7-20

動作記述回路を用いた順序テスト生成およびテスト容易化バインディング

日大生産工(院)　○井上　諒一　　日大生産工　細川　利典
奈良先端大　藤原　秀雄

1　はじめに

近年，半導体微細化技術の進歩に伴い，回路の大規模化，複雑化が急速に進んでいる．組合せ回路に関しては，大規模回路に対しても高い故障検出率を得られるテスト生成手法が提案されている[1]．しかしながら，順序回路に関しては，テスト生成時間が膨大になり，現実的な時間で高い故障検出率を達成することが困難である．

また，近年 LSI の設計生産性向上のために，より抽象度の高い，動作記述レベル[2]と呼ばれる記述での設計が提案され，普及が進んでいる．動作記述レベルで設計された回路は，動作合成[2]と呼ばれる技術を用いてレジスタ転送レベル（Register Transfer Level: RTL）の回路に合成される．動作合成は，スケジューリング[2]と呼ばれる，回路をサイクル精度な構造にする処理と，バインディング[2]と呼ばれる，演算器とレジスタの共有化を実行する処理から構成されている．スケジューリングまで完了し出力した回路を，機能記述 RTL という．スケジューリングとバインディングが完了し出力した回路を，構造記述 RTL という．

構造記述 RTL に対して階層テスト生成[8]に基づくテスト容易化設計が提案されている[3]．しかし，このレベルの回路はコントローラとデータパスが分離されている．したがって，データパスとコントローラに対して，別々のテスト容易化設計が必要になり，面積オーバーヘッドが大きくなる．

そこで，機能記述 RTL を対象に，縮退故障のテスト生成を行う手法が提案されている[4,5,6]．これらの手法は，機能記述 RTL 回路を割当決定図（Assignment Decision Diagram: ADD）[7]で表現したものを対象にテスト系列を生成している．具体的には，各機能要素に対して階層テスト生成を行い，テスト環境[8]を生成する．次に，各機能要素のゲートレベル回路に対する縮退故障テストパターンをテスト環境に代入することでテスト系列を生成している．これらの手法は，コントローラとデータパスを分離せずにテストが可能である．さらに，従来のゲートレベルテスト生成と比較し高速にテスト生成ができている．

しかしながら，これらの手法はバインディングを論理合成にまかせており，生成したテスト系列がゲートレベル回路に対して高い故障検出率を得られる保証はない．本論文では，ADD を用いて生成したテスト環境を考慮してバインディング[2]を実行することで，ゲートレベル回路に対して高い故障検出率を達成することが可能な手法を提案する．

2　動作合成

動作合成とは，C 言語などで記述した動作記述から，ハードウェア記述言語である VHDL[9]や Verilog HDL[10]などの RTL 回路記述を生成するものである．動作合成の手順は 4 つに分けられる：グラフ生成，スケジューリング，バインディング，RTL 回路記述生成（図 1）．

A Sequential Test Generation Method and a Binding Method for Testability Using Behavioral Description Circuits

Ryoichi INOUE, Toshinori HOSOKAWA, and Hideo FUJIWARA

図 1：動作合成

グラフ生成では，与えられた動作記述を割当決定図（Assignment Decision Diagram: ADD）[7]やコントロール/データフローグラフ（Control/Data-Flow Graph: CDFG）[2]と呼ばれるグラフで表現する．スケジューリングでは，各演算操作をどの時刻に実行するかを決定する．バインディングでは，スケジューリングした情報をもとに，各演算操作や変数に，具体的な演算器やレジスタを割り当てる処理を行う．RTL 回路記述生成では，割り当てられた演算器やレジスタ間の結線を行いデータパスを生成する．また，制御信号を生成するコントローラを生成する．

ADD を用いた動作合成では，スケジューリングやバインディングの処理をグラフ構造を変化させることで実現する．スケジューリング済みの ADD を動作 ADD，バインディング済みの ADD を構造 ADD と定義する．

3 ADD

本論文の対象である割当決定図（Assignment Decision Diagram: ADD）にていて説明する．ADD は，図 2 のように 4 つの要素から構成されている：割当値，割当条件，割当決定，割当対象．さらにこれらは，4 つのノードで構成されている：演算ノード，読込みノード，書込みノード，割当決定ノード（Assignment Decision Node: ADN）．ADD の動作は以下の通りである．論理演算で計算される割当条件により，ADN の条件入力の一つが真になる．条件入力に対応する値入力が選択され，論理演算で計算された割当値が割当対象に入力される．

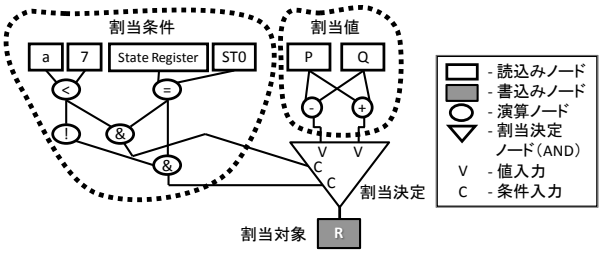

図 2：ADD

ADD は本来，動作合成内部の処理で用いるために提案されたものであるが，機能記述 RTL 回路を表現するデータ構造として用いることもできる．コントローラとデータパスを一様に表現できるため，コントローラとデータパスを分離せずにテスト環境生成を行うモデルとして適している．

4 ADD を用いた順序テスト生成

ADD を構成する各ノードに対して，任意の値を外部入力である読込みノードから正当化する経路を正当化経路と呼ぶ．また，そのノードの出力値を外部出力である書込みノードまで伝搬する経路を伝搬経路と呼ぶ．ADD を用いた順序テスト生成は，正当化経路と伝搬経路を有効にする値の時系列であるテスト環境を生成する．この際用いる値は，文献[4]で提案された 9 値の代数を基本に，文献[5][6]では代数を拡張したものが用いられている．次に，あらかじめライブラリ化しておいた各ノードのテストパターンをテスト環境に代入することで，テスト系列を生成する．このテストパターンは，ノードの種類ごとに論理合成し，生成したゲートレベル回路に対し，組合せテスト生成を実行して生成したものである．ただし，回路に対して動作合成や論理合成を実行することで，各演算がライブラリで想定したゲートレベル回路と異なっている場合がある．従って，論理合成後に故障シミュレーションを実行し，故障検出率を確認する必要がある．

5 バインディングを考慮した ADD を用いた順序テスト生成

本テスト生成法のねらいについて 5.1 で述べ，5.2 で本テスト生成法の手順について述べる．5.3 で提案するテスト容易化バインディングを

説明する．

5.1 ねらい

ADD の全ノード数に対し，テスト環境が生成されたノード数の割合であるテスト環境生成率[6]が高いほど，ゲートレベルでの故障検出率も高くなる．しかしながら，テスト環境生成率が最も高い文献[6]の手法でも，ゲートレベルのテスト生成手法と比較して高い故障検出率を得られているとは言えない．この理由としては以下のことが挙げられる．従来法では動作合成の手順の一つであるバインディングを考慮しておらず，バインディングが実行された際に，一つの演算器に対して，テスト環境が生成できない演算のみが割り当てられる可能性がある．この場合，動作合成後の RTL 回路においてテスト環境が存在しない演算器が残る．したがって，論理合成後のゲートレベル回路に対しても，生成したテスト系列で検出できない故障が残り，故障検出率が低下していると考えられる．

そこで，本手法では，ある演算器に対して，テスト環境が生成できない演算のみが割り当てられないようにすることで，故障検出率を向上させるバインディング手法を提案する．

5.2 テスト生成手順

本テスト生成法の手順について説明する．テスト生成フローを図 3 に示す．

(STEP1)
機能 ADD を入力とし，文献[6]の手法を用いてテスト環境を生成する．
(STEP2)
STEP1 でテスト環境の生成が不可能なノードが存在した場合，与えられた演算器数制約のもとでテスト容易化を考慮したバインディングを行い，構造 ADD を生成する．テスト環境の生成が不可能なノードが存在しなかった場合は，与えられた演算器数制約のもとでバインディングを行い，構造 ADD を生成する．
(STEP3)
STEP2 でバインディングを実行することで ADD 上に ADN が追加されるので，追加された ADN に対してテスト環境を生成する．
(STEP4)
STEP3 で変更したテスト環境に被テストノードのテストパターンを代入し，テスト系列を生成する．
(STEP5)
STEP2 で生成した構造 ADD を論理合成し，ゲートレベル回路を生成する．
(STEP6)
STEP4 で故障シミュレーションを実行し，生成したテスト系列の故障検出率を算出する．

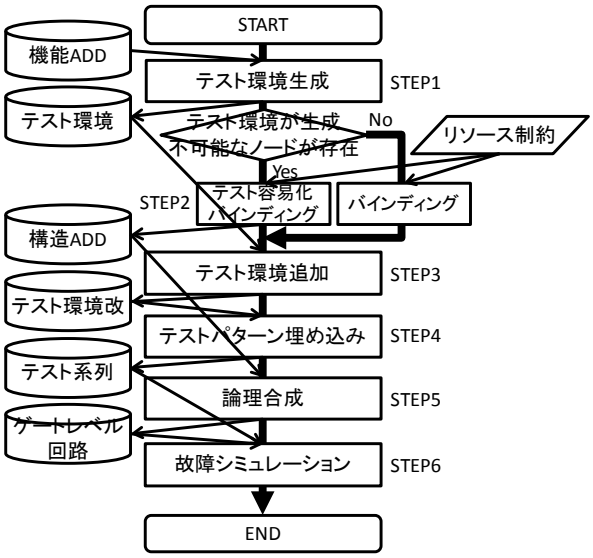

図 3：テスト生成フロー

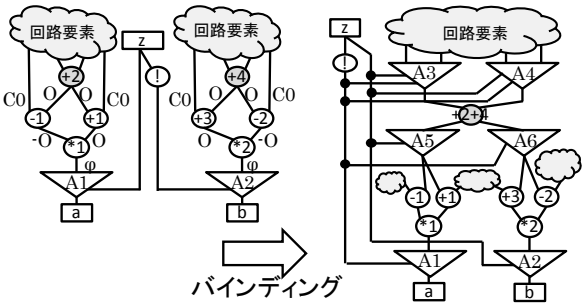

図 4：バインディング例 1

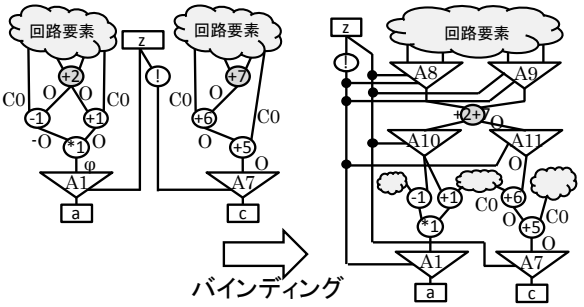

図 5：バインディング例 2

5.3 テスト容易化バインディング

提案するテスト容易化バインディングを，例を用いて説明する．図 4 と図 5 にバインディングを行った例を示す．加算+2 と加算+4 に対して，テスト環境を生成した場合，故障の影響を観測できるという意味の代数 O[4]が，それぞれ乗算*1 と乗算*2 の出力に伝搬できず，テスト環境の生成に失敗する．加算+7 に関してはテスト環境を生成できている．加算+2 と加算+4 を同じの加算器にバインディングした場合（図 4），この加算器に対するテスト環境は存在せず，テストができない．一方，加算+2 と加算+7 を一つの加算器にバインディングした場合（図 5），加算+7 に対して生成したテスト環境を使用することができ，この加算器に対するテストが可能である．本手法では，リソース制約を満たす限り，このようなバインディングを行う．ただし，バインディングを行ったことにより，ADN（A8, A9, A10, A11）が追加される．したがって，これらの ADN に対して，追加でテスト環境を生成する必要がある．

6 おわりに

本論文では，ADD を用いて生成したテスト環境を考慮してバインディングを実行することで，ゲートレベル回路に対して高い故障検出率を達成することが可能な手法を提案した．

今後の課題として，本手法を実装し，生成されたテスト系列でどれだけ故障を検出できるかを検証する必要がある．

「参考文献」

[1] E. Gizdarski and H. Fujiwara, “SPIRIT: a highly robust combinational test generation algorithm,” IEEE Trans. on Computer Aided Design for Integrated Circuits and Systems, Vol. 21, No. 12, pp. 1446-1458, Dec. 2002.

[2] Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, and Steve Y-L Lin, HIGH-LEVEL SYNTHESIS Introduction to Chip and System Design, Kluwer Academic Publisher, 1992.

[3] H. Wada, T. Masuzawa, K. K. Saluja and H. Fujiwara, "Design for strong testability of RTL data paths to provide complete fault efficiency," Proc. of 13th International Conf. on VLSI Design, pp. 300-305, Jan. 2000.

[4] I. Ghosh, M. Fujita, "Automatic Test Pattern Generation for Functional RTL Circuits Using Assignment Decision Diagrams," Proc. of ACM/IEEE Design Automation Conference, pp.43-48, Jun. 2000.

[5] L. Zhang, I. Gosh, and M. Hsiao, “Efficient sequential atpg for functional rtl circuits,” in Proc. International Test Conference, pp. 290-298, Sep. 2003.

[6] H. Fujiwara, C. Y. Ooi and Y. Shimizu, "Enhancement of test environment generation for assignment decision diagrams," 9th IEEE Workshop on RTL and High Level Testing (WRTLT'08), pp.45-50, Nov. 2008.

[7] V. Chaiyakul, D. D. Gajski, and L. Ramachandran, "High-Level Transformations for Minimizing Syntactic Variances," in Proc. Design Automation Conference, pp. 413-428, Jun. 1993.

[8] B. T. Murray and J. P. Hayes, "Hierarchical test generation using pre computed tests for modules," IEEE Trans. Computer-Aided Design, Integrated Circuits & Syst., vol.9, no.6, pp.594-603, Jun. 1990.

[9] 中 幸政, VHDL と CPLD によるロジック設計入門, CQ 出版株式会社, 2005.

[10] 並木 秀明,前田 智美,宮尾 正大, ディジタル回路と Verilog-HDL, 株式会社技術評論社, 1996.