論　文

無閉路部分スキャン設計に基づくデータパスのテスト容易化高位合成におけるバインディング手法

高崎　智也[†]　　井上　智生[††]　　藤原　秀雄[†]

A Binding Method in High-Level Synthesis for Testable Data Paths Based on Acyclic Partial Scan Design

Tomoya TAKASAKI[†], Tomoo INOUE[††], and Hideo FUJIWARA[†]

あらまし　本論文では，無閉路構造に基づく部分スキャン設計のための，データパスのテスト容易化高位合成手法を提案する．スケジュールされた動作記述（データフローグラフ）に対して，面積（リソース数）の最小性を満たしながら，無閉路化のためのスキャンレジスタ数を最小にする演算器とレジスタのバインディング法を提案する．本手法は，テスト容易性を考慮しない従来手法と比較して，リソース数を増やすことなく，無閉路化のためのスキャンレジスタ数の小さいレジスタ転送レベルデータパスを合成することができる．

キーワード　高位合成，部分スキャン設計，無閉路構造，最小クリーク分割，データパス

1. まえがき

近年の VLSI の高集積化，大規模化に伴い，回路のテストはますます重要でかつ困難な問題となっている [1]．テストの費用を削減するために，設計の初期の段階からテスト容易性を考慮することが必要とされている．抽象度の高い動作記述からレジスタ転送レベル（RTL）の回路を合成する高位合成の段階でテスト容易性を考慮することにより，回路の面積・性能とともにテスト容易性も含めた最適化及び設計費用の削減ができるものと期待されている．本論文では，テスト容易性を考慮した高位合成（テスト容易化高位合成）の一手法として，無閉路構造に基づく部分スキャン設計のためのデータパスのテスト容易化高位合成法を考察する．

一部のフリップフロップをスキャン可能なフリップフロップ（スキャンフリップフロップ）に置き換える部分スキャン設計は，小さいハードウェアオーバヘッドでテスト容易な回路を実現するための重要な技術の一つである．用いるテスト生成アルゴリズムによって，部分スキャン設計法は大きく二つの手法に分けられる．一つは順序回路用テスト生成アルゴリズムを用いることを前提とした部分スキャン設計法で，文献 [4], [5] ではスキャンフリップフロップによってセルフループを除いたフィードバックループを切断する手法が提案されている．もう一つは組合せ回路用テスト生成アルゴリズムを用いることを前提とした部分スキャン設計法である [6]〜[9]．この部分スキャン設計法において共通することは，RTL 回路の一部のレジスタ（フリップフロップの組）をスキャンレジスタに置き換え，スキャンレジスタによってセルフループを含むすべてのフィードバックループを切断することにより，無閉路構造を実現する無閉路部分スキャン設計である．本論文での高位合成法は，無閉路部分スキャン設計のスキャンレジスタ数が最小になる RTL データパスを合成し，組合せ回路用のテスト生成アルゴリズムを適用することを目的とする．

部分スキャン設計を指向したデータパスのテスト容易化高位合成法として，これまでに多くの手法が提案されている．文献 [10] では一部のレジスタをスキャンレジスタに割り当て，レジスタの可制御性/可観測性を向上させるためのデータパスの合成法が提案されている．文献 [11] では小さい数のスキャンレジスタを用い

[†] 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan
[††] 広島市立大学情報科学部，広島市
Faculty of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan

てセルフループ以外のすべてのフィードバックループを切断するデータパスの合成法が提案されている．文献 [13] ではセルフループ以外のすべてのフィードバックループを切断するスキャンレジスタ数を小さくするためのレジスタのバインディング法が提案されている．また，文献 [12] では部分スキャン設計を想定しないが，フィードバックループ数の小さいデータパスの合成法が提案されている．これらの手法はいずれも順序回路用テスト生成アルゴリズムを用いることを前提にしており，必ずしもすべてのフィードバックループを切断する手法ではない．よって，組合せ回路用テスト生成アルゴリズムのための無閉路部分スキャン設計を指向した合成法にそのまま適用することができない．

本論文では，高位合成の部分問題として，スケジュールされた動作記述（データフローグラフ）に対して，リソース数（演算器数，レジスタ数）の最小性を満たしながら，生成される RTL データパスで無閉路化（セルフループを含むすべてのフィードバックループを切断）のためのスキャンレジスタ数を最小にする演算器とレジスタのバインディング法を提案する．提案するバインディング法は，(1) 動作レベルでセルフループを構成する変数は，RTL データパスでスキャンレジスタに割り当てなければならない，(2) 動作レベルでフィードバックループが発生しても，レジスタを効率良く共有できれば，RTL データパスでスキャンレジスタ数を減らすことができる，という二つの事実に着目し，(1) 演算器・レジスタの共有によってセルフループの発生をできるだけ回避する，(2) できるだけ多くのフィードバックループが同じレジスタを通るように演算器・レジスタを割り当てるものである．

以下，**2.** で提案するバインディング法の全体の流れを示し，**3.** で演算器とレジスタバインディングの発見的手法について詳細を説明する．**4.** で最小クリーク分割を用いたバインディングのヒューリスティックアルゴリズムについて示し，**5.** で動作記述のベンチマークに対する実験結果より提案手法の有効性を示す．

2. 全体の流れ

提案するバインディング法は，リソース数（演算器数，レジスタ数）を最小にする一般的なアルゴリズムを，無閉路部分スキャン設計のためのスキャンレジスタ数が最小になるように変更したものである．もととなるスキャンレジスタ数最小化を指向しない一般のバインディングアルゴリズムとして，文献 [2], [3] にあるような，両立グラフを用いた手法を採用した．ここではまず，そのアルゴリズムについて説明する．なお，スケジューリングとアロケーションは既に行われているものとする．入力となるデータフローグラフ（DFG）は以下のように定義される．

［定義 1］ スケジュール済み DFG（SDFG）は有向グラフ $G_{sD} = (V_D, E_D, t, s)$ である．ここで，V_D は外部入出力を含む演算を頂点とする集合，$E_D \subset V_D \times V_D$ は変数を辺とする集合，$t : V_D \to \{op_1, op_2, \ldots, op_n\}$ は演算の型，$s : V_D \to Z^+ \cup \{0\}$（非負整数）は演算が実行される制御ステップを表す．

ここでは簡単のため，各演算の実行遅延は 1 制御ステップと仮定する．

バインディングの主要な手続きは，SDFG 中の演算を演算器に割り当てる演算器バインディングと変数をレジスタに割り当てるレジスタバインディングからなる．一般には演算器バインディングとレジスタバインディングに分けて問題を解く．ここでは演算器バインディング，レジスタバインディングの順に行う手法を考える．各バインディングついて，最小個の演算器・レジスタの割当てを行うため，両立グラフに対して最小クリーク分割（クリーク数最小のクリーク分割）を解く．ここで各クリークは共有された演算器またはレジスタに対応している．本論文で扱う両立グラフとして，以下で定義する演算/レジスタ両立グラフを用いる．

SDFG 中の二つの演算が同じ制御ステップで実行されず，同じ型の演算器で実現できるとき，これらの演算は両立可能であるという．

［定義 2］ SDFG G_{sD} に対する演算両立グラフ（OCG）は，無向グラフ $G_O = (V_O, E_O)$ である．ここで，頂点 $v \in V_O$ は SDFG G_{sD} の演算，辺 $(u, v) \in E_O \subset V_O \times V_O$ は頂点 u, v に対応する演算が両立可能であることを表す．

［例 1］ 図 1 の SDFG の加算に対する演算両立グラフは図 2 のようになる．例えば，+1 と +2 はそれぞれステップ 1, 3 でスケジュールされているので，両立可能な辺をもつ．+2 と +3 は同じ制御ステップでスケジュールされているので，その間に辺は存在しない．

SDFG 中の二つの変数のライフタイム（変数が使用されている時間）に重複がないとき，これらの変数は両立可能であるという．

［定義 3］ SDFG G_{sD} に対するレジスタ両立グラフ（RCG）は，無向グラフ $G_R = (V_R, E_R)$ である．ここで，頂点 $v \in V_R$ は SDFG G_{sD} の変数，辺

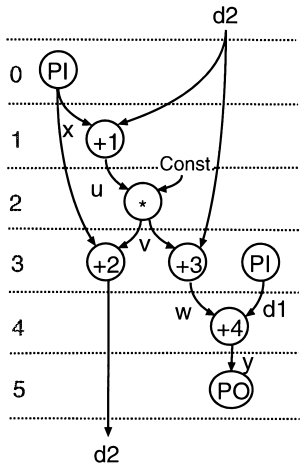

図 1　スケジュール済み DFG G_{sD}
Fig. 1　Scheduled DFG G_{sD}.

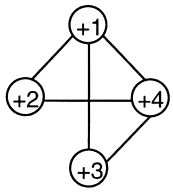

図 2　G_{sD} に対する演算両立グラフ G_O
Fig. 2　Operation compatibility graph G_O for G_{sD}.

$(u, v) \in E_R \subset V_R \times V_R$ は頂点 u, v に対応する変数が両立可能であることを表す．

［例 2］ 図 1 の SDFG に対するレジスタ両立グラフは図 3 のようになる．例えば，変数 u と v はライフタイムが異なるので，両立可能な辺をもつ．一方，変数 $d2$ はすべての制御ステップにわたって利用されているので，それと両立可能な辺は存在しない．

以上で定義した演算/レジスタ両立グラフを用いて，最小クリーク分割により最適なバインディングを求める．最小クリーク分割を求めるとき，演算器数またはレジスタ数に関して等価なバインディングは複数存在することが考えられる．しかし，それらは無閉路化のためのスキャンレジスタ数について必ずしも等価であるとは限らない．複数の最小クリーク分割の解の中からスキャンレジスタ数を最小にする解を選択するために，スキャンレジスタの必要性をクリークの重みで表す．求めたいバインディングをすべての最小クリーク分割の中で重みが最小になる最小クリーク分割を求める問題として扱うことにする．次の章では演算器とレジスタバインディングのための両立グラフのクリークの重みとその重みを利用した最小クリーク分割について述べる．

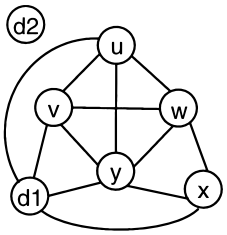

図 3　G_{sD} に対するレジスタ両立グラフ G_R
Fig. 3　Register compatibility graph G_R for G_{sD}.

3. 無閉路部分スキャン設計を指向したバインディング

3.1 演算器バインディング

ここでは演算の共有によってできるループを少なくし，それらのループがこの後のレジスタバインディングで互いにスキャンレジスタを共有しやすくすることを考える．

SDFG で両立可能な二つの演算間に経路が存在し，それらの演算を一つの演算器として共有すれば，もとの演算間の経路は共有した演算器を通るループとなる．よって，その演算間の経路上にあるいずれかの変数はループを切断するためのスキャンレジスタに割り当てなければならない．両立可能な演算間の経路の長さ，すなわちその経路上にある変数の数が大きければ，そのうちいずれか一つをスキャンレジスタに割り当てればよいので，スキャンレジスタを選択する自由度は大きくなる．一般に両立可能な二つの演算間には複数の経路が存在する．簡単のため，ここでは最短経路を複数の経路の代表として扱うことにする．複数ある経路の中で最短経路上の変数は最もスキャンレジスタの共有が行いにくいと考えられる．また，スキャンレジスタに割り当てられる変数のライフタイムが長ければ，スキャンレジスタは共有しにくい．ライフタイムの正確な見積もりはレジスタバインディングで行うとして，ここではライフタイムの代わりに経路をもつ両立可能な二つの演算間の時刻差を単純に評価することにする．

以上のことを考慮して，演算両立グラフの各辺に，以下の重みを付ける．

3.1.1 演算両立グラフで付ける重み

［演算両立グラフの辺 (u, v) の重み］

（1） SDFG で対応する演算 u と v の間に経路が存在しない（$u \to v$, $v \to u$ のいずれの方向にも経路が存在しない）とき

$$w_o(u,v) = 0$$

（2） SDFG で対応する演算 u と v の間に経路が存在するとき

u と v の最短経路を p（$u \to v$, $v \to u$ の両方向で経路が存在するとき，それらすべての経路の中で最短），p の長さと時間（時刻差）をそれぞれ $l(p)$, $t(p)$ とすると，

$$w_o(u,v) = t(p) \times K_{l(p)}$$

ここで，$K_{l(p)}$ は $K_{l(p)} > K_{l(p)+1}$ を満たす十分大きな数とする．

上の式を用いると，経路が短いほど大きな重みが与えられる．特に，最短経路長が 1 のときにできるセルフループは，演算の共有によってできる限り作りたくないことを表している．また，重みにかけている最短経路の時間は，同じ長さの最短経路の中だけで差がつくようにしたものである．同じ最短経路長の共有の組合せがあれば，時間の短いものが優先されることを表している．

この重みにおいて，演算間の経路が長い共有については，経路が短い共有と比べて，小さい値の重みが与えられる．したがって実際には，ある程度の長さの経路の短い共有だけで評価に差が十分現れるものと考えられる．実験では長さ 5 のものまで評価した（**5.** 参照）．

［例 3］ 図 1 の SDFG の加算についての演算両立グラフ図 2 に対して，各辺に対応する演算間の最短経路長とその時間を図 4 の括弧の中に示す．ここで，現在のステップ 5 と次のステップ 0 が同じ制御ステップ内で行われるものと仮定する．$K_3 = 1$, $K_2 = 10$, $K_1 = 100$ とするとき，各辺の重みは図 4 の数値のようになる．例えば，+3 と +4 を一つの演算器として共有すると，共有した演算器はセルフループを構成し，変数 w はスキャンレジスタに割り当てなければならない．同様に +1 と +2 を共有すると $d2$ はスキャンレジスタとなるが，これは w よりも利用されている時間が長い．一方，+1 と +3 を共有すると，セルフループでないループができ，u か v のいずれかがスキャンレジスタに割り当てられる．スキャンのための共有に関しては，$(+3,+4)$ は $(+1,+2)$ よりもよく，

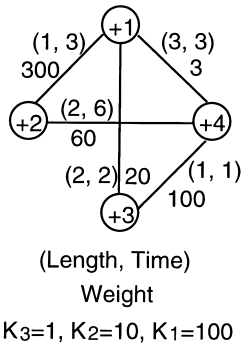

図 4 演算両立グラフ G_O の重み付け
Fig. 4 Weighted operation compatibility graph for G_O.

$(+1,+3)$ は $(+3,+4)$ よりもよい．図 4 の両立グラフの各辺の重みはこの順位を表したものになっている．

3.1.2 クリークの重み

一つの演算器の共有に関するスキャンレジスタの必要性を表す尺度としてクリークの重みを考える．ここでクリークは共有された一つの演算器を表している．演算両立グラフでクリーク（演算器）を構成するとき，重みの大きい辺（共有）が少ないことが望ましい．よって，クリーク $C_i = (V_i, E_i)$ の重み $W_o(C_i)$ を以下のように定義する．

$$W_o(C_i) = \sum_{e \in E_i} w_o(e)$$

3.1.3 重み和最小クリーク分割問題

演算器バインディングに関するスキャンレジスタの必要性を表す尺度としてクリーク分割の重みを考える．クリーク分割では重みの大きいクリークが少ないことが望ましいことから，クリーク分割の重みをその中に含まれているクリークの重みの総和で表す．演算器数最小のもとで，スキャンレジスタ数を最小にする演算器バインディングとして，以下の問題を考えることができる．

［重み和最小クリーク分割問題］
入力：演算両立グラフ $G_O = (V_O, E_O)$
出力：クリークの重みの総和 $\sum_{i=1}^{n} W_o(C_i)$ が最小となるクリーク分割 $\pi = \{C_1, C_2, \ldots, C_n\}$
（クリーク $C_i = (V_i, E_i)$ とすると，$V_O = V_1 \cup V_2 \cup \ldots \cup V_n$ かつ $V_i \cap V_j = \emptyset, \forall i \neq j$）
条件：クリーク数 n が最小

以上のように演算両立グラフに対してクリーク分割

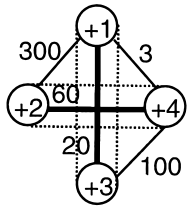

図 5　G_O に対する重み和最小クリーク分割
Fig. 5　Weighted minimum clique partitioning for G_O.

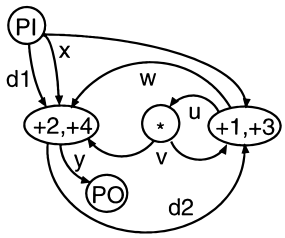

図 6　演算共有グラフ G_{oD}
Fig. 6　Operation bound graph G_{oD}.

を行った結果，演算器の共有を表したグラフとして，同じ演算器に割り当てられる複数の演算に対応するSDFGの頂点を一つの頂点として併合したグラフをつくる．このグラフを演算共有グラフという．

［例 4］ 図 4 の演算両立グラフに対して，最小となるクリーク数 2 の分割は，$\{\{+1,+2\},\{+3,+4\}\}$, $\{\{+1,+3\},\{+2,+4\}\}$, $\{\{+1,+2,+4\},\{+3\}\}$, $\{\{+1,+3,+4\},\{+2\}\}$ の 4 通り存在する．このうち図 5 に示す $\{\{+1,+3\},\{+2,+4\}\}$ がクリークの重みの和が $20+60=80$ で最小となる分割である．このときの演算共有グラフは図 6 のようになる．結果として，この解はセルフループを含んでいない．一方，ほかの最小クリーク分割の解はすべてセルフループが発生する辺 $(+1,+2)$ または $(+3,+4)$ を含んでいる．

3.2 レジスタバインディング

ここでは先に得られた演算共有グラフで変数の共有によってできるループを少なくし，それらのループがスキャンレジスタをできる限り共有することを考える．

演算共有グラフで両立可能な二つの変数間に経路が存在し，それらの変数を一つのレジスタとして共有すれば，もとの変数間の経路は共有したレジスタを通るループとなる．よって，その変数自身も含める経路上にあるいずれかの変数はスキャンレジスタに割り当てなければならない．特に，演算共有グラフで隣接している二つの変数を一つのレジスタとして共有すれば，もとの経路は共有したレジスタを通るセルフループとなるので，そのレジスタは必ずスキャンレジスタにしなければならない．演算共有グラフで共有する二つの変数間に隣接以外の経路がある場合は，共有したレジスタがスキャンレジスタになるかどうかは，経路中のほかの変数がスキャンレジスタに割り当てられるかどうかに影響する．両立可能な二つの変数を一つのレジスタとして共有したとき，そのレジスタがスキャンレジスタになるかどうかをレジスタ両立グラフの辺の重みで表現する．

また，ほかのどの変数との共有に関係なく，演算共有グラフでセルフループを構成している変数は必ずスキャンレジスタに割り当てなければならない．これをレジスタ両立グラフの頂点の重みで表現する．

3.2.1 レジスタ両立グラフで付ける重み

レジスタ両立グラフの各辺に以下の重みを付ける．

［レジスタ両立グラフの辺 (u,v) の重み］

（1） 演算共有グラフで対応する変数 u と v の間に経路が存在しない（$u \to v$, $v \to u$ のいずれの方向にも経路が存在しない）とき：$w_{er}(u,v)=0$

（2） 演算共有グラフで対応する変数 u と v の間に経路が存在する（$u \to v$, $v \to u$ のいずれかの方向で経路が存在する）とき：

（2-1） 変数 u, v が隣接しているとき：$w_{er}(u,v)=1$

（2-1） 変数 u, v が隣接していないとき：$w_{er}(u,v)=\omega$（ただし，$0<\omega<1$）

レジスタ両立グラフの各頂点（変数）に以下の重みを付ける．

［レジスタ両立グラフの頂点 v の重み］

（1） 演算共有グラフで対応する変数 v がセルフループを構成しているとき：$w_{vr}(v)=1$

（2） （1）以外のとき：$w_{vr}(v)=0$

［例 5］ 図 3 のレジスタ両立グラフに対して，図 6 の演算共有グラフにある演算器バインディングが行われたときの各辺と各頂点に重みを付けたレジスタ両立グラフは図 7 のようになる．ここで，$\omega=0.5$ とする．図 6 の演算共有グラフでは，それ自身がセルフループを構成している変数が存在しないので，レジスタ両立グラフの各頂点の重みはすべて 0 となる．各辺の重みについて考えると，例えば，w と x は一つのレジスタとして共有するとセルフループができ，共有したレジスタはスキャンレジスタになる．$d1$ と x は共有し

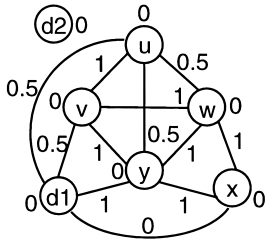

図 7　レジスタ両立グラフ G_R の重み付け
Fig. 7　Weighted register compatibility graph for G_R.

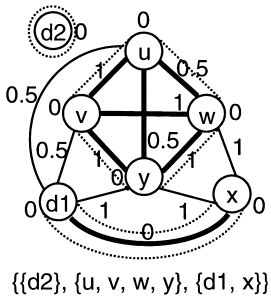

図 8　G_R に対する重み和最小クリーク分割
Fig. 8　Weighted minimum clique partitioning for G_R.

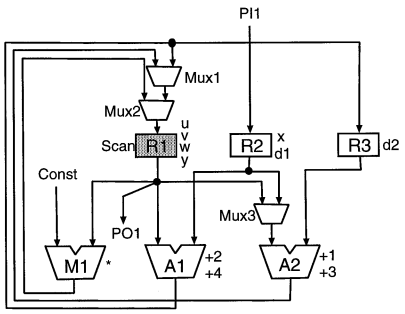

図 9　合成された RTL データパス（本手法）
Fig. 9　A synthesized RTL data path. (our method)

てもループができることはなく，スキャンレジスタの必要がない．スキャンのための共有に関しては，$(d1, x)$ は (w, x) よりはるかによい．このように，レジスタ両立グラフの各辺の重みは変数を共有することによるスキャンレジスタの必要性を表している．

3. 2. 2　クリークの重み

一つのレジスタの共有に関するスキャンレジスタの必要性を表す尺度としてクリークの重みを考える．ここでクリークは共有された一つのレジスタを表している．レジスタ両立グラフでクリークを構成するとき，共有したレジスタ（クリーク）がセルフループができる変数の共有（重み 1 の辺）やセルフループを構成する変数（重み 1 の頂点）を一つでも含んでいれば，そのレジスタはスキャンレジスタでなければならない．クリーク $C_i = (V_i, E_i)$ の重み $W_r(C_i)$ を以下のように定義する．

$$W_r(C_i) = \max\{\max_{e \in E_i} w_{er}(e), \max_{v \in V_i} w_{vr}(v)\}$$

クリークの重みはその中に含まれているすべての変数を一つのレジスタとして共有したとき，そのレジスタがスキャンレジスタになるかどうかを表している．

3. 2. 3　重み和最小クリーク分割問題

レジスタバインディングに関してスキャンレジスタの必要性を表す尺度としてクリーク分割の重みを考える．クリーク分割では重みの大きいクリーク，特にスキャンレジスタに割り当てなければならない重み 1 のクリークが少ないことが望ましい．よって，クリーク分割の重みをその中に含まれているクリークの重みの総和で表す．この重みが小さくなるようにクリーク分割を行えば，必要なスキャンレジスタ数を小さくすることができると考えられる．実際，この重みは結果として RTL データパスで必要なおおよそのスキャンレジスタ数を表している．レジスタ数最小のもとで，スキャンレジスタ数を最小にするレジスタバインディングとして，以下の問題を考えることができる．

［重み和最小クリーク分割問題］
入力：レジスタ両立グラフ $G_R = (V_R, E_R)$
出力：クリークの重みの総和 $\sum_{i=1}^{n} W_r(C_i)$ が最小となるクリーク分割 $\pi = \{C_1, C_2, \ldots, C_n\}$
（クリーク $C_i = (V_i, E_i)$ とすると，$V_R = V_1 \cup V_2 \cup \ldots \cup V_n$ かつ $V_i \cap V_j = \emptyset, \forall i \neq j$）
条件：クリーク数 n が最小

［例 6］ 図 7 のレジスタ両立グラフに対して，最小となるクリーク数 3 の分割は，$\{\{d2\}, \{u, v, w\}, \{d1, x, y\}\}$，$\{\{d2\}, \{u, v, w, y\}, \{d1, x\}\}$，$\{\{d2\}, \{u, v, d1\}, \{w, x, y\}\}$，$\{\{d2\}, \{u, v, y, d1\}, \{w, x\}\}$ の 4 通り存在する．このうち図 8 に示す $\{\{d2\}, \{u, v, w, y\}, \{d1, x\}\}$ がクリークの重みの和が $0 + 1 + 0 = 1$ で最小となる分割である．このとき，セルフループを構成するクリーク $\{u, v, w, y\}$ によって，少なくとも一つのスキャンレジスタが必要である．これに対し，ほかの最小クリーク分割の解ではセルフ

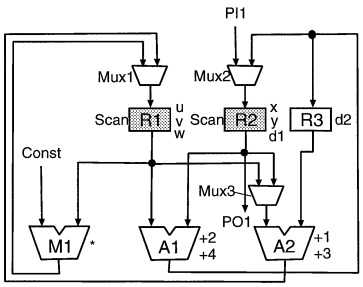

図 10　合成された RTL データパス（クリークの重みの和が最小でない場合）
Fig. 10　A synthesized RTL data path. (sum of weights of cliques is not minimum)

ループを構成するクリーク（この例では重み 1 の辺を含んでいるクリーク）が二つできるため，少なくとも二つのスキャンレジスタが必要となる．

［例 7］ 例 4，例 6（図 5，図 8）の演算器・レジスタバインディングの結果，合成された RTL データパスは図 9 のようになり，無閉路部分スキャン設計のための最小のスキャンレジスタ数は 1 となる．これは図 1 の SDFG から合成される RTL データパスの中で，最小のスキャンレジスタ数である．一方，例 6 のレジスタバインディングでクリークの重みの和が最小にならないクリーク分割 $\{\{d2\}, \{u, v, w\}, \{d1, x, y\}\}$ では，RTL データパスは図 10 のようになり，$\{u, v, w\}$, $\{d1, x, y\}$ に対応するレジスタがセルフループを構成することから，少なくとも二つのスキャンレジスタが必要となる．

4. ヒューリスティックアルゴリズム

ここでは前章で述べた重み和最小クリーク分割を解くヒューリスティックアルゴリズムを示す．これは重みなしの最小クリーク分割を求めるヒューリスティックアルゴリズム [3] をもとにして，クリーク数最小を求めながらスキャンレジスタ数が最小となるように重みを組み込んだものである．クリークの重みの計算を変えることで，演算器とレジスタのバインディングの両方に同じアルゴリズムを適用することができる．アルゴリズム `weighted_min_clique` を図 11 に示す．

演算/レジスタ両立グラフ `Gc` に対して，`weighted_min_clique` はクリーク数最小に関して等価な複数の解の中からスキャンレジスタ数を最小にするクリーク分割 `C_best` を選択する．このアルゴリズムではまずはじめに各頂点にクリークを割り当てる．両立グラフの辺は共有できるクリークの組を表している．

```
weighted_min_clique(Gc = (Vc, Ec, w_v, w_e))
{
  while (L < Lmax) {
    ++L;
    C = {Vc};
    /* First assign a clique to each vertex */
    (Cs1, Cs2) = select_start_pair(C);
    /* Select a start pair of sharing cliques
       on heuristic function H1 */
    merge(Cs1, Cs2);
    while(Exist a pair of sharing cliques) {
      (Ci, Cj) = select_clique_pair(C);
      /* Select a pair of sharing cliques
         on heuristic function H2 */
      merge(Ci, Cj);
    }
    if ((number(C) < number(C_best)) ||
       ((number(C) == number(C_best)) &&
        (Ws(C) < Ws(C_best)))) {
      C_best = C;
      /* Update solution */
    }
  }
}
```

- Input : compatibility graph `Gc = (Vc, Ec, w_v, w_e)`
  - `Vc` : operation/variable in SDFG
  - `Ec` : sharing possible relation between operations/variables
  - `w_e` : weight of edge
  - `w_v` : weight of vertex
- Output : optimal clique partitioning `C_best`
- `number(C)` : number of cliuqes
- `Ws(C)` : sum of weights of cliques

図 11　重み和最小クリーク分割のヒューリスティックアルゴリズム
Fig. 11　Heuristic algorithm for weighted minimum clique partitioning.

最適なクリーク分割をより広く探索するために，`select_start_pair(C)` でクリーク分割を複数回繰り返して求め，それらの分割の中からクリーク数が最小でクリークの重みの和が最小になるものを最適解として出力する．`select_start_pair(C)` では評価関数 H_1 ではじめに同じクリークにできる組を選択し，それらのクリークを共有する．この評価関数は両立グラフの辺の重みが小さいものを優先する．クリーク分割を求める繰返しは前よりもクリーク数が小さいかまたはクリーク数が同じでクリークの重みの和が小さい分割が求められる限り，続けられる．クリーク分割の解を求める全体の繰返しの数 `Lmax` については，実験的に評価する．

クリーク分割を求める各繰返しについては，評価関

数 $H_2(h_c, h_w)$ で同じクリークにできる組を選択し，それらのクリークを共有する．これを同じクリークにできる組がなくなるまで行う．ここでの評価関数は，クリーク数を最小にしながらクリークの重みの和が小さいクリーク分割を求めるのに利用される．具体的には，以下のように考える．

評価尺度 h_c はクリーク数最小の分割を求めるために適用される [3] もので，(α, β) の 2 項組からなる．選択した共有するクリークの組によって，α はほかに共有できる可能性のあるクリーク数，β は共有できなくなるクリーク数を表している．同じクリークにできる組を選択するとき，α は大きい方が望ましく，β は小さい方が望ましい．評価尺度 h_c は α, β の優先順位に従って評価を行う．

評価尺度 h_w はクリークの重みの和を小さくするためのもので，評価尺度 h_c に関して複数の等価な候補の中から一つを選択するために適用される．共有してもクリークの重みが増えないものを優先する．演算器バインディングでは（選択する両立グラフの辺の重み）−（選択した辺によって共有できなくなる両立グラフの辺の重みの和）を計算し，この値が最も小さいものを選択する．選択した辺によって共有できなくなる両立グラフの辺は，結果としてできるクリーク分割の中に含まれない辺である．この辺の重みを足した値が大きい方が，辺の重みの総和をクリークの重みとするとき，クリークの重みの和を小さくすることができる．レジスタバインディングでは（共有する 2 頂点の重みの和）−（選択する両立グラフの辺の重みと共有する 2 頂点の重みの最大値）を計算し，この値が最も小さいものを選択する．この評価式の第 1 項，第 2 項はそれぞれ共有前と後のクリークの重みの和を表している．この値が 0 以下であれば，クリークの重みが増えることはない．共有前後のクリークの重み和の差が小さいものを優先して選択することにより，辺と頂点の重みの最大値をクリークの重みとするとき，クリークの重みの和を小さくすることを指向している．

5. 実験結果

提案手法の有効性を示すために，前章で述べたヒューリスティックアルゴリズムを実装し，いくつかの動作記述のベンチマークに対して演算器とレジスタのバインディングを求める実験を行った．実験に利用したベンチマークは 3rd Lattice Wave Filter（LWF，図 1），Tseng [10]，Paulin [10]，4th Jaumann Wave Filter（JWF），4th IIR Cascade Filter (IIR)，5th Elliptic Wave Filer（EWF）の 6 種類の DFG に対して何通りかのスケジューリングを試みたスケジュール済み DFG（SDFG）である．回路名に付いている ‘.1’ などは同じ DFG でスケジューリングを適当に変化させたものを示している．表 1 に使用したベンチマークのレイテンシ（l），外部入力数（#PI），外部出力数（#PO），外部入出力以外の演算数（#Op），変数の個数（#Var）を示している．

ここで演算器バインディングにおいて，演算両立グラフの重みは長さ 5 の最短経路まで評価した．よって，$l(p)$ を最短経路 p の長さとするとき，時間にかける定数は $K_{l(p)} = 32^{5-l(p)}$ とした．レジスタバインディングにおいて，重みに付ける 0 と 1 の間の定数 ω については，0.5 とおいて評価した．また，各バインディングでクリーク分割を求める繰返し数 Lmax については，繰返しの中で最良の解が得られる回数の適切な値を調べるために，特定の値を設定せずに演算/レジスタ両立グラフの辺数とした．

バインディングの結果得られた RTL 回路に対して，文献 [14] のアルゴリズムを用いて無閉路化に必要な最小個のスキャンレジスタを求めた．その結果を表 2 に ST として示す．表 2 において，#OU, #Mux, #Reg, #Scan はそれぞれ合成された RTL の演算器数，2 入力マルチプレクサ数，レジスタ数，スキャンレジスタ数を表している．CPU は SUN Ultra30 で演算器・レジスタのバインディングに対して Lmax 回の繰返しで

表 1　ベンチマーク特性
Table 1　Benchmark characteristics.

Bench.	l	#PI	#PO	#Op	#Var
LWF	5	2	1	5	7
LWF.1					
Tseng	5	3	1	8	11
Tseng.1	6				
Tseng.2					
Paulin	5	4	3	10	11
Paulin.1					
JWF	9	1	1	17	20
JWF.1					
JWF.2					
JWF.3					
IIR	7	1	1	17	22
IIR.1					
IIR.2	8				
IIR.3					
EWF	16	1	1	34	38
EWF.1					
EWF.2					

表 2　ヒューリスティックアルゴリズムによる実験結果
Table 2　Experimental results with heuristic algorithms.

Bench.	M	RTL Characteristics				CPU
		#OU	#Mux	#Reg	#Scan	[s]
LWF	NT	3	6	3	3	<0.1
	ST	3	3	3	1	<0.1
LWF.1	NT	3	5	4	3	<0.1
	ST	3	3	4	1	<0.1
Tseng	NT	7	6	6	4	<0.1
	ST	7	8	5	2	<0.1
Tseng.1	NT	6	8	6	4	<0.1
	ST	6	7	6	3	<0.1
Tseng.2	NT	6	7	6	5	<0.1
	ST	6	10	5	3	<0.1
Paulin	NT	4	17	6	6	<0.1
	ST	4	12	6	4	<0.1
Paulin.1	NT	5	16	7	7	<0.1
	ST	5	13	7	5	<0.1
JWF	NT	3	15	7	7	<0.1
	ST	3	14	7	5	<0.1
JWF.1	NT	3	16	7	7	<0.1
	ST	3	17	7	5	<0.1
JWF.2	NT	3	17	7	7	<0.1
	ST	3	17	7	6	<0.1
JWF.3	NT	4	17	8	8	<0.1
	ST	4	17	8	4	<0.1
IIR	NT	5	21	7	7	<0.1
	ST	5	16	7	4	<0.1
IIR.1	NT	5	23	7	7	1.0
	ST	5	20	7	4	1.0
IIR.2	NT	4	22	7	7	1.0
	ST	4	20	7	4	1.0
IIR.3	NT	4	21	7	7	1.0
	ST	4	13	7	4	1.0
EWF	NT	4	39	11	11	56.0
	ST	4	33	11	7	53.0
EWF.1	NT	4	35	11	11	56.0
	ST	4	37	11	7	53.0
EWF.2	NT	4	35	11	11	56.0
	ST	4	34	11	7	53.0

クリーク分割の解を求めるのにかかった CPU 時間を示している．CPU における ‘<0.1’ は CPU 時間が 0.1 秒未満であったことを示している．

演算/レジスタ両立グラフの重みの効果を調べるために，スキャンレジスタ数最大化を指向する手法 NT も実験した．その結果も併せて表 2 に示してある．NT はクリーク分割の重み（クリークの重みの和）が最大になるようにヒューリスティックアルゴリズムに変更を加えて求めたものである．表 2 の結果より，すべてのベンチマークに対して，本手法 ST は手法 NT よりも少ないスキャンレジスタ数が得られた．これらからわかるように，提案した演算器・レジスタバインディングにおける重みは無閉路化のスキャンレジスタ数に

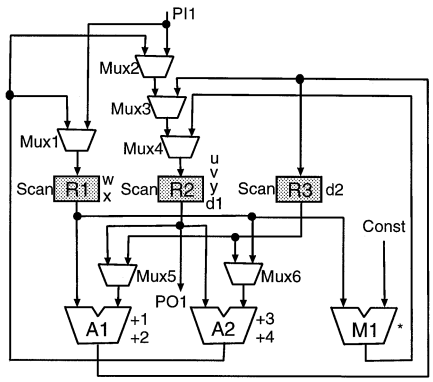

図 12　手法 NT による合成結果（LWF）
Fig. 12　Result of synthesized RTL data path by method NT. (LWF)

相関があるといえる．参考のため，図 12 に LWF の手法 NT のときの RTL データパスを示す．本手法 ST のときの RTL データパスは図 9 に対応している．これらの結果の RTL データパスを見てもわかるように，セルフループ数が減り，複数のループが共通のレジスタを通るようにスキャンレジスタが効率良く共有されて，スキャンレジスタ数が小さくなっている．

クリーク分割の繰返しの中で最良の解が得られる回数に関しては，ほとんどが 10 以下の少ない回数で，規模の大きい EWT でも両立グラフの辺数 330 の約半分程度の回数でよいことがわかった．

次に，本バインディング法の有効性を調べるために，演算/レジスタ両立グラフで重みを付けずにヒューリスティックアルゴリズムを適用した手法 NW も実験した．その結果を表 3 に示す．表 2 と表 3 の結果を比較してみると，ほとんどすべてのベンチマークに対して，NW のスキャンレジスタ数は ST と NT の間の値をとった．ST は NW よりも同じか小さいスキャンレジスタ数が得られていることがわかる．提案した演算器・レジスタバインディング法は有効であるといえる．

更に，ヒューリスティックアルゴリズムの精度を調べるために，表 2 の実験とは別に，EWF 以外の例に対して，起こり得るすべてのクリーク分割の中から時間をかけて重みが最小の解を求めることを試みた．そのうち文献 [14] のアルゴリズムで求めた最小数のスキャンレジスタが表 2 の ST の結果より更に小さくなったものを表 4 に示す．それ以外のものについては，ヒューリスティックでも演算器数，レジスタ数，及びスキャンレジスタ数は等しく小さい解を得ることがで

表 3　重みなしの実験結果
Table 3　Experimental results with no weights.

Bench.	RTL Characteristics				CPU
	#OU	#Mux	#Reg	#Scan	[s]
LWF	3	4	3	3	<0.1
LWF.1	3	5	4	3	<0.1
Tseng	7	7	5	2	<0.1
Tseng.1	6	6	6	3	<0.1
Tseng.2	6	8	5	4	<0.1
Paulin	4	12	6	5	<0.1
Paulin.1	5	16	7	5	<0.1
JWF	3	15	7	6	<0.1
JWF.1	3	17	7	7	<0.1
JWF.2	3	13	7	7	<0.1
JWF.3	4	16	8	6	<0.1
IIR	5	17	7	5	<0.1
IIR.1	5	20	7	5	1.0
IIR.2	4	20	7	5	1.0
IIR.3	4	17	7	6	1.0
EWF	4	32	12	10	56.0
EWF.1	4	34	12	10	56.0
EWF.2	4	34	11	10	56.0

表 4　全探索による実験結果
Table 4　Experimental results with all search.

Bench.	RTL Characteristics				CPU
	#OU	#Mux	#Reg	#Scan	[s]
Tseng.1	6	7	5	2	<0.1
Tseng.2	6	10	5	2	<0.1
IIR	5	20	7	3	3394.0
IIR.1	5	17	7	3	24059.0
IIR.2	4	10	7	3	16199.0

きた．この結果からわかるように，Tseng.1 に対する ST（表 2）は，スキャンレジスタ数だけでなく，レジスタ数も最小の解を得られていなかった（レジスタ数 6）．しかし，スキャンレジスタに関する重みを付けずに求めた結果 NW（表 3）でもレジスタ数は 6 になっていることから，レジスタ数並びにスキャンレジスタ数の増加は，提案するアルゴリズムのもととした最小クリーク分割のヒューリスティック [3] によるものと思われる．また，IIR, IIR.1, IIR.2 では，ST の解となったレジスタ両立グラフに対するクリーク分割の重みが，表 4 に示す解の重みよりわずかに大きい，すなわち，ST のヒューリスティックが最小の重みのクリーク分割を得ていないことがわかった．したがって，重みのわずかな差も表現できるようにヒューリスティックを改良することが課題といえるが，これらのスキャンレジスタ数は ST においていずれも 4 で，最小の 3 に近い解を選択していることがわかった．Tseng.2 については，$\omega = 0.5$ のもとで，表 4 で示したスキャンレジスタ数が 2 となるクリーク分割の重みは，ST で求めたスキャンレジスタ数が 3 となるクリーク分割の重みと等しいことがわかった．すなわち，Tseng.2 については，今回の実験で設定した $\omega = 0.5$ のもとで得られる重み和最小クリーク分割の解は，必ずしもスキャンレジスタ数最小に対応していなかったといえる．この例では ω を 0.5 よりも小さい値にして重みを計算すると，スキャンレジスタ数が 2 のときのみがクリークの重み和が最小となり，ST によってその解が得られることがわかった．ω の適切な値については若干の調整が求められるが，多くの場合，0.5 に設定するだけでよい結果が得られることがわかった．

以上のように，提案する重み付けとヒューリスティックアルゴリズムによって，無閉路化のためのスキャンレジスタ数は最小か若しくはそれに近い値を得ることができるといえる．

6. むすび

本論文では，スケジューリング処理後の動作記述（データフローグラフ）に対して，テスト容易性を考慮しない従来手法と比較して，演算器数，レジスタ数のリソース数を増やすことなく，無閉路化のためのスキャンレジスタ数の小さいレジスタ転送レベルのデータパスを合成するテスト容易化高位合成手法を提案した．更に，提案手法を小規模ではあるが動作記述のベンチマークに適用し，その有効性を示した．提案手法はリソース数の最小性を満たしながら，生成される RTL データパスで無閉路部分スキャン設計に必要となるスキャンレジスタ数を最小にする演算器とレジスタのバインディングが得られる．

今後はバインディング手法だけでなく，スキャンレジスタ数最小化のためのスケジューリング手法についても提案する必要がある．更に，データパスだけでなくコントローラも含めた合成手法についても検討していきたい．

謝辞　本研究に関し，多くの貴重な意見を頂いた本学の増澤利光助教授，井上美智子助手はじめ情報論理学講座の諸氏に感謝する．本研究は一部，(株）半導体理工学研究センター（STARC）との共同研究，及び文部省科学技術研究費補助金・基盤研究 B(2)（課題番号 09480054）の研究助成による．

文　献

[1] H. Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985.
[2] G. De Micheli, Synthesis and Optimization of Digital

Circuits, McGraw-Hill, Inc., 1995.
[3] P. Michiel, U. Lauther, and P. Duzy, The Synthesis Approach to Digital System Design, Kluwer Acedemic Publishers, 1992.
[4] K. Cheng and V.D. Agrawal, “A partial scan method for sequential circuits with feedback,” IEEE Trans. Comput., vol.39, no.4, pp.544–548, April 1990.
[5] D.H. Lee and S.M. Reddy, “On determining scan flip-flops in partial-scan design approach,” Proc. Int. Conf. Computer-Aided Design, pp.322–325, 1990.
[6] R. Gupta, R. Gupta, and M.A. Breuer, “The BALLAST methodology for structured partial scan design,” IEEE Trans. Comput., vol.39, no.4, pp.538–544, April 1990.
[7] 藤原秀雄，大竹哲史，高崎智也，“組合せテスト生成複雑度でテスト生成可能な順序回路構造とその応用，” 信学論（D-I），vol.J80-D-I, no.2, pp.155–163, Feb. 1997.
[8] 高崎智也，井上智生，藤原秀雄，“内部平衡構造に基づく部分スキャン設計法の考察，” 信学論（D-I），vol.J81-D-I, no.3, pp.318–327, March 1998.
[9] T. Inoue, T. Hosokawa, and H. Fujiwara, “An optimal time expansion model based on combinational ATPG for RT level circuits,” Proc. IEEE the 7th Asian Test Symposium, pp.190–197, Dec. 1998.
[10] T.C. Lee, N.K. Jha, and W.H. Wolf, “Behavioral synthesis of highly testable data paths under the non-scan and partial scan environments,” Proc. Design Automation Conf., pp.292–297, 1993.
[11] M. Potkonjak, S. Dey, and R.K. Roy, “Behavioral synthesis of area-efficient testable designs using interaction between hardware sharing and partial scan,” IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst., vol.14, no.9, pp.1141–1154, 1995.
[12] A. Mujumdar, R. Jain, and K. Saluja, “Behavioral synthesis of testable designs,” Proc. IEEE Int. Symp. on Fault-Torelant Computing, pp.436–445, 1994.
[13] V. Fernandez and P. Sanchez, “Partial scan high-level synthesis,” Proc. European Design and Test Conf., pp.481–485, 1996.
[14] S.T. Chakradhar, A. Balakrishman, and V.D. Agrawal, “An exact algorithm for selecting partial scan design,” Proc. Design Automation Conf., pp.81–86, 1994.

（平成 11 年 4 月 26 日受付，9 月 6 日再受付）

高崎　智也　（学生員）

平 7 創価大・工・情報システム卒．平 9 奈良先端大博士前期課程了．現在奈良先端大博士後期課程に在学中．テスト容易化設計，テスト容易化高位合成に関する研究に従事．

井上　智生　（正員）

昭 63 明大・工・電子通信卒．平 2 同大大学院博士前期課程了．同年松下電器産業（株）入社．明大大学院博士後期課程を経て，平 5 奈良先端大情報科学研究科助手．平 11 より広島市立大学情報科学部助教授．松下電気電器産業（株）においてマイクロプロセッサの研究開発に従事．明治大，奈良先端大，広島市大において，テスト生成，並列処理，テスト容易化設計に関する研究に従事．博士（工学）．IEEE，情報処理学会各会員．

藤原　秀雄　（正員）

昭 44 阪大・工・電子卒．昭 49 同大大学院博士後期課程了．阪大工学部助手，明治大理工学部教授を経て，現在奈良先端大情報科学研究科教授．昭 56 ウォータールー大客員助教授．昭 59 マッギル大客員準教授．論理設計，高信頼設計，設計自動化，テスト容易化設計，テスト生成，並列処理，計算複雑度に関する研究に従事．著書に “Logic Testing and Design for Testability”（The MIT Press）など．工博．情報処理学会会員．IEEE Fellow，IEEE Golden Core Member.