ISSN 2186-5647

2-28

テスト容易化機能的時間展開モデル生成のためのバインディング法

日大生産工(院)　○佐藤 護　　日大生産工(院)　増田 哲也　日大生産工(院)　西間木 淳
日大生産工　細川 利典　　大阪学院大学　藤原 秀雄

1. はじめに

近年，半導体集積技術の発展に伴い，設計される大規模集積回路(Large Scale Integrated circuits:LSI)の大規模化，複雑化が急速に進展している[1]．これにより，LSIの設計コストの増大が問題視されている．LSIの設計生産性を向上させる手段として，動作レベルの回路記述からレジスタ転送レベル(Register Transfer Level:RTL)の回路を生成する動作合成が提案されている[1][2]．動作レベルの回路記述は，RTLの回路記述と比較して，高い抽象度で記述されるため，設計生産性に優れる．

一方，LSIの高速化，高集積化に伴い，一般的なLSIテスト方法であるスキャンテスト[3]の実行時間が増加している．それに加え，フルスキャン設計[3]が施された回路に対するサイドチャネル攻撃手法が提案されており[4]，セキュリティ面でも問題があることが報告されている[4]．そのため，非スキャンテストで品質の高いテストを実行することが理想的である．非スキャンテストに基づくテスト容易性を考慮した動作合成が提案されている[5-7]．これらの手法は，RTL回路のデータパスのテスト容易性に着目した手法である．そのため，データパスに対する高い故障検出率を実現するためには，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は付加回路が必要である．一方，データパスとコントローラを分離しないことを前提としたRTL回路に対するテスト容易化設計手法も提案されている[8]．

文献[8]では，データパスのテスト容易な構造に基づいて，テスト時にその動作を制御可能とするようにコントローラを拡大する．しかしながら，従来の順序回路のテスト生成アルゴリズムは，回路構造にのみ着目してテスト系列を生成する．すなわち，テスト容易な構造に基づいてテスト系列が生成されるとは限らない．文献[9]は，機能検証用のテスト系列からコントローラの状態遷移を網羅するようにデータパスの機能的時間展開モデル[9]を生成し，テスト生成を行う方法が提案されている．文献[9]では，テスト生成時に回路の機能動作時における各時刻の制御信号．状態信号の値を制約として付与する．それにより，回路の機能動作に基づいてテスト生成を実行できる．

文献[10]ではデータパスのテスト容易な構造に着目したテスト容易化機能的時間展開モデル[10]を生成し，コントローラ拡大を適用し，テスト生成を行う方法が提案されている．文献[10]の手法では，まずデータパスのテスト容易な構造に基づいてテスト生成を行うためのテスト容易化機能的時間展開モデルの動作を制御するためにコントローラを拡大する．テスト生成時は，拡大したコントローラの機能に着目し，生成したテスト容易化機能的時間展開モデルの動作を制御するように各時刻の制御信号・状態信号値を制約として与える．文献[10]の手法は，テスト時にコントローラとデータパスを分離せずにテスト容易な動作を制御できる．さらに，テスト生成時には各時刻の制御信号・状態信号値が制約として与えられるため，比較的小さな時間展開数で，かつ解空間の狭いモデルを生成することができ，テスト生成の効率化が図れる．

本論文では，生成されるテスト容易化機能的時間展開モデルの時間展開数をさらに削減するために演算器の入力や出力の順序深度を削減するバインディング法を提案し，故障検出率の向上，テスト生成時間の削減を目指す．

2. 実験方法および測定方法

本章では，テスト容易化機能的時間展開モデルを用いたテスト生成[10]の説明を行う．

2.1. データパスのテスト容易な動作を制御するためのコントローラ拡大

コントローラとデータパスを分離せずにテストすることを前提としたテスト容易化手法として，コントローラを拡大する手法が提案されている[8]．文献[8]では，データパスのテスト容易な構造を基に，テスト容易な動作を実現するため，コントローラに状態遷移や状態を追加する．しかしながらテスト生成時に，データパスのテスト容易な動作をコントローラが制御するとは限らず，テスト生成が困難となる可能性がある．

A Binding Method to Generate Easily Testable Functional Time Expansion Models

Mamoru SATO , Tetsuya MASUDA , Jun NISHIMAKI , Toshinori Hosokawa , Hideo Fujiwara

2.2. 機能的時間展開モデルを用いたテスト生成

機能的時間展開モデルを用いたテスト生成[9]では，機能検証用のテスト系列から抽出したコントローラからデータパスへ入力される制御信号系列と，データパスからコントローラへ出力する状態信号系列を制約値として，指定された時間分展開された時間展開モデルに対してテスト生成を行う．機能的時間展開モデルとは，時間展開数が有限であり，さらに状態信号系列と制御信号系列が制約として付与された時間展開モデルである．そのため，制約が付与されない時間展開モデル（構造的時間展開モデル）を用いたテスト生成と比較して，テスト生成が高速になる．しかしながら，機能動作レイテンシが大きくなると時間展開数が大きくなり，そのモデルの構造がテスト容易であるとは限らないという問題点がある．

2.3. テスト容易化機能的時間展開モデルを用いたテスト生成

テスト容易化機能的時間展開モデル[10]では，データパスのテスト容易な構造に着目し，テスト容易化機能的時間展開モデルの生成を行う．さらに，生成されたテスト容易化機能的時間展開モデルの動作を制御するために，必要に応じて状態遷移を追加し，コントローラを拡大する．テスト生成時には，制御信号系列および状態信号系列を制約値として，指定された時間展開数分展開されたテスト容易化機能的時間展開モデルに対してテスト生成を行う．そのため，テスト生成が高速化・効率化される．また，テスト容易化機能的時間展開モデルの時間展開数は，データパスの機能動作レイテンシに依存しない．よって，時間展開数の小さなテスト容易化機能的時間展開モデルを生成することが可能となる．テスト容易化機能的時間展開モデルは，データパスのテスト容易な動作に着目して生成される．そのため，あるテスト対象モジュールに対して時間展開数 k 以下のテスト容易化機能的時間展開モデルを生成する場合，そのテスト対象モジュールを含むモデルが生成できない可能性があるという問題点がある．また，入力が定数のみの演算器を含むテスト容易化機能的時間展開モデルは，時間展開数が k 以下であっても，テスト生成が困難となる可能性がある．

3. 諸定義

本章では，テスト容易化機能的時間展開モデル生成のためのバインディングで使用する用語の定義を行う．

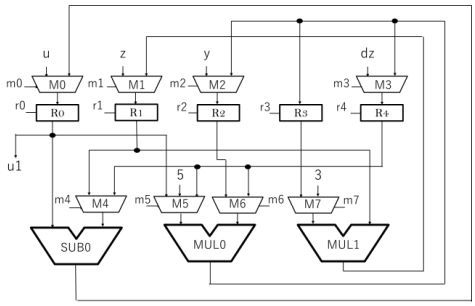

図3.1-1　データパス図

3.1. 演算器入力順序深度

演算器の入力から，任意の外部入力に到達するまでの各経路における最小のレジスタ数を演算器入力順序深度という．特に，演算器の左（右）入力の演算器入力順序深度を，演算器左（右）入力順序深度という．図3.1-1は演算器SUB0，MUL0，MUL１，レジスタR0～R4, マルチプレクサM0～M7から構成されるデータパス例である．

MUL1の演算器左入力順序深度は，MUL1の左入力→M7→R3→MUL0→M6→R2→M2→yの経路に存在するレジスタ数が2個で, MUL1の左入力までの全経路中で最小であるので，MUL1の演算器左入力順序深度は2となる．MUL1の演算器右入力順序深度は，MUL1の右入力→R1→M1→zの経路に存在するレジスタ数が1個で, MUL1の右入力までの全経路中で最小であるので，MUL1の演算器右入力順序深度は1となる．

3.2. 演算器出力順序深度

演算器の出力から，任意の外部出力に到達するまでの各経路における最小のレジスタ数を演算器出力順序深度という．図3.1-1を用いて説明を行う．MUL1の出力順序深度は，MUL1の出力→M1→R1→M4→SUB0→M0→R0→u1の経路に存在するレジスタ数が1個でMUL１の出力からの全経路中で最小であるので，MUL1の演算器出力順序深度は1となる．

3.3. 演算器バインディンググラフ

演算器バインディングに，無向グラフG（V,E,w）を用いる（演算器バインディンググラフ）．頂点v（v∈V）は演算器であり，辺（u,v）（(u,v)∈E, u,v∈V, u≠v）は隣接頂点である演算器uとvが共有可能であることを示す．各辺は重みがラベル付され，重みw : E→Z+(Z+は非負の整数)で示される．辺(u,v)の重みw(u,v)は演算器uとvを共有化した際の，演算器入出力順序深度の削減数である．演算器入力順序深度が無限大であるとき，その値は十分に大きな整数値（例．100）を用いて重みを計算する．

図3.3-1は，演算器バインディンググラフの例であり，5個の乗算器（*0～*4）の共有可能性を表現している．表3.3-1に，図3.3-1の各辺の重みを示す．

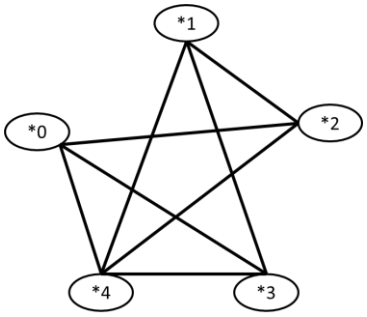

図3.3-1 演算器バインディンググラフ

表3.3-1　重み一覧

辺	重み	辺	重み
(*1,*2)	100	(*2,*4)	2
(*1,*3)	1	(*3,*0)	100
(*1,*4)	102	(*3,*4)	101
(*2,*0)	3	(*4,*0)	3

3.4. レジスタバインディンググラフ

レジスタバインディングに，無向グラフG(V,E,w)を用いる（レジスタバインディンググラフ）．頂点v（v∈V）はレジスタであり，辺（u,v）（(u,v)∈E, u,v∈V, u≠v）は隣接頂点であるレジスタuとvが共有可能であることを示す．各辺は重みがラベル付され，重みw：E→Z+ (Z+は非負の整数)で示される．辺(u,v)の重みw(u,v)はレジスタuとvを共有化した際の，全演算器の演算器入出力順序深度の削減数の総和である．

図3.4-1は，レジスタバインディンググラフの例であり，11個レジスタの共有可能性を表現している．表3.4-1は，図3.3-1に対して演算器バインディング（演算*0,*3,*4を共有化，演算*1,*2を共有化）が完了した後の，図3.4-1の辺の重みを示す．表3.4-1に，演算器入力および出力順序深度を削減できる辺の重みのみを示す．

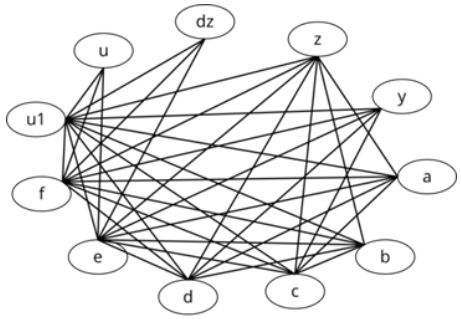

図3.4-1　レジスタバインディンググラフ

表3.4-1　重み一覧

辺	重み	辺	重み	辺	重み	辺	重み
(b,u1)	1	(d,e)	1	(c,y)	2	(c,a)	1
(c,u1)	1	(d,f)	1	(d,u1)	2	(f,y)	2
(f,u1)	1	(c,u)	2	(c,b)	1	(f,a)	1
(a,u1)	2	(c,dz)	2	(f,u)	2	(f,b)	1
(a,e)	1	(c,z)	2	(f,dz)	2	(f,z)	2
(a,f)	1						

3.5. 演算器バイディンググラフの重み和最大クリーク分割問題

入力：演算器バインディンググラフG
出力：演算器バインディングによって，どの演算がどの演算器に割当てられたかを示す情報である演算器バインディング情報
制約：演算器の個数
最適化：各クリークを構成する辺の重みの総和を最大化する，すなわち演算器バインディングが完了した時点での全演算器の演算器入出力順序深度の削減数の総和を最大にする．

3.6. レジスタバインディンググラフの重み和最大クリーク分割問題

入力：レジスタバインディンググラフG
出力：レジスタバインディングによって，どの変数がどのレジスタに割当てられたかを示す情報であるレジスタバインディング情報
制約：レジスタの個数
最適化：各クリークを構成する辺の重みの総和を最大化する，すなわちレジスタバインディングが完了した時点での全演算器の演算器入出力順序深度の削減数の総和を最大にする．

4.　実験結果

本論文では，実験対象回路を定数入力が含まれるEX2[11]，ARF[12]，BPF[12]，FFT[12]を対象に，提案するテスト容易化機能的時間展開モデル生成を考慮したバインディング法の有効性を示すために評価実験を行った．本実験では，提案するバインディング法と，従来手法として，小面積指向バインディング法であるレフトエッジアルゴリズム[2]によって生成された2通りのRTL回路を合成し比較実験を行った．テスト生成ツールはテスト容易化機能的時間展開モデル[10]のテスト生成が可能な内製のFTEM_ATPGを使用した．論理合成にはDesign Compilerを使用し，データパスのビット幅は32ビットとして合成し，対象故障は単一縮退故障，故障個所は演算器内に設定した．評価基準には，演算器入出力順序深度，テスト生成時間，故障検出率，テスト系列長，データパスとコントローラを含む回路全体の総面積を用いた．演算器入出力順序深度に対する実験結果を表4-2に，テスト生成時間，故障検出率，テスト系列長に対する実験結果を表4-1に示す．

表 4-1 テスト生成結果

	総故障数		故障検出率	
回路名	従来手法	提案手法	従来手法	提案手法
EX2	52682	52746	99.99	100.00
ARF	105552	105552	99.90	100.00
BPF	56806	56668	97.16	100.00
FFT	66706	113592	98.75	100.00
	テスト生成時間		テスト系列長	
回路名	従来手法	提案手法	従来手法	提案手法
EX2	2295.07	1312.19	99	104
ARF	16020.91	4899.20	291	326
BPF	28820.42	1728.52	257	168
FFT	4620.53	8305.04	328	268

表 4-2　演算器入出力順序深度

従来手法					提案手法				
回路名	演算器名	左入力	右入力	出力	回路名	演算器名	左入力	右入力	出力
ex2	MUL0	1	1	1	ex2	MUL0	1	1	1
	MUL1	2	1	1		MUL1	1	1	1
	SUB0	1	2	1		SUB0	1	1	1
ARF	MUL0	1	1	2	ARF	MUL0	1	1	1
	MUL1	1	1	1		MUL1	1	1	1
	MUL2	1	1	1		MUL2	1	1	1
	MUL3	1	1	1		MUL3	1	1	1
	ADD0	1	1	1		ADD0	1	1	1
	ADD1	1	1	1		ADD1	1	1	1
BPF	MUL0	1	1	1	BPF	MUL0	1	1	1
	MUL1	1	1	1		MUL1	1	1	1
	SUB0	1	1	2		SUB0	1	1	1
	SUB1	1	1	1		SUB1	1	1	1
	ADD0	1	1	1		ADD0	1	1	1
	ADD1	1	1	1		ADD1	1	1	1
FFT	MUL0	1	1	1	FFT	MUL0	1	1	1
	MUL1	1	1	1		MUL1	1	1	1
	MUL2	1	∞	1		MUL2	1	1	1
	MUL3	1	∞	1		MUL3	1	1	1
	ADD0	1	1	1		ADD0	1	1	1
	ADD1	1	1	1		ADD1	1	1	1
	ADD2	1	1	1		ADD2	1	1	1
	ADD3	1	1	1		ADD3	1	1	1
	SUB0	1	1	1		SUB0	1	1	1
	SUB1	1	1	1		SUB1	1	1	1
	SUB2	∞	1	1		SUB2	1	1	1
	SUB3	∞	1	1		SUB3	1	1	1

演算器入出力順序深度の比較では，従来手法に存在していた2以上の値がすべて1となった．これにより，生成されるテスト容易化機能的時間展開モデルの時間展開数が小さくなり，テスト容易となり，故障検出率やテスト生成時間で効果的な結果を得ることができた．特に，演算器入出力順序深度∞が無くなったFFTでは故障検出率，乗算器の出力順序深度が2から1となったARFではテスト生成時間において，大きな改善を得ることができた. テスト生成結果では, 従来手法と比べ, 提案手法では故障検出率は平均1.05%，最大2.84%向上した. テスト生成時間は, 平均10.70%，最大94.00%削減された. テスト系列長は, 平均9.00%, 最大34.63%削減された．実験結果から，提案するバインディング手法は，テスト容易化機能的時間展開モデルのテスト容易性を向上させるものであり，有用であると考えられる．

5.　むすび

本論文では，演算器入出力順序深度を定義し，テスト容易化機能的時間展開モデル生成のためのテスト容易化バインディングを提案した．評価実験では，すべての実験対象回路全てに対して，より高い故障検出率を達成することができた. テスト生成時間では3つの回路に対して効果的であった．今後の課題は，CDFGを対象とした実験や，より大きなDFGでの実験，コントローラ拡大における状態遷移の追加数を削減するようなテスト容易化バインディングの提案などが挙げられる．

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] D. D. Gajski, N. D. Dutt, A. C-H Wu, and S. Y-L Lin, High-Level Synthesis: Introduction to Chip and System Design, Kluwer Academic Publishers, 1992.

[3] H. Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985.

[4] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, “Scan-Based Side-Channel Attack Against RSA Crypto-Systems Using Scan Signatures”, IEICE Trans. on Fundamentals, Vol. E93-A, No. 12, pp2481-2489, 2010.

[5] M.T.-C. Lee, W. H. Wolf, and N. K. Jha, “Behavioral Synthesis for Easy Testability in Data Path Scheduling”, in Proc. Int. Conf. on Computer-Aided Design, pp.616 -619, 1992.

[6] M.T.-C. Lee, W. H. Wolf, and N. K. Jha, “Behavioral Synthesis for Easy Testability in Data Path Allocation”, in Proc. Int. Conf. Computer Design, pp. 29 -32, 1992.

[7] M.T.-C. Lee, N. K. Jha, and W. H. Wolf, “Behavioral Synthesis for Highly Testable Datapaths Under the Non-Scan and Partial Scan Environments”, in Proc. Design Automation Conf., pp.292-297, 1993.

[8] L.M.FLottes , B.Rouzeyre , L.Volpe,”A Controller reSynthesis Based Methos For Improving Datapath Testability”, IEEE International Symposium on Circuits and Systems, pp. 347 -350, May 2000.

[9] K. Sugiki, T. Hosokawa, and M.Yoshimura, ”A Test Generation Method for Datapath Circuits Using Functional Time Expansion Models”, 14th IEEE Workshop on RTL and High Level Testing (WRTLT’08), pp.39-44, Nov. 2008.

[10] T. Masuda, J. Nishimaki, T. Hosokawa and H. Fujiwara, "A Test Generation Method for Datapaths Using Easily Testable Functional Time Expansion Models and Controller Augmentation," IEEE the 24th Asian Test Symposium (ATS'15), pp. 37-42, Nov. 2015.

[11] M.T.-C.Lee, High-Level Test Synthesis of Digital VLSI Circuits, Artech House Publishers, 1997.

[12] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra, Low-Power High-Level Synthesis for Nanoscale CMOS Circuits”, Springer, 2008.