7-18

ADD を用いた機能 RTL 回路に対する順序深度削減のためのバインディング法

日大生産工（学部）　○藤原 浩顕　　日大生産工　細川　利典
日大生産工(院)　長　孝昭

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 Integration:LSI)が大規模化, 高集積化してきている. 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, 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.

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]が提案され, その効果が報告されている.

手法[5]では動作記述をグラフ化するために DFG(Data Flow Graph)[4]を用いている. DFG はデータパスしか表現することができないためコントローラとデータパスが完全に分離されている回路にしか適用されない. コントローラとデータパスが混在している回路をグラフ化したものに ADD(Assignment Decision Diagram) [6]というグラフが存在する.

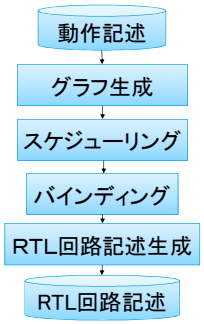

図 1. 動作合成の処理

本論文では ADD でグラフ化された機能 RTL に対してのテスト容易化用バインディングを提案する. 機能 RTL とは, スケジューリングまで完了した RTL 回路である.

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[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, 入力して動作記述ファイル及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り当てる.

各ステップの詳細については次章以降で説明する.

A binding method to reduce sequential depths for functional RTL Circuits Using Assignment Decision Diagram

Hiroaki FUJIWARA, Toshinori HOSOKAWA, and Takaaki CHO

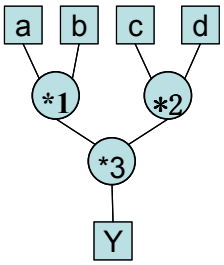

図 2. ADD 生成

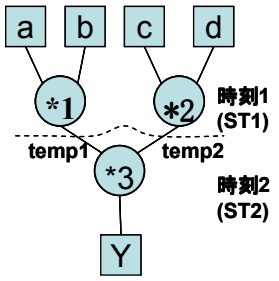

図 3. 時刻割当て

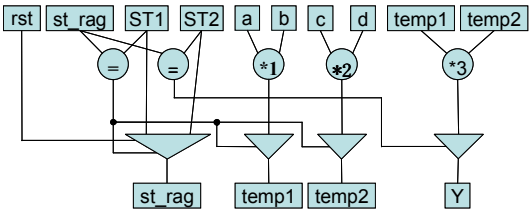

図 4. スケジューリング済み ADD

2-1. グラフ生成

動作記述から，グラフを生成する．グラフの表現方法には様々な種類が存在する[3]が，本研究ではグラフにデータパスとコントローラが混在するモデルを表現できる ADD を用いた．例として，動作記述を(1)で示す．

$$Y=(a*b)*(c*d) \quad \cdots(1)$$

図 2 に式(1)を表現した ADD を示す．

2-2. スケジューリング

スケジューリングとは，ADD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 において，演算{*1}と演算{*2}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{temp1}，{temp2}に割当てている．さらに，演算{*3}が時刻 2 に割当てられ，演算結果を変数{Y}に格納する．図 4 に図 2 の ADD を図 3 に示すようにスケジューリングした ADD を示す．

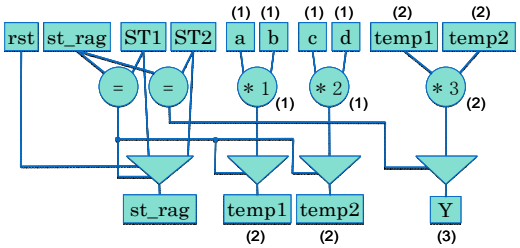

図 5. ライフタイム割当て済み ADD

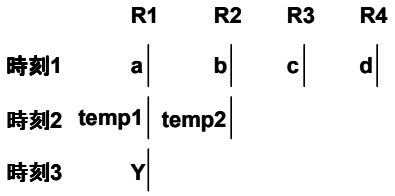

図 6. 面積最小化レジスタバインディング

2-3. バインディング

スケジューリングされたADDに対して，各演算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の各演算と変数のライフタイムを解析した結果を図5に示す．ただし，各変数，演算子に対するライフタイムを(A,B,C,…)と表現する．A,B,Cは変数や演算子を使用する時刻である．

図5から，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(1)，{*3}のライフタイムが(2)なので，ライフタイムが重ならない演算同士を同じ演算器に割当てる．本論文では{*1}と{*3}を乗算器mul1に，{*2}を乗算器mul2に割当てた．

次にレジスタバインディングを行う．図6に図5を基にレジスタ数(面積)最小化を指向したレジスタバインディング結果を示す．

図5より各変数のライフタイムを計算する．図5の変数{a,b,c,d}のライフタイムが(1)，変数{temp1,temp2}のライフタイムが(2)，変数{Y}のライフタイムが(3)である．ライフタイムが重ならない変数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6は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,temp1,Y}をレジスタR1に，変数{b,temp2}をレジスタR2に，変数{c}をレジスタR3に変数{d}をレジスタR4にそれぞれ割当てた．

面積最小化バインディング結果からRTL回路を合成した結果が図7である．

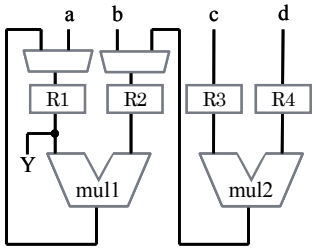

図 7. 面積最小化バインディング RTL 回路

3. 順序深度削減バインディング

動作合成, 論理合成後回路のテストを容易にするために, 面積を可能な限り最小化しつつバインディング時に可制御費と可観測費を強化し, 制御も観測も困難である信号線を可能な限り減らす.

図 8 は図 7 のデータパスの順序深度[3]を解析するためのグラフある. グラフの頂点がレジスタ, 辺がレジスタ間の接続関係を示す. 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をたどり, 間に存在するレジスタの数が各入力レジスタの順序深度である.

順序深度削減バインディング[3]とは，テスト容易化バインディングの手法の一つである. 各入力レジスタの順序深度を解析し, 順序深度を可能な限り最小にする.

図 8 において，入力レジスタは{R1,R2,R3,R4}で, 出力レジスタは{R1}である．各入力レジスタに対して順序深度を調解析すると，R1→Y は 0, R2→Y は 1，R3→Y は 2，R4→Y は 2 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 2 となる.

次に, 順序深度を考慮したバインディングを行う. 演算器バインディングでは{*1}と{*3}を同じ演算器に割当てたので，演算{*1}の結果である変数{temp1}は演算{*3}の出力{Y}で観測することができる．また，{*2}と{*3}は別の演算器に割当てたため, {temp2}は{Y}で観測することができない. そこで，{temp1}を{R2}に割当て，代わりに{temp2}を{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は図 9 のようになる．図 9 を基に RTL 回路記述を合成した結果を図 10 に示す.

図 10 の回路は，面積最小化バインディングから合成された図 7 と比べると, 演算器{mul2}と接続するレジスタが{R2}から{R1}になり，演算器{mul1}と接続するレジスタが{R1}一つから{R1}と{R2}の二つになる．また，マルチプレクサの数や信号線の本数が増えたため回路面積が増したが, 回路全体の面積に占める割合は非常に小さいため考慮には値しない.

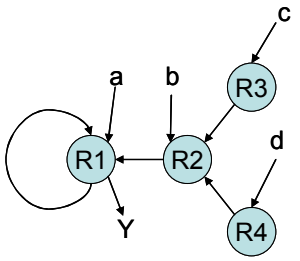

図 8. 面積最小化バインディングの順序深度

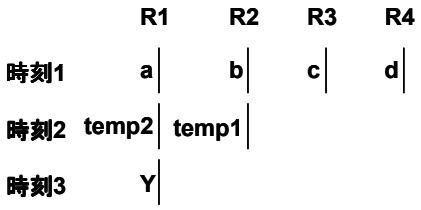

図 9. 順序深度削減レジスタバインディング

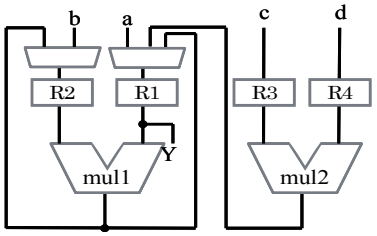

図 10. 順序深度削減バインディング RTL 回路

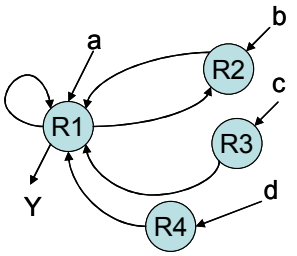

図 11. 順序深度削減バインディングの順序深度

図 10 のデータパスの，順序深度を解析するためのグラフを図 11 に示す.

図 11 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. 各入力レジスタそれぞれに対して順序深度を調べると R1→Y は 0，R2→Y は 1，R3→Y は 1，R4→Y 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 10 の回路は図 7 の回路に比べ, 順序深度を削減することができ，テストが容易になると考えられる.

4. 実験結果

本論文では動作記述 ex01(2), ex02(3)の 16bit,

32bit それぞれに面積最小化バインディングと順序深度削減バインディングを行い, 生成された回路に対して ATPG ツールを用いて実験を行った. 表 1 はその実験結果である.

$$Y=(a*b)*(c*d) \quad \cdots(2)$$

$$Y=(a*b)*(c*d)*(e+f) \quad \cdots(3)$$

表 1 より, すべての回路において順序深度削減手法を用いたことにより故障検出率もテスト長も面積最小化バインディングを上回った.

5. おわりに

本論文では, 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の実装を検討し, テスト容易性の評価を行った. その結果, 故障検出率もテスト長も面積最小化バインディングを上回った.

参考文献

[1] IEEE Standard 1076: Verilog Language Reference Manual, IEEE, 2001.

[2] IEEE Standard 1076: 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 C-H Wu, and Steve Y-L Lin, : HIGH-LEVEL SYNTHESIS Introdu-ction 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,29-32,1992.

[6] V.Chaiyakul and D.D. Gajski, "Assignment Decision Diagram and for high-level Synthesis, " Technical Report #92-103, Unive rsity of Calfornia Irvine, Oct.1992.

[7] F.J.Kurdahi and A.C.Parker, REAL: A program for register allocation, In Proc. Design Sutomation Conf.. 210-215, 1987.

表 1. 実験結果

回路	バインディング	レジスタ割当て	演算器割当て	テスト検出効率	故障検出率	総故障数	面積	テスト長
ex1_16b	面積最小化	R1=(a,temp1,Y) R2=(b,temp2) R3=(c) R4=(d)	mul1=(*1,*3) mul2=(*2)	97.72%	97.72%	8700	1940	2596.7
	順序深度削減	R1=(a,temp2,Y) R2=(b,temp1) R3=(c) R4=(d)	mul1=(*1,*3) mul2=(*2)	99.33%	99.33%	8882	2008	447.99
ex2_16b	面積最小化	R1=(a,temp1,temp3,Y) R2=(b,temp2,temp4) R3=(c,e) R4=(d,f)	mul1=(*1,*3,*4) mul2=(*2) add1=(+1)	98.77%	98.77%	9730	2142	1581.68
	順序深度削減	R1=(a,temp2,temp3,Y) R2=(b,temp1,temp4) R3=(c,e) R4=(d,f)	mul1=(*1,*3,*4) mul2=(*2) add1=(+1)	99.63%	99.63%	9972	2210	1477.54
ex1_32b	面積最小化	R1=(a,temp1,Y) R2=(b,temp2) R3=(c) R4=(d)	mul1=(*1,*3) mul2=(*2)	99.37%	99.37%	34508	7336	6848.12
	順序深度削減	R1=(a,temp2,Y) R2=(b,temp1) R3=(c) R4=(d)	mul1=(*1,*3) mul2=(*2)	99.71%	99.71%	34974	7468	1968.26
ex2_32b	面積最小化	R1=(a,temp1,temp3,Y) R2=(b,temp2,temp4) R3=(c,e) R4=(d,f)	mul1=(*1,*3,*4) mul2=(*2) add1=(+1)	99.53%	99.53%	36594	7746	5239.91
	順序深度削減	R1=(a,temp2,temp3,Y) R2=(b,temp1,temp4) R3=(c,e) R4=(d,f)	mul1=(*1,*3,*4) mul2=(*2) add1=(+1)	99.83%	99.83%	37060	7878	2294.09