7-48

ループ数削減指向テスト容易化バインディングの評価

日大生産工(学部)　○森田　菜留美
日大生産工　　細川　　利典

1. はじめに

近年，半導体技術の急速な進歩により，LSI (Large Scale Integrated circuits)が大規模化し，回路が複雑化してきている．従来 LSI はレジスタ転送レベル(Register Transfer Level: RTL)での設計が主流であるが，LSI の大規模化に伴い， RTL での設計が困難になってきている[1]. それゆえ，RTL より抽象度の高い動作レベルで設計された回路を RTL に変換する技術である動作合成[2]が注目されている．

また，LSI の大規模化，複雑化に伴い，LSI のテストはますます重要でかつ困難な問題となっている．そのため，動作合成の段階でテスト容易性を考慮することにより，回路の本来の機能を保ちつつテスト容易性も含めた全体的な最適化が可能になるものと期待されている[2].

LSI は順序回路であり，回路中にフィードバックループ[3]が存在する．順序回路に対して，レジスタの値を任意に設定及び観測することが困難であるため，テスト生成が困難である[1].

順序回路のテスト容易化の設計手法として，一部のレジスタをスキャン可能なレジスタ(スキャンレジスタ)に置き換える部分スキャン設計[4]が提案されている．部分スキャン設計を指向したデータパスのテスト容易化動作合成法として，これまで多くの手法が提案されている[2][3].

本稿では，文献[2]のテスト容易化を考慮した動作合成法として無閉路部分スキャン設計を指向したデータパスのバインディング手法[2]の実装を検討し，回路中のセルフループ数を削減した回路のテスト容易性の評価を報告する．評価の比較対象として，面積最小化を考慮したデータパスのバンディング手法であるレフトエッジアルゴリズム(LEA)[4]を使用する．

2. 動作合成

動作合成とは，動作記述から RTL 回路を合成する技術である[2]．動作合成にはグラフ生成，スケジューリング，バインディング，RTL 回路記述生成の４つのステップがある(図 1).

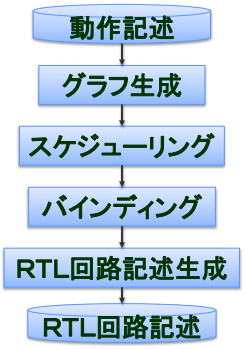

図 1. 動作合成の流れ

グラフ生成では与えられた動作記述をグラフで表現する．グラフには DFG(Data Flow Graph)[1]を用いる．スケジューリングでは各演算の依存性を保ちながら，各時刻に演算操作を割当てる．バインディングではスケジューリング済みの DFG(SDFG)を基に各演算操作や変数に具体的な演算器やレジスタを割当てる．RTL 回路記述生成は割当てられた演算器やレジスタ間を接続し，RTL 回路を生成する．

2.1. スケジューリング

スケジューリングの基本的な手法として、ASAP(As Soon As Possible)[4]や ALAP(As Late As Possible)[4]がある．ASAP は可能な限り早い時刻に演算操作を割当てるスケジューリング手法である．ALAP は可能な限り遅い時間に演算操作を割当てるスケジューリング手法である．ASAP および ALAP では，使用リソース数を考慮していないため，多くの演算操作を同時刻に割当て，面積のオーバーヘッドが大きくなることがある．面積効率のよい SDFG を得るア

Evaluation of binding methods for testability
to reduce the number of feedback loops

Narumi MORITA and Toshinori HOSOKAWA

ルゴリズムとして FDS(Forse Directed Scheduling) [5]が提案されている．本稿では，スケジューリング手法に FDS を用いる．図 2 は SDFG である．

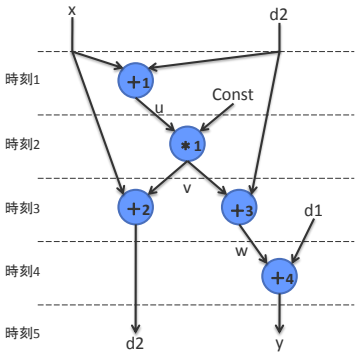

図 2. スケジューリング済み DFG

2.2. バインディング

SDFG において，各変数をレジスタに割当てる操作や，各演算を演算器に割当てる操作をバインディングという．SDFG から演算操作使用時刻や変数使用時刻を表すライフタイムを求め，ライフタイムが衝突しないように演算操作を演算器に，変数をレジスタに割当てる．

バインディングの基本的な手法として、レフトエッジアルゴリズム(LEA)が提案されている．LEA は面積最小化を考慮したバインディング手法である．

図 3 に図 2 の SDFG のバインディングを行った RTL 回路を示す．

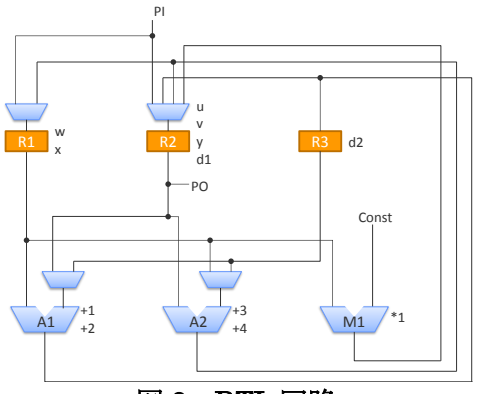

図 3. RTL 回路

3. フィードバックループ

部分スキャン設計は小さい面積・遅延・消費電力オーバーヘッドでテスト容易な回路を実現するための重要な技術の一つである．部分スキャン設計の一つの方法として、無閉路部分スキャン設計[6]が提案されている．順序回路において、回路内のフィードバックループにはセルフループとグローバルループがある．セルフループは同じレジスタの出力から入力までの経路が存在し，その経路中には他のレジスタが存在しないループである．グローバルループはレジスタの出力から入力までの経路が存在し，その経路中に他のレジスタが存在するループである．無閉路部分スキャンとはループ内のレジスタをスキャンレジスタに置き換えることで回路内のループを切断し，テストを容易にする設計法である．テスト時にスキャンレジスタは外部入出力とみなすことができ，値を任意に設定することが容易になる．セルフループを構成している場合はセルフループを構成しているレジスタ，グローバルループを構成している場合はグローバルループ内の一つのレジスタをスキャンレジスタに置き換える．

フィードバックループは順序回路のテスト生成を困難にする．フィードバックループが存在することで順序深度[7]が大きくなり，一般的な順序回路のテスト生成方法である時間展開モデルの時間展開数が有限化できない．順序回路を無閉路構造にすることで回路の順序深度を 1 に有限化でき，テスト容易化される[3].

図 4 の RTL 回路でのセルフループは，レジスタ R1 と加算器 A2 から構成されるセルフループ，レジスタ R2 と加算器 A1 から構成されるセルフループ，レジスタ R3 と加算器 A1 から構成されるセルフループの 3 つが存在する．

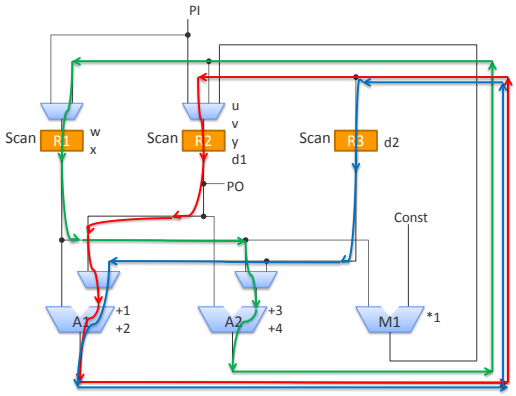

図 4. RTL 回路のセルフループ

4. テスト容易化バインディング

テスト容易化バインディングの手法として、無閉路部分スキャン設計に基づくデータパスのテスト容易化バインディング手法[2]がある．順序回路を無閉路構造にするためには，フィードバックループ内のレジスタを少なくとも一つはスキャンレジスタに置き換える必要がある．セルフループはレジスタを一つのみ通るループであるため，セルフループ内のレジスタはスキャンレジスタに置き換える必要がある．よって，セルフループが多い場合，スキャンレジスタ数も多くなる．手法[2]では，RTL データパス回路内に発生するセルフループをできる限り削減し，残ったセルフループやグローバルループに対してはスキャンレジスタに置き換えることで，少ないスキャンレジスタ数で無閉路構造を構成し，テスト容易化を実現する．

この手法では演算両立グラフ，レジスタ両立グラフでクリーク数最小のクリーク分割を行うことで面積最小化を実現する．また，両立グラフの各辺や各頂点に重みをつけ，重み和最小クリーク分割を行うことで，面積最小化かつテスト容易化を考慮したバインディングを行う．両立可能な 2 つの演算間に経路が存在すると，その演算の共有によってループができる．その経路上の少なくとも一つの変数はスキャンレジスタに割当てられる変数(スキャン変数)となる．経路の長さ(経路上の変数の数)が大きいほど，スキャン変数は選択しやすくなる．よって，演算両立グラフの辺の重みは，最短経路長が長いほど小さな重みをつける．重みをつけた演算両立グラフを重み和最小クリーク分割を行い，それを基に演算共有グラフ[2]を生成する．演算共有グラフで両立可能な 2 つの変数間に経路が存在し，それらの変数を 1 つのレジスタで共有すると，変数間の経路は共有したレジスタを通るループとなる．よって，その経路内のいずれかの変数はスキャンレジスタに割当てなければならない．特に，経路が隣接している場合は，セルフループが生じるので，そのレジスタはスキャンレジスタに割当てなければならない．また，演算共有グラフでセルフループを構成している変数もスキャンレジスタに割当てる必要がある．レジスタ両立グラフの頂点の重みは演算共有グラフで変数がセルフループを構成しているときに大きな重みをつける．レジスタ両立グラフの辺の重みは演算共有グラフで変数間に経路が存在するときに大きな重みをつける．また，経路が隣接しているときはさらに大きな重みをつける．

図 5 に無閉路部分スキャン設計を指向したデータパスのバインディング手法によりバインディングを行った結果の RTL 回路を示す．

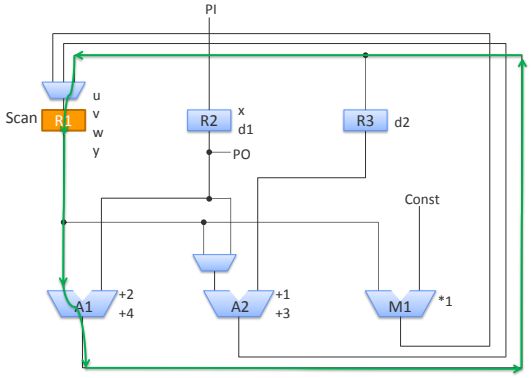

図 5. RTL 回路(テスト容易化)

図 5 の RTL 回路内のセルフループ数はレジスタ R1 と加算器 A1 から構成されるループ内のセルフループ数は 1 つに削減でき，LEA よりもセルフループ数を削減することができた．

5. 実験結果

ATPG ツールは synopsys 社の TetraMAX と STAGY[8] を使用する．TetraMAX は必要に応じて故障毎に時間展開数をかえてテスト系列を生成する．STAGY はすべての故障に対して同じ時間展開モデルで，時間展開数を指定してテスト系列を生成する．

表 1 に実験で使用した回路の特性を示す．本稿では二つの DFG から LEA と手法[2](PR1) でバインディングを行い，回路を生成した．表の＃PI は外部入力，＃PO は外部出力，＃reg はレジスタ数，＃add は加算器の数，＃sub は減算器の数，＃mlt は乗算器の数，＃MUX はマルチプレクサの入力総数，＃self_loop は回路内のセルフループ数，＃feedback は回路内のグローバルループ数，＃depth は順序深度，＃area は回路

内のゲート数(面積)である．手法[2]ではセルフループ削減のためのバインディング手法であったため，LEA に比べセルフループの数が削減されている．

表 2 にテスト生成結果を示す．ex2 の回路では LEA に比べると，手法 PR1 でテスト生成時間がかかり，故障検出率も低くなっている．self の回路では手法 PR1 の方がテスト生成時間は短く，故障検出率も高くなった．二つの回路においてフィードバックループ数の多い方が，テスト生成時間がかかり故障検出率も低くなっている．

6. おわりに

本稿では小規模な回路で従来法との比較実験を行った．２つの回路においてフィードバックループの存在がテスト生成を困難にしている可能性があることがわかった．

今後の予定として，対象回路を増やし，どのような回路で手法PR1が有効であるか検証したのち，本稿で対象としたテスト容易化バインディング手法の他にグローバルループを削減するバインディング手法や順序深度を削減するバインディング手法との比較実験を行う予定である．

参考文献

1) 藤原 秀雄,"ディジタルシステムの設計とテスト", 工学図書株式会社, 2004.

2) 高崎智也,井上智生,藤原秀雄,"無閉路部分スキャン設計に基づくデータパスのテスト容易化高位合成におけるバインディング手法", 電子情報通信学会論文誌(DI),Vol.J83-D-I No.2 ,2000.

3) V.Femandez and P.Shchez,"Partial Scan High-Level Synthesis", 1996 European Design and Test Conference (ED&TC '96),1996.

4) Giovanni De Micheli, "SYNTHESIS AND OPTIMIZATION OF DIGITAL CIRCUITS", McGraw-Hill,inc, 1994.

5) M. C. McFarland, A. C. Parker, R. Camposano,"The high-level synthesis of digital systems", Proc.IEEE, 1990.

6) H.Fujiwara, "Logic Testing and Design for Testability", The MIT Press, 1985.

7) Tien-Chien Lee, Wayne H. Wolf, Niraj K. J-ha, John M. Acken: "Behavioral Synthesis for Easy Testability in Data Path Allocation", Int. Conf.Computer Design,1992.

8) Kazuya SUGIKI, Toshinori HOSOKAWA, Masayoshi YOSHIMURA," A Test Generation Method for Datapath Circuits Using Functional Time Expansion Models", WRTLT'08, 2008.

表 1. 回路特性

	#PI	#PO	#reg	#add	#sub	#mlt	#MUX	#self loop	#global	#depth	#area
ex2(LEA)	4	1	5	0	1	2	16	3	6	3	1088
ex2(PR1)	4	1	5	0	1	2	17	1	8	3	1004
self(LEA)	5	1	5	1	0	1	12	3	3	3	1429
self(PR1)	5	1	5	1	0	1	11	2	1	2	1414

表 2. テスト生成結果

ATPGツール	Tetra Max				STAGY			
	総故障数	故障検出率(%)	故障検出効率(%)	テスト生成時間(s)	総故障数	故障検出率(%)	故障検出効率(%)	テスト生成時間(s)
ex2(LEA)	6018	98.54%	98.60%	3.3	1957	99.75%	99.75%	7.97
ex2(PR1)	5588	96.67%	96.67%	7.44	1822	94.24%	94.24%	375.09
self(LEA)	7496	99.04%	99.21%	790.37	3886	99.56%	99.59%	41.77
self(PR1)	7344	99.73%	99.80%	613.85	3944	99.90%	99.92%	10.02