ISSN 2186-5647

コントロール／データフローグラフを用いた順序深度削減指向バインディング法の評価

日大生産工（学部）　○兒玉　雄佑　　日大生産工　細川　利典

1. はじめに

近年，半導体技術の進展に伴い，大規模集積回路(Large Scale Integrated circuits: LSI)の規模や複雑度が飛躍的に増大している．これにより，設計される LSI が急速に大規模化しており，設計が困難になっている．この問題を解決するためには，LSI の設計生産性を向上させる必要があり，抽象度の高い動作レベルでの設計が提案されてきた[1]．動作記述により設計された回路を，レジスタ転送レベル（Register Transfer Level: RTL）回路に合成する技術として動作合成[1][2][3]が存在する．

設計された LSI は，不良品であるか否かをテストし，良品であると判定された LSI のみを出荷しなければならない．しかしながら， LSI の大規模化に伴い LSI 設計に普及しているスキャン設計された LSI のテストコストが劇的に増加している[4]．これにより，非スキャンに基づく，テスト容易性を考慮した動作合成が必要とされている[3]．

これまでの研究でテスト容易性を考慮した動作合成アルゴリズムとして，バインディング時に順序深度を削減することにより，テスト容易化を実現する方法が提案されている[5]．しかしながら，文献[5]の手法は回路のデータパスのみに着目した手法となっており，データパスとコントローラがテスト時に分離されていることが前提となる．テスト時にデータパスとコントローラを分離するために必要な回路面積などのハードウェアオーバヘッドは大きく，実際分離できない場合もある．それゆえ，コントローラを無視してデータパスのみのテスト容易化設計を行っても，データパスとコントローラが分離されない回路に対してテスト容易性の向上が得られない場合があると考えられる．

本論文では，コントローラの動作を含めたテスト容易化バインディング手法の研究の前段階として，従来のデータパスのテスト容易化バインディング手法がデータパスとコントローラが分離されていない回路に対してもテスト容易性の向上効果があるか否かを評価する．

2. 動作合成

動作合成とは動作記述を読込み，RTL の回路を合成する技術である[1][2][3]. 図 1 に動作合成の流れを示す．動作合成には，コントロールデータフローグラフ(Control Data Flow Graph:CDFG)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 つの主なステップがある．

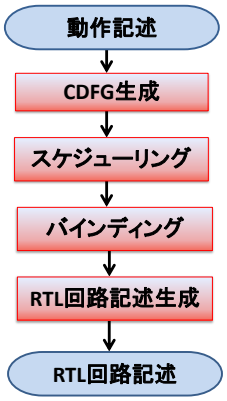

図 1. 動作合成の流れ

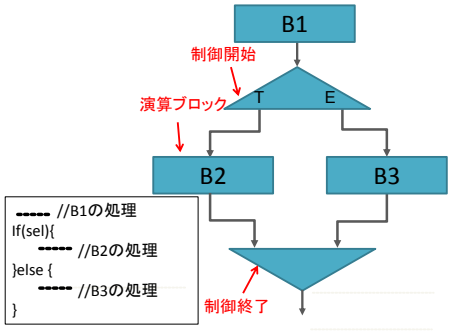

図 2. if 文の CFG と動作記述

2.1. CDFG 生成

CDFG 生成とは与えられた動作記述から CDFG を作成することである．CDFG の表現方法は，コントロールフローグラフ（Control Flow Graph: CFG）とデータフローグラフ（Data Flow Graph: DFG）を分けて表現する方法と，CFG と DFG を 1 つのグラフで表現する方法の 2 種類がある．CFG とは動作記述の制御文の構造を示すグラフであり，DFG とは演算操作の構造を示すグラフである．

2.1.1 CFG

図 2 に if 文の CFG を示す. CFG は演算ブロック，制御開始，制御終了の 3 つの要素から構成される．演算ブロックとは DFG の部分であり，演算処理のみで構成される部分を表す．図 2 では B1，B2，B3 が

Evaluation of Binding Methods to Reduce Sequential Depths Using Control Data Flow Graphs

Yusuke Kodama,Toshinori Hosokawa

```
1.  g=(a+b)+(d+e)
2.  break;
```

図 3. B2 の動作記述例

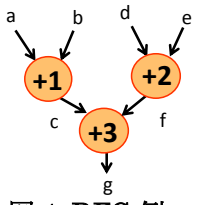

図 4. DFG 例

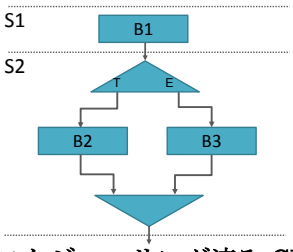

図 5. スケジューリング済み CFG

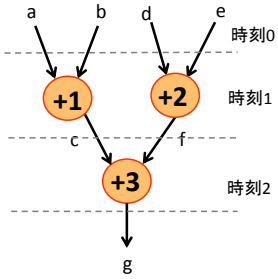

図 6. スケジューリング済み DFG

該当する. 制御開始とは, 制御文の開始部分を表す. if 文や switch 文などの条件分岐や，for 文や while 文のようなループ処理の判定が該当する．図 2 では if 文の条件が真(Then)であった場合に T の処理が行われ, 偽(Else)であった場合は E の処理が行われる. 制御終了とは，条件分岐やループ処理の終了部分を表す.

2.1.2. DFG

DFG は LSI における演算処理を表現するグラフで，入力変数，出力変数，内部変数，演算操作から構成される．図 2 の B2 の処理が図 3 の動作記述であった場合の DFG を図 4 に示す.

2.2. スケジューリング

スケジューリングとは CFG，DFG に対して，各演算操作をどの時刻で実行するかを割当てる処理である．基本的なスケジューリングアルゴリズムとして, ASAP(As Soon As Possible)[6]や ALAP(As Late As Possible)[6]などがある. 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

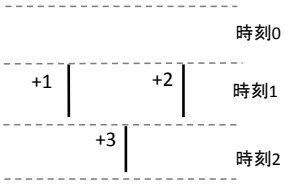

図 7. 演算（加算）のライフタイム

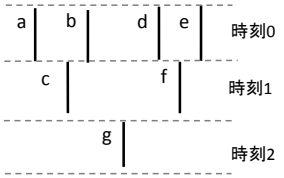

図 8. 変数のライフタイム

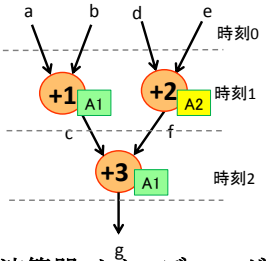

図 9. 演算器バインディング例

ある．ALAP は可能な限り遅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．図 5 に図 2 の CFG をスケジューリングリングした例を，図 6 に図 4 の DFG をスケジューリングした例をそれぞれ示す. なお，スケジューリング済みの DFG を SDFG（Scheduled Data Flow Graph）[1]と呼ぶ.

2.3. バインディング

バインディングとは，SDFG に存在する演算操作および変数にそれぞれ演算器やレジスタを実際に割当てる処理である．演算器やレジスタを最適に割当てるためには各演算操作や変数に対してライフタイムを求める必要がある．ライフタイムとは，演算処理の始まりから終わりまでの時刻と，変数が値を保持し始める時刻から，値を保持しなくなる間の時刻である．ライフタイムが重ならない演算操作や変数を，同じ演算器やレジスタで共有することができる. ライフタイムには演算処理に関するライフタイムと，変数に関するライフタイムがある．例として，図 7 に加算器のライフタイムを，図 8 に変数のライフタイムを示す.

2.3.1 演算器バインディング

図 7 に対して，演算器バインディングを行う．演算{+1,+2}のライフタイムが時刻 1，{+3}のライフタイムが時刻 2 なので，ライフタイムが重ならない演

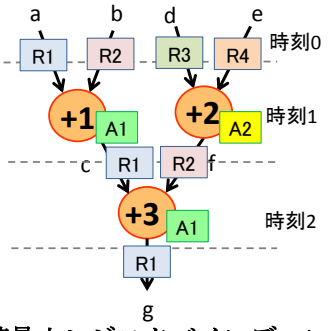

図 10. 面積最小レジスタバインディング例

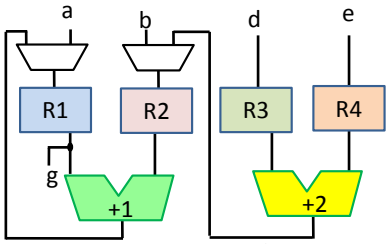

図 11. 面積最小化バインディング RTL データパス回路

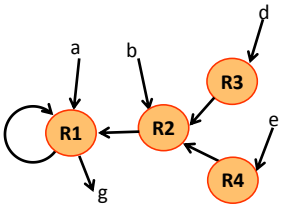

図 12. 面積最小化バインディングの順序深度

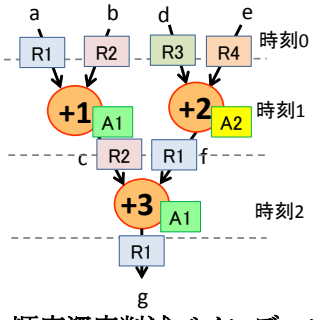

図 13. 順序深度削減バインディング例

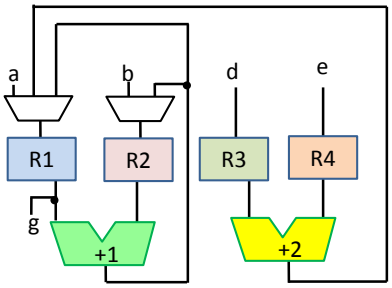

図 14. 順序深度削減バインディング RTL データパス回路

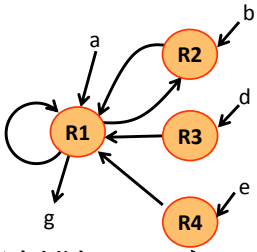

図 15. 順序深度削減バインディングの順序深度

算どうしを同じ演算器に割当てる. 図 9 に{+1}と{+3}を加算器 A1 に，{+2}を加算器 A2 に割当てた結果を示す.

2.3.2. 面積最小化レジスタバインディング

図 8 に対してそれぞれの変数のライフタイムを計算する. 図 8 の変数{a,b,d,e}のライフタイムが時刻 0，変数{c,f}のライフタイムが時刻 1， 変数{g}のライフタイムが時刻 2 である. ライフタイムが重ならない変数どうしを同じレジスタに割当てるようにバインディングを行う. 図 10 に面積最小化を目的としたレフトエッジアルゴリズム[7]を用いたバインディング結果を示す. 変数{a,c,g}をレジスタ R1 に，変数{b,f}をレジスタ R2 に，変数{d}をレジスタ R3 に，変数{e}をレジスタ R4 にそれぞれ割当てた. また，図 10 のバインディング情報をもとに RTL データパス回路生成した結果を図 11 に示す.

2.3.3. テスト容易化レジスタバインディング

テスト容易化バインディング[8]として，レジスタバインディング時に，可制御性および可観測性の向上を行い，制御も観測も困難なレジスタを削減することを試みる. これにより，可制御性と可観測性が高い回路が合成される. また，順序深度[8]を削減し，RTL データパス回路のテスト容易化を試みる. 順序深度とは，入力レジスタから出力までの最短経路におけるレジスタの段数で，各入力レジスタの最大の順序深度がデータパスの順序深度と定義する. 図 12 に図 11 のデータパスの順序深度を調べるためのグラフを示す. グラフの頂点がレジスタ，辺が演算器とマルチプレクサのみを通ってレジスタに到達できることを示す. 各入力変数が割当てられたレジスタ（入力レジスタ）から出力までの経路間に存在する頂点の数が各入力レジスタの順序深度である. 図 12 では入力レジスタは{R1,R2,R3,R4}で,出力レジスタは{R1}である. 入力レジスタそれぞれに対して順序深度を調べると，R1→g は 0，R2→g は 1，R3→g は 2，R4→g は 2 となり，この回路の順序深度は各レジスタの順序深度のうち最大値である 2 となる.

次に，順序深度削減を目指したバインディングを

表 1. 実験結果

	バインディング手法	TC(%)	FC(%)
FSMなし	LEA	99.61	99.10
	順序深度削減	99.85	99.34
FSMあり	LEA	79.95	5.12
	順序深度削減	97.98	36.43

行う．2.3.1 節における演算器バインディングでは，演算{+1}と{+3}を同じ演算器に割当てたので，演算{+1}の出力である変数{c}は演算{+3}の出力{g}で観測することができる．また，演算{+2}と{+3}は別の演算器に割当てたため，変数{f}は出力{g}で観測することができない．そこで，変数{c}をレジスタ{R2}に割当て，代わりに変数{f}をレジスタ{R1}に割当てる．順序深度を考慮したレジスタバインディング結果を図 13 に，図 13 のバインディング情報をもとに RTL データパス回路生成した結果を図 14 に，図 14 のデータパスの順序深度を調べるためのグラフを図 15 に示す．図 15 での入力レジスタは{R1,R2,R3,R4}で，出力レジスタは{R1}である．入力レジスタそれぞれに対して順序深度を調べると R1→g は 0，R2→g は 1，R3→g は 1，R4→g は 1 となる．この回路の順序深度は，各レジスタの順序深度の最大値から 1 である．よって，図 14 の回路は図 10 の回路に比べ，順序深度を削減することができた．

3. 実験結果

本論文では，文献[8]に掲載されている回路を参考にした実験回路に対して，レフトエッジアルゴリズムと順序深度削減バインディングを適用し，データパスのみで合成した回路と，コントローラを含めた回路全体で合成した回路に対し，テスト生成を行った．ATPG ツールは Synopsys 社の TetraMax を用いた．実験結果を表 1 に示す．なお，表 1 の TC は故障検出効率，FC は故障検出率を表している．

データパスのみで合成した回路の結果に注目すると，故障検出効率が 99.61%から 99.85%に値が上昇しているので，順序深度削減の効果が得られたことがわかる．次にコントローラを含む回路全体で合成した結果を見てみると 79.95%から 97.98%に値が上昇し，本論文で使用した回路において順序深度削減の効果が得られた．

4. おわりに

本論文では，データパスのテスト容易化バインディング手法がデータパスとコントローラが分離されていない回路に対してもテスト容易性の向上効果があるか否かを実験し，評価した．

実験結果より，データパスとコントローラが分離されていない回路に対しても，ある程度テスト容易性の向上効果が見られた．また，コントローラを含む回路の故障検出効率は，コントローラを含まない回路の故障検出効率に比べて，1.87%～19.66%低下している．

今後の課題として，表 1 の実験結果の詳細な解析と，同様の実験を他の回路で行うことが挙げられる．

参考文献

[1]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.

[2]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

[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]Y. Sato, T. Ikeda, M. Nakao, and T. Nagumo, “A bist approach for very deepsub-micron (vdsm) defect, ”Proc. International Test Conference ，pp. 283-291，2000.

[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, pp29-32, 1992.

[6]GiovanniDe Micheli, “SYNTHESIS AND OPTIMIZATION OF DIGITAL CIRCUITS”, McGraw-Hill,inc, 1994.

[7]F.J.Kurdahi and A.C.Parker, REAL:A program for register allocation, In Proc. Design Sutomation Conf.. pp210-215, 1987.

[8]Mike Tien-Chien Lee “High-Level Test Synthesis of Digital VLSI Circuits”,Artech House Publishers,1997.