

機能的時間展開モデルを用いたテスト生成のための バインディング法

日大生産工(院) ○増田 哲也 NEC エンジニアリング(株) 西間木 淳
日大生産工 細川 利典 大阪学院大 藤原 秀雄

1. まえがき

近年、半導体集積技術の発達に伴い、設計される大規模集積回路(Large Scale Integrated circuits : LSI)の高集積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法として、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[2]. RTL回路は、回路の主な演算処理を実行するデータパスと回路の動作を制御するコントローラで構成される. また、LSIの高集積化に伴い、一般的なLSIテスト手法であるスキャンテストのテスト実行時間は急激に増加している. そのため、非スキャンテストに基づくテスト容易化設計[3]およびテスト生成法[4]が提案されている. 文献[3]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させるようにコントローラを拡大する. しかしながら、これまで提案されてきた順序回路のテスト生成アルゴリズム[5]は、回路構造のみからテスト系列を生成する. それゆえ、テスト容易な構造に基づいて拡大したコントローラの機能通りにテスト生成するとは限らない. 文献[4]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を生成し、テスト生成を行う方法が提案されている. 文献[4]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を生成する. 次に、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コントローラを拡大する. データパスのテスト生成時は、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制御信号・状態信号系列を制約として与える. 文献[4]の手法により、テスト生成時における時間展開数の有限化およびテスト生成の効率化を図ることができる. 文献[4]の手法で用いるテスト容易化機能的時間展開モデルはデータパスの構造に依存して生成される. したがって、データ

パスのテスト容易性が低い場合、テスト容易性の高いテスト容易化機能的時間展開モデルの生成が困難となる可能性がある. また、文献[4]では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コントローラが制御できない場合、その動作を実現するためにコントローラの拡大が必要となる. その結果、コントローラの面積オーバーヘッドが増大する. さらに機能動作と複数のテスト動作を区別するための外部入力追加が必要となるため、外部入力ピンのオーバーヘッドが増大が考えられる. それゆえ、文献[4]の手法においては、機能動作時に制御可能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望ましい. 加えて、高い故障検出率を達成するためには、データパスにテスト容易な構造が存在する必要がある.

動作合成におけるバインディングは、回路の構造を決定する処理であり、生成される回路の面積や性能はバインディング結果に大きく影響する. したがって、バインディングにおいて、テスト容易化機能的時間展開モデル生成を考慮することで、面積オーバーヘッドの増大を抑え、テスト容易性の高いテスト容易化機能的時間展開モデル生成が可能となると考えられる.

本論文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のためのバインディング法を提案する. 実験では例題回路に対して本手法を適用し、提案手法の有効性を示す.

2. 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

文献[4]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法が提案されている. 文献[4]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、必要に応じてコントローラを拡大する. テスト生成時にはテスト容易化機能的時間展開モデルに基づき、制御信号系列および状態信号系列を制約と

A Binding Method for Test Generation Using Functional Time Expansion Models

Tetsuya MASUDA, Jun NISHIMAKI, Toshinori HOSOKAWA
and Hideo FUJIWARA

して付加する．そのため，通常のテスト生成と比較してテスト生成が高速化・効率化される．さらに制御信号・状態信号系列を制約として付与することでデータパスの動作が制御されるため，テスト生成時は必ずコントローラの拡大通りのテスト生成を実行できる．そのため，あるハードウェア要素のテストのために定義するテスト用の制御動作は1つで十分である．したがって，文献[3]の手法と比較して，コントローラ拡大による面積オーバーヘッドの増大を抑えることができる．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のレイテンシに依存しない．そのため，時間展開数の小さいテスト容易化機能的時間展開モデルを生成することで，短いテスト系列で高速なテスト生成が可能となる．さらに，生成されるテスト容易化機能的時間展開モデルのテスト容易性，コントローラ拡大による面積オーバーヘッドを考慮することで，より効率的なデータパスのテスト生成が実現できる．なお本論文で説明するテスト生成において，テスト対象となるハードウェア要素はデータパス内の演算器とする．以後，テスト容易化機能的時間展開モデルをテスト容易化 FTEM と記述する．

図1にRTLデータパス例を示す．図1において， u ， z ， y ， dz は外部入力， $u1$ は外部出力， $R0$ から $R4$ はレジスタ， $ADD0$ は加算器， $SUB0$ は減算器， $MUL0$ ， $MUL1$ は乗算器， $m1$ から $m7$ ， $r1$ から $r3$ は制御信号， s は外部入力かつ状態信号である．図2にコントローラ例を示す．コントローラは入力，出力，状態，状態遷移で構成された状態遷移機械として表現される．図2において， $S0$ から $S7$ は状態， $ST0$ から $ST8$ は状態遷移を表す． $S0$ はリセット状態とする．図3は図1よりテスト容易な構造に着目して生成されたテスト容易化 FTEM 例である．

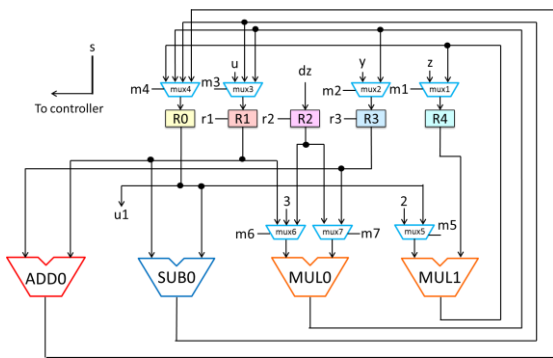


図 1. RTL データパス例

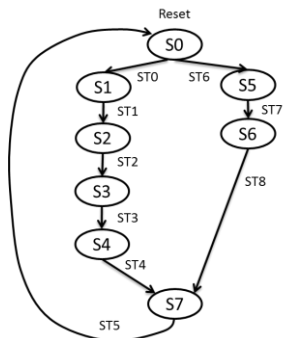


図 2. コントローラ例

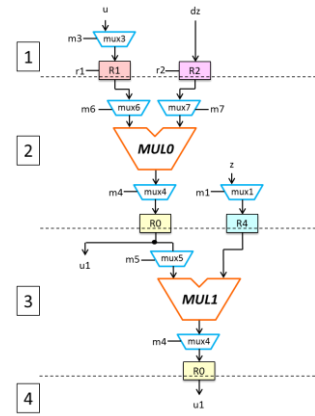


図 3. テスト容易化 FTEM 例

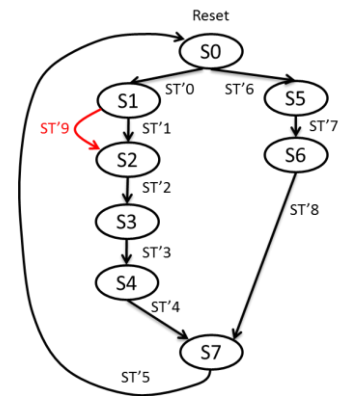


図 4. 拡大コントローラ例

図3は $MUL0$ ， $MUL1$ ， $R0$ ， $R1$ ， $R2$ ， $R4$ ， $mux1$ ， $mux3$ ， $mux4$ ， $mux5$ ， $mux6$ ， $mux7$ のハードウェア要素が動作する4時間展開モデルである．各ハードウェア要素は外部入力から何らかの値をその入力に伝搬する経路が存在し，かつその出力の値を外部出力まで伝搬する経路が存在する．したがって，上記のハードウェア要素は図3のモデルを用いることでテスト生成が可能といえる．テスト生成時には図3のモデルを制御するための制御信号値および状態信号値を制約として回路に付与する．図4は図3のモデルを動作させるために機能を拡大したコントローラである．図3のモデルの動作が機能動作時において実現不可能であったと仮定する．この時，図3のモデルの動作を実現するために状態 $S1$ ， $S2$ 間に状態遷移 $ST'9$ を追加し，コントローラの拡大を実行している．図4の例では図3の動作を状態遷移 $ST'0$ ， $ST'9$ ， $ST'2$ ， $ST'3$ が出力する制御信号によって実現する．

3. テスト容易化 FTEM を用いたテスト生成のためのバインディング法

本章ではテスト容易化 FTEM を用いたテスト生成を考慮したバインディング法について述べる．テスト容易化 FTEM を用いたテスト生成においては，生成されるテスト容易化 FTEM のテスト容易性が高く，かつ可能な限りコントローラの拡大を実行せず，機能動作上でテスト時の動作が実現できることが望ましい．テスト容易性が高く，追加状態遷移数が少ないテスト容

易化 FTEM が生成されるようにバインディング時に考慮することで、小さいオーバーヘッドで高故障検出効率を達成することが可能になると考えられる。例題回路として図 5 にスケジューリング済みのコントロールデータフローグラフ(Control Data Flow Graph : CDFG)を示す。図 5 の CDFG は外部入力 u , dz , z , y , $status$, 外部出力 $u1$, 定数入力 3, 5, 乗算, 減算で構成される。 $status$ が 1 の時, 図 5(a)の演算を実行し, $status$ が 0 の時, 図 5(b)の演算を実行する。

バインディングは主に演算器バインディング, レジスタバインディングの 2 つに分けることができる。本章で説明する手法では, 演算器バインディング時にデータパスのテスト容易性の向上を指向し, レジスタバインディング時に機能動作上で動作可能となるテスト容易化 FTEM 生成を指向する。

3.1. 演算器バインディング

演算器バインディングとは CDFG 上に存在する演算に対して具体的なハードウェア要素である演算器を割当てて処理である。提案手法では演算器バインディング時にデータパスのテスト容易性の向上を指向する。テスト容易化 FTEM のテスト容易性に最も悪影響を与える要因として, 定数のみの制御をうける演算器が挙げられる[6]。したがって, 本手法では定数制御演算器の削減を指向した演算器バインディングを実行する。定数制御演算器削減を実現するために, 演算器バインディング時には, 各演算に演算入出力距離を定義する。演算入出力距離とはグラフ上に存在する演算の可制御性, 可観測性を評価する尺度である。演算入力距離は外部入力から演算の入力まで到達するための最短のエッジ数で表現され, 演算出力距離は演算の出力から外部出力までの最短のエッジ数で表現される。ただし, ある演算の入力が定数となっている場合, 演算入力距離は ∞ と定義する。また, 1 本以上のエッジを

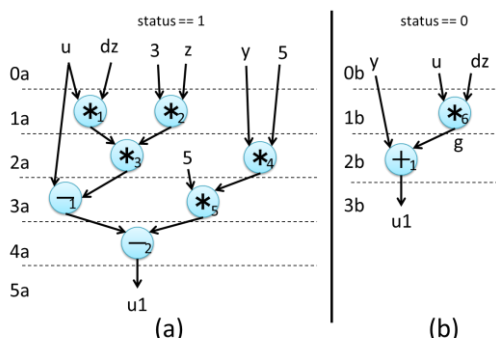


図 5. 例題回路の CDFG

表 1. 図 5 の演算入出力距離

演算	演算入出力距離		
	左入力距離	右入力距離	出力距離
*1	0	0	3
*2	∞	0	3
*3	1	50	2
*4	0	∞	2
*5	∞	50	1
*6	0	0	1
+1	0	1	0
-1	0	2	1
-2	50	50	0

表 2. 演算器共有後の演算入出力距離

演算	演算入出力距離			演算器
	左入力距離	右入力距離	出力距離	
{*1,*3,*5,*6}	0	0	1	MUL0
{*2,*4}	0	0	2	MUL1
{+1}	0	1	0	ADD0
{-1,-2}	0	50	0	SUB0

定数入力に到達する場合は可制御性の評価を悪くするために本論文では 50 と定義する。表 1 に図 5 の CDFG 上に存在する各演算の演算入出力距離を示す。演算入出力距離は演算器バインディングにおける演算器の共有化によって, 同じ演算器に割当てられる演算の入出力距離の最小値を評価値とすることができる。本手法では演算器バインディングによって演算入力距離 ∞ となる演算の削減を最優先に実行し, その後に演算器共有後の演算入出力距離の総和が最小となるように共有化を行う。表 2 に上記を考慮して演算器の共有化を実行した結果を示す。表 2 で示したように演算器バインディングを実行することで, 定数のみを入力とする演算器数を削減することができ, かつ外部入力, 外部出力までの距離が小さくなり可制御性・可観測性の高いデータパスが生成できる。

3.2. レジスタバインディング

レジスタバインディングとは CDFG 上に存在する変数に対して具体的なハードウェア要素であるレジスタを割当てて処理である。本手法のレジスタバインディングでは機能動作上で動作可能となるテスト容易化 FTEM の生成を指向する。本手法では初めに一つの変数に対して一つのレジスタを割当てて。図 6 に 3.1 節で示した演算器バインディングと各変数にレジスタを割当てた結果を示す。すべての変数にレジスタを割当てた後, テスト対象となる各演算器において, 機能動作時に何らかの値を外部入力からその入力まで伝搬およびその出力から外部出力まで伝搬する経路が存在するか解析する。図 6 において解析すると, $status$ が 0 の時の 4 サイクルの機能動作において MUL0 と ADD0 が外部入力 u , y , z よりそれらの入力まで値を伝搬, それらの出力から外部出力 $u1$ まで値を伝搬できることがわかる。さらに, 図 1 において存在する定数入力の演算器や, 再収斂構造は図 6 上には存在せず, 時間展開数は 4 に有限化される。したがって, MUL0, ADD0 は図 6(b)に示した経路をテスト容易化 FTEM としてモデル化することで, テスト容易性が高く, 追加状態遷移を必要とせずにテスト生成できることがわかる。しかしながら, MUL1, SUB0 においては前述した経路は存在しないため, レジスタバインディングによって改善する必要がある。初めに SUB0 と比較して回路規模の大きい MUL1 に着目する。MUL1 は構造的に外部入力 y , z から直接入力まで値を伝搬する経路がする。しかしながら機能動作時において y と z の値を同時に MUL1 の入力に伝搬する処理は定義されていない。この時, CDFG 上で MUL1 がアイドル状態となる時刻に着目する。図 6 より MUL1 は 3a, 4a, 5a, 1b, 2b, 3b の時刻でアイドル状態となっていることがわかる。

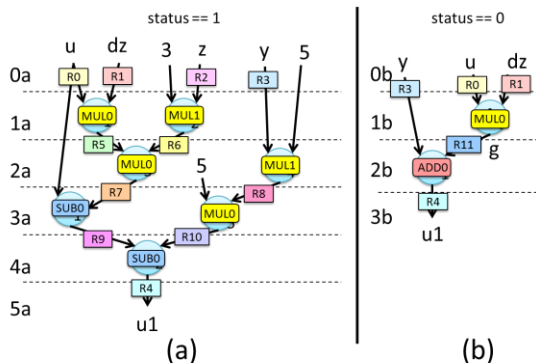


図 6. 演算器・レジスタバインディング結果 1

特に 1b においては状態遷移を追加することなく, R3, R2 へロードされた外部入力 y, z の値を MUL1 の入力まで伝搬させることができる. さらに, 1b において R4 はアイドル状態となっている. したがって, 1b で MUL1 の出力を R4 へロードする制御信号を与えることで, MUL1 をテストするためのテスト容易化 FTEM を状態遷移の追加をせずに生成することが可能となる. しかしながら, MUL1 の出力に割当てられているレジスタは R7 と R10 であるため, R4 へ出力をロードすることはできない. これを解決するために, R4 と R7 または R4 と R10 の共有化を行う. 本例では R10 と R4 の共有化を選択する. このように MUL1 および R4 のアイドル状態を利用することで, 新たに状態遷移を追加せずに MUL1 のテスト容易化 FTEM が生成できる. SUB0 についても同様に SUB0 と R4 がアイドル状態となる時刻に着目し, レジスタバインディングを実行する. 本例では SUB0 がアイドル状態となる時刻 1a に着目し, R7 と R3 の共有化を行った. これにより, 時刻 1a において外部入力 u, y から値を入力まで伝搬し, 出力をアイドル状態である R4 へロードすることで機能動作を用いて SUB0 のテスト容易化 FTEM が生成可能となる. このようにアイドル状態の演算器, レジスタを考慮してバインディングを実行することで, 機能動作上でテスト容易化 FTEM を生成できる可能性が増える. 本例ではすべての演算器に対して機能動作上でのテスト容易化 FTEM の生成が可能となったため, 他の共有可能なレジスタに関しては可能な限り共有化し, 回路面積の削減を図った. 図 7 にレジスタバインディング後の CDFG を示す.

4. 実験結果

本論文では例題回路に対して提案したバインディング法と文献[7]の面積最小化指向のバインディング法を適用し, 文献[4]の手法でテスト生成実験を行った. 表 3 より, テスト容易化 FTEM を用いたテスト生成を考慮したバインディングを実行することで, 文献[4]のテスト容易化 FTEM 生成の際, 状態遷移を追加せずに機能動作のみで制御できるテスト容易化 FTEM が生成されたことがわかる. また, 故障検出率は 100%を達成している. 実験により, バインディング時に定数制御演算器削減, およびアイドル状態の演算器, レジ

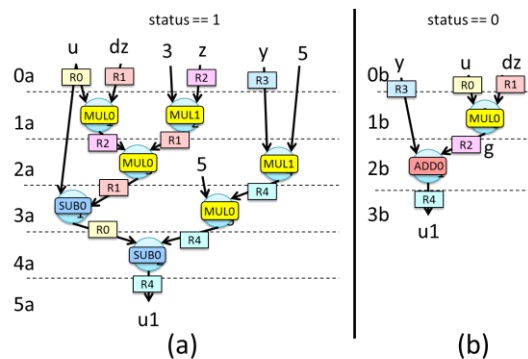


図 7. 演算器・レジスタバインディング結果 2

表 3. テスト容易化 FTEM およびテスト生成結果

手法	演算器数	レジスタ数	テスト容易化 FTEM 数	テスト容易化 FTEM 数詳細		追加状態遷移数	故障検出率 (%)
				k=3	k=4		
文献[8]	4	4	3	1	2	2	100
提案手法	4	5	2	1	1	0	100

スタを考慮することで, テスト容易化 FTEM を用いたテスト生成手法に対する有効性を示すことができた. しかしながら, 提案手法は文献[8]と比較してレジスタ数が 1 個多い結果となった. 提案手法は外部入力, 外部出力と接続されているレジスタがアイドル状態となる時刻を確保した状態で共有化を実行したためと考えられる.

5. おわりに

本論文ではテスト容易化 FTEM を用いたテスト生成を考慮したバインディング法を提案し, その有効性を実験により評価した. 今後の課題として, より大規模なベンチマーク回路を用いた評価実験と, バインディングの具体的なアルゴリズム設計などが挙げられる.

参考文献

- [1] 藤原 秀雄, “デジタルシステムの設計とテスト”, 工学図書株式会社, 2004.
- [2] M.C.McFarland, A.C.Parker, R.Camposano, “The high-level synthesis of digital systems”, Proc. IEEE, pp.301-318, 1990.
- [3] L.M.FLottes, B.Rouzeyre, L.Volpe, “A CONTROLLER RESYNTHESIS BASED METHODS FOR IMPROVING DATAPATH TESTABILITY”, IEEE International Symposium on Circuits and Systems, pp. 347 -350, May 2000.
- [4] Yusuke Kodama, Jun Nishimaki, Tetsuya Masuda, Toshinori Hosokawa and Hideo Fujiwara, “A Controller Augmentation Method to Generate Easily Testable Functionl k-Time Expansion Models for Data Path Circuits”, The Ninth Workshop on RTL and High Level Testing(WRTL’13).
- [5] W.T.Cheng, “The back algorithm for sequential test generation”, Proc. 1988 IEEE Int. Conf. on Computer Design, pp. 66-69, Oct. 1988
- [6] 増田哲也, 西間木淳, 細川利典, 藤原秀雄, “機能的 k 時間展開モデルのテスト容易性評価”, 信学技報, vol114, no.99, DC2014-17, pp.45-50, 2014 年 6 月.
- [7] 石井英明, 細川利典, テスト容易化インタフェースを設けた動作合成システム PICTHY の開発, 第 62 回 FTC 研究会, 2010.