7-49

平衡再収斂構造に着目したテスト容易化バインディング法

日大生産工（院）　○藤原 浩顕　　日大生産工(院) 井上諒一
日大生産工 細川　利典

1. はじめに

近年，半導体微細化技術の進歩に伴い大規模集積回路(Large Scale Integrated circuits:LSI)が大規模化，複雑化してきている．従来 LSI の設計において Verilog-HDL[1]や VHDL[2]などのハードウェア記述言語を用いてレジスタ転送レベル（Register Transfer Level:RTL）で開発するのが一般的だが，LSI の回路規模の増加に比べ設計生産性が向上していないため，従来の設計方法では設計が困難になってきている[3]．そこで，より抽象度の高い動作レベルで設計し，RTL の回路を合成する動作合成[4]という設計方法論が注目されてきている[4]．

動作記述は RTL 記述での設計に比べ，記述量が少ないため，設計生産性に優れるが，その反面，レジスタや演算器の割当てなどを動作合成ツールが決定するため，そのツールの性能が合成後の回路の性能に大きく影響する[4]．

一方，設計・製造された LSI は，不良品か否かを判定するためのテストが行われ，良品のみ市場に出荷されるが，LSI の大規模化，複雑化に伴い，テストも困難になってきている[3]．そのため，設計の初期段階でテスト容易化した設計が必要となってきている[3]．特に，動作合成のスケジューリングやバインディング時にテスト容易化を考慮する方法が提案されている[4]．

テスト容易化を考慮した動作合成手法として順序深度削減バインディング[5]やループ数削減バインディング[6]が提案され，その効果が報告されている．また，論理回路レベルでは，平衡再収斂構造[7]を削減することでテストを容易にできることが報告されている[7]．

本稿では，動作合成時に RTL での平衡再収斂構造削減を指向したバインディングを提案し，一般的に用いられている面積最小化バインディング法であるレフトエッジアルゴリズム(Left Edge Algorithm :LEA)[4]とテスト容易性について比較をした．

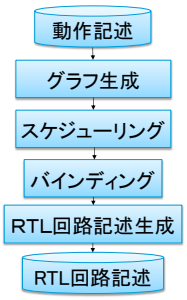

図 1．動作合成の処理

2. 動作合成

動作合成とは動作記述を読込み，RTL 回路を合成する技術である[4]．動作合成には図 1 のようにグラフ生成，スケジューリング，バインディング，RTL 回路記述生成といった 4 つのステップに分かれており，動作記述および演算器やレジスタの制約数を入力する．グラフ生成というステップでは，与えられた動作記述を変換しグラフを生成する．スケジューリングというステップでは，グラフに対して，与えられた制約に従って演算操作や変数をどの時刻に割当てるかを決定する．バインディングというステップでは，スケジューリングされたグラフに対して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．

各ステップの詳細については次章以降で説明する．

2-1. グラフ生成

動作記述から，グラフを生成する．例として，動作記述を以下に示す．

v=((a-((a*b)*(c*d)))*(b*(e*f)))*g　…(1)
w=(a-((a*b)*(c*d)))*(b*(e*f))　…(2)
x=f+(g*h)　…(3)
z=j+k　…(4)
y=i+(j+k)　…(5)

High Level Synthesis For Testability Based on Balanced Reconvergence Structures

Hiroaki FUJIWARA, Ryoichi INOUE, and Toshinori HOSOKAWA,

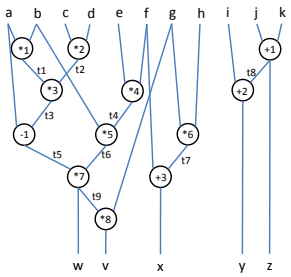

図 2. DFG

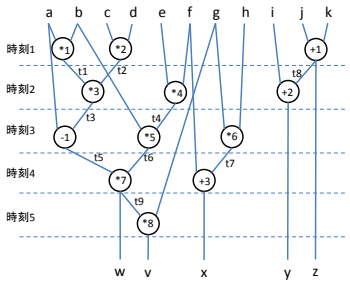

図 3. SDFG

式(1)～(5)をデータフローグラフ(Data Flow Graph:DFG)[4]で表現したものが図 2 となる．頂点が演算操作，辺が中間変数または外部入力，外部出力となる．各演算操作と中間変数には通し番号をつける．

2-2. スケジューリング

スケジューリングとは，DFGに対して，与えられた制約に従って演算操作や変数をどの時刻に割当てるかを決める処理である．図3は図2をスケジューリングしたスケジューリング済みDFG(Schedulinged Data Flow Graph:SDFG)である．

図 3 において，演算{*1,*2,+1}が時刻 1 にスケジューリングされ，それぞれの演算結果を格納する内部変数が{t1,t2,t8}に割当てられている．同様に，すべての演算が各時刻にスケジューリングされている．

2-3. バインディング

SDFG に対して，各演算や変数がどの時刻の間，値を保持しているか(ライフタイム)を解析する．
図 4，図 5 は変数と演算のライフタイムである．

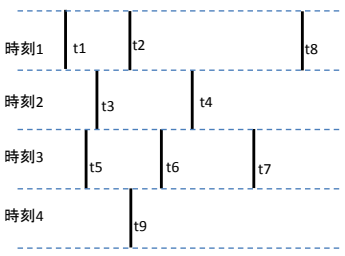

図 4. 変数ライフタイム

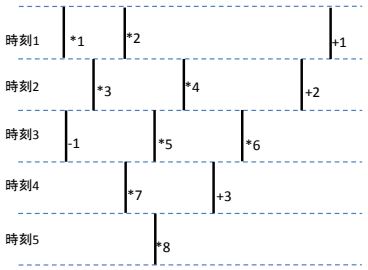

図 5. 演算ライフタイム

変数，演算それぞれについて，ライフタイムが異なる変数どうしを同じレジスタに，ライフタイムが異なり演算の種類が同じ演算どうしを同じ演算器に割当てる．例えば図4より{t1,t3,t5,t9}はライフタイムが異なるため，同じレジスタR1に割当てることができる．図5より，演算{*1,*3,*5,*7,*8}は同種の演算ではライフタイムが異なるため，同じ演算器MUL1に割当てることができる．このように割り当て可能な変数や演算を左に詰めて割当てを行うアルゴリズムがLEAである．LEAを用いて，他の変数，演算についても同様に割当てを行うと，表1のようになる．

表1. バインディング結果(LEA)

演算器:演算	レジスタ:変数
MUL1:*1,*3,*5,*7	R1:t1,t3,t5,t9
MUL2:*2,*4,*6	R2:t2,t4,t6
ADD1:+1,+2,+3	R3:t7,t8
SUB1:-1	

バインディング結果から各レジスタ，演算器，外部入力，外部出力を結線しRTL回路を合成した結果が図7である．

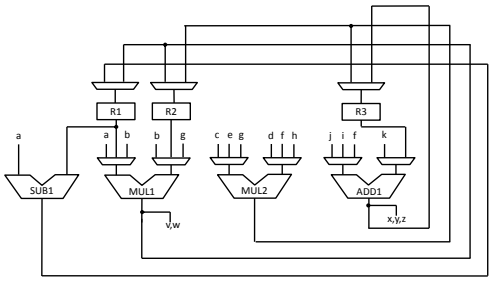

図 7. RTL 回路(LEA)

3. RTL 平衡再収斂構造

RTL平衡再収斂構造とはデータパスのレジスタからレジスタまでの経路上に同一個数のレジスタを通過する経路が複数存在している構造のことをいう．例えば図 8 は分岐点であるレジスタ FF1 から再収斂点 FF2 までに j 個のレジスタを通過する経路が k 本存在する「j 段 k 重平衡再収斂構造」である．

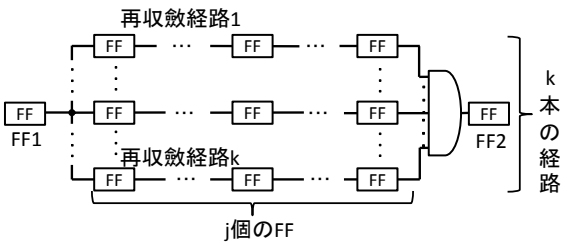

図 8. j 段 k 重 RTL 平衡再収斂構造

データパスから RTL 平衡再収斂構造を抽出する方法を以下に示す.

図 7 より，各レジスタの出力から演算器を通過して接続されるレジスタは以下のようになる.

$$R1 \rightarrow R1,R2 \quad \cdots(6)$$
$$R2 \rightarrow R1,R2 \quad \cdots(7)$$
$$R3 \rightarrow R3 \quad \cdots(8)$$

次に図 4 より，最後に使われる変数が 4 時刻目にあるため, 式(6)～(8)は図 9 のようなレジスタ関連グラフになる．1 時刻目に使われるレジスタを{$R1_A$,$R2_A$,$R3_A$}とし，2 時刻目のレジスタを{$R1_B$,$R2_B$,$R3_B$}，3 時刻目を{$R1_C$,$R2_C$,$R3_C$}，4 時刻目を{$R1_D$,$R2_D$,$R3_D$}とする.

図 9 から，平衡再収斂構造を抽出する．$R1_A$ を分岐点とすると, $R1_B$ と $R2_B$ を通過して R1c を再収斂点とする 1 段 2 重再収斂構造が構成されている．以降，{分岐点，再収斂点，段数，経路数}の形に省略して{$R1_A$,R1c,1,2}のように表す．他の再収斂構造を探索すると以下のようになる.

{$R1_A$,$R2_c$,1,2},{$R1_A$,$R1_D$,2,4},{$R1_A$,$R2_D$,2,4}
{$R2_A$,R1c,1,2},{$R2_A$,$R2_c$,1,2},{$R2_A$,$R1_D$,2,4}
{$R2_A$,$R2_D$,2,4},{$R1_B$,$R1_D$,1,2},{$R1_B$,$R2_D$,1,2}
{$R2_B$,$R1_D$,1,2},{$R2_B$,$R2_D$,1,2}

以上から，図 7 の回路には 1 段 2 重再収斂構造が 8 つ，2 段 4 重再収斂構造が 4 つ構成されていることがわかる.

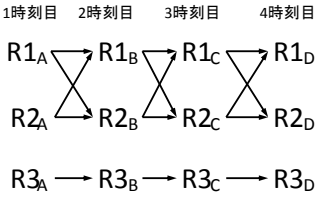

図 9. レジスタ関連グラフ(LEA)

4. 平衡再収斂構造削減指向バインディング

回路中に平衡再収斂構造が存在すると，テスト生成が困難になる可能性がある[7]．そこで，平衡再収斂数を削減するバインディングを考える．図 9 のレジスタ関連グラフより，レジスタから複数のレジスタへの接続があると分岐点や再収斂点が生まれ，平衡再収斂構造を構成する可能性が出てくる．これを RTL で考える．レジスタから複数の演算器への接続や，演算器から複数のレジスタへの接続があると，平衡再収斂の分岐点や再収斂点が発生することがわかる.

図 3 と表 1 の変数割当てから，変数 t5 と変数 t6 を考える．変数 t5 は演算 SUB1 の出力である．演算 SUB1 の出力は t5 のみであり，他には存在しない．変数 t6 は演算器 MUL1 の出力である．演算器 MUL1 の出力はほかに t1，t3，t9 が存在し，これらはすべてレジスタ R1 に割当てられている.

よって，t5 を R2 に，t6 を R1 に割当てることによって，演算器から R1 への接続を減らし平衡再収斂構造を削減できると考える．演算，変数の割当ては表 2 のようになる.

表 2. バインディング結果(平衡再収斂構造削減)

演算器:演算	レジスタ:変数
MUL1:*1,*3,*5,*7	R1:t1,t3,t6,t9
MUL2:*2,*4,*6	R2:t2,t4,t5
ADD1:+1,+2,+3	R3:t7,t8
SUB1:-1	

表 2 のバインディング結果から RTL を合成すると図 10 のようになる.

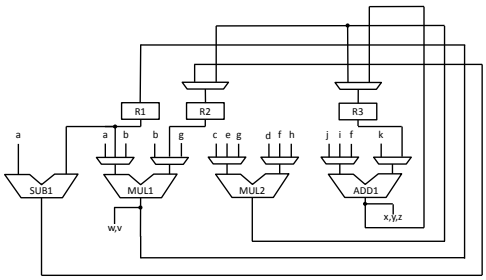

図 10. RTL 回路(平衡再収斂構造削減)

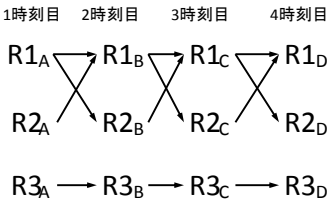

図 11. レジスタ関連グラフ(平衡再収斂構造削減)

図 10 から，平衡再収斂構造を抽出する．各レジスタの出力方向から演算器を通過して接続されるレジスタは以下のようになる．

R1→R1,R2 …(9)
R2→R2 …(10)
R3→R3 …(11)

式(7)と式(11)を比較すると，R2 から R1 への経路が削減されていることがわかる．式(9)～(11)からレジスタ関連グラフを生成すると図 11 となる．

図 11 から平衡再収斂構造を抽出すると

{R1A,R1c,1,2},{R1B,R1D,1,2}

となり，1 段 2 重再収斂構造が 2 つ構成されていることがわかる．LEA では 1 段 2 重再収斂構造が 8 つ，2 段 4 重再収斂構造が 4 つ構成されていたので，平衡再収斂削減指向バインディングを行うことにより，平衡再収斂構造数を削減することができた．

5. 実験結果

本論文では複数の回路に対してそれぞれ LEA を用いたバインディングと平衡再収斂削減指向バインディングを行い，生成された回路に対して Synopsos 社による ATPG ツール TetraMax を用いて実験を行った．表 3 はその実験結果である．

表 3. 実験結果

回路名	手法	故障検出率(%)	故障検出効率(%)	ATPG時間(s)
ex00	LEA	99.98	99.98	100.85
	提案手法	99.03	99.03	1444.31
ex01	LEA	99.89	99.89	157.34
	提案手法	99.59	99.59	1064.53

表 3 より，平衡再収斂削減指向バインディングを用いることにより LEA よりも平衡再収斂構造数が削減できたが，故障検出率は LEA が上回った．これは，今回は小規模回路で実験を行ったため，平衡再収斂構造を削減してもテスト容易性にあまり関係がなく，逆に割当てを変更したことでレジスタを任意の値に制御することが困難になったためと思われる．大規模回路で実験をすることで平衡再収斂構造削減の効果を得ることができ，テストが容易になると思われる．

6. おわりに

本論文では，平衡再収斂構造削減バインディングの実装を検討し，テスト容易性の評価を行った．今後の予定として，大規模回路での実装，ループ削減や順序深度削減などの他のテスト容易化バインディング法との比較・検証などが挙げられる．

参考文献

[1] Verilog Language Reference Manual, IEEE, 2001.
[2] VHDL Language Reference Manual, IEEE, 1987.
[3] 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.
[4] Daniel D. Gajski, Nikil D. Dutt, Allen CH Wu, and Steve Y-L Lin: HIGH-LEVEL SYNTHESIS Introduction to Chip and System Design, Kluwer Academic Publisher, 1992.
[5] Tien-Chien Lee, Wayne H. Wolf, Niraj K. Jha, John M. Acken: Behavioral Synthesis for Easy Testability in Data Path Allocation, Int. Conf. Computer Design,1992.
[6] TAKASAKI Tomoya ,INOUE Tomoo, FUJIWARA Hideo : A Binding Method in High-Level Synthesis for Testable Data Paths Based on Acyclic Partial Scan Design,IEICE, 2000.
[7]Toshinori Hosokawa, Toshihiro Hiraoka, Mitsuyasu Ohta,: A Testability Evaluation of Balanced Reconvergence Structures Based on Partial Scan Design, Information Processing Society of Japan, 1999.