as described in section 5.2. The example of RTL circuit with inserted test points is depicted in figure 5.3. In this figure Nodes N9_TEST and N16_TEST are new primary inputs allowing to directly control nodes N9 and N16. Nodes N10 and N7 were directly made primary outputs in order to be better observable.

More details on the test controller for the data-path with inserted test points can be found in [DBBM00].

5.3.2 Scan design architecture

The main characteristics of our scan approach are:

- Data-path separated from the original circuit controller as described in section 5.2.
- Scan transformation process is not limited to registers, but is extended to all sequential FUs existing in scan equivalent (DFT reuse concept).
- Multiple scan chain is supported.

The example of a scan circuit architecture obtained by applying our scan technique to the circuit of figure 5.1 is depicted in figure 5.4. This architecture contains one scan chain starting from external TPG, passing through the test controller, selected scan registers and ending in external TRC. More details on the test controller for the scan data-path can be found in [DBBM00] and in [DB99].

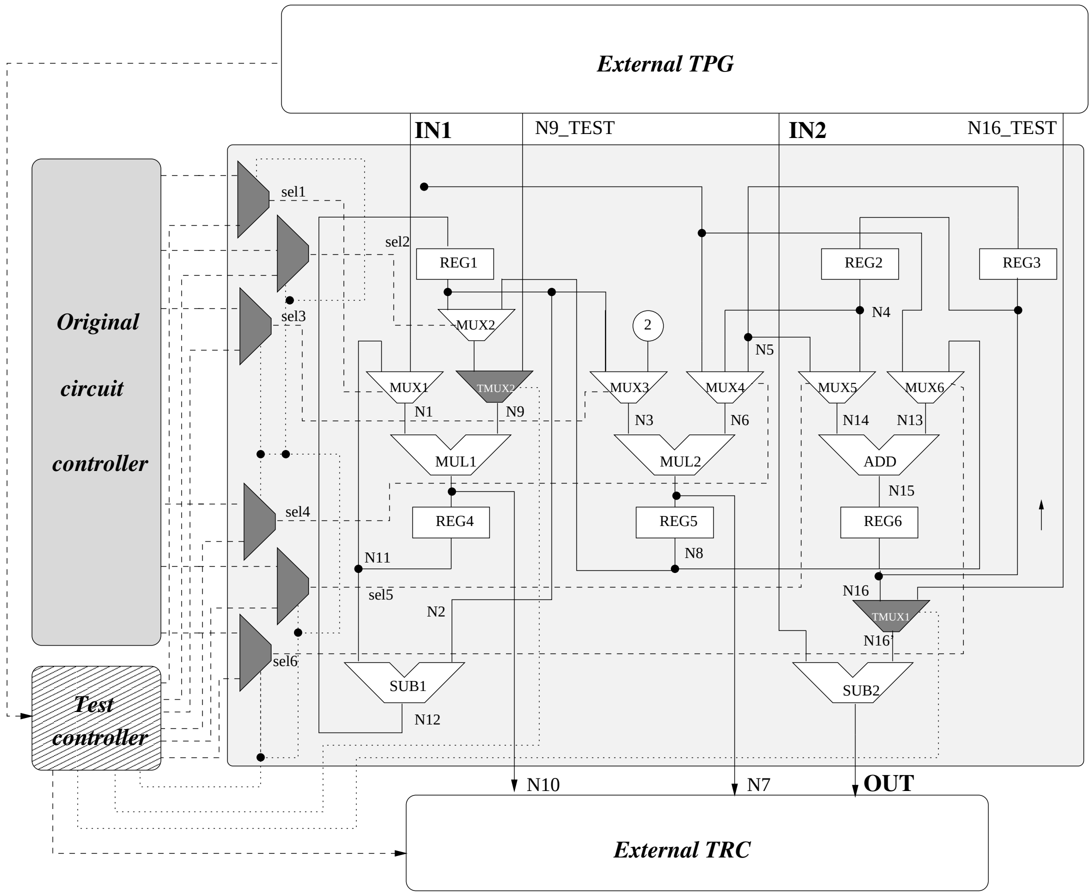


Figure 5.3: RTL architecture of circuit Diffeq with inserted test points.

5.3.3 BIST design architecture

Main features of our BIST approach are as follows:

- Data-path separated from the original circuit controller as described in section 5.2.
- 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

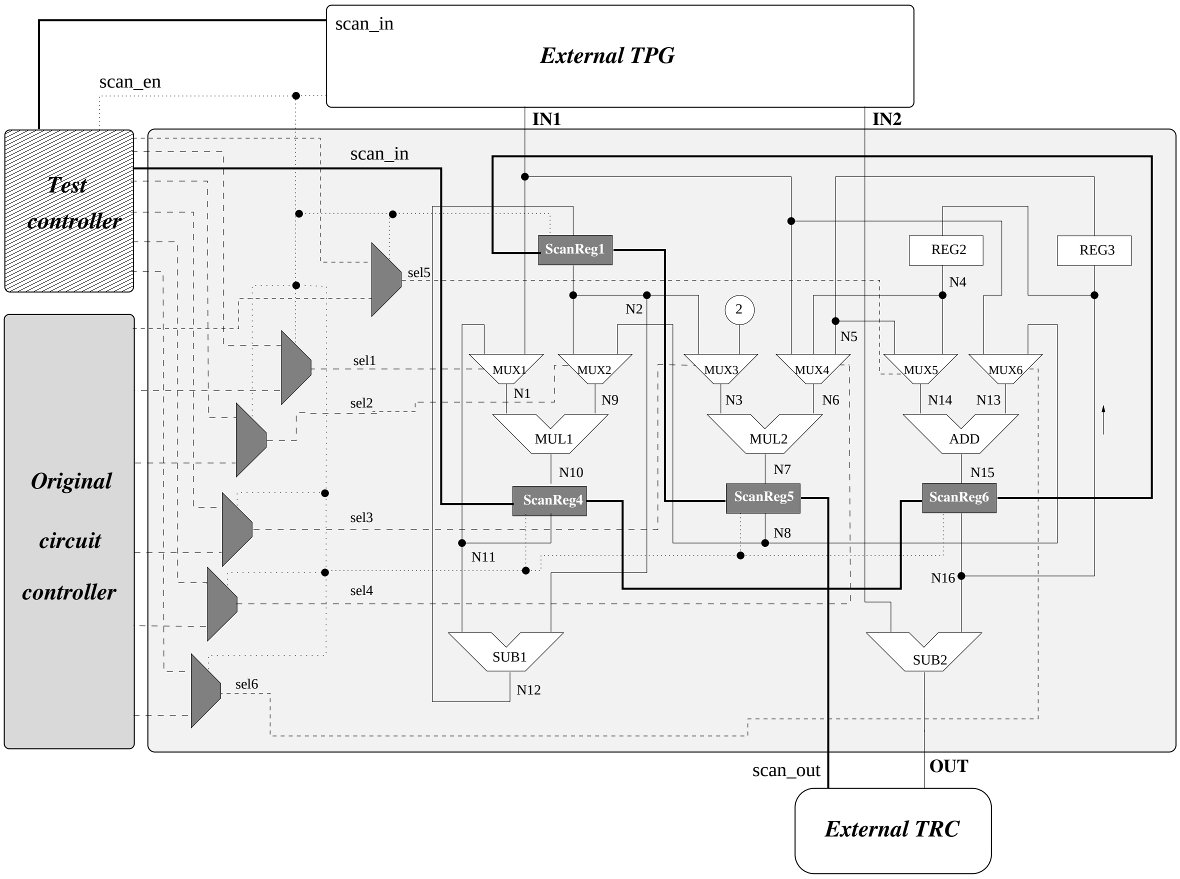


Figure 5.4: RTL architecture of circuit Diffeq with inserted scan chain.

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 ([DBBM00]). This generator takes as input the test structure parameters given by IDAT. Other kind of test structures, e.g. based on cellular automata [CSdAAC99], 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.

The advantage of such a concept is that internal test is not limited only to the test of selected components, but also includes component interconnection test. When we summarize our BIST approach we assume to obtain quickly testable circuit with parameters corresponding to user requirements by:

- connecting of TPG and TRC to selected FUs
- transformation of selected FUs into their BIST equivalents, if such equivalents exist in the library
- combination of both above approaches

- creation of one or more scan chains to test interconnections not tested during FUs tests.

The example of BIST circuit obtained by applying our technique to the circuit of figure 5.1 is depicted in figure 5.5. More details on test controller for the BIST data-path can be found in [DBBM00] and in [DB99].

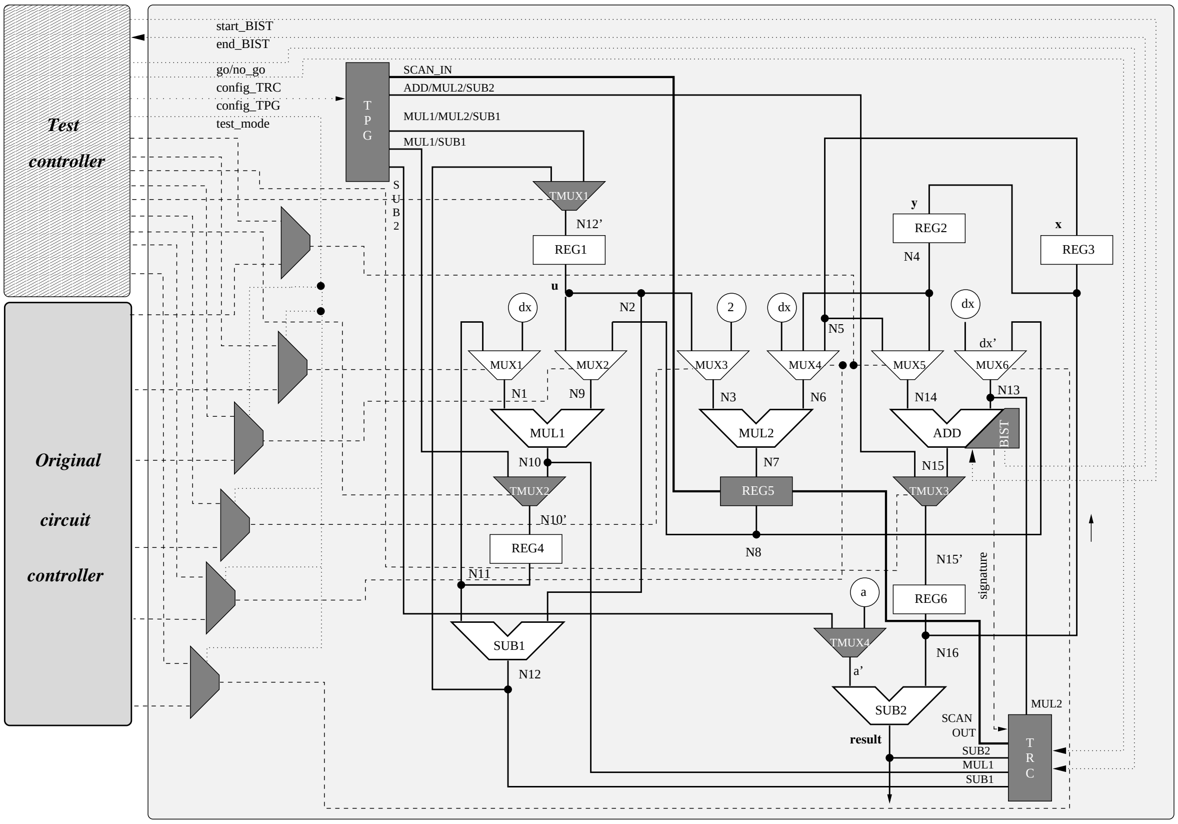


Figure 5.5: BIST architecture created by IDAT.

5.4 Trade-off overview

Trade-off mechanisms to solve a *selection problem* are used in many domains of human activity and their principles are well known. A selection problem belongs to a broader class of problems, referred to as *multiple criteria decision making* [HY81]. Multiple criteria decision making methods have been studied by many researchers in various disciplines such as operational research, management science, economics, marketing research, applied mathematics, decision theory and psychology.

5.4.1 Problem definition

Let us have a circuit data-path composed of interconnected components. For each component certain number of testable versions can exist in library. Each version is described by a set of attributes.

The goal is, according to the constraints related to chosen testability methodology and weighed requirements for the final circuit:

- To find an optimal version for each component such that the final circuit composed of these versions will best accomplish user requirements.
- To add corresponding test structures allowing that allocated testable versions function correctly in both normal and test modes.

5.4.2 Solution outline

A trade-off mechanism has even been particularly used for a purpose similar to ours, but at a lower level of abstraction, namely at the gate level. Such research activities have been developed in the late eighties by Zhu from University of Southern California, under the direction of Melvin Breuer. Their *Testable Design Expert System* (TDES) ([Zhu86] and [ZB88]) was developed for PLAs, RAMs, ROMs, and blocks of combinational logic in order to make these architectures testable. Ten years later, the state of the art has allowed to gain more than one level of abstraction and we are in our own work solving the similar type of problems at the RTL. In our case, their approach of *linear weight system* has been particularly useful, and we have adapted it to our context.

Our trade-off process first of all searches to satisfy the user requirements. If this 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 component, which is identified as easily testable, may show testability problems when it is embedded into a data-path. To avoid this situation, our trade-off process runs in two steps:

- Initially the *local trade-off* is executed. Local trade-off acts at the component level. During this trade-off, the attributes of each component version are compared with corresponding user requirements resulting in the list of all versions ordered according to the level of satisfaction of user requirements.
- Then *global trade-off* follows at the level of the data-path. During this trade-off, in each iteration step the global (circuit) attributes are calculated and compared with corresponding requirements. According to the results of this comparison, the circuit modifications in order to satisfy user requirements are performed.

More details on our trade-off algorithm can be found in chapter 8.

5.5 Related publications

- P. Bukovjan and M. Marzouki and W. Maroufi, *Allocation for Testability in High-Level Synthesis Process*, Proceedings from Electronic Devices and Systems International Conference, Brno, Czech Republic, pp. 153-156, June, 1998
- P. Bukovjan and M. Marzouki and W. Maroufi, *Design for Testability Reuse in Synthesis for Testability*, Proceedings of XII Symposium on Integrated Circuits and System Design, Natal, Brazil, pp. 206-209, September, 1999
- L. Ducerf-Bourbon and P. Bukovjan and M. Marzouki, *TACOS: A Testability Allocation and Control System*, Compendium of Papers from IEEE European Test Workshop, Cascais, Portugal, May 2000

Chapter **6**

Testability analysis

The test of the whole circuit means the test of each circuit node, to check if it is not stuck at 0 or at 1. To test a node it is necessary:

- In the first phase to set any specified value to this node by using of primary inputs, scan components outputs or a TPG outputs connected directly or through one or more components to this node. This process is called controlling of a node and the metrics used to express the difficulty to perform this is called *controllability*.
- In the second phase it is necessary to check if this value was set correctly. This is performed by using of primary outputs, scan components inputs or TRC inputs connected directly or through one or more components to this node. This checking process is called observing of a node and the difficulty to perform this checking is traditionally reflected by the *observability* metrics.
- If a node can be controlled and observed then this node is called a testable node. The *testability* metric is a combination of the node controllability and the node observability metrics.

In this chapter the testability analysis of general RTL data-path, which is driven in the test mode by added test controller, is proposed for early prediction of testability bottlenecks. Proposed testability analysis is used by all allocation for testability techniques namely test point insertion, scan and BIST, in the following way:

- In the test point insertion approach the testability analysis selects the points to be controlled and to be observed.
- In the scan approach the testability analysis is used, as in the test point insertion approach, to select the points, which should be controlled resp. observed. This is done by selecting the nearest sequential components using a special algorithm and then transforming it into its scan equivalent from the library.
- In the BIST approach the testability analysis guides the selection of the set of FUs which should be tested.

This chapter is organized as follows. After the detailed definition of used testability metrics in section 6.1 and equations to calculate these metrics in section 6.2, the testability analysis algorithm is explained in section 6.3. In section 6.4 the demonstration of this algorithm on a simple example is performed, followed, in section 6.5, by the evaluation of our method in the form of the comparison of our controllability and observability results with similar ones obtained by another existing testability analysis method. Finally some directions for future amelioration of our testability analysis method are provided in section 6.6. Contributions of presented testability analysis are summarized in section 6.7.

6.1 Testability definition

According to [AS88] the *testability* is the property of a circuit that makes it easy and sometimes possible to test. To be useful, testability analysis should be simpler than actual test generation or fault simulation. The goal of the testability analysis is to identify the most hard-to-test parts of the circuit data-path.

The main testability metrics defined in this section and used in the trade-off process to focus allocation for testability activities to hard-to-test parts are the *controllability exigency* and the *observability exigency* of a node, which are based on controllability and observability metrics. The controllability and observability metrics should reflect both transparency and feed-back problems.[1] In order to reflect component transparency to test vectors we use the controllability transfer factor and the observability transfer factor.

Although some concepts are common to other approaches presented in literature and summarized in chapter 4 (especially to the approaches of Gu et al. [GKP94] [Gu96]and Pires et al.[FPR96] [Pir98]), our testability analysis process differs from them by the fact that we suppose the creation of the test controller for all testable architectures (see chapter 5). Also the formulas to calculate testability metrics for each component port are different. In addition we calculate the testability metrics for all circuit nodes including control nodes.

This section is organized as follows. Our definitions of node controllability, observability and testability metrics are given in subsection 6.1.1. The discussion about controllability and observability transfer factors is made in subsection 6.1.2 followed by the definition of controllability, observability and testability exigencies of a node. Finally the formulas to calculate testability metrics for each port of various components are provided in subsection 6.2.1.

[1] In the future version of our system we plan to also include the solution for reconvergence of component inputs.

6.1.1 Controllability, observability and testability of a node

Our testability analysis process is based on the calculation of controllability, observability and testability of each node. The controllability, observability and testability values range from 0 to 1. The higher those values are, the better they are.

Controllability

The *controllability* (C) of node i measures the difficulty to set a desired value to this node from the circuit primary inputs, outputs of scan components or TPG outputs. The general formula to calculate controllability of node i was taken from [FPR96]:

$$C_i = \frac{y}{N}, \tag{6.1}$$

where y is the number of test patterns that can be propagated to this node from the primary inputs regardless of the values of patterns. N is the total number of possible values, which can appear at this node during the normal operation of the circuit. Usually $N = 2^n$, where n is the bit-width of node i.

Most of the circuit nodes are signals connecting the outputs of components to the inputs of other ones. To calculate component output controllability, it is necessary to know, how difficult it is to set a desired value to the component output from the component inputs. This difficulty is expressed, for the case where the component is not embedded into a circuit, by the component *controllability transfer factor* discussed in subsection 6.1.2.

Observability

The *observability* (O) of node i measures the difficulty to observe a desired value from this node, which is usually the input of one ore more components, on the circuit primary outputs, scan components inputs or TRC inputs. The general formula to calculate observability of node i was taken from [FPR96]:

$$O_i = \frac{y}{N}, \tag{6.2}$$

where y is the number of pairs of faulty and fault-free responses that can be propagated and differentiated at the primary outputs regardless of their values. N is the total number of possible pairs. Usually $N = 2^n \times (2^n - 1)$, where n is the bit-width of node i.

To calculate component input observability, it is necessary to know how difficult it is to observe a desired value from the component input at the component output node by setting the other component inputs into specified values. This difficulty is expressed, for the case where component is not embedded into a circuit, by the component *observability transfer factor* defined in subsection 6.1.2.

Testability

After that, the *testability* (T) of node i is calculated in order to compact the controllability and observability values of this node to one value expressing overall testability of this node. As is stated above a node is not testable if is not controllable and observable.

- The testability of node i, which is not an output of a constant generator, is simply the product[2] of node controllability and node observability values
$$T_i = C_i \times O_i. \tag{6.3}$$
- If node i is the output of a constant generator, such a node is not controllable. However, in this case it does not mean automatically, that such a node in not testable. This node is considered testable, if it is possible to observe it, because we suppose that in normal mode only the value of connected constant can appear at this node. In this way, for node i connected to a constant its testability is equal to its observability:
$$T_i = O_i. \tag{6.4}$$

Composed and decomposed nodes

If node i is *composed* (see figure 6.1a) from the bits of different controllabilities, its controllability is estimated as the average of all bits controllabilities[3]:

$$C_i = \frac{1}{w}\sum_{k=1}^{w} C_{i_k},$$

where w is node i bit-width.

The observability of a node composed of the bits of different observabilities is estimated in the same way:

$$O_i = \frac{1}{w}\sum_{k=1}^{w} O_{i_k},$$

where w is node i bit-width.

Flag "constant" of node i (explained in the sequel of this chapter) is set to YES, only if all bits are connected to constants.

If node i is *decomposed* (see figure 6.1b) into its bits, each of these bits inherits all testability attributes of its original node.

[2]The product is only one possible way of combining controllability and observability metrics. In our case, we have adopted a pessimistic approach, which requires both non-zero controllability and non-zero observability to consider a node accessible for test. Some authors uses different approaches to combine these metrics for example average or sum of these values.

[3]This assumption is a simplification. In fact the "high" bits have generally worse controllability and observability than "low" bits. The better solution can be to introduce CTF and OTF (explained in the sequel of this chapter) for each component output bit. Total component output CTF (resp. OTF) will be then the average of all bits' CTFs (resp. OTFs).

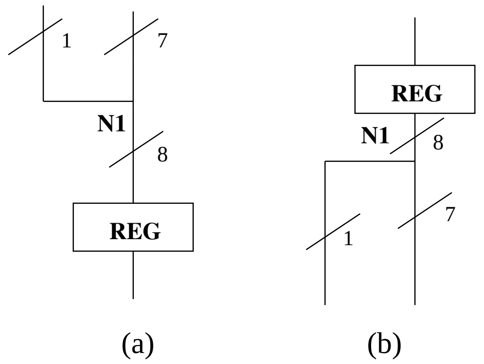


Figure 6.1: (a) Node N1 composed of more bits, which can have different controllabilities. (b) Node N1 decomposed into bits, which can have different observabilities.

6.1.2 Controllability and observability transfer factors

Our definition of FU transfer factors is adapted from Gu definition [Gu96]. In the current version of IDAT we suppose, as also Gu supposes, only FUs with one data output. The extension to multi-output FUs is discussed in section 6.6.

The *controllability transfer factor (CTF)* of a FU reflects the difficulty to control FU outputs by its inputs, and the *observability transfer factor (OTF)* reflects the difficulty to observe FU inputs at its outputs. These transfer factors are measured by a *probabilistic method*. However, they reflect not only the difficulties in random test generation but also the difficulties in deterministic test generation. Both CTF and OTF are in the range from 0 to 1, where 1 represents the best controllability and observability transfer factor of a FU. For further studies the exact formula to calculate CTF and OTF can be found in [Gu96].

Different FUs have different $CTFs$ and $OTFs$. These factors only depend on the function performed by the FU and FU I/Os bit width.

The CTF and OTF for each FU are calculated only once and stored in a library together with other testability attributes of a FU. As was already indicated, the transfer factors are useful only for FU. For register or multiplexer the CTF and OTF are considered to be 1 and are not used in the formulas to calculate controllability and observability described in the next section.

Transfer factors of scan FUs

The CTF of a FU with implemented scan structure reflects the transparency from the last layer of internal scan registers to the components data outputs. In the same way the OTF of a FU with implemented scan structure reflects the transparency from the components data inputs to the first layer of internal scan registers. In the test mode the part of the component between these layers of registers is never used in the context of

FU inputs or outputs.

The example of a scan FU is depicted in figure 6.2. For this FU the CTF is equal to the worst CTF of both input internal FUs. The OTF of FU from figure 6.2 is equal to 1, because there is no combinational logic between the output of scan register 3 and the primary output Q.

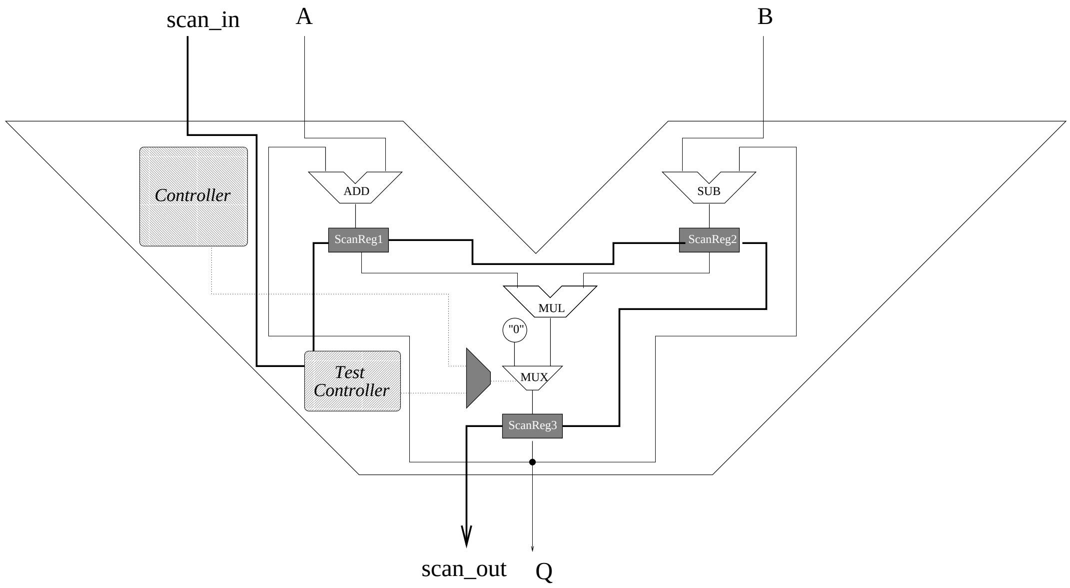


Figure 6.2: FU with implemented scan chain.

Transfer factors of BIST FUs

In the context of a FU with implemented BIST structure, the CTF and OTF are used in the following way:

- If a BIST mode is activated, the FU isolates itself from the rest of the circuit and test itself. The CTF and OTF are not used in this mode.
- If BIST FU does not test itself, its CTF and OTF are the same as the CTF and OTF of the original FU and are used in the same way.

6.1.3 Testability exigency of a node

As defined above, the controllability of a node reflects the lack of components transparency for controlling the component output and the problems related to inputs reconvergence.[4] The components input observability refers to the lack of transparency to observe component inputs from its outputs with respect to the reconvergence of other

[4] Detection of component inputs reconvergence is not yet implemented.

component inputs.

Traditionally controllability and observability metrics of each circuit node are supposed to guide the allocation of testability resources. This is undoubtedly valid at gate level, where the nodes are connected to components with nearly the same complexity. However at RTL, the controllability and observability metrics can not reflect the importance to control or to observe the nodes with respect to the actual test plan. Usually in such a plan the test of complex components namely FUs takes often more than 90% of total circuit test time, while the test of less complex components and components interconnections takes the rest. From this point of view to select a node, which should be controlled or observed with maximal effect to overall circuit testability it is necessary to consider also the complexity to test the components connected to this node.

Our defined controllability exigency and the observability exigency of a node attempts to overcome this drawback of traditional controllability and observability metrics. The test complexity of a component is simply modelized by its *area* supposing that the test of a large component takes usually more time than test a small one. The exigency to control or to observe a node can be also understand as some necessity to improve controllability or observability exactly of this node. The values range of exigency metrics varies from 0 to infinity, where bigger value refers to stronger necessity.

6.1.4 Controllability exigency definition

The *controllability exigency* (CE) of node i depends on the controllability of this node and the test complexity of the components driven by this node:

$$CE_i = cpx_{trg_i} \times (1 - C_i), \tag{6.5}$$

where cpx_{trg_i} is the sum of complexities of the components driven by node i (target components).

Formula 6.5 is not valid for the following nodes:

- *circuit primary outputs (resp. TRC inputs), which are not inputs of any component:* During the circuit test the inputs of all components are controlled in order to impose the responses at theirs outputs, which will be observed and evaluated. The circuit primary outputs, which are not inputs of any data-path components are not supposed to be controlled. Similar conclusion can be made also for a BIST circuit, where the primary outputs are the TRC inputs.

- *constant output:* We also do not control the output of a constant generator, because we suppose that the value at this output never change.

From these reasons circuit primary outputs, TRC inputs and outputs of constant generators have controllability exigency equal to 0.

6.1.5 Observability exigency definition

The *observability exigency* (OE) of node i depends on the observability of this node and the test complexity of the component driving this node:

$$OE_i = cpx_{src_i} \times (1 - O_i), \tag{6.6}$$

where cpx_{src_i} is the complexity of the component driving node i (source component).

This formula is not valid for *circuit primary inputs and scan components outputs (resp. TPG outputs)*. During the circuit test the outputs of all components are observed in order to evaluate the responses to applied test data from the components inputs. Because circuit primary inputs are not outputs of any data-path components, we suppose that we do not need to observe it. Related deduction can be made for a BIST circuit, where the role of primary inputs is transformed to the TPG outputs. From these reasons circuit primary inputs and TPG outputs have observability exigency equal to 0.

6.1.6 Testability exigency definition

In order to express the overall testability of node i the *testability exigency* (TE) is defined as the sum of controllability and observability exigencies:

$$TE_i = CE_i + OE_i. \tag{6.7}$$

6.2 Component's outputs controllabilities and inputs observabilities

6.2.1 Multiplexer

The multiplexer output *controllability* is calculated by the following formula:

$$C_{out} = C_{sel} \times \max C_{in}, \tag{6.8}$$

where C_{sel} is the controllability of multiplexer selection input. As was stated in chapter 5 for all circuit test architectures approaches we suppose that testing algorithm is independent from the original circuit controller and each multiplexer in the circuit can be controlled in an independent way by added test controller. For this reason, in order to have the best possible controllability of the multiplexer output, we can select the best controllable data input. The selection input is connected directly to the test controller and is considered fully controllable ($C_{sel} = 1$). More discussions concerning this approach can be found in section 6.6.

The *observability* of any multiplexer data input is simply:

$$O_{in} = C_{sel} \times O_{out}.$$

To calculate the observability of multiplexer selection input we have adopted the pessimistic approach. According to this approach the observability of the selection input is determined by the worst controllability of multiplexer inputs. The selection input is observable only if all data inputs, except the inputs connected to a constant (see section 6.3), are controllable:

$$O_{sel} = O_{out} \times \prod_{i=1}^{n} C_{in_i},$$

where n is the number of multiplexer data inputs not connected to a constant.

6.2.2 Register

The register output *controllability* is equal to the controllability of the register data input multiplied by the controllability of the clock input:

$$C_{out} = C_{clk} \times C_{in}. \tag{6.9}$$

The *observability* of the register data input is in the same way defined as:

$$O_{in} = C_{clk} \times O_{out}. \tag{6.10}$$

The controllability of the clock input is, in current version of IDAT, always supposed to be equal to 1. In future versions of our system we may implement the checking procedure if the circuit clock signal goes directly from its primary input to the input of all sequential components without passing through any combinational logic.

The observability of the register clock input is obtained by application of following formula:

$$O_{clk} = C_{in} \times O_{out}. \tag{6.11}$$

6.2.3 Functional unit

In the same way as for the calculation of the multiplexer selection input observability, the calculation of FU outputs controllability and inputs observability we have adopted the pessimistic approach.

Controllability

The *controllability* of a FU output is strongly dependent on the worst input controllability. If any FU input is both not controllable and not connected to a constant generator[5], the FU output is also not controllable. From this reason the formula to calculate FU output controllability is defined as follows:

$$C_{out} = CTF \times \prod_{i=1}^{n} C_{in_i}, \tag{6.12}$$

[5]Node connected to a constant generator is not necessary to control (see subsection 6.3.1)

where CTF reflects the transparency problems caused by selected implemented function and n is the number of FU inputs not connected to a constant.

Observability

The *observability* of a FU input depends on the observability of the worst observable FU output and also on the worst controllable FU input different from observed input:

$$O_{in} = O_{out} \times OTF \times \prod_{i=1,\ i \neq in}^{n} C_i, \tag{6.13}$$

where OTF reflects the transparency problems caused by implemented function, n is the number of FU data inputs not connected to a constant and input i is not connected to a constant (see section 6.3).

To calculate the observability of possible selection input of the FU, OTF is not necessary:

$$O_{sel} = O_{out} \times \prod_{i=1}^{n} C_{in_i},$$

where n is the number of FU inputs not connected to a constant and input i is not connected to a constant. O_{sel} is the observability of selected FU output.

6.3 Testability analysis algorithm

The *propagation algorithm* has been developed in order to progressively calculate the controllability and observability value of each node in RTL data-path.

6.3.1 Calculation of controllabilities

Initialization

The controllability evaluation starts:

- for external test from the primary inputs and the scan components' outputs (if any). The controllability of primary inputs and scan registers outputs is set to 1 (the best possible value).
- for internal test from TPG outputs and the scan components' outputs (if any). The controllability of TPG outputs and scan registers outputs is set to 1 (the best possible value).

The controllability of the scan FU outputs is equal to the CTF of this FU (see section 6.1.2). All other nodes have their controllability set to 0 (the worst possible value). They are initially considered as uncontrollable.

Propagation

The controllability calculation then progresses:

- for external test towards primary outputs and scan components' inputs
- for internal test towards TRC inputs and scan components' inputs (if any).

The controllabilities of each component output can be modified only if all component inputs are either controllable or connected to a constant (see the next paragraph). However, this condition doesn't apply if the component is multiplexer. For multiplexers the controllability of the multiplexer output is set to its best input controllability.

Existence of constants

The existence of constants in data-path requires special attention. It is not surprising that the output of a constant generator is absolutely uncontrollable. We can prove this statement by using of general formula 6.1 to calculate the controllability of a node. In our case the number of values that can be set at this node (y) by using of primary inputs, scan component outputs or TPG outputs is 0. The number of possible values which can appear at this node during the normal operation of the circuit (N) is 1. The controllability of a constant generator output is then equal to 0.

We can however make profit from the existence of such "constant nodes". First of all it is necessary to be aware of the feature that the value at the output of a constant generator is still the same and can not change during the normal operation of the circuit. From this point of view it is not necessary to check if we can set an another value at this node. To declare this node testable it is sufficient to observe it (see formula 6.4). If such a node is connected to an input of a FU and all the other inputs are controllable, it is possible to calculate the FU output controllability. Such a situation is depicted in figure 6.3 and described in the following paragraph.

Let us assume that selection input of multiplexer in figure 6.3 is fully controllable. In spite of uncontrollable node N5 and FU input N2, which is in the first cycle of propagation algorithm also uncontrollable, the controllability of FU output N3 is calculated by the following formula:

$$C_{N3} = CTF_{ADD} \times C_{N1}. \tag{6.14}$$

C_{N2} is not taken into account. This special treatment is allowed by the introduction of *"constant" flag*, which is assigned to each node in the circuit. If this flag is set to YES, the corresponding node is considered to be connected to a constant and then this node is not used in the controllability/observability calculations. In our case, flag "constant" is set to YES for nodes N5 and N2.

The information about connection of the node to a constant expires:

- on the output of a FU, if at least one FU input is not connected to a constant,

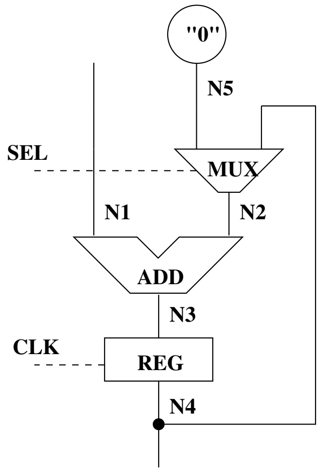


Figure 6.3: An example of a circuit where one FU input is connected through the multiplexer to constant "0".

- on the output of a multiplexer, if a controllable input of this multiplexer is selected. Example: controllability of node N2 from figure 6.3 is calculated at two steps. In the first step we connect MUX output N2 to the input N5, which is the output of a constant generator. The controllability of N2 remains 0 and flag "constant" of this node is set to YES. In the second step MUX output N2 is connected to input N4. Controllability of node N2 is set to controllability of N4, which is higher than 0 and flag "constant" of node N2 is set to NO.
- on the output of a register with RESET/SET input, when its data input is already controllable (see the next paragraph).

More discussion concerning the existence of constants in data-path can be found in section 6.6.

Component with RESET/SET input

In the case where a component with the reset/set possibility is embedded in some hard-to-test circuit part, such a component can be considered as the:

- *temporary generator of constant 0* when RESET input exists and is used
- *temporary generator of constant* $2^n - 1$ *(resp. -1)*, where n is the bit-width of component data output signal, when SET input is active.

Sometimes such a possibility greatly improves the testability of related hard-to-test parts. For example in the circuit from figure 6.4 the possibility to reset the register enables the test algorithm to initialize input B of the FU. However a component with the reset or set possibility can not be considered as the real generator of constant. The reason is that such a component has, beyond the RESET/SET input, also one or more data inputs. Returning to the circuit from figure 6.4, during the normal operation of this circuit, at the output of the register also different values than 0 or $2^n - 1$ (resp. -1) can appear.

The conclusion is that, when one or more inputs of a FU are connected to the output of a component with RESET/SET possibility, it is not enough (and sometimes impossible) to apply the reset resp. set during the whole test procedure, to declare the FU output controllable. Returning to the circuit of figure 6.4, the reset input of the register is used only to initialize the feedback and then the FU inputs need to be considered as reconvergent.

Currently, the controllability of a component output with RESET input, when this RESET input is applied is, according to formula 6.1 equal to:

$$C_{out} = \frac{1}{N}, \tag{6.15}$$

where N is the number of all possible values, which can appear in the normal functional mode at this component output. In other words, with the RESET or SET input of a component we are able to set at component output only one value from all possible values which can be set at this output during the normal operation of the circuit. Returning to the circuit from figure 6.4 the controllability of output B, when the reset is active, is:

$$C_B = \frac{1}{2^8} = 0.0039.$$

If a component has both RESET and SET inputs then

$$C_{out} = \frac{2}{N}.$$

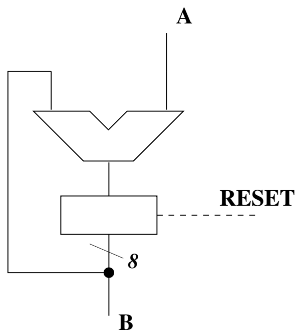


Figure 6.4: An example of a circuit with initializable feedback.

From above reasons we can not make profit of using the flag “constant” and we use the RESET/SET ability of a component only when all component inputs are uncontrollable in order to initialize the first value at the component output. In the circuit of figure 6.4 the final controllability of register output B will be better than controllability calculated in the first step using formula 6.15, because of reconvergent FU inputs.[6]

[6]In the current state of IDAT we are not able to test this example, because the reconvergence treatment is not yet implemented to corresponding algorithms (see proposition in section 6.6).

Controllability improvement in next iteration steps

Controllability of any internal node appearing in a loop can be improved in some of the next iteration steps of our testability analysis algorithm.

Let us return back to the example of figure 6.3. In the second iteration of propagation algorithm, the multiplexer is switched to connect N4 to N2. In this case, $C_{N2} = C_{N4}$ and "constant" flag for N2 is set to NO. New C_{N3} is calculated. If this new controllability is better than the old one, the controllability of N3 is set to this better value:

$$C_{N3} = C_{N3_{new}}.$$

Otherwise, C_{N3} remains unchanged.

6.3.2 Calculation of observabilities

When the calculation of controllabilities is accomplished the calculation of observabilities is performed.

Initialization

The observability evaluation, symmetrically to controllability evaluation, starts:

- for external test, from the primary outputs and the scan components' inputs (if any). The observability of primary outputs and scan registers inputs is set to 1 (the best possible value). The observability of the scan FU inputs is equal to the OTF of this FU (see section 6.1.2).
- for internal test, from TRC inputs and the scan components' inputs (if any). The observability of TRC inputs and scan registers outputs is set to 1 (the best possible value).

The observability of all other nodes is set to 0. They are initially considered as unobservable.

Propagation

The observability calculation then progresses backwards towards the:

- primary inputs and the scan components' outputs (if any) in external test mode
- TPG outputs and the scan components' outputs (if any) in internal test mode.

During the iteration algorithm, the observability of a component input can be calculated only when the component output is observable and the component inputs, other than the currently processed one, are either controllable or connected to a constant. This condition doesn't apply when the component is multiplexer.

6.4 Example

Supposing an external test, let us apply above described testability analysis algorithm on a simple data-path example from figure 6.3. All calculated controllabilities and observabilities together with controllability, observability and testability exigency of nodes are finally summarized in table 6.1.

Node	C	O	T	CE	OE	TE
N1	1	0.709	0.709	0	0	0
N4	0.817	1	0.817	2673.3	0	2673.3
CLK	1	0.817	0.817	0	0	0
SEL	1	0.709	0.709	0	0	0
N2	0.817	0.868	0.709	7873.3	1925.7	9799.0
N3	0.817	1	0.817	5163.4	0	5163.4
N5	0	0.868	0.868	0	3297.5	3297.5

Table 6.1: Testability analysis results for circuit from figure 6.3.

6.4.1 Calculation of controllabilities

To calculate controllability of all nodes it is necessary to do the following steps:

1. For primary inputs N1, SEL and CLK, the controllability is set to 1. The controllability of all other nodes is set to 0.

2. Assuming MUX driven by the added test controller. In this way we are free to switch the multiplexer to input N5, which is connected to a constant $C_{N2} = C_{N5} = 0$. The flag “constant” is copied from N5 to N2.

3. The CTF of two inputs 8 bits adder is 0.817. Because N1 is the primary input ($C_{N1} = 1$) and N2 is connected to the constant, the controllability of its output is according to formula 6.14 resulting in 0.817.

4. Because $C_{CLK} = 1$, the controllability of REG data input N4 is simply copied to its output: $C_{N4} = C_{N3} = 0.817$.

5. The new controllability value of N2 is obtained by setting $C_{N2_{new}} = C_{N4} = 0.817$.

6. Because $(C_{N2} = 0) < (C_{N2_{new}} = 0.817)$ then $C_{N2} = C_{N2_{new}} = 0.817$.

7. $C_{N3_{new}}$ is recalculated by means of formula 6.12:

$$C_{N3_{new}} = CTF_{ADD} \times C_{N1} \times C_{N2} = 0.817 \times 1 \times 0.817 = 0.667.$$

8. Because $C_{N3} > C_{N3_{new}}$ controllability of N3 remains unchanged.

6.4.2 Calculation of observabilities

When the calculation of controllability for each node is finished, the observability of each node is obtained by following steps:

1. For primary output N4 the observability is set to 1.

2. The observability of REG input N3 is calculated by formula 6.10 as

$$O_{N3} = C_{CLK} \times O_{N4} = 1 \times 1 = 1.$$

The observability of REG clock input is calculated according to formula 6.11 as

$$O_{clk} = C_{N3} \times O_{N4} = 0.817 \times 1 = 0.817.$$

3. The observabilities of both FU inputs are obtained by applying formula 6.13:

$$O_{N1} = O_{N3} \times OTF_{ADD} = 1 \times 0.868 = 0.868,$$

where C_{N2} is not considered in above equation, because we suppose the connection of N2 to the constant.

$$O_{N2} = O_{N3} \times C_{N1} \times OTF_{ADD} = 1 \times 1 \times 0.868 = 0.868.$$

4. The observability of MUX SEL input is

$$O_{SEL} = O_{N2} \times C_{N4} = 0.868 \times 0.817 = 0.709,$$

where C_{N5} is not considered, because N5 is connected to the constant.

5. The observability of the constant node N5 is

$$O_{N5} = C_{SEL} \times O_{N2} = 1 \times 0.868 = 0.868.$$

6. New observability of N4 is $O_{N4_{new}} = O_{N2} = 0.868$.

7. Because $(O_{N4} = 1) > (O_{N4_{new}} = 0.434)$ the observability of N4 remains unchanged.

6.4.3 Calculation of controllability and observability exigencies

The test complexities (modelized by areas) of all components from circuit in figure 6.3 are: $cpx_{MUX} = 14600$, $cpx_{ADD} = 43000$ and $cpx_{REG} = 28200$.[7]

[7] The unit of test complexity is the same as the unit for area (Synopsys units).

Controllability exigencies

The controllability exigencies of all circuit nodes are calculated as follows:

1. $CE_{N1} = cpx_{ADD} \times (1 - C_{N1}) = 43000 \times (1 - 1) = 0$.
2. Because node N5 is the output of a constant generator, we do not need to control it: $CE_{N5} = 0$.
3. $CE_{N2} = cpx_{ADD} \times (1 - C_{N2}) = 43000 \times (1 - 0.817) = 7873.3$.
4. $CE_{N3} = cpx_{REG} \times (1 - C_{N3}) = 28200 \times (1 - 0.817) = 5163.4$.
5. $CE_{N4} = cpx_{MUX} \times (1 - C_{N4}) = 14600 \times (1 - 0.817) = 2673.3$.
6. $CE_{SEL} = cpx_{MUX} \times (1 - C_{SEL}) = 14600 \times (1 - 1) = 0$.
7. $CE_{CLK} = cpx_{REG} \times (1 - C_{CLK}) = 28200 \times (1 - 1) = 0$.

Observability exigencies

The observability exigencies of all circuit nodes are calculated as follows:

1. $OE_{N4} = cpx_{REG} \times (1 - O_{N4}) = 28200 \times (1 - 1) = 0$.
2. $OE_{N3} = cpx_{ADD} \times (1 - O_{N3}) = 43000 \times (1 - 1) = 0$.
3. Because N1, SEL and CLK are the primary inputs, we do not need to observe them: $OE_{N1} = OE_{SEL} = OE_{CLK} = 0$.
4. $OE_{N2} = cpx_{MUX} \times (1 - O_{N2}) = 14600 \times (1 - 0.868) = 1925.7$.
5. Because N5 is the output of the constant generator, we do not need to observe it: $OE_{N5} = 0$.

Testability exigencies

The testability exigency of each circuit node is simply calculated by using formula 6.7 and summarized in table 6.1.

6.5 Evaluation

In order to compare the results from our testability analysis process with the results from another approach we have applied our method to the example depicted in figure 6.5. This circuit was originally used to demonstrate testability analysis method developed at Linköping University by Gu et al. Their controllability and observability results were extracted from [Gu96].

As was already stated in chapter 6, our testability analysis approach was inspired by Gu et al. method and we also use the same CTFs and OTFs for each component. In contrast of Gu et al. testability definition, we have dropped the use of sequential factors, because we suppose that they are not useful in our proposed test design architectures with the independent test controller. Analysis concerning test time are performed by the test scheduler tool cooperating with IDAT.

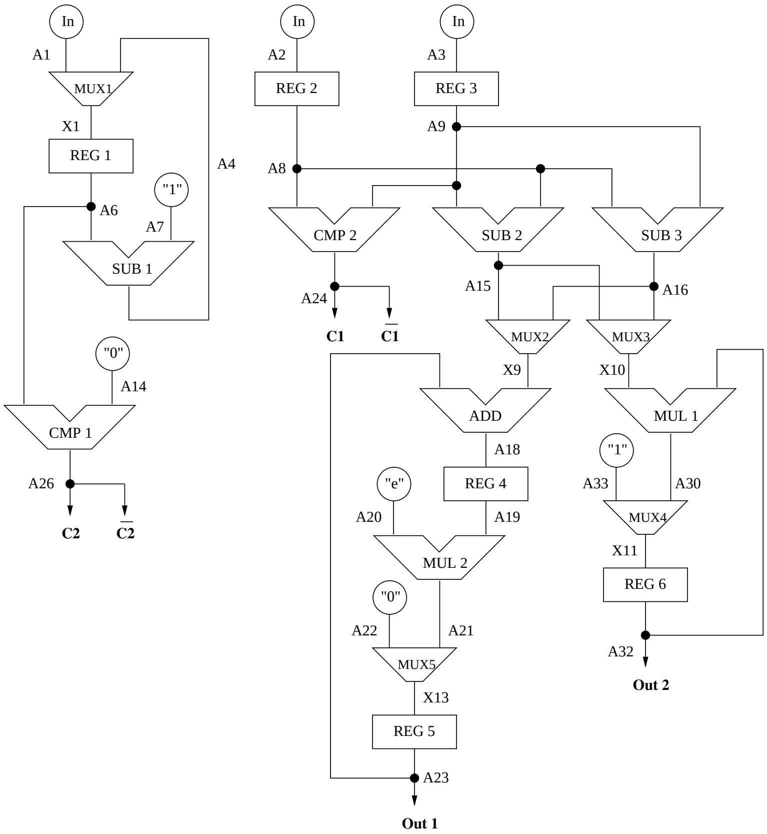


Figure 6.5: Example of a data-path from Gu thesis [Gu96].

Table 6.2 summarizes the testability analysis results. Our controllability and observability values are compared with Gu et al. combinational controllability and combinational observability values. The correlation between these results is for controllability 98.21% and for observability 76.53%.

The difference is explained by the fact that Gu's method is based on the reusing of already existing instructions from the original circuit controller to test the data-path, while we suppose the creation of a new test controller, which, during the test, controls the circuit data-path independently from the original controller. The other difference is that we use different formulas to calculate testability metrics. For example, to be able to calculate the controllability of FU output, our testability analysis algorithm requires that all FU inputs are either controllable or connected to a constant. In other case,

Node	Gu's		IDAT					
	C	O	C	O	T	CE	OE	TE
A1	1.00	0.55	1	0.642	0.642	0	0	0
A2	1.00	0.36	1	0.642	0.642	0	0	0
A3	1.00	0.36	1	0.642	0.642	0	0	0
A4	0.84	0.55	0.816	0.642	0.524	2686.4	16826.0	19512.4
A6	1.00	0.55	1	0.642	0.642	0	10095.6	10095.6
A7	-	0.48	0	0.557	0.557	0	11067.0	11067.0
A8	1.00	0.36	1	0.642	0.642	0	10095.6	10095.6
A9	1.00	0.36	1	0.642	0.642	0	10095.6	10095.6
A14	-	0.55	0	0.642	0.642	0	8950.0	8950.0
A15	0.72	0.51	0.816	0.502	0.410	5372.8	23389.4	28762.2
A16	0.72	0.51	0.816	0.502	0.410	5372.8	23389.4	28762.2
A18	0.52	0.75	0.667	0.802	0.534	9402.2	8531.2	17933.4
A19	0.52	0.75	0.667	0.802	0.534	137964.9	5594.9	143559.8
A20	-	0.57	0	0.534	0.534	0	11641.5	11641.5
A21	0.42	0.86	0.512	1	0.512	7125.7	0	7125.7
A22	-	0.86	0	1	1	0	0	0
A23	0.36	1.00	0.512	1	0.512	20986.5	0	20986.5
A24	0.86	0.44	0.860	1	0.860	0	0	0
A26	0.86	0.86	0.860	1	0.860	0	0	0
A30	0.50	0.86	0.627	1	0.627	5450.4	0	5450.4
A32	0.43	1.00	0.627	1	0.627	154476.5	0	154476.5
A33	-	0.86	0	1	1	0	0	0

Table 6.2: Comparison of Gu's and IDAT's testability analysis results for the circuit data-path of figure 6.5.

according to our algorithm, it is impossible to proceed to this FU output. However, Gu's algorithm in this case accepts also uncontrollable FU inputs.

6.6 Discussion and future work

Some parts of our testability analysis method, need more detailed approach. Besides the improvement of presented formulas accuracy and inclusion of a solution for reconvergence of component inputs, it is also necessary to identify embedded loops.

6.6.1 New attribution of transfer factors

Formulas 6.12 and 6.13 are in fact only valid for FUs with one output. Usually each output of a FU with more outputs is used by different function implemented by this FU. For example FU in figure 6.6 is the 8 bits adder with carry. In this case addition and carry are two different functions with different $CTFs$. From this reason it is necessary to attribute this factor not to FU, but to each FU output. Formula 6.12 should be in form:

$$C_{out} = CTF_{out} \times \prod_{i=1}^{n} C_{in_i},$$

where n is the number of FU inputs not connected to a constant.

For multi-output FU, each FU input can be observed by using of various FU output. From this reason we will attribute to each FU input the OTF for each component output. In other words, each FU input will be attributed the list of $OTFs$. In the example of figure 6.6, each input of adder can be observed by using either output of addition or output of carry. It is obvious that the observation of 8 bit input A or B by using of 1 bit carry output is not very effective.

For multi-outputs FU formula 6.13 should be in form:

$$O_{in_{out}} = O_{out} \times OTF_{in_{out}} \times \prod_{i=1,\, i \neq in}^{n} C_i,$$

where n is the number of FU data inputs not connected to a constant and input i is not connected to a constant.

The observability of an FU input is equal to the best observability obtained by the observation of this input from various FU outputs:

$$O_{in} = \max(O_{in_{out}}).$$

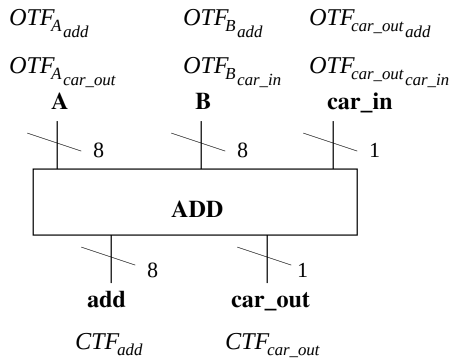


Figure 6.6: New attribution of transfer factors: each FU output will be attributed its CTF and each FU input will be attributed the list of $OTFs$.

6.6.2 Shared selection inputs mode

As was stated in chapter 5, in the current IDAT version, in order to not care about the state dependencies in original circuit controller, the testing algorithm of the data-path is independent from the original circuit controller and is driven by the separate test controller. In some cases, the area caused by the implementation of new test structures can be critical already after the first modification, so the application of related allocation for testability technique become impossible. The solution can be *shared selection inputs mode* planned for any future version of IDAT.

In the shared selection inputs mode, in order to not completely change the philosophy of our testability analysis, we will keep the concept of separate test controller. The difference will be that we will not allocate new multiplexer for the selection input of each multiplexer already existing in the data-path in order to make this multiplexer controllable in an independent way. In contrast to normal mode, we will allocate new multiplexer only to *each control signal from original circuit controller*. Usually it has the same effect as to allocate the new multiplexer for the selection input of each multiplexer already existing in data-path. However, sometimes the multiplexers selection inputs are shared as depicted in figure 6.7. In such a case, the controllabilities of the multiplexers outputs cannot always be set automatically to its best input controllability multiplied by C_{sel} (according to formula 6.8). In this case a *shared selection inputs trade-off algorithm* will be executed, in order to find the higher average output controllability value of all multiplexers sharing one selection pin (see figure 6.7).

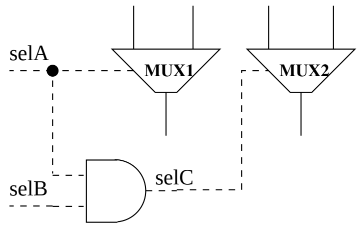


Figure 6.7: Two multiplexers with shared selection inputs. If $selA = 0$ then $selC = 0$.

6.6.3 Shared reset inputs

The shared reset inputs, connected for example as in the circuit from figure 6.8, should not cause problems to the testability analysis algorithm. However, the test scheduler will cope with more complex test synchronization task.

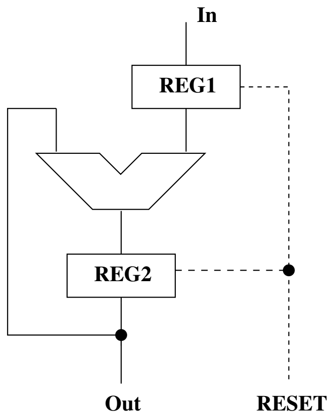


Figure 6.8: Example of a circuit with shared reset inputs.

6.6.4 Component input connected to a constant

Formula 6.14 is in fact only a simplification valid in the case where the FU is type adder or subtracter. More accurate form of this equation can be

$$C_{N3} = CTF_{ADD} \times C_{N1} \times k,$$

where k is the coefficient related to the type of the function implemented by the FU and the value of constant at the FU input.

Imagine that FU is type multiplier. Then for an even value of the constant we can observe at the FU output only even values. If the constant is equal to 0, the FU output is nearly uncontrollable.

6.6.5 Reconvergence of component inputs

An inspiration for the inclusion of a solution for reconvergence of component inputs can be found in [FPR97]. They include reconvergence definition into the definition of controllability transparency coefficient (T_C) (used for similar purpose like our CTF[8]):

$$T_C = \frac{(C_2 - C_1) \times p}{n} + C_1, \tag{6.16}$$

where

- n is the total bit width of the input ports
- p is the number of common bits between the input ports
- C_1 is the proportion of values that can be obtained on the FU output, given that its input ports are *not connected* to each other ($p = 0$)
- C_2 is the proportion of values that can be obtained on the FU output, given that its input ports are *connected* to each other ($p = n$).

6.6.6 Embedded loops

The identification of embedded loops is also a challenge for future research. In the case of embedded loops, presented testability analysis method is unable to determine the degree of nesting of certain component (all nodes inside the loops are considered as completely uncontrollable). This feature of our testability analysis can bring a problem in allocation for testability algorithm. When many embedded loops occur in the circuit, it is necessary to accurately determine the most embedded sequential component, which should be made testable first, because it best improves total circuit testability. In the circuit of figure 6.9 it is register R3.

[8] One of the differences between our CTF and T_C is that C_1 and C_2 from formula 6.16 are stored in the library like FU attributes.

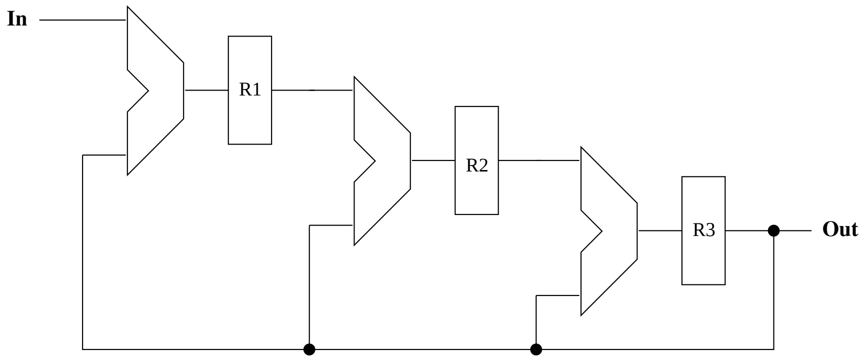


Figure 6.9: Example of a simple circuit with embedded loops.

6.7 Conclusion

The main contributions of presented testability analysis approach are:

- Introduction of metrics node testability, which cumulates the controllability and observability values of a node into one value expressing the overall testability of this node.
- The way how we make profit of the existence of constant generators in designed circuit to calculate controllability of FU outputs. For example Gu et al. and also other authors do not introduce any special treatment to cope with the existence of "constant" node.
- The introduction of controllability exigency and observability exigency metrics, which better reflects actual test plan than classical controllability and observability metrics.

6.8 Related publications

- P. Bukovjan and M. Marzouki and W. Maroufi, *Testability Analysis in High-Level Synthesis for Testability*, Proceedings from 10th European Workshop on Dependable Computing, Vienna, Austria, pp. 71-75, May 1999
- P. Bukovjan and M. Marzouki and W. Maroufi, *Testability Analysis and Cost/Quality Trade-off in Synthesis for Testability*, Compendium of Papers from IEEE European Test Workshop, Constance, Germany, May 1999

Chapter 7

Attributes

The trade-off process is 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 required *design attributes* for final circuit (area, maximal delays, etc.) and available means in the components library is performed during the allocation step of a classical HLS process, the proposed cost/quality trade-off is made between required *testability attributes* for designed circuit (area overhead, delays overheads, testability, etc.) and available means in the library of testable components during the allocation for testability process, considering also testability methodology constraints.

As was already stated in chapter 5, the trade-off process is made at two levels: at the local level (among original component and its testable versions stored in the library) and at the global level (data-path architecture). From this point of view the attributes can also be distinguished as the *local attributes* represented by the *characteristic vector* of each circuit component and the *global attributes*, which are calculated from the local attributes and then saved to the *global attributes vector*. Some of the attributes exist at both local and global level.

The definitions of all local attributes are given in section 7.1, followed by the definitions of all global attributes in section 7.2. The properties of different groups of attributes are summarized in section 7.3. The component RVA specification proposed for the next version of IDAT is provided in section 7.4.

7.1 Local attributes

Each component in Rich Virtual Architecture (RVA) specification is described by the set of design and testability attributes. These attributes, which are set by a specialist (results of synthesis of each component, etc), are quantified using generic units. An example of the component RVA specification is provided in the following lines:

```
adder
(type FU) (CTF 0.8169)(OTF 0.8681) (IDkey ADD) (area 430)
 (delays (a 50 q)(b 50 q)) (init NO) (testability 0.99))
(
 a(7 downto 0) IN DATA
 b(7 downto 0) IN DATA
 q(7 downto 0) OUT DATA
)
```

The component RVA specification proposed for the next version of IDAT can be found in section 7.4.

7.1.1 Component design attributes

- $type$ - Currently IDAT supports component types summarized in table 7.1.

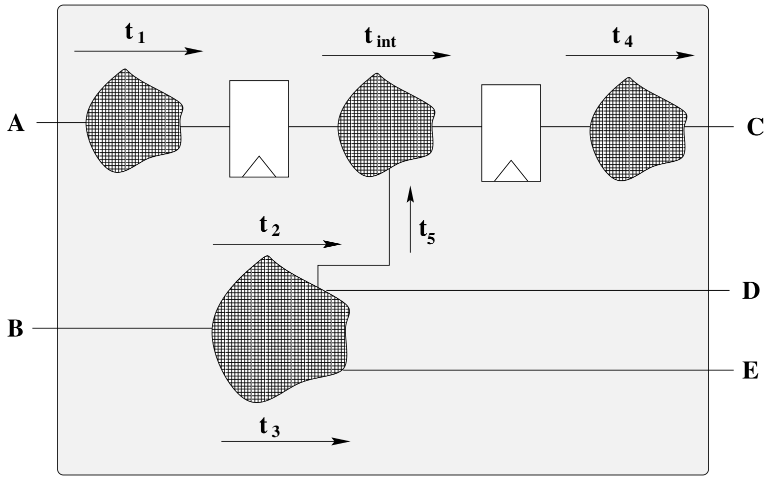
component	symbol
multiplexer	MUX
register	REG
constant generator	CONST
test pattern generator	TPG
test results checker	TRC
functional unit	FU

Table 7.1: Component types supported by IDAT.

- Component type FU has also additional attributes *controllability transfer factor* (CTF) and *observability transfer factor* (OTF). These attributes are determined by the function implemented by this FU, the number of FU I/O ports, their type and their bit width. The future attribution and the new syntax of CTF and OTF attributes are depicted in section 7.4.
- $IDkey$ - This user defined string attribute is used as the identification key to enable DFT reuse. Only library components with the same $IDkey$ as the original component can replace original non-testable component.
- $area$ - This attribute describes total component area.
- $delays$ - This attribute is the composed attribute reflecting transfer time between:
  - each input and corresponding output
  - each input and connected internal register
  - two internal registers
  - each internal register and connected output.

  An example is depicted in figure 7.1. The dark parts are some interconnected combinational block separated by registers. In fact the delay in the circuit is composed by the delays of a signal passing through such a combinational block. Returning to figure 7.1, signal entering to input B is modified by the connected combinational block and directly affects the primary outputs D and E, where the new information appears in time t_2 resp. t_3. In contrast, input A does not simultaneously affect any primary outputs, but the input of the register with the delay equal to t_1. In the following clock cycle the information from the input of the first register is transfered to the input of the second register through some combinational block with the delay t_{int}. In the same way, in the third clock cycle the output of the second register sends its data through the combination block with delay t_4 to output C.

To avoid the misunderstanding it is necessary to emphasize that not all delays between each internal nodes are listed behind keyword `delay`. Only the longest internal delay between two sequential components (between two successive clock states) is put to the list together with the delays related to primary inputs and outputs.




```
(delays t_int ((A t1)(B t2 D)(B t3 E)(B t5)(C t4)))
```

Figure 7.1: Transfer times inside an imaginary component composing its *delays* attribute.

- *init* - This attribute, which can be set either to YES or to NO, reflects the ability of a sequential component to be initialized. From the testability point of view it is better if the component can be initialized.
- *testability* - At the component level this attribute reflects the testability problems caused by component internal architecture. This attribute is used only to distinguish different testable versions in the library according to their testability (this attribute does not enter in the calculation of global attribute of the same name). It can be set for example to component fault coverage value, if this value is available. Of course, the values used to express components testability must be consistent

with components inside one library.

The component (circuit) port attributes are:

- $range$ - Described in the same style as in VHDL (with *to* and *downto*).
- $direction$ - Port directions IN, OUT and $INOUT$ are supported.
- $type$ - Supported port types are summarized in table 7.2 and are used both by the testability analysis algorithm and in order to correctly connect the testability structures when a component is replaced by its testable version.

symbol	**port type**
DATA	data I/O port
CONTROL	I/O port connected to circuit controller
CLK	clock input
SCAN_IN	input of a scan chain
SCAN_OUT	output of a scan chain
SCAN_EN	input for switching of scan components to test mode

Table 7.2: Port types supported by IDAT.

7.1.2 Component testability attributes

Component testability attributes are in addition to certain design attributes used for a testable component stored in the library of testable components.

- $test\_method$ - This attribute, reflecting test methodologies of testable component stored in the library, can be set in the current version of IDAT either to SCAN or BIST.
- $scan\_style$ - This attribute is valid only if $test\_method$ is set to SCAN and corresponds to implemented scan style. As is known, there exist various scan styles (LSSD, scan path, etc.) and not all of them are easy compatible.

7.1.3 Characteristic vector

The characteristic vector of each original non-testable component i extracted from RVA description is composed of following attributes:

$$V_i = (area,\ testability,\ delays,\ init).$$

The characteristic vector of each component version j from the library of testable components is composed of following attributes:

$$V_j = (test\_method,\ area,\ testability,\ delays,\ scan\_style,\ init).$$

7.2 Global attributes

The global attributes are calculated from the local attributes of allocated component instances, taking into account existing connections among these instances, including new allocated test structure.

The *global attributes vector* contains following circuit attributes described in the following subsection:

$$GA = (area\_over,\ testability,\ controllab,\ observab,\ full\_scan,\ delay\_over,$$
$$test\_cycles,\ untest\_nodes,\ scan\_style,\ scan\_chains,\ init,\ extraIO).$$

7.2.1 Attributes $controllab$, $observab$ and $testability$

Attributes controllability, observability and testability are defined in order to reflect the ability of a circuit to be tested. As was explained in chapter 6, the ability of a circuit to be tested does not depend only on the values of each node controllability and observability, but also on the test complexity of the components to which these nodes are connected. In the same way as in chapter 6, this complexity is modelized by the area of these components supposing that a bigger component is more complex to test than a smaller one.

The global controllability is defined as:

$$controllab = \frac{\sum_{i=1}^{n} cpx_{src_i} \times C_i}{\sum_{i=1}^{n} cpx_{src_i}} \tag{7.1}$$

$$observab = \frac{\sum_{i=1}^{n} cpx_{trg_i} \times O_i}{\sum_{i=1}^{n} cpx_{trg_i}} \tag{7.2}$$

and

$$testability = \frac{\sum_{i=1}^{n} (cpx_{src_i} + cpx_{trg_i}) \times T_i}{\sum_{i=1}^{n} (cpx_{src_i} + cpx_{trg_i})}, \tag{7.3}$$

where n is the number of all data signals in the circuit and test complexities of the connected components (cpx_{src} and cpx_{trg}) are defined in chapter 6. We do not include control signals into these calculations, because we suppose that test structures will not be allocated in order to improve the testability of these signals. Control signal are partially used to drive the test process (for example *select* signal of a multiplexer is, during the test mode, used to select the input from which test pattern will be set into its output).

7.2.2 Attribute $area\_over$

The global area is estimated by the following formula:

$$area_{global} = \sum_{i=1}^{n} area_{inst_i} + \sum_{j=1}^{m} bw_{sig_j} \times a_{sig} + \sum_{k=1}^{p} bw_{cont_k} \times a_{cont} + \sum_{l=1}^{q} bw_{pad_l} \times a_{pad},$$

where

- n is the number of allocated components instances
- bw is the bit width
- m is the number of allocated internal signals multiplied by the area coefficient for internal signals (a_{sig})
- p is the number of data-path I/O ports connected to the controller multiplied by area coefficient for circuit ports connected to the controller (a_{cont})
- q is the number of allocated data-path I/O ports connected to pads multiplied by the area coefficient for circuit pads (a_{pad}).

The above formula gives quite coarse estimation, because at RTL, circuit topology is not yet known. We have decided to distinguish not only circuit internal signals and pads, but also the signals connecting the data-path and the controller (data-path I/O ports from/to the original and test controller). The reason is that data-path internal signals routing is usually regular (for the bits of one signal) and short comparing to connections between data-path and controller, which are usually less regular and longer. From these reasons we suppose that $a_{pad} >> a_{cont} \geq a_{sig}$.

Area overhead is measured in percents and is calculated by following formula:

$$area\_over = \frac{area_{global_{modified}} - area_{global_{original}}}{area_{global_{original}}} \times 100.$$

7.2.3 Attribute $test\_cycles$

The attribute $test\_cycles$ corresponds to the total number of test cycles necessary to test the whole data-path. This attribute is estimated by the test scheduler running in parallel with IDAT. More details concerning the proposed scheduler concepts can be found in [DBBM00] and in [DB99].

7.2.4 Attribute $untest\_nodes$

The attribute $untest\_nodes$ reflects the percentage of untestable nodes in the circuit and is calculated by formula:

$$untest\_nodes = \frac{n_{untest}}{N} \times 100,$$

where n_{untest} is the number of untestable nodes in the data-path and N is total number of nodes in the circuit. An untestable node is identified according to its testability attribute (T), which is equal to 0.

7.2.5 Attributes max_delay and $delay\_over$

The internal attribute max_delay reflects the circuit critical path and is taken as maximal delay in the data-path. The delays in a data-path can be (see also the definition of local attribute $delays$ in section 7.1) the transfer times between:

- each primary input and corresponding primary output
- each primary input and connected sequential component (mostly register)
- two sequential components
- each sequential component and connected primary output.

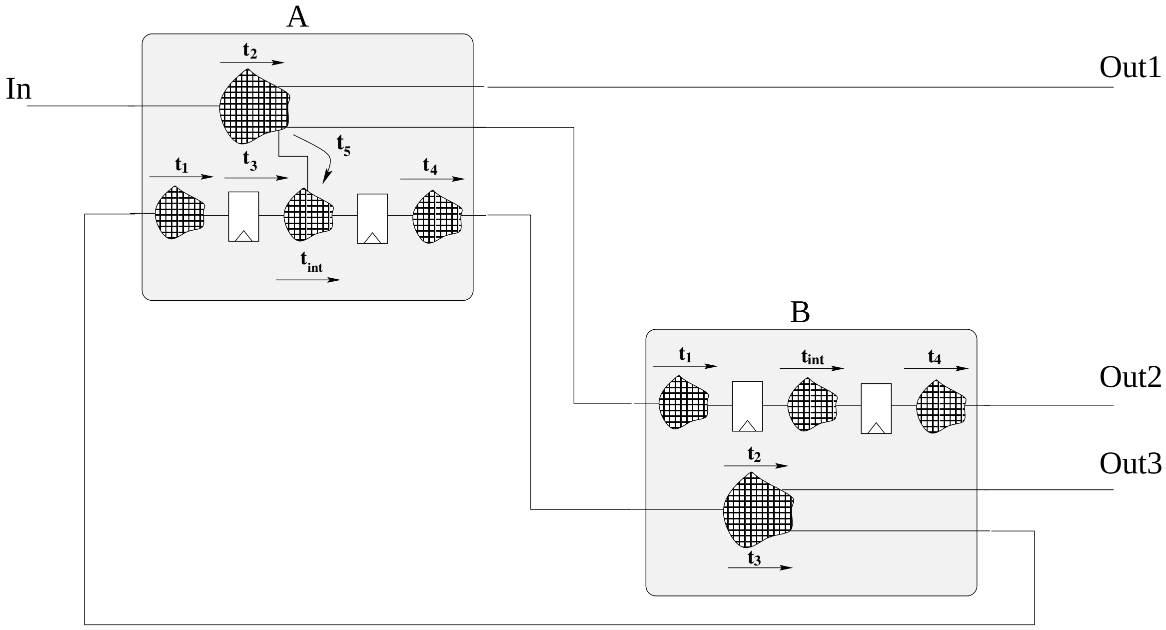


Figure 7.2: Example of a circuit with two complex interconnected components to demonstrate the calculation of the internal *global maximal delay* attribute.

These delays taken from the circuit in figure 7.2 are summarized in table 7.3. The max_delay is then the biggest one. A more realistic example is depicted in figure 7.3.

The attribute $delay\_over$ is related to internal *global maximal delay* attribute and reflects the change of the circuit critical path. This attribute is calculated by formula:

$$delay\_over = \frac{max\_delay_{modified} - max\_delay_{original}}{max\_delay_{original}} \times 100.$$

7.2.6 Attributes $full\_scan$, $scan\_style$ and $scan\_chains$

Attribute $full\_scan$ set to YES reflects that related data-path is already transformed in its test mode to the blocks of combinational circuit. This is done by conversion of all se-

From	**time**	**To**
In	t_{A_2}	Out1
In	t_{A_5}	
	$t_{A_{int}}$	
	$t_{A_4} + t_{B_3} + t_{A_1}$	
In	$t_{A_3} + t_{B_1}$	
	$t_{B_{int}}$	
	t_{B_4}	Out2
	$t_{A_4} + t_{B_2}$	Out2

Table 7.3: The delays in the circuit from figure 7.2. Each empty cell in column **From** or **To** corresponds to the connection to a register.

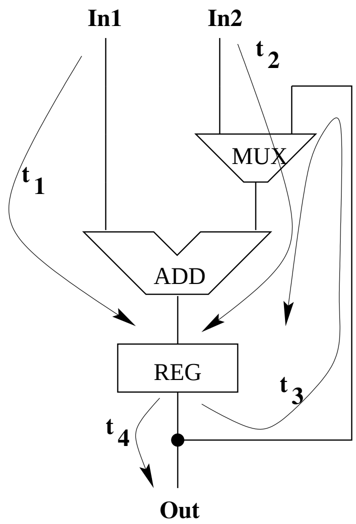


Figure 7.3: Example of a simple circuit with depicted delays.

quential components (registers, sequential FU existing in its scan equivalent) into their scan versions (full scan). Circuit transformed to full scan needs only combinational ATPG to generate test vectors, which is less complex than sequential ATPG used with a partial scan design.

Attribute $scan\_style$ corresponds to selected scan style. There exists number of scan styles (LSSD, scan path, etc.) and not all of them are easy compatible.

Attribute $scan\_chains$ reflects the number of scan chains in the circuit.

7.2.7 Attribute $extraIO$

Number of extra I/O ports caused by inserted test structures is represented by attribute $extraIO$.

7.3 Properties of attributes

The properties of circuit attributes are described in details by Zhu in [Zhu86] and this section is only modified extract from her work, although this work is concerned with a lower level of abstraction.

Each component/circuit attribute corresponds to one subset in the set of all attributes (numeric or symbolic one). To compare different values of an attribute, it is necessary to define an *ordering relation* for every subset. The ordering relation cannot be defined in the same way for all subsets. For example, for some attributes such as $testability$, a larger value is preferable, but for others such as $area\_over$, $test\_cycles$, etc., smaller values are preferred.

7.3.1 Numeric and symbolic attributes

An attribute is *numeric* if its values are in the domain of real or integer numbers. Otherwise it is a *symbolic* attribute ($scan\_style$). Symbolic attributes which only take logical values are *logical* (or boolean) attributes ($full\_scan$, $init$).

7.3.2 Upper-bound and lower-bound attributes

A numeric attribute att_i is an *upper-bound* attribute, if the requirement r_i specifies an upper bound for its value. For example $area\_over$ is an upper-bound attribute since by requiring 25%, it is implied that the actual area overhead should be no more than 25%. For an upper-bound numeric attribute a value x_1 is better than x_2 if $x_1 < x_2$. An attribute value satisfies a requirement if $att_i \leq r_i$.

A numeric attribute att_i is a *lower-bound* attribute, if the requirement r_i specifies a lower bound for its value. For example, $testability$ is a lower bound attribute, because when requesting 95%, it is implied that the actual fault testability should be no less than 95%. For a lower-bound numeric attribute, a value x_1 is better than x_2 if $x_1 > x_2$. An attribute value satisfies a requirement if $att_i \geq r_i$.

One logical value is stronger than another if the former is more favorable. A logical attribute att_i is an *upper-bound* attribute, if for att_i YES is stronger than NO. In the same way, a logical attribute att_i is a *lower-bound* attribute, if for att_i NO is stronger than YES. For example $init$ is an upper-bound attribute since it is usually more desirable to have an initializable circuit.

7.4 Future work

The component RVA specification proposed for the next version of IDAT is as follows and corresponds to the new attribution of transfer factors in the future version of IDAT described in section 6.6 in chapter 6.

```
adder_with_carry
((type FU) (IDkey ADD_CARRY)(area 430)
 (delays (a 50 q)(b 50 q)) (init NO)(testability 0.99))
(
 a(7 downto 0) IN  DATA  (OTF (q 0.8681) (cr_out 0.05))
 b(7 downto 0) IN  DATA  (OTF (q 0.8681) (cr_out 0.05))
 cr_in         IN  DATA  (OTF (q 0.99)   (cr_out 0.85))
 q(7 downto 0) OUT DATA  (CTF 0.8169)
 cr_out        OUT DATA  (CTF 0.9)
)
```

7.5 Related publication

- P. Bukovjan and M. Marzouki and W. Maroufi, *Cost/Quality Trade-off in High-Level Synthesis for Testability*, Proceedings from Design and Diagnostics of Electronic Circuits and Systems Workshop, Szczyrk, Poland, September 1998

Chapter **8**

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 into account other criteria including the mandatory requirements imposed by the chosen testability methodology.

The trade-off process searches first of all to satisfy the user requirements. If it is impossible with the existing library then a dialogue starts between our system (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 level, and the *global trade-off*, which operates at the data-path level.

This chapter is organized as follows: The trade off criteria are summarized in section 8.1, followed by the definitions of used trade-off functions in section 8.2. Section 8.3 details the local trade-off process. The various global trade-offs for test point insertion, for scan and for BIST testable architectures are explained in section 8.4.

8.1 Trade-off criteria

The criteria of proposed cost/quality trade-off process are following:

- *Local attributes* (attributes of originally allocated components and extracted components from the library). These attributes, which were detailed in chapter 7, are stored in the *characteristic vector* of each component.

- *Global attributes* (data-path attributes). These attributes, also detailed in chapter 7, are calculated for each circuit configuration and are stored in the *global attributes vector*.

- *User requirements* for the circuit are the explicit declarations of the testability and design goals for the circuit. To get what user wants in his design, he must specify what he wants. The user requirements drive the allocation for testability process. They are classified into three groups according to their importance in a component or data-path cost/quality (from the final testability point of view) ratio. This classification also corresponds to the user expertise in the field of HL-SFT and DFT: class 1 - beginner user, class 2 - advanced user and class 3 - expert user (see table 8.1). User requirements are summarized in the *user requirements vector* and weighed by corresponding *weighing factors*.

- *Weighing factors* allow the user to give his preferences, ranking his requirements by order of importance. The relative value of the $w_i^{'}$ weight is converted into normalized weights w_i using the known mathematical formula

$$w_i = \frac{w_i^{'}}{\sqrt{\sum_{j=1}^{c} w_j^{'2}}},$$

where c is the number of all user requirements related to the chosen global testability method.

- *Test methodology constraints* are imposed by selected test methodology. They act in the process of optimal components version selection and during the creation of additional test structures. In scan test methodology a test methodology constraint is for example that creation of new scan chain not only save the test time, but also brings a new area overhead imposed by new I/O pads.

requirement	Class 1	Class 2	Class 3
$area\_over$	•	•	•
$testability$	•	•	•
$full\_scan$	•	•	•
$delay\_over$		•	•
$extraIO$		•	•
$test\_cycles$		•	•
$scan\_style$			•
$scan\_chains$			•
$init$			•
$controllab$			•
$observab$			•

Table 8.1: Representation of user requirements in user requirements classes.

8.2 Trade-off functions

To accomplish the given task, our trade-off algorithm uses certain functions namely:

- *Distance Function (DF)* - for calculation of the distance between the value of an attribute and corresponding user requirement.
- *Penalty-Credit Function (PCF)* - which personalizes DFs to each attribute and normalizes the resulting values to a comparable level.
- *Score function* - calculating the score of each component version.

These functions are inspired by the work presented in [Zhu86] and [ZB88], since the same kind of problems have to be solved.

8.2.1 Distance function

The *distance function* returns the distance between the value of local attribute att_{ij} and the value of corresponding requirement r_j, where:

- index i determines the i-th component version in the set of all versions of the given component
- index j determines the j-th local testability attribute or j-th requirement.

The definition of DF depends on the properties of the corresponding attribute and the requirement. These properties were already described in chapter 7. For attributes from a characteristic vector, DF is defined as follows:

- if the value of r_j is not important then $DF(r_j, att_{ij}) = 0$
- if att_{ij} is upper-bound numeric then $DF(r_j, att_{ij}) = r_j - att_{ij}$
- if att_{ij} is lower-bound numeric then $DF(r_j, att_{ij}) = att_{ij} - r_j$
- if att_{ij} is upper-bound logical and $r_j = YES$, $att_{ij} = NO$ then $DF(r_j, att_{ij}) = -1$
- if att_{ij} is upper-bound logical and $r_j = att_{ij}$ then $DF(r_j, att_{ij}) = 0$
- if att_{ij} is upper-bound or lower-bound logical and $r_j = NO$, $att_{ij} = YES$ then $DF(r_j, att_{ij}) = 1$

Equivalent functions are valid also for attributes from a global attributes vector.

8.2.2 Penalty-Credit Function

DF, defined above, determines the distance between the requirement value and the value of corresponding attribute. In many cases, not the magnitude of the distance but the effect of the difference determines the selection. The actual effect may not be proportional to the distance. In general, there are several typical cases for an attribute value:

- value is absolutely unacceptable

- value may be accepted with penalty
- value is acceptable without penalty
- value is accepted with credit

In our case, first all attributes from each characteristic vector, then all attributes from the global attributes vector, are compared with the user requirements, using the related PCFs. A trade-off is necessary when a conflict is identified by the PCF result for the characteristic vector of each component version (resp. a global attributes vector) and the requirements vector.

The *PCF for a characteristic vector* is defined by the following formula:

$$PCF_j(r_j,\ att_{ij}) = DF(r_j,\ att_{ij}) \times sc_j,$$

where:

- index i determines the i-th component version in the set of all versions of a given component
- index j determines the j-th component attribute att resp. j-th requirement r
- $DF(r_j, att_{ij})$ - as defined above, is the distance between r_j and att_{ij} values. Positive distance gives a credit, negative distance gives a penalty or leads to unacceptable value of PCF.

In the same way the *PCF for a global attributes vector* is defined like:

$$PCF_j(r_j,\ ga_j) = DF(r_j,\ ga_j) \times sc_j,$$

where index j determines j-th global attribute ga resp. j-th requirement r.

In both formulas the *scale vector* for each requirement j (sc_j) is used to normalize DF values to a comparable level and is set by an expert.

Default PCFs for numeric attributes

For numeric attributes, the default PCF is defined below (these formulas are taken from [ZB88] and generalized by adding of tolerance coefficient). A, B, C and D are constants. These constants can be changed by an expert (see chapter 9). The graphic illustration of such a default PCF is shown in figure 8.1. For numeric attributes from a characteristic vector, the PCFs are defined in the following way:

- if $DF(r_j,\ att_{ij}) > r \times e$ then $PCF_j(r_j,\ att_{ij}) = C$
- if $0 < DF(r_j,\ att_{ij}) \leq r \times e$ then $PCF_j(r_j,\ att_{ij}) = A + \frac{C-A}{r_j \times e} \times DF(r_j,\ att_{ij})$
- if $DF(r_j,\ att_{ij}) = 0$ then $PCF_j(r_j,\ att_{ij}) = A$
- if $0 > DF(r_j,\ att_{ij}) \geq -(r \times e)$ then $PCF_j(r_j,\ att_{ij}) = B + \frac{B-D}{r_j \times e} \times DF(r_j,\ att_{ij})$

- if $DF(r_j,\ att_{ij}) < -(r \times e)$ then $PCF_j(r_j,\ att_{ij}) = E$, where $E << 0$

Factor e in above expressions is the *tolerance coefficient* ranging from 0 to 1. This coefficient is also modifiable by an expert.

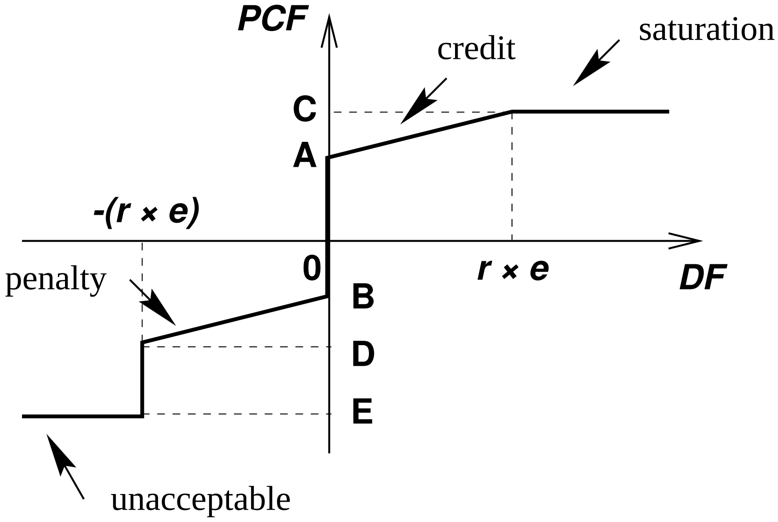


Figure 8.1: Default PCF for numeric attributes.

8.2.3 Score function

Our *cost function* called *score function* has been also adapted from [Zhu86]. The score function computes the numeric score for a component version V_i from the set of all versions of the given component, making use of PCFs results of the component attributes weighted by the corresponding weights. In the current state of work, we have adapted the *linear weight system* - the weights are constant during the whole trade-off process. In such a way the score of version V_i is defined as:

$$score(V_i) = \sum_{j=1}^{c} PCF_j(r_j,\ att_{ij}) \times w_j,$$

where

- index j determines j-th attribute corresponding to j-th requirement
- c being the number of user requirements for chosen global testability method
- w factors are the normalized weights.

8.3 Local trade-off

When a circuit data-path is extracted, IDAT starts browsing the connected library of testable components. Testable components, whose $IDkey$ corresponds to the original (non-testable) components and which test methodology attribute corresponds to required global test methodology, are extracted and put to the *container* related to corresponding original component.

During the local trade-off the attributes of original component and the attributes of each component version (stored in the corresponding characteristic vector) are compared with corresponding user requirements from the user requirements vector results in the list of all versions ordered according to the level of satisfaction of user requirements. In other words, in the local trade-off phase the ordering of versions in component containers is performed.

8.4 Global trade-off

The goal of the global trade-off and related modifications is to satisfy the user requirements in the frame of selected global testability methodology.

The global trade-off is interactive. If it is impossible to satisfy the user requirements with available resources, then a dialogue starts between IDAT and the user, who is asked to modify certain of his requirements or at least to change requirements preferences, being guided by the system.

8.4.1 Global trade-off for test point insertion

A classical test point insertion process is based on the searching of the worst controllable nodes and the worst observable nodes in the data-path, in order to improve their controllability resp. observability by direct transformation of selected points to either primary input or to the primary outputs.

The overview of our testable architecture with inserted test points was already presented in chapter 5. The main differences between classical test point insertion approach and our approach are that:

- In the test mode the data-path is isolated from the original controller. Circuit test is driven by added test controller.

- In order to reflect different test effort to test different circuit parts, we use the concept of FUs test, known from BIST, also in the test point insertion process. This is done by our allocation of testability method based on the knowledge of controllability and observability exigencies of each circuit node (see chapter 6). The controllability and observability exigencies are strongly dependent on the complexity of components to which the selected node is connected. In other words, a complex FU attracts the test point insertion to be made in its close neighborhood, when necessary.

8.4.2 Global trade-off for scan

A classical partial scan concept is based on searching the worst controllable nodes and the worst observable nodes in the circuit in order to improve their controllability resp. observability by replacing the selected registers with their scan equivalents.

The overview of scan design architecture was already presented in Chapter 5. The main differences between classical allocation for scan approach and our approach are that:

- In the test mode the data-path is isolated from the original controller. Circuit test is driven by added test controller.
- In our approach, we also take into account the possibility of existing scanable components (FUs) in the data-path. This is done by our allocation of testability method based on the knowledge of controllability and observability exigency of each circuit node. The controllability and observability exigency is strongly dependent on the complexity of components to which selected node is connected. In other words, a complex FU, which inputs are not "well enough" controllable and/or which outputs are not "well enough" observable, either attracts allocation of scan resources in its close neighborhood or is directly transformed into its scan equivalent, when such an equivalent of this FU exists.

The allocation of a scan resource is made in following steps:

1. Identify signal or primary port with the worst testability exigency.
2. Compare their controllability exigency with observability exigency.
   - $CE \geq OE$: If identified node is the input of a FU, which can be transformed into a scan version, this FU is selected. If this node is connected to the input of more FUs, which can be transformed into scan versions, the FU with the biggest test complexity is selected. In other case a register or a FU, which can be transformed into a scan version and which is driven by the identified node is searched.
   - $CE < OE$: If identified node is the output of a FU, which can be transformed into a scan version, this FU is selected. In other case a register or a FU, which can be transformed into a scan version and which drives the identified node is searched.
3. Selected component is transformed into its scan equivalent and appropriately connected into the data-path.

The decision on how many scan chains will be created depends on the trade-off between required maximal number of test cycles, minimal area overhead and maximal number of additional I/Os.

8.4.3 Example of scan branch-and-bound session

Let us consider the circuit of figure 8.2. Scan equivalents exist in the library for both FUs. FU1 area in scan version is bigger than FU2 area in scan version. Requirement for circuit testability is more than 90% and for area overhead is less than 11%. The branch-and-bound steps are depicted in table 8.2.

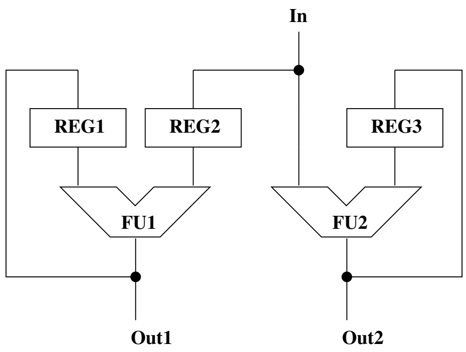


Figure 8.2: Circuit example for scan branch-and-bound session.

step	executed operation	testability	area overhead	FU1	FU2	REG1	REG2	REG3
1	replace FU1	<u>80%</u>	10 %	×				
2	replace FU2	99%	<u>20%</u>	×	×			
3	return FU2 to its original	<u>80%</u>	10 %	×				
4	replace REG3	<u>85%</u>	<u>12%</u>	×				×
5	return REG3 to its original	<u>80%</u>	10 %	×				
6	return FU1 to its original	<u>50%</u>	0 %					
7	replace REG1	<u>60%</u>	2 %			×		
8	replace FU2	90%	10 %		×	×		

Table 8.2: Example of scan branch-and-bound session. Underlined values do not correspond to user requirements. Symbol × corresponds to the components transformed into their scan versions.

This example demonstrates the situation when, despite existence of FU1 in scan version, it is more advantageous, from the final area overhead point of view, to test this FU by other means. This test is performed by transforming REG1 into a scan register, leading to a very small area overhead, and by using primary input In. The problem is that sometimes, when tested FU contains many closed loops, such an "external" (with respect to the FU) test is not very efficient and it is necessary, in order to fulfill the testability requirement, to allocate an FU in its scan version.

8.4.4 Global trade-off for BIST

A typical BIST concept is based on using either LFSR or BILBO like TPG and MISR (usually with a ROM to store compacted correct signatures) or again BILBO like TRC to test complex FUs.

The overview of our BIST design architecture was already presented in chapter 5. In our approach we combine above classical BIST concept with the scan path approach. The scan chain starting from TPG and ending in TRC is used to test the interconnections not covered by FUs test.

Our TPG and TRC are reconfigurable. This means that the number of TPG data outputs is created according to the FU with maximal number of inputs, which is required to be tested from this TPG. In the same way the number of TRC data inputs is created according to the FU to test by using of this TRC with maximal number of data outputs. More details concerning our TPG and TRC internal architecture can be found in [DB99].

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. An example of test paths for one FU is depicted in figure 8.3.

  In figure 8.3, the selected FU to test is MUL1, and together with the test of MUL1 also the following interconnections are tested: TMUX1 $\rightarrow$ REG1, REG1 $\rightarrow$ MUX2, MUX2 $\rightarrow$ MUL1, TMUX2 $\rightarrow$ REG4, REG4 $\rightarrow$ MUX1 and MUX1 $\rightarrow$ 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 but 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.

Example 1

The first trade-off, among all FUs in order to select the set of FUs for which the test will have maximal effect to overall circuit testability, is demonstrated in the following example. Let us have a circuit with four FUs summarized in table 8.3. None of them has a BIST version, so each of them, when selected to be tested, must be connected to TPG and TRC. User requirements are following: testability more than 80%, area overhead less than 20%. The branch-and-bound algorithm steps are depicted in table 8.4.

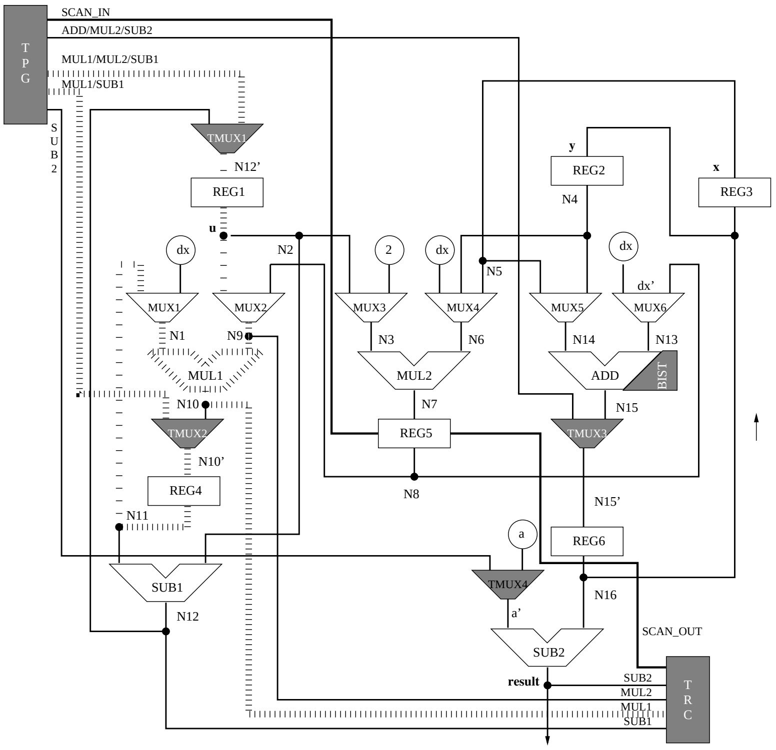


Figure 8.3: Data-path of the circuit from figure 5.5 with emphasized test paths to test MUL1.

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

Table 8.3: List of FUs originally allocated in the data-path from example 1.

BBA step	executed operation	% of testability	% of area overhead	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 8.4: The branch-and-bound algorithm steps for example 1. Underlined values does not correspond to user requirements.

The result of the branch-and-bound algorithm execution are: FU1, FU3 and FU4 connected to the TPG and TRC. 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 connected FU bit-widths), resulting in an unsatisfactory total area overhead.

Example 2

The second trade-off, between possibilities to either transform original FU into its BIST version, if such a version exists, or to test original FU by connection to TPG and TRC, is demonstrated in the following example. Let us have a circuit with three FUs summarized in table 8.5. FU3 has in library its BIST equivalent. User requirements are the following: testability more than 80%, area overhead less than 20%. The branch-and-bound algorithm steps are depicted in table 8.6.

FU name	% of global area	input #	output #
FU1	40	16	8
FU2	20	16	8
FU3	15	16	8

Table 8.5: List of FUs originally allocated in the data-path from example 2.

step	executed operation	testability	area overhead	test time	FU1	FU2	FU3
1	create TPG and TRC	60%	5%	40			
2	connect FU1	70%	8%	10	•		
3	connect FU2	75%	10%	20	•	•	
4	replace FU3	79%	21%	15	•	•	×
5	return to original FU3	75%	10%	20	•	•	
6	connect FU3	80%	12%	25	•	•	•

Table 8.6: The branch-and-bound algorithm steps for example 2. Underlined values does not correspond to user requirements. Symbol • refers to the FUs connected to TPG and TRC and symbol × corresponds to the FU transformed into its BIST version.

This example shows the situation when, in spite of existence of FU3 in its BIST version, it is more advantageous, from the final area overhead point of view, to test this FU by connection to TPG and TRC. The problem is that sometimes, when tested FU contains many close loops, the pseudo-random test performed by TPG in the form of LFSR together with TRC in the form of MISR is not very efficient. To solve this problem it is necessary, in order to fulfill the requirement to testability, either to use deterministic TPG, which cause much more bigger area overhead than LFSR with MISR, or to allocate related FU in its BIST version.

8.5 Related publications

- P. Bukovjan and L. Ducerf Bourbon and M. Marzouki, *Cost/Quality Trade-Off in Synthesis for BIST*, 1st IEEE Latin America Test Workshop, Rio de Janeiro, Brazil, pp. 110-115, March 2000
- P. Bukovjan and L. Ducerf Bourbon and M. Marzouki, *Cost/Quality Trade-Off in Synthesis for Scan*, Proceedings from Design and Diagnostics of Electronic Circuits and Systems Workshop, Smolenica, Slovakia, pp. 30-33, April 2000

Chapter 9

IDAT Tool

Our allocation for testability approach presented in previous chapters is implemented into *Interactive Design for Test reuse in Allocation for Testability* (IDAT) tool. This tool takes a part of our allocation for testability environment composed of three tools: IDAT tool, Test Scheduler (TesS) and generator of test structures for BIST (BSG). For more details concerning the integration of IDAT to a global test environment see [DBBM00].

IDAT transforms circuit data-path into its testable equivalent making use of the library of components with already implemented testability structures. IDAT collaborates with TesS, in order to complement the testable data-path with the test controller, and calls BSG to generates dynamically reconfigurable TPG and TRC when BIST architecture of designed circuit is required.

The whole allocation for testability environment is not strictly related to certain HLS system, but in order to test the performance of our approach we have decided to attach it to UGH (User-Guided High-Level) synthesis system. UGH is currently under development in the framework of the European ESPRIT project COSY/OMI and will be included into ALLIANCE VLSI CAD System [Ali99] created in our laboratory (LIP6).

The overview of IDAT directories is provided in section 9.1. The necessary steps to install IDAT are described in section 9.2 followed by section 9.3, where all possible options and all input and output files are described. Some typical session examples are presented in section 9.4.

9.1 IDAT directories

IDAT is originally designed for UNIX platform but its current version can be easily modified to be compilable also on any other platform. In order to be open to possible modifications, IDAT is delivered in the form of ASCII standard C++ source code including corresponding makefiles. IDAT package also includes sets of input files including examples of user requirements for test point, scan and BIST final architecture, few RVA examples, simple library of testable components and default technology file. The main directory `IDAT` contains following subdirectories:

- `bin` - to store binary executable files of IDAT
- `doc` - to store IDAT documentation
- `examples` - to store RVA examples. This directory is usually used as IDAT working directory, where also resulting files are stored.
- `idat` - to store the main source file
- `mkutils` - to store some scripts that generate makefiles and configuration files for program compiling
- `sce` - to store C++ source code of all IDAT classes
- `tools` - to store generators of TPG and TRC
- `utils` - to store library files (`.lib`) and files containing user requirements (`.req`)

9.2 Installation

IDAT tool is delivered in `idat.tgz` file. To install it, it is necessary to perform the following steps:

1. Select directory where `IDAT` subdirectory should be created.
2. `gunzip idat.tgz` followed by `tar -xvf idat.tar` to unpack the whole system into desired directory.
3. In `IDAT` directory enter `SetUp` to start the compilation process.
4. For frequent users of IDAT: include the path towards executable file `IDAT/bin/idat*` into `PATH` variable in the corresponding shell file.

In the future version of IDAT we plan to simplify this installation process.

9.3 Running

When IDAT is successfully compiled, it can be run (usually from directory `examples`) by command:

```
idat [option(s)] <inputCircuitName>.rva [<techFileName>.tec]
```

The default technology file name is `default.tec`.

9.3.1 Options

It is supposed that the future version of IDAT will be driven by a graphical interface. In order to have IDAT core as independent as possible from this graphical interface, IDAT communicates with its environment by the set of files and by the set of options set when IDAT is started. The possible options, which are listed when `idat` command with no attributes is entered, are the following:

- `-o`: Information *about the program* version, authors and contact address will be printed.
- IDAT can work in two modes, automatic and manual:
  - In the *automatic mode*, which is set as default, the whole allocation for testability process is performed automatically. User is asked for help only when it is impossible to satisfy his requirements with available means and it is necessary to change some of his requirements or to change requirements preferences.
  - In the *manual allocation for testability mode*, selected by option `-m`, IDAT only proposes solutions. The final decision is left to the user, who can accept proposed solution or select the another one. This mode is designated for more experienced users, or for learning purposes.
- The options concerning required global test methodology are:
  - `-t` only calculation of global attributes including testability analysis of original circuit is performed
  - `-p` for test point insertion
  - `-s` for scan
  - `-b` for BIST
  - Option `-f` (full scan) invokes transformation of all sequential components into their scan equivalent, if this equivalent exist in the library. The trade-off is made only between the test time (depended on the number of scan chains) and the area overhead imposed by inserted new test primary ports.[1]

  Of course only one of these options can be selected. When any of above options is entered, user is asked to enter any.
- The default user level is beginner. To change it, it is possible to use either `-a` option for the advanced user or `-e` option for the expert user.
- `-x`: This option invokes the output of testability analysis results in the format of LaTeX table.
- `-d`: If this option is selected, the debug mode for IDAT is executed. This option is often followed by specification of C++ classes to debug. The detail description of this mode is out of scope of this document.

[1] In the current version of IDAT, the option for full scan is not yet implemented.

- `-g`: This option will be used by the IDAT graphical interface in order to re-direct the output text messages into this interface.

9.3.2 Input files

The following files compose the input to IDAT:

- *RVA circuit description* (`.rva`) described in chapter 5. The RVA descriptions of one circuit data-path is provided in appendix B.
- *Technology file* (`.tec`) necessary to calculate certain global attributes. In the current version of IDAT the file contains area coefficients (a_{sig}, a_{cont} and a_{pad}) described in chapter 7 and "breaking" points for PCFs of each global attribute.
- *User requirements* (`.req`) described in chapter 8.
- *Testable components library* is composed from the following files.
  - `Arith.lib` - containing the interfaces of all arithmetic-logic units (ALUs)
  - `Components.lib` - contains the interfaces of all other components different from ALUs: registers and multiplexers
  - `SynthesisParam.lib` - contains for each component from `Arith.lib` and `Components.lib` the list of selected attributes obtained from synthesis process
  - `FunctionalParam.lib` - contains for each component from `Arith.lib` and `Components.lib` the list of selected attributes depending on the function implemented by each component or on the attributes related to testability features of each component

9.3.3 Output files

The following files compose the output from IDAT:

- *VHDL description of original data-path* (`.vhd`) is only simple transformation of the structural part of input RVA description into the VHDL format.
- *VHDL description of modified data-path* (`.vhd`) is the main IDAT output file containing modified data-path including new allocated test structures.
- *List of all data-path modifications* (`.chg`) containing transformations of instances, insertions of new instances, components, primary ports and signals together with the list of created scan chains etc.
- *Global attributes of original data-path* (`.gat`) file summarizes the calculated attributes of original data-path. This file can give to the user the first idea about the requirements values ranges.

- *Global attributes of modified data-path* (.gat) files summarize all the attributes of the final data-path.
- *Testability analysis results of original data-path*, where the original controller is separated from the test process (.cot) file contains for each data-path node its testability attributes values. When the option -x is selected, when IDAT started, these results are saved also in the format of LaTeX table (.tex).
- *Testability analysis results of modified data-path* (.cot) file has the same format as the testability analysis results of original data-path. When the option -x is selected these results are saved also in the format of LaTeX table (.tex).

9.4 Session examples

In the following session examples, IDAT will be demonstrated in its testability analysis mode and automatic allocation for testability mode. The manual allocation for testability mode is very similar to the automatic allocation for testability mode. The difference is that in manual mode all data-path modifications performed by the tool must be confirmed by the user, who can change the IDAT decision.

These demonstrations will be made on Diffeq benchmark taken from [PKG86]. Diffeq benchmark is originally described at the behavioral level. We have extracted its RTL architecture from [PKG86]. This simplified RTL, which is the result from HLS system HAL (described in the same article), modified by adding necessary labels to components and signals and converting the output comparator to subtractor SUB2, is depicted in figure 9.1.

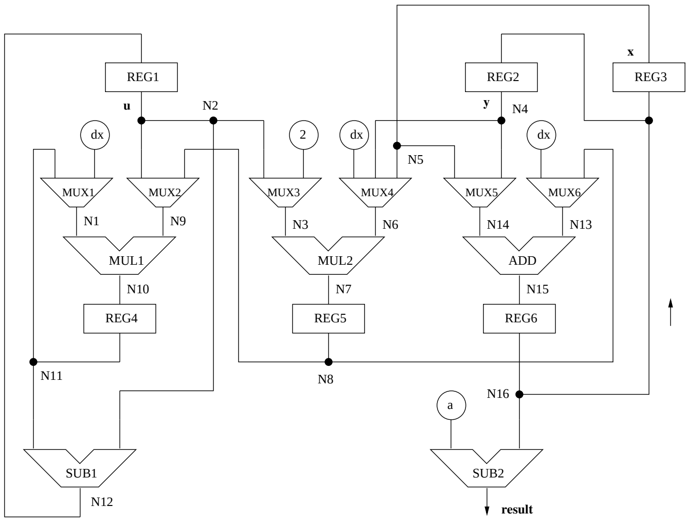


Figure 9.1: Diffeq data-path generated by HAL.

The results of the following presented sessions are analyzed in chapter 10.

9.4.1 Calculation of global attributes

To start calculation of global attributes process for data-path Diffeq we have used command:

idat -t diffeq.rva

which produces following short output:

```
-------------------------
| Welcome to IDAT system |
-------------------------

Circuit diffeq was successfully extracted from diffeq.rva file.

VHDL description of circuit diffeq was saved to diffeq_orig.vhd file.

Technology coeficients were initialized from ../utils/default.tec file.

Testability analysis results were saved to diffeq_orig.cot file.

Global attributes were saved to diffeq_orig.gat file.

Goodbye.
```

The testability analysis results stored in `tseng_orig.cot` and `tseng_orig.gat` files are analyzed in chapter 10.

9.4.2 Automatic allocation for testability

The automatic allocation for testability process is demonstrated for BIST methodology. To start automatic allocation for data-path Diffeq we have used command:

idat -b diffeq.rva

which produces following output:

```
-------------------------
| Welcome to IDAT system |
-------------------------

Circuit diffeq was successfully extracted from diffeq.rva file.

VHDL description of circuit diffeq was saved to diffeq_orig.vhd file.

Technology coeficients were initialized from ../utils/default.tec file.

User requirements and weights were extracted from:
/home/bukovjan/IDAT/bin/../utils/diffeqBIST.req

Weighed requirements to BIST circuit diffeq:
```

```
testability                79.00 %          8
area overhead              24.00 %          7
untestable nodes           10.00 %          4

List of the original components for which corresponding
version(s) were extracted from connected library:
- CompCont_register_d : 1 corresponding version(s)

In order to isolate original circuit controller from circuit
under test and connect this circuit to new created test controller
a new multiplexer is allocated to each control signal.

-----------------------------------------------------------------------
Estimated global attributes ordered according to your preferences:
testability                 0.00 %    <- !!!
area overhead               6.96 %
untestable nodes          100.00 %    <- !!!

The worst global failing attribute is testability.

Global attributes were saved to diffeq_orig.gat file.

VHDL description of circuit diffeq was saved to diffeq_orig.vhd file.

IDAT decision: connection of instance mul1 to TPG and to TRC
by the following way:
Input n1 is by signal n10 connected to TPG.
Input n9 is by signal n12 connected to TPG.
Output n10 is directly connected to TRC.

-----------------------------------------------------------------------
Estimated global attributes ordered according to your preferences:
testability                34.50 %    <- !!!
area overhead               9.28 %
untestable nodes           75.00 %    <- !!!

The worst global failing attribute is testability.

IDAT decision: connection of instance mul2 to TPG and to TRC
by the following way:
Input n3 is already by signal n12 connected to TPG.
Input n6 is by signal n15 connected to TPG.
Output n7 is by signal n13 connected to TRC.

-----------------------------------------------------------------------
Estimated global attributes ordered according to your preferences:
testability                74.70 %    <- !!!
area overhead              10.44 %
untestable nodes           26.09 %    <- !!!

The worst global failing attribute is untestable nodes.

IDAT decision: connection of instance sub2 to TPG and to TRC
by the following way:
Input a is directly connected to TPG.
Input n16 is already by signal n15 connected to TPG.
Output r is directly connected to TRC.

-----------------------------------------------------------------------
Estimated global attributes ordered according to your preferences:
testability                76.97 %    <- !!!
area overhead              11.60 %
untestable nodes           24.49 %    <- !!!

The worst global failing attribute is untestable nodes.
```

```
IDAT decision: connection of instance sub1 to TPG and to TRC
by the following way:
Input n11 is already by signal n10 connected to TPG.
Input n2 is already by signal n12 connected to TPG.
Output n12 is directly connected to TRC.

-----------------------------------------------------------------------
Estimated global attributes ordered according to your preferences:
testability                77.66 %    <- !!!
area overhead              11.60 %
untestable nodes           24.49 %    <- !!!

The worst global failing attribute is untestable nodes.

IDAT decision: connection of instance add to TPG and to TRC
by the following way:
Input n14 is already by signal n15 connected to TPG.
Input n13 is by signal dx connected to TPG.
Output n15 is directly connected to TRC.

-----------------------------------------------------------------------
Estimated global attributes ordered according to your preferences:
testability                79.93 %
area overhead              12.76 %
untestable nodes            7.69 %

Circuit diffeq has no failing attribute.

Global attributes were saved to diffeqBIST.gat file.

Testability analysis results were saved to diffeqBIST.cot file.

VHDL description of circuit diffeq was saved to diffeqBIST.vhd file.

Changes in circuit diffeq was saved to diffeqBIST.chg file.

Goodbye.
```

Chapter **10**

Experimental Results

This chapter presents the experimental results obtained by application of our allocation for testability system IDAT to three data-path benchmarks. For each data-path all possible testability architectures are created and evaluated.

This chapter is organized as follows: section 10.1, demonstrates the considerable improvement of the cost/quality relation for the circuit with many hard-to-test parts - benchmark Diffeq. The resulting global attributes of Tseng circuit from IDAT sessions presented in chapter 9 are summarized and analyzed in section 10.2, For the same data-path allocated for scan the IDAT testability values are compared with the real values of the fault coverage obtained by using Mentor Graphics $DFTAdvisor$ and $FlexTest$ tools described in chapter 4. Section 10.3 summarizes the results of application of our testability analysis techniques to Bert data-path. Finally, further results are provided in section 10.4. Contrarily to experimental results provided in the first three sections of this chapter, where the case studies are well chosen to demonstrate the whole capability of IDAT tool, these further results show the general evolution of testability increase which can be obtained with IDAT, with respect to area overhead evolution.

In each case, configuration 0, created by the earliest step of each allocation for testability methodology, corresponds to the original data-path with the allocated set of multiplexers separating, in the test mode, this data-path from the original controller (see figure 5.2).

Although the considered circuits are often used for benchmarking, it has been difficult to compare our results with those reported in [AM94], [Gup94], [PR94], [WD96] and [Lee97] in terms of fault coverage and test generation time. One reason is that we do not 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 these benchmarks, and it has been checked that the resulting data-paths, where components are re-allocated under global testability constraints, are actually more testable that the original data-paths provided by the HLS system without considering testability issues and that the overall obtained results are very satisfactory.

10.1 Diffeq benchmark

The origin of Diffeq benchmark was already described in chapter 9 and its RTL architecture is depicted in figure 9.1.

10.1.1 Allocation for external test

As can be observed from testability analysis results summarized in table 10.1, all Diffeq data-path nodes are initially untestable.

Test point insertion

User requirement for testability is more than 75%, for area overhead less than 12% and for maximal number of untestable nodes 0%. The selected global attributes for each Diffeq data-path modification step are depicted in the diagram of figure 10.1(a). Performed modifications are as follows:

1. N10 is made primary output
2. N16 is made primary input
3. N9 is made primary input
4. N7 is made primary output
5. N2 is made primary input.

Allocation for scan

User requirement for testability is more than 80%, for area overhead less than 25% and for maximal number of untestable nodes 0%. The selected global attributes for each Diffeq data-path modification step are depicted in the diagram of figure 10.1(b). Following registers are subsequently transformed into their scan equivalents and connected to the same scan chain: REG4, REG6, REG1 and REG5.

Kotásek et al. in [KRH00] obtained similar results by applying their formal approach to the same benchmark. They have selected either set REG1, REG4 and REG6 or set REG1, REG3 and REG4 as the best candidates for scan transformation.

Node	C	O	T	CE	OE	TE
a	1	0	0	0	0	0
dx	1	0	0	0	0	0
clk	1	0	0	0	0	0
sel1	1	0	0	0	0	0
sel2	1	0	0	0	0	0
sel3	1	0	0	0	0	0
sel4	1	0	0	0	0	0
sel5	1	0	0	0	0	0
sel6	1	0	0	0	0	0
r	0	1	0	0	0	0
sel_test	1	0	0	0	0	0
sel1_TEST	1	0	0	0	0	0
sel2_TEST	1	0	0	0	0	0
sel3_TEST	1	0	0	0	0	0
sel4_TEST	1	0	0	0	0	0
sel5_TEST	1	0	0	0	0	0
sel6_TEST	1	0	0	0	0	0
const	0	0	0	0	25000.0	25000.0
n1	1	0	0	0	14600.0	14600.0
n2	0	0	0	76200.0	28200.0	104400.0
n3	0	0	0	0	14600.0	14600.0
n4	0	0.754	0	39300.0	6948.5	46248.5
n5	0	0.754	0	39300.0	6948.5	46248.5
n6	1	0	0	0	24700.0	24700.0
n7	0.768	0	0	6542.4	413800.0	420342.4
n8	0.768	0	0	6774.4	28200.0	34974.4
n9	0.768	0	0	96001.6	14600.0	110601.6
n10	0.590	0	0	11567.0	413800.0	425367.0
n11	0.590	0	0	25266.8	28200.0	53466.8
n12	0	0	0	28200.0	47000.0	75200.0
n13	1	0	0	0	14600.0	14600.0
n14	0	0.754	0	43000.0	3597.5	46597.5
n15	0	0.868	0	28200.0	5671.7	33871.7
n16	0	0.868	0	103400.0	3719.6	107119.6
sel1′	1	0	0	0	14600.0	14600.0
sel2′	1	0	0	0	14600.0	14600.0
sel3′	1	0	0	0	14600.0	14600.0
sel4′	1	0	0	0	14600.0	14600.0
sel5′	1	0	0	0	14600.0	14600.0
sel6′	1	0	0	0	14600.0	14600.0

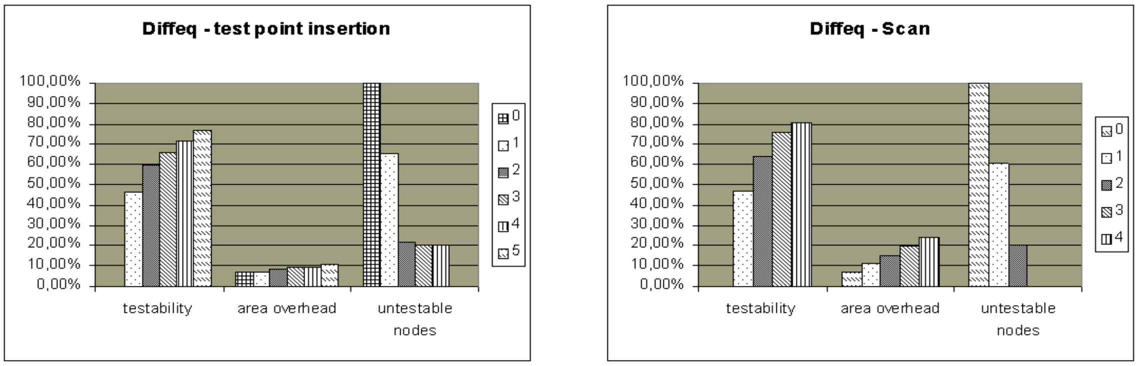
Table 10.1: Testability analysis results for original circuit diffeq supposing external test.

10.1.2 Allocation for BIST

User requirement for testability is more than 79%, for area overhead less than 24% and for maximal number of untestable nodes less than 15%. The selected global attributes for each modification step performed by IDAT are depicted in the diagram of figure 10.2. Performed modifications are as follows:

1. MUL1 selected: input N1 is by signal N10 connected to TPG, input N9 is by signal N12 connected to TPG, output N10 is directly connected to TRC,

2. MUL2 selected: input N3 is by signal N12 already connected to TPG, input N6 is by signal N15 connected to TPG, output N13 is directly connected to TRC,



(a) (b)

Figure 10.1: Selected resulting attributes for different data-path configurations of benchmark Diffeq (a) with inserted test points, (b) allocated for scan.

3. SUB2 selected: primary input IN2 is directly connected to TPG, input N15 is by signal N15 already connected to TPG, primary output OUT is directly connected to TRC,

4. SUB1 selected: input N11 is by signal N10 already connected to TPG, input N2 is by signal N12 already connected to TPG, output N12 is directly connected to TRC,

5. ADD selected: input N14 is by signal N15 already connected to TPG, input N13 is by primary input IN1 connected to TPG, output N15 is directly connected to TRC.

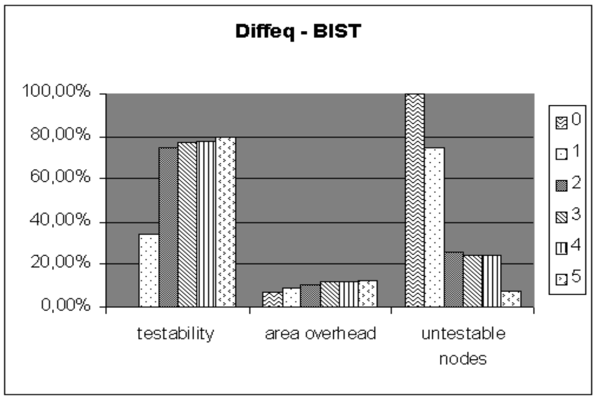


Figure 10.2: Selected resulting attributes for different data-path configurations of benchmark Diffeq allocated for BIST.

10.2 Tseng benchmark

Tseng is originally described at the behavioral level, while the input of IDAT is an RVA architecture. We have extracted the data-path architecture from a simplified RTL one from [PKG86] (see figure 10.3) and modified it by adding the necessary labels to components and nodes. In order to obtain the RVA description we have enriched this simplified architecture by required design and testability attributes.

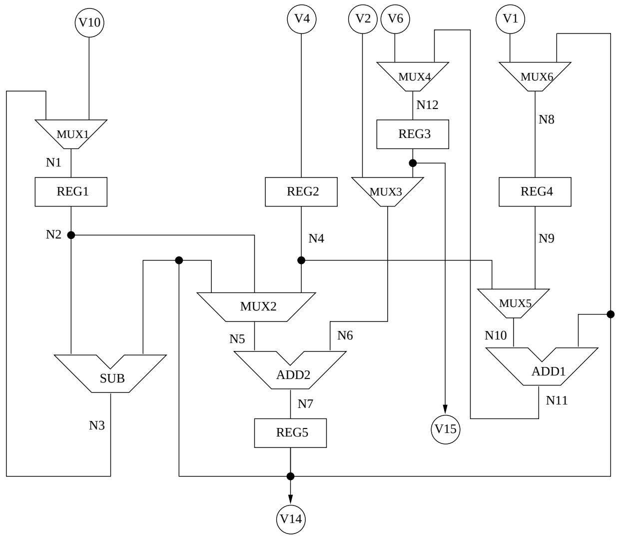


Figure 10.3: Data-path architecture of Tseng benchmark.

10.2.1 Allocation for external test

As can be observed from testability analysis results summarized in table 10.2, for an external test, there are no untestable nodes in original circuit Tseng.

Test point insertion

User requirement for testability is more than 90% and for area overhead less than 40%. The selected global attributes for each Tseng data-path modification step are depicted in the diagram of figure 10.4(a). Performed modifications are as follows:

1. V14 is made primary input
2. N3 is made primary output
3. N7 is made primary input
4. N2 is made primary output
5. N4 is made primary output.

Node	C	O	T	CE	OE	TE
V10	1	0.868	0.868	0	0	0
V4	1	0.868	0.868	0	0	0
V2	1	0.868	0.868	0	0	0
V6	1	1	1	0	0	0
V1	1	0.709	0.709	0	0	0
V14	0.817	1	0.817	23674.8	0	23674.8
V15	1	1	1	0	0	0
sel1	1	0.579	0.579	0	0	0
sel2	1	0.709	0.709	0	0	0
sel3	1	0.868	0.868	0	0	0
sel4	1	0.667	0.667	0	0	0
sel5	1	0.709	0.709	0	0	0
sel6	1	0.579	0.579	0	0	0
clk	1	1	1	0	0	0
sel_test	1	0.868	0.868	0	0	0
sel1_TEST	1	0.579	0.579	0	0	0
sel2_TEST	1	0.709	0.709	0	0	0
sel3_TEST	1	0.868	0.868	0	0	0
sel4_TEST	1	0.667	0.667	0	0	0
sel5_TEST	1	0.709	0.709	0	0	0
sel6_TEST	1	0.579	0.579	0	0	0
n1	1	0.868	0.868	0	1925.7	1925.7
n2	1	0.868	0.868	0	3719.6	3719.6
n3	0.667	0.868	0.579	4867.8	6199.3	11067.1
n4	1	0.868	0.868	0	3719.6	3719.6
n5	1	0.868	0.868	0	3257.9	3257.9
n6	1	0.868	0.868	0	1925.7	1925.7
n7	0.817	1	0.817	5163.4	0	5163.4
n8	1	0.709	0.709	0	4246.4	4246.4
n9	1	0.709	0.709	0	8201.9	8201.9
n10	1	0.709	0.709	0	4246.4	4246.4
n11	0.667	1	0.667	4857.0	0	4857.0
n12	1	1	1	0	0	0
sel1′	1	0.579	0.579	0	6151.5	6151.5
sel2′	1	0.709	0.709	0	4246.4	4246.4
sel3′	1	0.868	0.868	0	1925.7	1925.7
sel4′	1	0.667	0.667	0	4857.0	4857.0
sel5′	1	0.709	0.709	0	4246.4	4246.4
sel6′	1	0.579	0.579	0	6142.1	6142.1

Table 10.2: Testability analysis results for original circuit tseng supposing external test.

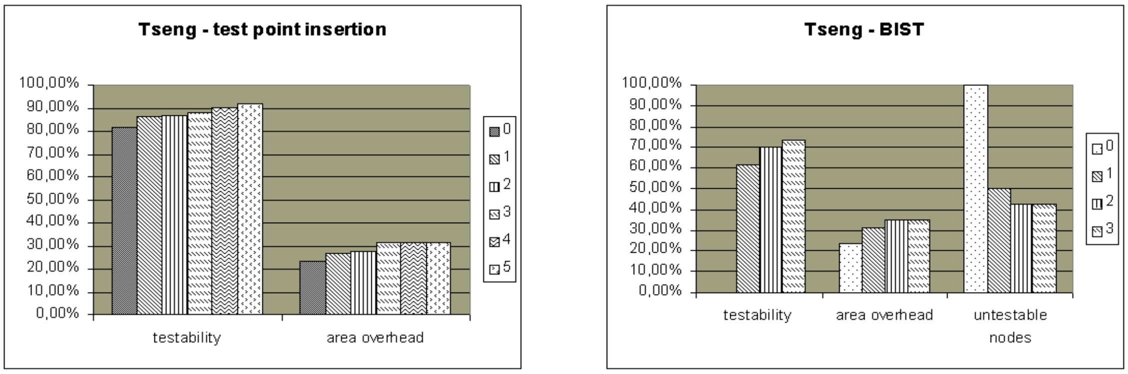
Allocation for scan

The selected global attributes for each Tseng data-path modification step are depicted in the diagram of figure 10.5(a). User requirement for testability is more than 89% and for area overhead less than 60%. Performed modifications are:

1. register REG5 is transformed to its scan equivalent and connected to the new created scan chain

2. registers REG1 is transformed to its scan equivalent and connected to the same scan chain.

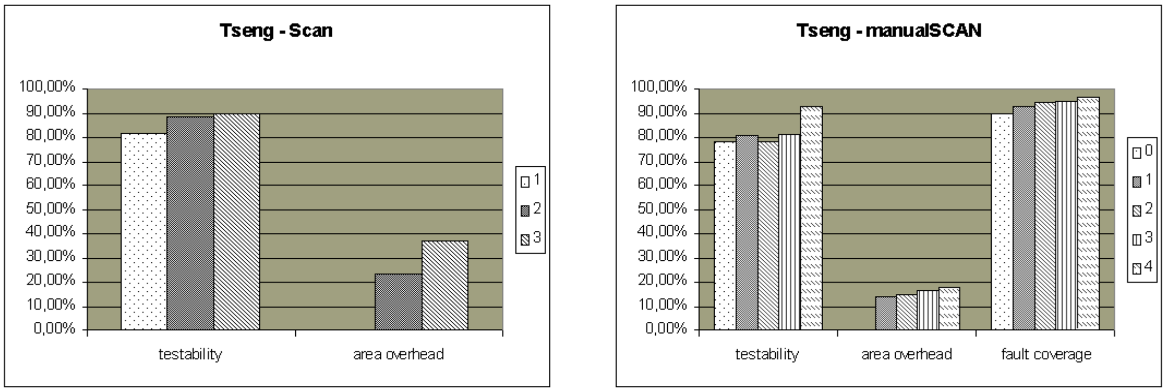
In order to compare our circuit testability attribute with a real fault coverage, we have decided to select manually certain registers to be converted into their scan equiv-



(a) (b)

Figure 10.4: Selected resulting attributes for different data-path configurations of benchmark Tseng (a) with inserted test points, (b) allocated for BIST.

alents and observe the correlation between calculated testability values and measured fault coverage values. The evolution of global attributes after each data-path modification is depicted in diagram of figure 10.5(b). Configurations from 1 to 4 correspond to the data-path, where registers REG5, REG2, REG4 and REG1 are successively added to one scan chain.



(a) (b)

Figure 10.5: Selected resulting attributes for different data-path configurations of benchmark Tseng allocated for scan (a) automatically with IDAT, (b) manually.

According to our expectation the values of *fault coverage*, obtained by using of Mentor Graphics $DFTAdvisor$ and $FlexTest$ tools, improve with increasing number of registers connected to the scan chain. The correlation between selected testability metrics calculated by IDAT and fault coverage obtained by using $FlexTest$ is 96.23 %.

10.2.2 Allocation for BIST

For circuit Tseng the whole automatic allocation for BIST session is shown in chapter 9. User requirement for testability is more than 73%, for area overhead less than 35% and for maximal number of untestable nodes less than 45%. The evolution of global attributes after each modification is depicted in diagram of figure 10.4(b). The created configurations are the following:

1. SUB selected: for input N2 primary input V10 is connected to TPG, for input V14 signal N7 is connected to TPG, output N3 is directly connected to TRC,

2. ADD2 selected: for input N5 signal N7 is already connected to TPG, for input N6 primary input V6 is connected to TPG, output N7 is directly connected to TRC,

3. ADD1 selected: for input N10 and input V14 signal N7 is already connected to TPG, there is no direct reconvergence because signal N7 is connected through one register (REG5) to V14 and through two registers (REG5 and REG4) to input N10 - different sequential depth, for input N11 signal N6 is connected to TRC.

10.3 Circuit Bert

The RTL data-path example Bert is taken from [BFR98] and is reprinted in figure 10.6 adding necessary label to the nodes. In order to obtain the RVA description of this data-path we have enriched the extracted RTL architecture by required design and testability attributes.

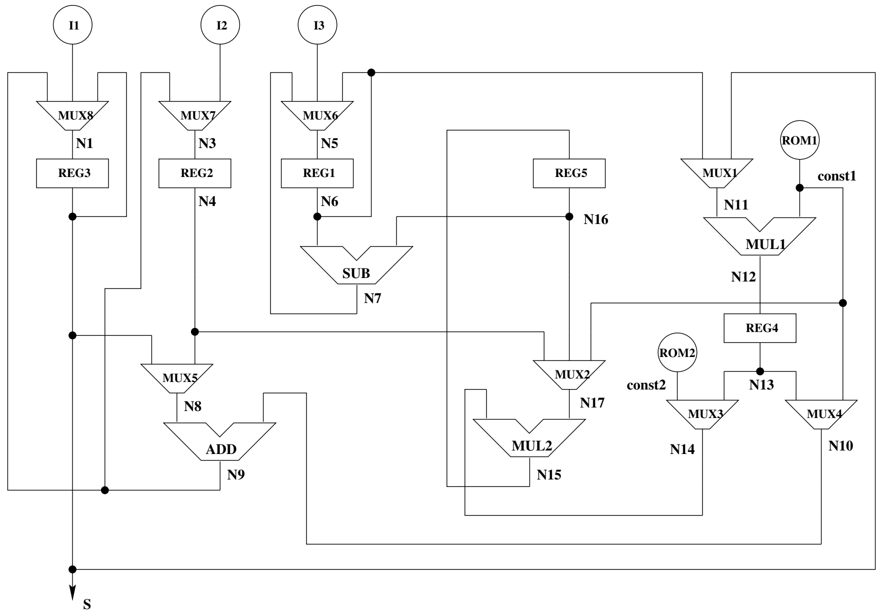


Figure 10.6: Original architecture of circuit Bert.

10.3.1 Allocation for external test

As can be observed from testability analysis results summarized in table 10.3, for an external test, there are no untestable nodes in original circuit Bert.

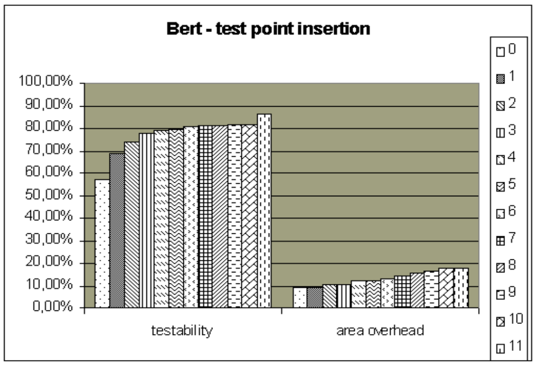
Node	C	O	T	CE	OE	TE
I1	1	1	1	0	0	0
I2	1	0.667	0.667	0	0	0
I3	1	0.696	0.696	0	0	0
clk	1	1	1	0	0	0
sel1	1	0.696	0.696	0	0	0
sel2	1	0.286	0.286	0	0	0
sel3	1	0.372	0.372	0	0	0
sel4	1	0.667	0.667	0	0	0
sel5	1	0.667	0.667	0	0	0
sel6	1	0.436	0.436	0	0	0
sel7	1	0.545	0.545	0	0	0
sel8	1	0.817	0.817	0	0	0
S	1	1	1	0	0	0
sel_test	1	0.817	0.817	0	0	0
sel1_TEST	1	0.696	0.696	0	0	0
sel2_TEST	1	0.286	0.286	0	0	0
sel3_TEST	1	0.372	0.372	0	0	0
sel4_TEST	1	0.667	0.667	0	0	0
sel5_TEST	1	0.667	0.667	0	0	0
sel6_TEST	1	0.436	0.436	0	0	0
sel7_TEST	1	0.545	0.545	0	0	0
sel8_TEST	1	0.817	0.817	0	0	0
const1	0	0.868	0.868	0	3297.5	3297.5
const2	0	0.484	0.484	0	12894.2	12894.2
n1	1	1	1	0	0	0
n3	1	0.667	0.667	0	4866.2	4866.2
n4	1	0.667	0.667	0	9399.0	9399.0
n5	1	0.696	0.696	0	7512.0	7512.0
n6	1	0.696	0.696	0	8576.5	8576.5
n7	0.627	0.696	0.436	9220.8	14294.2	23515.0
n8	1	0.667	0.667	0	4866.2	4866.2
n9	0.817	1	0.817	7195.8	0	7195.8
n10	0.768	0.868	0.667	9976.0	1925.7	11901.7
n11	1	0.696	0.696	0	4440.3	4440.3
n12	0.768	0.868	0.667	6542.4	54580.2	61122.6
n13	0.768	0.868	0.667	6774.4	3719.6	10494.0
n14	0.768	0.484	0.372	96001.6	7530.2	103531.8
n15	0.768	0.604	0.464	6542.4	163830.1	170372.5
n16	0.768	0.604	0.464	16634.4	11164.8	27799.2
n17	1	0.372	0.372	0	15514.3	15514.3
sel1′	1	0.696	0.696	0	4440.3	4440.3
sel2′	1	0.286	0.286	0	10430.1	10430.1
sel3′	1	0.372	0.372	0	9170.4	9170.4
sel4′	1	0.667	0.667	0	4866.2	4866.2
sel5′	1	0.667	0.667	0	4866.2	4866.2
sel6′	1	0.436	0.436	0	8233.0	8233.0
sel7′	1	0.545	0.545	0	6648.4	6648.4
sel8′	1	0.817	0.817	0	2673.3	2673.3

Table 10.3: Testability analysis results for original circuit bert supposing external test.

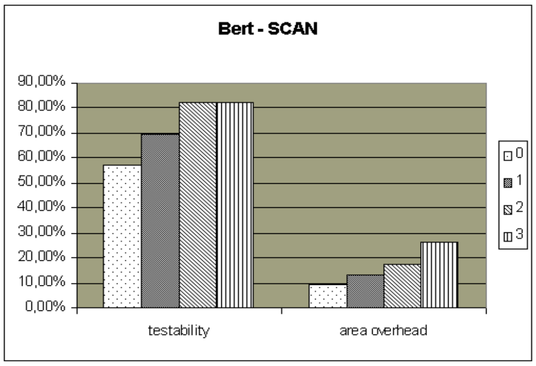
Test point insertion

User requirement for testability is more than 86% and for area overhead less than 20%. The selected global attributes for each Bert data-path modification step are depicted in the diagram of figure 10.7(a). Performed modifications are as follows:

1. N15 is made primary output
2. N14 is made primary input
3. N12 is made primary output
4. N16 is made primary input
5. N7 is made primary output
6. N10 is made primary input
7. N13 is made primary input
8. N9 is made primary input
9. N15_POINT is made primary input
10. N12_POINT is made primary input
11. N17 is made primary output.



(a)



(b)

Figure 10.7: Selected resulting attributes for different data-path configurations of circuit Bert (a) with inserted test points, (b) allocated for scan.

Allocation for scan

User requirement for testability is more than 82%, for area overhead less than 20% and for maximal number of untestable nodes 0%. The selected global attributes for each Bert data-path modification step are depicted in the diagram of figure 10.7(b). Performed modifications are: register REG5 is transformed to its scan equivalent and connected to new created scan chain and then registers REG4, REG1 and REG2 are successively transformed to their scan equivalents and connected to the same scan chain.

10.3.2 Allocation for BIST

User requirement for testability is more than 79%, for area overhead less than 16% and for maximal number of untestable nodes less than 20%. The selected global attributes for each modification step performed by IDAT are depicted in the diagram of figure 10.8. Performed modifications are as follows:

1. MUL1 selected: input N11 is directly connected to TPG, input const1 is directly connected to TPG, output N12 is directly connected to TRC,

2. MUL2 selected: input N14 is directly connected to TPG, input N17 is by signal const1 already connected to TPG, output N15 is directly connected to TRC,

3. ADD selected: input N8 is directly connected to TPG, input N10 is by signal const1 already connected to TPG, output N9 is directly connected to TRC,

4. SUB selected: input N6 is by signal N7 already connected to TPG, input N16 is directly connected to TPG, output N7 is directly connected to TRC.

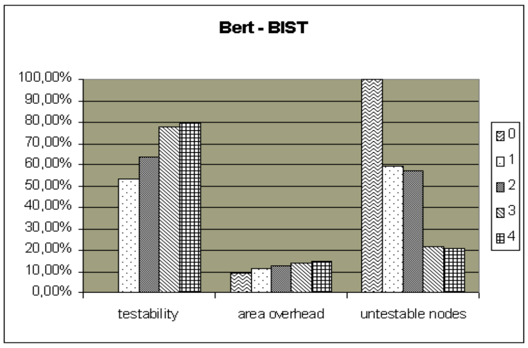


Figure 10.8: Selected resulting attributes for different data-path configurations of circuit Bert allocated for BIST.

10.4 Further results with new IDAT version

The IDAT software is currently being updated with new features, as indicated in relevant chapters of this thesis document. Some recent optimization of IDAT software has allowed us to compute the general evolution of the testability attribute with respect to the area overhead attribute. Results provided in this section show, for two benchmarks (Tseng and Diffeq) and the three possible DFT modes (test point insertion, Scan and

BIST), the best testability which can be obtained when there is no restriction imposed on the area overhead.

Although it is usually impossible, with any test method configuration, to obtain testability equal 100%, because of existence of FUs, which have generally CTF and OTF factors smaller than 1, the obtained results show how testability can be improved with relatively low area overhead, using an automatic allocation for testability tool like IDAT.

10.4.1 Test point insertion mode

The testability evolution with respect to area overhead is shown for test point insertion mode on figures 10.9 and 10.10, for benchmarks Tseng and Diffeq, respectively. A peculiarity of the test point insertion mode is that the total area overhead is very high (78% for Tseng and 152% for Diffeq). This is due to the fact that the necessary area overhead for new pad design has been taken into account in both figures. Obviously, in this mode, each additionnaly inserted test point involves a new pad in the circuit, and pads are known to be highly area consuming, specially when compared to the core area of small benchmarks like Tseng and Diffeq.

For Tseng benchmark, achievable testability reaches 96.71%, for a maximum area overhead of 78.24% (including pad area). The area overhead reached for this testability value when excluding pad area is only about 5%.

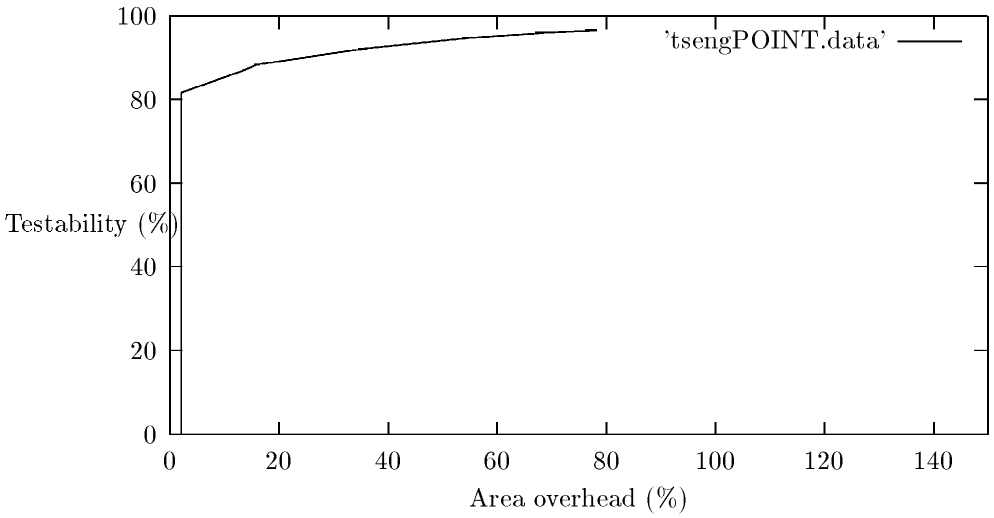


Figure 10.9: Tseng: Testability increase w.r.t. area overhead (Test point insertion mode)

For Diffeq benchmark, achievable testability reaches a value of 93.71%, which is obtained by transformation of nearly each node to either control or observe point. In this case the obtained area overhead is 152.68%, while the area overhead of the core of the benchmark (excluding pad area in both original and testable versions) is only about 2%.

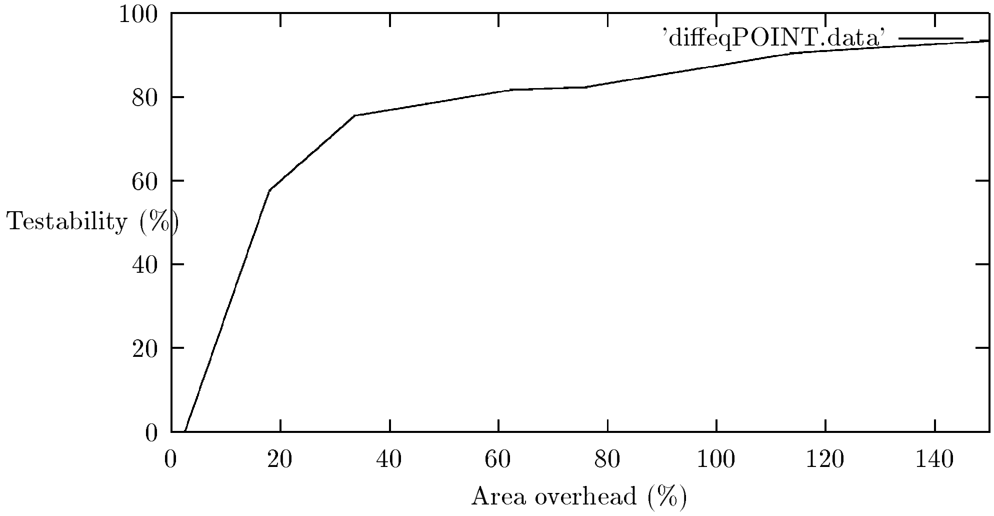


Figure 10.10: Diffeq: Testability increase w.r.t. area overhead (Test point insertion mode)

10.4.2 Scan mode

The testability evolution with respect to area overhead is shown for Scan mode on figures 10.11 and 10.12, for benchmarks Tseng and Diffeq, respectively. For Tseng benchmark, achievable testability reaches 89.98%, for a maximum area overhead of 9.66%. For Diffeq benchmark, achievable testability reaches 81.82%, for a maximum area overhead of 11.74%.

10.4.3 BIST mode

The testability evolution with respect to area overhead is shown for BIST mode on figures 10.13 and 10.14, for benchmarks Tseng and Diffeq, respectively. For Tseng benchmark, achievable testability reaches 73.35%, for a maximum area overhead of 9.36%. For Diffeq benchmark, achievable testability reaches 80.59%, for a maximum area overhead of 17.33%.

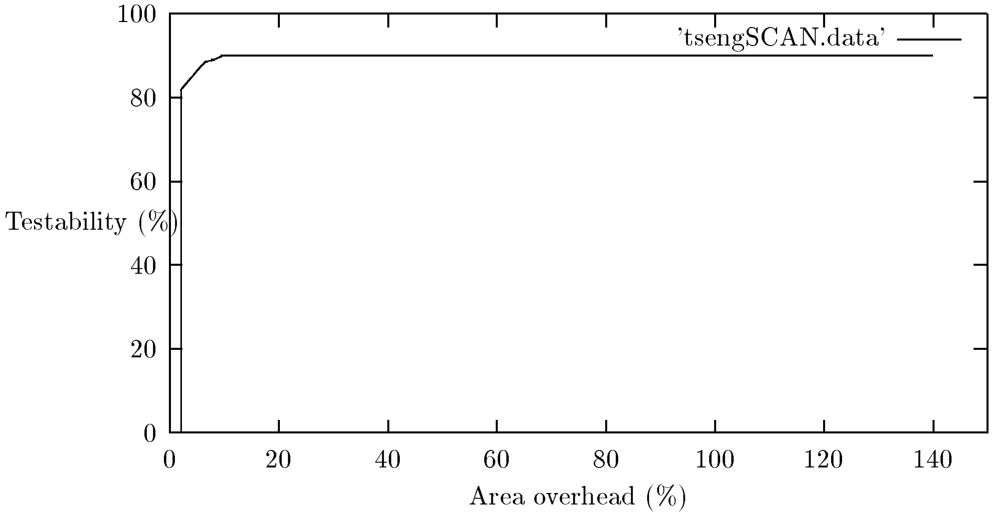


Figure 10.11: Tseng: Testability increase w.r.t. area overhead (Scan mode)

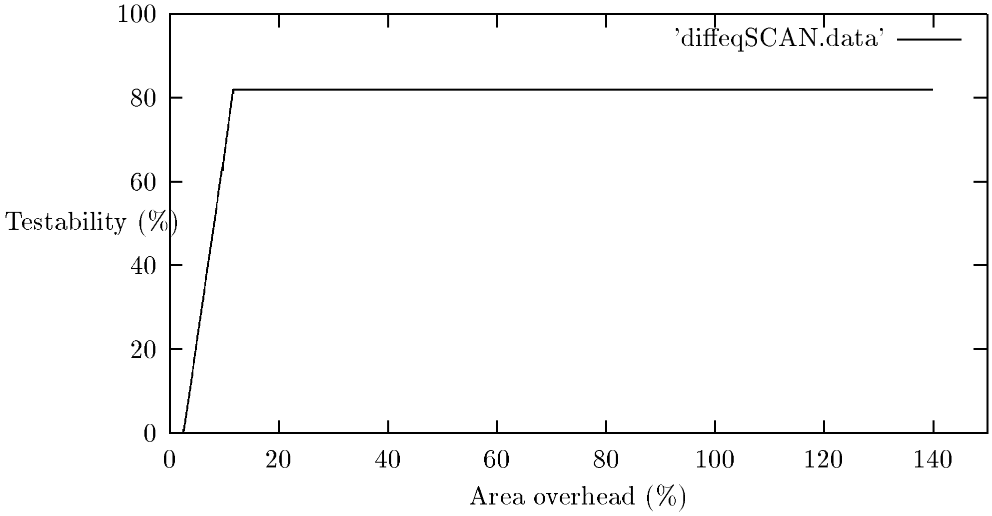


Figure 10.12: Diffeq: Testability increase w.r.t. area overhead (Scan mode)

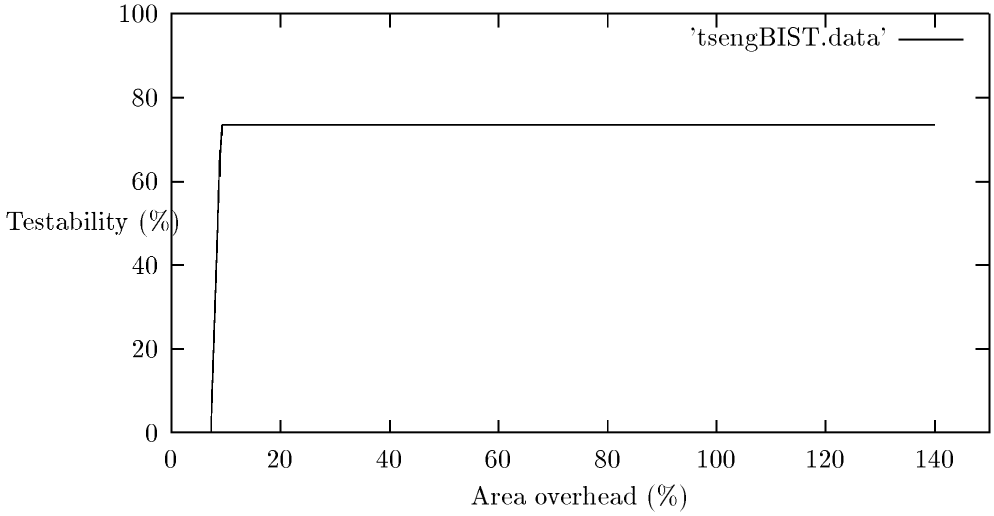


Figure 10.13: Tseng: Testability increase w.r.t. area overhead (BIST mode)

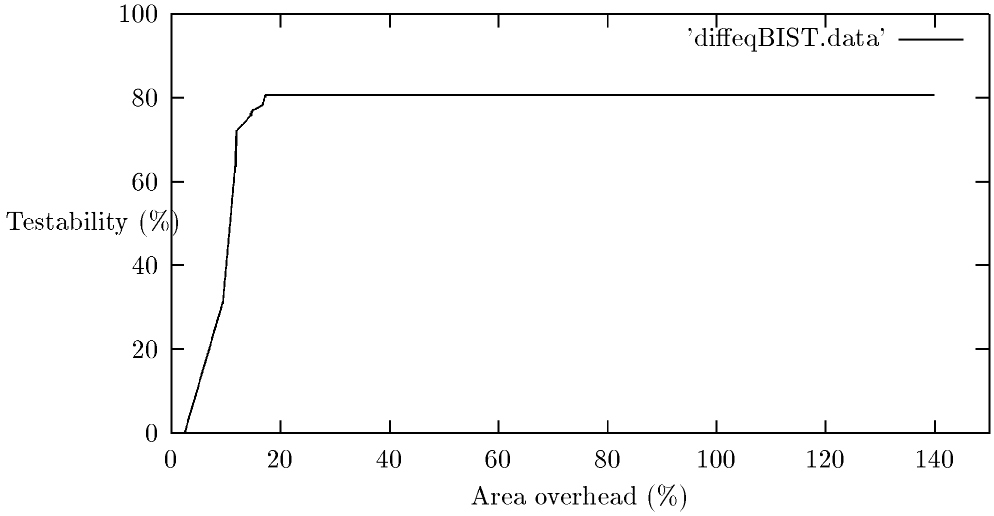


Figure 10.14: Diffeq: Testability increase w.r.t. area overhead (BIST mode)

Chapter **11**

Conclusions and Future Work

11.1 Conclusions

The aim of this thesis was to propose an interactive allocation for testability method for an RTL data-path taking into account intended implementation of the test controller.

As is known, usually a modification of a design to make it more testable leads to some cost and performance degradations. Presented trade-off allocation method, driven by testability analysis results and selected circuit design attributes, attempts to reach the best cost/quality trade-off of designed circuit. Our allocation method was implemented into our *Interactive DFT in Allocation for Testability* (IDAT) tool taking into account the availability of a test controller, synthesized using a complementary tool. The main features of IDAT system are:

- Adaptability of our approach to most already existing HLS systems. Currently IDAT is being implemented in conjunction with the HLS system UGH, in the framework of Alliance VLSI CAD system [Ali99].

- Consolidation of test point insertion, scan and BIST approach into one tool, which brings the user more freedom in the test methodology selection. Most of estimated circuit attributes, like area overhead and testability, are comparable among all of these circuit architectures.

- Reusing of already existing components with implemented testability structures - DFT reuse concept.

- New RTL data-path testability analysis approach taking into account the availability of a test controller in all proposed allocation for testability approaches. Presented testability analysis algorithm calculates for each circuit node its controllability exigency and observability exigency values, reflecting not only test vectors transparency problems through different components, but also the test complexities of these components.

- The testability analysis guides our interactive cost/quality trade-off algorithm. Presented trade-off algorithms for different test methodologies take into the account user experience in DFT area, his requirements to certain design and test attributes of final circuit, available means from the library of testable components and imposed test methodology constraints. The trade-off algorithms are implemented in the form of the branch-and-bound optimizations.

11.2 Future work

There remains some parts of our work both in the research area and in the area of IDAT algorithms optimization, which need more detailed approach:

- Connection of IDAT with the tool creating global test schedule and implementing required test controller.
- Implementation of the *shared selection inputs mode*. In this mode, in order to not completely change the philosophy of our testability analysis, we will keep the concept of separate test controller. The difference will be that we will not allocate new multiplexer for the selection input of each multiplexer already existing in the data-path in order to make this multiplexer controllable in an independent way. In contrast to normal mode, we will allocate new multiplexer only to each control signal from original circuit controller.
- Modification of testability analysis method in order to introduce a solution for reconvergence of component inputs and a solution to analyze embedded loops. More directions for future research concerning these subjects can be found in chapter 6.
- Testing and implementation of the algorithm to calculate global attribute max_delay. The proposed solution can be found in chapter 7.
- Development of a graphical interface to encapsulate IDAT system, tool which generates TPGs and TRCs according to IDAT's requirements and system implementing required test controller.

11.3 Long term perspectives

This thesis has addressed the domain of synthesis for testability of integrated circuits, made up of functional units (FUs). These FUs may be of various kind, size and complexity, and one of the main interest of our approach is that no assumption is made on these FUs, which are considered as black-boxes. This kind of approach may then be extended at higher level of complexity and integration, namely at System-on-a-Chip level.

Advances in design methods and fabrication technologies now enable the design of complex system chips (so-called System on a Chip, or SoC). However, time-to-market pressure, together with return on investment constraints, make the design from scratch style difficult for such complex systems. Thus, the new design style trend is more and more based on embedding reusable modules (called cores), and estimations are foreseeing at 50% the number of core-based embedded systems in year 2000, and at 90% in 2013.

Typically, SoCs are made up of a CPU core, one or more dedicated co-processors, one or more memory elements, optionnaly some analog cores, and some logic defined by the system integrator.

Designing SoCs in view of testability has been confirmed as the unique way of detecting and locating their faulty behavior. This tendency is even stronger for the design of IP cores, which testing becomes very hard when the core is integrated in the SoC, unless if the core provider gives test means (e.g. test vectors), just like data-sheets are provided, and in a standard form.

Access to the cores can either be serial, based on a scan or boundary scan chain, or parallel, through a test bus. In any case, testing each core individually should be allowed, as well as the test of the system logic and the test of the system interconnections. The core itself should include its own testability features.

Among the many technical constraints implied by this level of system integration, one of them is related to the testability at the SoC level as well as at the embedded core level. Accessibility to the cores through the SoC pins is the main problem, and new language and architecture standards for testing are currently being investigated [Zor99].

The test group within LIP6 laboratory is currently investigating the testability of such systems, and a testable architecture has already been defined [MBM00].

However, one of the still unresolved, hot topics in this area is the issue of core test resources partitioning, according to various criteria: core external and core internal test resources, area overhead, delay overhead (specially with new technologies), etc. Reusing the concepts developed in the framework of this thesis and some parts of the IDAT tool may help solving this same kind of problems occurring at the system level, by considering cores as functional units.

Appendix **A**

List of Abbreviations

ASIC - Application Specific Integrated Circuits

ATPG - Automated Test Pattern Generation

BILBO - Built-In Logic-Block Observer

BIST - Build-in Self Test

DF - Distance Function

DFG - Data Flow Graph

DFT - Design for Test

FU - Functional Unit

HLS - High-Level Synthesis

HL-SFT - High-Level Synthesis for Testability

IDAT - Interactive Design for Test reuse in Allocation for Testability tool

LFSR - Linear Feedback Shift Register

MISR - Multiple Inputs Signature Analyzer

PCF - Penalty-Credit Function

RTL - Register-Transfer Level

RVA - Rich Virtual Architecture

SFT - Synthesis for Testability

TPG - Test Patterns Generator

TRC - Test Results Checker

VLSI - Very Large Scale Integrated

Appendix **B**

Benchmark Diffeq

Following description is an RVA description of circuit Diffeq depicted in figure 5.1 in chapter 5:

```
// Modified circuit Diffeq, CMP is substituted by SUB

(
  diffeq

  // Primary ports
  (a(7 downto 0)     IN  DATA
   dx(7 downto 0)    IN  DATA
   clk               IN  CLK
   sel1              IN  CONTROL
   sel2              IN  CONTROL
   sel3              IN  CONTROL
   sel4(1 downto 0)  IN  CONTROL
   sel5              IN  CONTROL
   sel6              IN  CONTROL
   r(7 downto 0)     OUT DATA)

  // Components
  (
   register_d REG
   ((area 28200)(max_delay 2)(power 500)(clock_cycles 1)(init NO)
    (CTF 1)(OTF 1)(testability 1))
   (
    q(7 downto 0) OUT DATA
    d(7 downto 0) IN  DATA
    ck IN CLK
   )

   adder ADD
   ((area 43000)(max_delay 100)(power 700)(clock_cycles 0)(init NO)
    (CTF 0.8169)(OTF 0.8681)(testability 0.99))
   (
    a(7 downto 0) IN  DATA
    b(7 downto 0) IN  DATA
    q(7 downto 0) OUT DATA
   )

   multiplexor2 MUX
   ((area 14600)(max_delay 22)(power 200)(clock_cycles 0)(init NO)
    (CTF 1)(OTF 1)(testability 1))
```

```
 (
  a(7 downto 0) IN  DATA
  b(7 downto 0) IN  DATA
  q(7 downto 0) OUT DATA
  sel IN CONTROL
 )

 subtracter SUB
 ((area 47000)(max_delay 190)(power 900)(clock_cycles 0)(init NO)
  (CTF 0.816)(OTF 0.8681)(testability 0.99))
 (
  a(7 downto 0) IN  DATA
  b(7 downto 0) IN  DATA
  q(7 downto 0) OUT DATA
 )

 multiplier8 MUL
 ((area 413800)(max_delay 1000)(power 900)(clock_cycles 0)(init NO)
  (CTF 0.768)(OTF 0.8016)(testability 0.98))
 (
  a(7 downto 0) IN  DATA
  b(7 downto 0) IN  DATA
  q(7 downto 0) OUT DATA
 )

 multiplexor3 MUX
 ((area 24700)(max_delay 22)(power 220)(clock_cycles 0)(init NO)
  (CTF 1)(OTF 1)(testability 1))
 (
  a(7 downto 0)   IN  DATA
  b(7 downto 0)   IN  DATA
  c(7 downto 0)   IN  DATA
  q(7 downto 0)   OUT DATA
  sel(1 downto 0) IN  CONTROL
 )

 const2 CONST
 ((area 25000)(max_delay 2)(power 400)(clock_cycles 0)(init NO)
  (CTF 0)(OTF 1)(testability 1))
 (
  q(7 downto 0) OUT DATA
 )
)


// Signals
(
 const(7 downto 0)
 n1(7 downto 0)
 n2(7 downto 0)
 n3(7 downto 0)
 n4(7 downto 0)
 n5(7 downto 0)
 n6(7 downto 0)
 n7(7 downto 0)
 n8(7 downto 0)
 n9(7 downto 0)
 n10(7 downto 0)
 n11(7 downto 0)
 n12(7 downto 0)
 n13(7 downto 0)
 n14(7 downto 0)
 n15(7 downto 0)
 n16(7 downto 0)
)
```

```
// Ports maps
 (
  reg3 register_d
  (
   q n5
   d n16
   ck clk
  )

  reg2 register_d
  (
   q n4
   d n16
   ck clk
  )

  sub2 subtracter
  (
   q r
   a a
   b n16
  )

  reg6 register_d
  (
   q n16
   d n15
   ck clk
  )

  add adder
  (
   b n13
   a n14
   q n15
  )

  mux5 multiplexor2
  (
   b n4
   q n14
   sel sel5
   a n5
  )

  reg1 register_d
  (
   q n2
   d n12
   ck clk
  )

  sub1 subtracter
  (
   b n2
   q n12
   a n11
  )

  reg4 register_d
  (
   q n11
   d n10
   ck clk
```

```
)

mul1 multiplier8
(
 a n1
 b n9
 q n10
)

mux2 multiplexor2
(
 a n2
 b n8
 q n9
 sel sel2
)

mux6 multiplexor2
(
 b n8
 q n13
 sel sel6
 a dx
)

reg5 register_d
(
 q n8
 d n7
 ck clk
)

mul2 multiplier8
(
 b n6
 q n7
 a n3
)

mux4 multiplexor3
(
 c n5
 b n4
 q n6
 a dx
 sel sel4
)

mux3 multiplexor2
(
 b const
 q n3
 sel sel3
 a n2
)

mux1 multiplexor2
(
 b dx
 q n1
 sel sel1
 a n11
)

cons const2
```

```
    (
     q const
    )
   )
)
```

Bibliography

[ABF90] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital System Testing and Testable Design*. Computer Science Press, New York, 1990. ISBN 0-7167-8179-4.

[ABF95] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital System Testing and Testable Design, Revised Printing*. IEEE Press, NJ, USA, 1995. ISBN 0-7803-1062-4.

[Ait95] R.C. Aitken. An overview of test synthesis tools. *IEEE Design and Test of Computers*, 12(2):8–15, Summer 1995.

[Ali99] Alliance vlsi cad system. http://www-asim.lip6.fr/alliance/, 1999.

[AM94] 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.

[AS88] V.D. Agrawal and S.C. Seth. *Tutorial - Test Generation for VLSI Chips*. IEEE Computer Society Press, 1988. ISBN 0-8186-8786-X.

[Avr91] L. Avra. Allocation and assignment in high-level synthesis for self-testable data path. In *Proceedings of the International Test Conference*, pages 463–472, 1991.

[BCM97] S. Barbagallo, F. Corno, and D. Medina. Testable synthesis through rt level dft rules. In *Proceedings of European Design and Test Conference*, 1997.

[BFR98] 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.

[BFR99] 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.

[BFR00] D. Berthelot, M.L. Flottes, and B. Rouzeyre. A method for trading test time, area and fault coverage in datapath bist synthesis. In *Compendium of Papers from IEEE European Test Workshop*, pages 219–224, Cascais, Portugal, May 2000.

[BMM99] 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.

[BMR81] R.G. Bennetts, C.M. Maunder, and G.D. Robinson. Camelot: A computer-aided measure for logic testability. *IEE Proceeding*, 128(pt.E(5)):177–189, 1981.

[Bre78] M.A. Breuer. New concepts in automated testing of digital circuits. In *Proceedings of EEEC Symposium on CAD of Digital Electronic Circuits and Systems*, pages 69–92, Brussels, 1978.

[Bre95] M.A. Breuer. Cad framework and tools for the synthesis of complex testable digital systems. Final progress report, University of Southern California, April 1995.

[BvEG95] F.P.M. Beenker, K.J.E. van Eerdewijk, and R.B.W. Gerritsen. *Testability Concepts for Digital ICs - The Macro Test Approach*. Kluwer Academic Publishers, Boston, 1995.

[CKS94] C.-H. Chen, T. Karnik, and D.G. Saab. Structural and behavioral synthesis for testability techniques. *Proceedings of IEEE Transactions on CAD of Integrated Circuits and Systems*, 13(6):777–785, June 1994.

[CLP94] V. Chickermane, J. Lee, and J.H. Patel. Addresing design fot testability at the architectural level. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 13(7):920–934, July 1994.

[CM89] C.H. Chen and P.R. Menon. An approach to functional level testability analysis. In *Proceedings of International Test Conference*, pages 373–380, Washington, 1989.

[Cox95] H. Cox. Synthesizing circuits with implicit testability constraints. *IEEE Design and Test of Computers*, 12(2):16–22, Summer 1995.

[CP91] S. Chiu and C.A. Papachristou. A design for testability scheme with applications to data path synthesis. In *Proceedings of 28th Design Automation Conference*, pages 271–277, June 1991.

[CP95] J. Carletta and C.A. Papachristou. Testability analysis and insertion for rtl circuit. In *Proceedings of International Conference on Computer Design*, 1995.

[CSdAAC99] 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.

[CWS91] C.-H. Chen, C. Wu, and D.G. Saab. Beta: Behavioral testability analysis. In *Proceedings of International Conference of Computer-Aided Design*, pages 202–205, 1991.

[DB99] L. Ducerf-Bourbon. Idat et son environnement logiciel: Etat des lieux. Technical report, Laboratory of Informatics at University of Paris 6, November 1999.

[DBBM00] L. Ducerf-Bourbon, P. Bukovjan, and M. Marzouki. Tacos: A testability allocation and control system. In *Compendium of Papers from IEEE European Test Workshop*, Cascais, Portugal, May 2000.

[DPR93] S. Dey, M. Potkonjak, and R. K. Roy. Exploiting hardware sharing in high-level synthesis for partial scan optimization. In *Proceedings of International Conference on Computer-Aided Design*, pages 20–24, Santa Clara, California, November 1993.

[FPR96] M.L. Flottes, R. Pires, and B. Rouzeyre. A testability analysis for driving architectural synthesis. In *Proceeding of High Level Design and Validatation Workshop*, Oackland, Novembre 1996.

[FPR97] 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.

[FPRV98] M.L. Flottes, R. Pires, B. Rouzeyre, and L. Volpe. Scanning datapaths : A fast and effective partial scan technique. In *Proceeding of Design Automation and Test in Europe (DATE 98),*, Paris, February 1998.

[FS96] V. Fernandez and P. Sanchez. Partial scan high-level synthesis. In *Proceedings of European Design and Test Conference*, 1996.

[Gaj94] D.D. Gajski. Introduction to high-level synthesis. *IEEE Design and Test of Computers*, pages 44–54, Winter 1994.

[GE92] C.H. Gebotys and M.I. Elmasry. *Optimal VLSI Architectural Synthesis*. Kluwer Academic Publishers, 1992. ISBN 0-7923-9223-X.

[GKP94] X. Gu, K. Kuchcinski, and Z. Peng. Testability analysis and improvement from vhdl behavioral specification. In *Proceedings of the European Design Automation Conferenece with EURO-VHDL*, pages 644–649, 1994.

[Gol80] L.H. Goldstein. Scoap: Sandia controllability/observability analysis program. In *Proceeedings of 17th Design Automation Conference*, Minnesota, June 1980.

[Gra79] J. Grason. Tmeas, a testability measurement program. In *Design Automation Conferences*, pages 156–161, 1979. n.

[GRJ95] A. Ghosh, A. Raghunathan, and N.K. Jha. Design for hierarchical testability of rtl circuits obtained by behavioral synthesis. In *Proceedings of International Conference on Computer Design*, pages 173–179, Austin, October 1995.

[Gu96] 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.

[Gup94] 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.

[HO94] I.G. Harris and A. Orailoglu. Syncbist: Synthesis for concurrent built-in self-testability. In *Proceedings of European Design Automation Conference*, pages 101–104, Cambridge, MA, Octobre 1994.

[HP93] 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.

[HPCN92] H. Harmanani, C.A. Papachristou, S. Chiu, and M. Nourani. Syntest: An environment for system-level design to test. In *European Design and Automation Conference*, pages 402–407, September 1992.

[HY81] C.L. Hwang and K. Yoon. *Multiple Attribute Decision Making - Methods and Applications: A State-of-art Survey*. Berlin/Heidelberg/New York, 1981.

[Ibm00] Ibm testbench system. http://www.chips.ibm.com/services/testbench/, July 2000.

[KCL94] T. Kim, K.-S. Chung, and C.L. Liu. A stepwise refinement data path synthesis procedure for easy testability. In *Proceedings of European Design and Test Conference*, pages 586–590, 1994.

[KRH00] Z. Kotasek, R. Ruzicka, and J. Hlavicka. Formal approach to the RTL testability analysis. In *1st IEEE Latin America Test Workshop*, pages 256–261, Rio de Janeiro, Brazil, March 2000.

[LCP93] J. Lee, V. Chickermane, and J.H. Patel. Impact of high level functional constraints on testability. In *Proceedings of IEEE VLSI Test Symposium*, pages 309–312, April 1993.

[Lee97] M.T.-C. Lee. *High-Level Test Synthesis of Digital VLSI Circuits*. Artech House Boston - London, 1997. ISBN 0-89006-907-7.

[Log99] Logic vision products. http://www.logicvision.com, November 1999.

[LP97] E. Larsson and Z. Peng. Early prediction of testability by analyzing behavioral vhdl specification. In *Proceedings of Norchip Conference*, Tallinn, November 1997.

[LPB97] K. Lai, C.A. Papachristou, and M. Baklashov. High level test synthesis across the boundary of behavioral and structural domains. In *Proceedings of International Conference on Computer Design*, 1997.

[MAA95] M. Marzouki, V.C. Alves, and A.R. Antunes. High-level synthesis for testability. In *38th IEEE Midwest Symposium on Circuits and Systems, Special Section on Testability Issues for Digital Circuits and Systems*, pages 718–721, Rio de Janeiro, Brazil, August 1995.

[MAAM96] 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.

[MBM00] W. Maroufi, M. Benabdenbi, and M. Marzouki. Controlling the cas-bus tam with ieee 1149.1 tap: A solution for systems-on-a-chip testing. In *4th IEEE International Workshop on Testing Embedded Core-based Systems*, pages 718–721, Montreal, Canada, May 2000.

[Men99] Mentor graphics dft products. http://www.mentorgraphics.com, October 1999.

[MJS94] A. Mujumdar, R. Jain, and K. Saluja. Behavioral synthesis of testable design. In *Proceedings of International Symposium of Fault-Tolerant Computing*, pages 436–445, Austin, June 1994.

[NAR00] T. Niculiu, C. Aktouf, and C. Robach. Behavioral and adaptable design for testability. In *Compendium of Papers from IEEE European Test Workshop*, pages 291–292, Cascais, Portugal, May 2000.

[PCH91] C.A. Papachristou, S. Chiu, and H. Harmanani. Syntest: A method fot high-level synthesis with self-testability. In *Proceedings of IEEE International Conference on Computer Design*, pages 458–462, October 1991.

[PGB95] 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.

[Pir98] R. Pires. *Test et Conception de Circuits Intégrés Digitaux : du niveau Comportemental au niveau Transferts de Registres*. PhD thesis, Université Montpellier II, July 1998. Ph.D. Thesis in french.

[PKG86] 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.

[PR94] 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.

[RS89] G. Russel and I.L. Sayers. *Advanced Simulation and Test Methodologies for VLSI Design*. Van Nostrand Reinhold (International), 1989. ISBN 0-7476-0001-5.

[Rut72] R.A. Rutman. Fault detection test generation for sequential logic heuristic tree search. *IEEE Computer Repository Paper*, 1972.

[SGCM93] J. Steensma, W. Geurts, F. Catthoor, and H. De Man. Testability analysis in high level data path. *Journal of Electronic Testing: Theory and Applications*, (4):43–56, 1993.

[Syn00] Synopsys test compiler family. http://www.synopsys.com, July 2000.

[TA89] K. Thearling and J. Abraham. An easily computed functional level testability measure. In *International Test Conference*, 1989.

[VTAA93] P. Vishakantaiah, T. Thomas, J.A. Abraham, and M.S. Abadir. Ambiant: Automatic generation of behavioral modification for testability. In *Proceeding of International Computer Design*, pages 63–66, Boston, October 1993.

[WC95] R.A. Walker and S. Chaudhuri. Introduction to the scheduling problem. *IEEE Design and Test of Computers*, 12(2):60–69, Summer 1995.

[WD96] 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.

[Wil90] B.R. Wilkins. *Testing Digital Circuits: An introduction*. Chapman and Hall, 1990. ISBN 0-412-38360-8.

[YP98] T. Yang and Z. Peng. Incremental testability analysis for partial scan selection and design transformations. In *Proceedings of IEEE European Test Workshop*, pages 107–112, Barcelona, Spain, May 1998.

[ZB88] X.A. Zhu and M.A. Breuer. A knowledge-based system for selecting test methodologies. In *IEEE Design and Test of Computers*, October 1988.

[Zhu86] 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.

[Zor99] Y. Zorian. Testing the monster chip. *IEEE Spectrum*, pages 54–60, July 1999.