

無閉路部分スキャン設計を指向したテスト容易化高位合成における スケジューリングの高速化

An Acceleration of a Scheduling Algorithm in High-Level Synthesis for Acyclic Partial Scan

岡 伸也¹, 市原 英行², 井上 智生²
Nobuya Oka, Hideyuki Ichihara, Tomoo Inoue

あらまし

一般に困難である順序回路に対するテスト生成を容易にするための無閉路部分スキャン設計において、オーバーヘッドとなるスキャンレジスタ数最小を目的とするテスト容易化高位合成が提案されている。本論文では、スキャンレジスタ数最小を指向したスケジューリングアルゴリズムで用いられるヒューリスティックである仮バインディングにおいて、演算間の仮両立性の不変性に着目し、仮バインディングの回数を削減することでスケジューリングの計算時間の削減を行う手法を提案する。実験結果は、スキャンレジスタ数を最小に保ったまま、スケジューリングの実行時間を削減できていることを示す。

1 背景

順序回路のテスト生成は一般に困難な問題であり、回路規模が大きくなると解けなくなる場合が多い。そのため、順序回路に対するテスト生成を容易に行うためのテスト容易化設計が考えられており、そのうちの1つとして、無閉路部分スキャン設計が提案されている[3]。無閉路部分スキャン設計とは、順序回路の一部のレジスタをスキャンレジスタと置き換えることで、順序回路内の閉路をすべて切断するテスト容易化設計である。閉路を切断した回路は、単一故障用の組合せ回路用テスト生成アルゴリズムを適用可能となる[3]。そのため、無閉路部分スキャン設計を行うことで、スキャン設計のオーバーヘッドであるスキャンレジスタ数を小さくでき、組合せ回路と同様にテスト生成が可能となる。

さらに文献[1][2]では、回路設計の一段階である高位合成で、無閉路部分スキャン設計におけるスキャンレジスタ数を考慮したテスト容易化高位合成が考えられており、その有効性が報告されている。

本論文では、無閉路部分スキャン設計におけるスキャンレジスタ数最小を指向したスケジューリングアルゴリズムで用いられるヒューリスティックである仮バインディング[1]において、仮バインディングを実行する回数を削減することでスケジューリング全体にかかる計算時間を削減する手法を提案する。提案手法では、仮バインディングにおける演算間の仮両立性の不変性に着目する。仮両立性の不変性の判定には、スケジューリングを決定する際に用いられる分散という値を調べることで行うため、削減のために新たな計算を行う必要がない。実験結果では、すべてのベンチマーク回路に対して、スキャンレジスタ数最小を保ったまま、仮バインディングの実行回数を削減できている。また、ほとんどの回路でスケジューリング全体の計算時間を削減できていることを示す。

2 無閉路部分スキャン設計を指向した高位合成

2.1 高位合成の流れ

一般に高位合成はスケジューリングとバインディングからなる。スケジューリングは、動作記述であるデータフローグラフ DFG (図1) の各演算を実行する時刻を決定する処理であり、結果は図2のようになる。バインディングは、スケジューリング結果をもとに、面積と性能の最適性を考慮し、図3のようなRTLデータパスを生成する処理である。本論文が対象とする無閉路部分スキャン設計を指向した高位合成では、スケジューリング[1]とバインディング[2]のそれぞれについて有効な手法が提案されている。図3の回路では、図1のDFGに[1][2]を適用した場合のRTLデータパスであり、6つのレジスタのうち2つのレジスタをスキャンレジスタとすることで無閉路となる。

文献[2]のバインディングは、結果として得られるRTLデータパスを、面積と性能の最適性を崩さない範囲で、無閉路化に必要なスキャンレジスタ数を削減するヒューリスティックアルゴリズムである。演算器バインディング、レジスタバインディングのそれぞれの手続きにおいて、演算/変数の共有によって生じるスキャンレジスタの必要度を両立グラフの重みとして表現し、クリーク数最小の中で重みが最小となるクリーク分割を行う。

文献[1]のスケジューリングでは、バインディング[2]の手続きを仮バインディングによって予測し、スキャンレジスタ数の削減を指向する。レイテンシ制約の下で演算器数の最小化を指向するフォースディレクテッドスケジューリング(FDS)アルゴリズム[4]をベースとし、フォースが最小となるスケジューリングの候補に対し、演算の仮バインディングを適用し、スキャンレジスタ数が最小となると予測されるものを選択するものである。以下、仮バインディングを用いたスキャンレジスタ数削減を指向したスケジューリングアルゴリズム[1]の詳細について述べる。

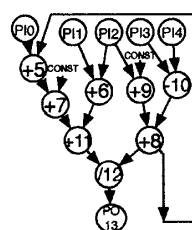


図1: DFG G_1

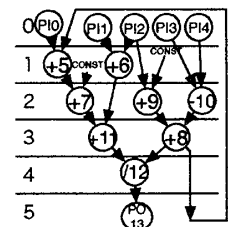
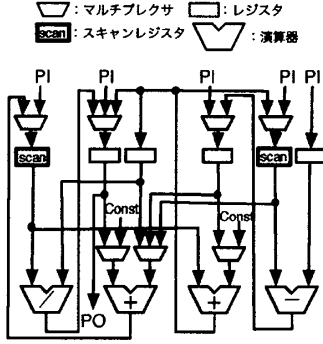


図2: DFG G_1 のスケジューリング結果

¹広島市立大学大学院情報科学研究科

²広島市立大学情報科学部

図3: DFG G_1 に対する RTL データパス

```

1  $fds\_pb(G, \lambda)$ {
2    $forall(v \in V)\{\varphi(v) := x;\}$ 
3    $while(\exists v \in V, \varphi(v) = x)\{$ 
4      $M := op\_with\_min\_force(G, \varphi, \lambda);$ 
5      $if(|M| = 1)\{(v_{best}, t_{best}) := element(M);\}$ 
6      $else\{(v_{best}, t_{best}) := sel\_by\_pb(G, \varphi, M);\}$ 
7      $\varphi := assign(G, \varphi, (v_{best}, t_{best}));$ 
8    $return \varphi;$ 

```

図4: スケジューリングアルゴリズム fds_pb

2.2 スケジューリングアルゴリズム [1]

図4にスケジューリングアルゴリズムを示す。このアルゴリズムは、与えられたデータフローグラフ (DFG) $G = (V, E, \tau)$ (V は演算の集合, E は演算の依存関係を表す辺の集合, $\tau: V \rightarrow \{1, 2, \dots, n_{res}\}$ は演算 $v \in V$ の型 $\tau(v)$ を表す。) とレイテンシ制約 λ に対し、各演算 $v \in V$ の実行開始時刻 $\varphi(v)$ ($\varphi: V \rightarrow \{0, 1, 2, \dots, \lambda\}$) を決定する。時刻の決定していない演算の時刻を x で表す。すなわち、このアルゴリズムが実行される前の各演算の開始時刻は、 $\forall v \in V, \varphi(v) = x$ となる (行2)。

アルゴリズム $fds_pb()$ では、 $op_with_min_force()$ により、演算に関するフォースに従って、フォース最小となる割当て (v, t) (演算 v の時刻 t へのスケジュール、つまり $\varphi(v) = t$) を求める (行4)。フォースとは、演算間にバネがあるとみなし、各時刻に割り当てられる演算数の期待値 (分散) をもとに、他の演算から受ける総合的な力 (フォース) を考えたものである。もし、得られたフォース最小の割当ての数が1ならばそれを選択し (行5)、2つ以上ならば、その中から、スキャンレジスタ数の最小化に有効と予測される割当てを仮バインディング $sel_by_pb()$ によって選択する (行6)。選択された割当て (v_{best}, t_{best}) によって、スケジュール φ を更新する (行7)。 $fds_pb()$ はこの処理を未スケジュールの演算がなくなるまで繰り返す (行3)。

このように $fds_pb()$ は、未スケジュールの演算を1つずつ選択し順にスケジュールを決定するアルゴリズムである。

2.3 仮バインディング

上述のように、アルゴリズム $fds_pb()$ は、演算器数最小の候補となる割当てが複数存在するとき、仮バイン

```

1  $sel\_by\_pb(G, \varphi, M)$ {
2    $forall((v, t) \in M)\{$ 
3      $\varphi^{(v, t)} := assign(G, \varphi, (v, t));$ 
4      $w_{total} := calc\_weight(G, \varphi^{(v, t)});$ 
5      $if(w_{total} < w_{best})\{w_{best} := w_{total};$ 
6        $(v_{best}, t_{best}) := (v, t);\}$ 
7    $return (v_{best}, t_{best});$ 

```

図5: 仮バインディングによる割当ての選択 sel_by_pb

ディングによって、スキャンレジスタ数最小の割当てを選択する ($sel_by_pb()$)。図5に仮バインディングによる割当ての選択の概略を示す。

アルゴリズム $sel_by_pb()$ では、フォース最小となる割当ての中からスキャンレジスタ数最小と予測される割当てを選択する。そのため、仮に割当て (v, t) を行ったスケジュール (すなわち、 $\varphi(v) = t$ とした) $\varphi^{(v, t)}$ を求め (行3)、[2] で提案されている重み付き演算器バインディングと同様のバインディングを仮に適用し、スキャンレジスタ数を見積もる。

同じ型の2つの演算 $u, v \in V$ について、それらが異なる時刻にスケジュールされているときはもちろん、どちらか一方の演算が未スケジュールのとき、すなわち、

$$\tau(u) = \tau(v) \wedge (\varphi(u) \neq \varphi(v) \vee \varphi(u) = x \vee \varphi(v) = x)$$

を満たすとき、2つの演算 u, v は仮に両立するという。以下では、スケジュール φ において演算 u, v が仮に両立するとき、 $comp(\varphi: u, v)$ と書く。一般性を失うことなく、2つの演算 u, v の型が異なれば ($\tau(u) \neq \tau(v)$)、 u と v は両立しないと考えることができる。以下では、特に断らない限り、両立性は型 k ごとについて議論する。

$sel_by_pb()$ は、この仮バインディングの結果に基づき、演算器数を最小化する割当ての候補 $(v, t) \in M$ の中から、重み和最小のもの (v_{best}, t_{best}) を1つ選択し (行4-6)、終了する。

2.4 適用例

例1: 入力を DFG G_1 とレイテンシ制約5とした場合の $fds_pb(G_1, 5)$ の実行例を示す。ここで与えられる演算器の型は加算器、減算器、除算器とする。何回かの繰り返しの結果、演算6, 9, 10のみ割当てが決定していない状態となる (これ以外の演算は図2のように割り当てられる)。このスケジュールを φ とする。このとき行4の実行結果として、 $M = \{(6, 1), (6, 2), (9, 1), (9, 2), (10, 1), (10, 2)\}$ が得られる。ここで、 $sel_by_pb(G_1, \varphi, M)$ の実行を考える。割当ての候補 M における仮割当てのうち、ここでは、 $\varphi^{(6, 1)}$, $\varphi^{(6, 2)}$ に対する仮バインディングの結果 (仮両立グラフに対する重み和最小クリーク分割) を図6に示す。なおここでは、議論を簡単にするため、加算 (+) の演算に関する両立性についてのみ考える。図6において、演算9はまだ割当てが決定していないため、すべての演算との間に辺が存在する。図6より、どちらについてもクリーク数が2で最小となっている。そこで、重み大を100、重み小を10とし重みを比べると、(6,1)の仮両立グラフの重み和は40となるのに対し、(6,2)の場合は、重み和が140となるため、重み和の小さい割当て (6,1)

が決定する。このような操作によって、 $sel_by_pb()$ の結果となる割当て (v_{best}, t_{best}) が決定される。

3 仮バインディングの実行回数削減法

3.1 不要な仮バインディングの実行

今、ある DFG G_2 に対する $fds_pb(G_2, 4)$ の実行の過程を考える。図7は演算5以外の演算の割当てが全て決定した状態のスケジュールを表す。このスケジュールを φ とする。このとき、図4、行4の実行結果として $M = \{(5, 1), (5, 2)\}$ が得られる。ここで、 $sel_by_pb(G_2, \varphi, M)$ の実行を考える。

仮割当て $(5, 1)$ を適用したときに得られるスケジュール $\varphi^{(5,1)}$ と、その仮割当てを適用する前のスケジュール φ を考えると、演算5に対するスケジュールは、それぞれ、 $\varphi^{(5,1)}(5) = 1, \varphi(5) = x$ となる。これらのスケジュール $\varphi^{(5,1)}, \varphi$ に対する仮両立グラフをそれぞれ考える (図8, 9) と、2つのグラフは同型であることがわかる。これは、仮割当て $(5, 1)$ を行っても、いずれの演算間の両立性には変化がなく、結果としてスキャンレジスタに関する見積もり (重み) も変化しないことを意味する。したがって、もし、仮割当てに対する両立グラフを作成する前に、演算の仮両立性が変化しないことがわかれば、仮バインディングを行わずに、その割当てに対する重みを得られる (そのまま利用できる) ので、全体として計算時間の短縮ができる。

3.2 仮両立性の不変性

まず、次式を満たす2つのスケジュール φ_1, φ_2 を考える。

$$\forall v \in V, \varphi_1(v) \neq x \Rightarrow \varphi_1(v) = \varphi_2(v)$$

ここで、 $S_1 = \{v : \varphi_1(v) \neq x\}, S_2 = \{v : \varphi_2(v) \neq x\}$ とすると、 $S_1 \subseteq S_2$ が言える。このとき、 φ_2 でのみスケジュールが決定している演算の集合 V_s は、 $V_s = S_2 - S_1$ と表せる。

ここで、任意の演算 $v_s \in V_s$ について、 $\varphi_1(v_s) = x$ なので、両立性の定義より、

$$\forall u \in V, u \neq v_s, \tau(u) = \tau(v_s) \Leftrightarrow comp(\varphi_1 : u, v_s)$$

は明らかである。一方、 $\forall v_s \in V_s, \varphi_2(v_s) \neq x$ であるから、両立性の定義より

$$\forall u \in S_2, u \neq v_s, \tau(u) = \tau(v_s) \wedge \varphi_2(u) \neq \varphi_2(v_s)$$

$$\Leftrightarrow comp(\varphi_2 : u, v_s)$$

である。したがって、次の事実を導くことができる。

性質1: スケジュール φ_2 において、時刻 t に割り当てられている型 k の演算の集合を $V_s(\varphi_2 : t, k)$ とする。 φ_2 でのみスケジュールが決定している演算 V_s のうち、演算の型が k である ($\tau(v_s) = k$) 任意の演算 v_s について、 $V_s(\varphi_2 : \varphi_2(v_s), k) = \{v_s\}$ ならば、かつ、そのときに限り、 φ_1 における型 k の両立性と φ_2 における型 k の両立性は等しい。

3.3 分散を用いた条件の判定法

先に述べた性質1における φ_1 と φ_2 の関係は、図4のアルゴリズムでは、 φ と、仮割当て (v, t) によって得られるスケジュール $\varphi^{(v,t)}$ との関係を表すことができる。そのため、性質1を用いることで、 φ と $\varphi^{(v,t)}$ における型 k の演算の両立性が等しいかどうかを判定することができる。

$\varphi^{(v,t)}$ において性質1を考えると、 $\varphi^{(v,t)}$ でのみスケジュールが決定している演算 V_s は、演算 v を含む。このとき、 V_s の要素が v のみであれば、性質1で示した $V_s(\varphi_2 : \varphi_2(v), k) = \{v\}$ を満たすかどうかを、 φ の情報を調べることで判定できる。

しかし、 $u \in V_s, u \neq v$ となる演算 u が存在する場合、仮割当て (v, t) によって割当てが決定する演算間で、仮両立性が等しくならない場合がある。例えば、図10では、仮割当て $(2, 2)$ により割当て $(3, 3)(4, 3)$ が決まった場合を表しているが、ここでは、演算3, 4の両立性が成り立たない。そのため、性質1を考える上で、 u が $V_s(\varphi^{(v,t)} : t, k)$ の要素でなければ、つまり、 $\tau(u) \neq k$ であれば、少なくとも型 k の両立性に影響を与えないことがいえる。

提案手法では、演算 v における仮両立性の変化を調べる道具として、 $op_with_min_force()$ でフォースを計算する際に用いる演算の型 k の時刻 $t (0 \leq t \leq \lambda)$ における分散 $dist(k, t)$ を利用する。分散とは、時刻 t に割り当てられる型 k の演算の個数の期待値を表す値である。例として、図7のDFGを用いて説明する。時刻3には2つの乗算の演算7, 8が割り当てられているので、乗算の分散は2となる。また、時刻1と2の乗算の分散は、演算5が時刻1と2の2時刻に割り当てられる可能性があるため、いずれも0.5となる。このように、演算の型ごとに、各時刻に対して演算が割り当てられている個数の期待値を求めたものが分散である。

分散の定義から、次のような性質を導くことができる。**性質2:**

$$dist(k, t) < 1 \Rightarrow \forall v \in V, \tau(v) \neq k \vee \varphi(v) \neq t$$

性質2から、仮割当て (v, t) の演算 v において、分散 $dist(\tau(v), t)$ を調べることで、性質1における $V_s(\varphi_2 : \varphi_2(v), k) = \{v\}$ を満たすかどうか調べることができる。

以上のことより、次の性質を導くことができる。

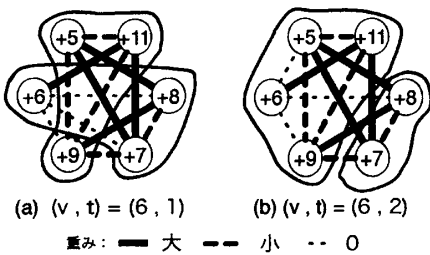


図6: DFG G_1 における仮両立グラフ

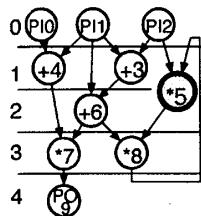


図7: DFG G_2 に対する部分的なスケジュール φ

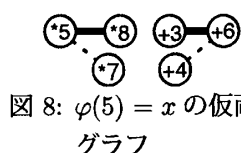


図8: $\varphi(5) = x$ の仮両立グラフ

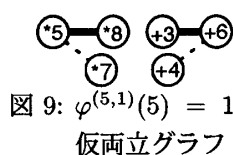


図9: $\varphi^{(5,1)}(5) = 1$ の仮両立グラフ

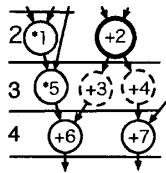
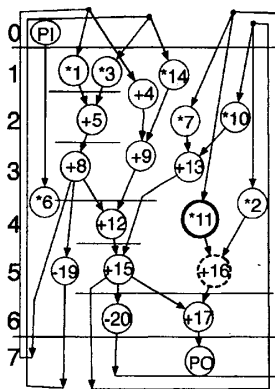
図 10: DFG G_3 (一部)

図 11: DFG IIR

性質 3: あるスケジュール φ に対して仮割当て (v, t) を行ったときのスケジュールを $\varphi^{(v,t)}$ とし, V_s を φ から $\varphi^{(v,t)}$ で新たに割り当てた演算の集合とすると, 以下の 2 つの条件を満たすならば, 型 k における φ と $\varphi^{(v,t)}$ の仮両立性は等しい.

$$(1) \text{dist}(\tau(v), \varphi(v)) < 1$$

$$(2) \forall u \in V_s, u \neq v \Rightarrow \tau(u) \neq k$$

この性質 3 を利用することで, スケジュール φ に対する仮バインディングの重みが計算済みの場合³, $\varphi^{(v,t)}$ に対する重みは, φ に対するそれを利用できる.

3.4 提案手法適用例

例 2: 図 11 において, 与えられる演算の型は加算 ($k=1$), 減算 ($k=2$), 乗算 ($k=3$) の 3 種類とし, 仮バインディング中で行う仮割当てを (11, 4) とし, そして, (11, 4) によって一意に決まる演算 $u (u \in V_s)$ は 16 とする. 乗算について考えると, 時刻 4 に乗算は割り当てられておらず, $\text{dist}(\tau(11), 4) = 0.667$ となり, 集合 V_s で $\tau(16) \neq 3$ となるため, 性質 3 より, 乗算における仮両立性は等しいといえる. 減算では, 集合 V_s で $\tau(16) \neq 2$ となるので, 減算においても仮両立性が等しいといえる. 加算では, $\tau(16) = 1$ であるため, 性質 3 より, 仮両立性が等しくならない状態であるといえる. 以上をまとめると, これまでは加算, 減算, 乗算と 3 つの重みの計算を行う必要があったが, 提案手法では, 加算についての重みを計算するだけでよいことがわかる.

4 実験結果

提案手法の有効性を調べるため, 実験を行った. 計算機は, SUN Blade2000 (UltraSPARC-III+, CPU 1.02GHz, メモリ 2GB) を用いた.

実験では, 提案手法である仮バインディング数削減を考えたスケジューリングと [1] のスケジューリングを比較した. 対象として 25 種類のベンチマーク回路を用い, ここでは表 1 に示す 9 つの結果について示す. 表 2 には, 従来法と提案手法それぞれの場合における各演算の仮バインディングによる重みの計算回数と, スケジューリングにかかった時間を示す.

表 2 より, すべての DFG で実行回数を削減できており, スケジューリングの計算時間についても削減できていることがわかる. 特に, 演算数が多い DFG (runge,

³ 図 4, 行 6 が実行されたとき.

trap) では, 仮バインディングの削減回数と, 計算時間の削減率が高い. これは, 演算数が大きくなるにつれ, スケジューリングの計算時間中の仮バインディングの計算時間の割合が高くなっているため, 仮バインディングの計算時間を削減することでスケジューリング全体の計算時間が削減できたためと考えられる.

なお, 2 つのスケジューリング法により得られたスケジュールは, [2] のバインディングを行い, RTL を合成した. その結果, スキャンレジスタ数とリソース数はすべてのベンチマーク回路で等しかった. よって, 提案手法によって仮バインディングを削減しても, 元々のスケジューリングアルゴリズムの有効性を損なうことがないことがわかる.

表 1: データフローグラフの特性

DFG	#Ops	#PIs	#POs	#Vars
EX2	8	5	1	14
EX3	11	5	1	25
PAULIN	10	4	3	11
IIR	17	1	1	22
EWf	34	1	1	38
nc_paulin	24	1	2	37
runge	44	1	2	72
trap	31	3	2	43

表 2: 仮バインディングの実行回数と計算時間の比較

DFG	実行回数 [回]		計算時間 [ms]	
	従来法	提案法	従来法	提案法
EX2	30	9	0.836	0.784
EX3	18	10	1.11	0.718
PAULIN	42	4	1.14	0.398
IIR	30	12	8.46	4.71
EWf	26	17	77.4	75.9
nc_paulin	48	9	5.77	2.34
runge	150	8	423	58.9
trap	52	6	114	21.3

5 まとめ

本論文では, 演算間の仮両立性の不変性に着目し, 無閉路部分スキャン設計を指向したスケジューリングアルゴリズムで用いられる仮バインディングの実行回数を削減することで, スケジューリング全体の計算時間を削減する手法を提案した. 実験結果から, 提案手法により仮バインディングの実行回数を削減でき, スケジューリング全体の計算時間を削減できることを示した. 今後の課題として, 仮両立性が等しくなるための条件をより一般化し, 必要十分となる条件を考察することが挙げられる.

参考文献

- [1] T. Inoue, T. Miura, A. Tamura, and H. Fujiwara, "A scheduling method in high-level synthesis for acyclic partial scan design," Proc. 11th IEEE Asian Test Symposium, pp. 128-133, Nov. 2002.
- [2] 高崎, 井上, 藤原, "無閉路部分スキャン設計に基づくデータベー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", 電子情報通信学会論文誌, Vol. J83-D-I, No. 2, pp. 282-292, 2000
- [3] T. Inoue, T. Hosokawa, T. Mihara, and H. Fujiwara, "An optimal time expansion model based on combinational ATPG for RT level circuits", Proc. IEEE the 7th Asian Test Symp., pp. 190-197, Dec. 1998.
- [4] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASIC's", IEEE Trans. on Computer-Aided Design, Vol. 8, No. 6, pp. 676-691, June, 1989.