社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-SLDM-138 (19)
2009 / 1 / 30

順序回路の上位設計記述における等価性指定の自動化手法

許　金美† 西原　佑† 松本　剛史†† 藤田　昌宏††,†††

† 東京大学大学院工学系研究科電子工学専攻 〒113-8656 東京都文京区本郷 7-3-1
†† 東京大学大規模集積システム設計教育研究センター 〒113-0032 東京都文京区弥生 2-11-16
††† 科学技術振興機構戦略的創造研究推進事業
E-mail: xujinmei@cad.t.u-tokyo.ac.jp, tasuku@cad.t.u-tokyo.ac.jp, matsumoto@cad.t.u-tokyo.ac.jp, fujita@ee.t.u-tokyo.ac.jp

あらまし 上位設計記述に対する等価性検証では、トランザクションごとに周期的な動作を繰り返す設計を検証する際に、レイテンシとスループットを用いて等価性を指定する手法やそれに基づく検証ツールが提案されている。しかし、その指定には、設計と指定法の両者についての知識が必要となる。本研究では、レイテンシとスループットによる等価性を自動的に推定する手法を提案する。提案手法では、ランダムシミュレーションの結果から入出力信号のレイテンシとスループットの値を絞り込む。加えて、より効率的な推定のために、タグシミュレーションによって、可能性のあるレイテンシとスループットを限定する手法も併せて提案する。いくつかの例題に対する実験において、タグシミュレーションによるレイテンシとスループットの範囲の絞込み、および、ランダムシミュレーションによる2つの設計間で成り立つ等価性の推定が行えることを示す。
キーワード 上位設計、等価性検証、等価性指定、順序回路

Automatic Equivalence Specification between Two Sequential Circuits in High-Level Design

Jinmei XU†, Tasuku NISHIHARA†, Takeshi MATSUMOTO††, and Masahiro FUJITA††,†††

† Department of Electronics Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 Japan
†† VLSI Design and Education Center, University of Tokyo
2-11-16 Yayoi, Bunkyo-ku, Tokyo, 113-0032 Japan
††† Core Research for Evolution Science and Technology, Japan Science and Technology Agency
E-mail: xujinmei@cad.t.u-tokyo.ac.jp, tasuku@cad.t.u-tokyo.ac.jp, matsumoto@cad.t.u-tokyo.ac.jp, fujita@ee.t.u-tokyo.ac.jp

Abstract There are several verifiers available that check the equivalence of high-level designs. In those tools, the equivalence to be checked is specified with the latency and throughput of the given designs. However, specifying them is not easy since it requires the users to have the detailed knowledge of the design and the equivalence specification method. In this work, we propose a method to infer the latency and throughput of a given high-level design. In the proposed method, using the results of random simulation, possbile latencies and throughputs are provided. In addition, to reduce the number of possible latencies and throughputs, we introduce a tag-simulation method where all possible latencies in the given design are statically derived. Through the experiments on example designs, we show that the proposed method is able to show correct latencies and throughputs.
Key words High-level design, equivalence checking, equivalence specification, sequential circuit

1. はじめに

VLSI設計では、後工程での誤り修正による手戻りを防ぐため、設計の初期における機能検証が重要となっており、Cベース言語のような設計言語によってシステム全体を記述する、上位設計と呼ばれる段階で誤りを検出・修正し、同時に、設計中

に新たな誤りを混入させないことが望まれている。後者のための有効な手段の一つとして等価性検証がある。等価性検証は与えられた2つの設計があらかじめ決められた意味(等価性の定義)で等価かどうかを判定する。従って、設計過程における新たな誤り混入を検出するために効果的である。

組合せ回路の等価性検証では、入力と出力の対応を指定し、対応する入力が等価なとき、対応する出力が等価になるかを判定する。一方、順序回路や上位設計の等価性検証では、入出力の対応を与えるだけでは正しく検証ができない場合がある。例えば、時間の概念のない上位設計記述では、入力を与えると所望の計算結果が即座に出力されると考えるが(つまり毎サイクル計算結果が出力されるが)、RTL記述ではハードウェア資源に合わせてスケジューリングが行われるため、その計算の結果が毎サイクル出力されるとは限らない。従って、その2つの設計が設計者が意図した通りに等価であったとしても、対応する出力を毎サイクル比較して検証すると「不等価である」と判定してしまう。そのため、順序回路や上位設計の等価性検証では、入出力が等価になるタイミングを考慮する必要がある。

上位設計の等価性検証ツールには、レイテンシとスループット(周期)によって等価性を指定するものがある。しかし、この等価性の指定には、設計と指定法についての理解が必要である。等価性の指定が難しい例として、複数のポートから並列出力する設計と1つのポートから逐次的に出力する設計、入出力の数が多い設計、などが挙げられる。さらに、他の設計者による設計など、設計に対する理解が不足している場合も成り立つべき等価性を正しく指定することは容易ではない。

以上を踏まえ、本稿では、2つの設計間のレイテンシとスループットで表される等価性を自動推定する手法を提案する。提案手法では、与えられた2つの設計に対してランダムシミュレーションを行い、レイテンシとスループットの値を推定する。提案手法では、(信号名, レイテンシ, スループット)の組を信号の時間属性と呼び、ある信号に有効な値(計算に使われる意味のある入力や計算結果の出力)が現れるタイミングをレイテンシとスループットによって表す。そして、ランダムシミュレーションを行うことによって、等価になる可能性のある信号の時間属性を絞り込む。しかし、入出力のポートが多い場合や、探索するレイテンシやスループットの範囲が広い場合には、この絞込みに要する計算量が大幅に増大してしまう。この問題を解決するため、設計記述に対してタグシミュレーションを行い、入出力間のタイミングを含めた依存関係を解析することにより、あり得る信号の時間属性の数を削減する手法も併せて提案する。

本稿の構成は以下の通りである。第2節では、関連研究として、上位設計における等価性指定、等価なレジスタを推定する手法、タグシミュレーション手法を紹介する。第3節では、提案する等価性の自動推定手法を述べる。第4節で設計例題に対する実験結果を示し、第5節でまとめと今後の課題を述べる。

2. 関連研究

2.1 上位設計における等価性指定

上位設計における等価性指定の必要性は、上位設計の普及に

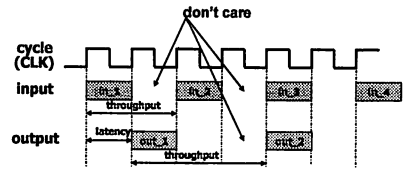

図1 レイテンシとスループット

従って高まっており、近年、いくつかの提案がなされている。文献[1]では、動作記述とRTL設計との間の等価性検証において、検証すべき等価性の指定法が提案されている。そこでは、"attribute statements"と呼ばれる文を動作記述中に挿入することにより、「どの変数の等価性を検証すべきか」「どのタイミングで等価性を検証すべきか」「記述中のどの範囲の等価性を検証すべきか」を指定する。また、Calypto社のSLEC(Sequential Logic Equivalence Checker)[2]は、システムレベル設計記述とRTL設計の間の等価性を検証する商用ツールである。信号処理などを行う順序回路の多くがトランザクションごとに周期的な動作を繰り返す点に着目し、与えられたレイテンシとスループットに従って等価性検証を行う。本研究で提案する手法で推定する等価性も、このレイテンシとスループットによって定義されるものである。レイテンシとスループットは図1のタイミングチャートを用いて説明できる。スループットは、ある入力が与えられてから次の入力が与えられるまでの時間である。一方、レイテンシは、一つの入力が与えられてから、その入力に対応する出力が出力されるまでの時間である。図1の例では、入力 *input* のスループットは2でレイテンシが0、出力 *output* のスループットは3でレイテンシが1である。

2.2 シミュレーションによるレジスター対応の推定

順序回路の等価性検証は、状態変数(フリップフロップ)の対応を取ることによって組合せ回路の等価性検証問題に帰着できることはよく知られている。その状態変数の対応を取る手法としては、シミュレーション結果を利用したもの、次状態関数によるもの、論理回路の構造によるもの、などがある[3]。なお、ここでの等価性とは、基本的に、全てのクロックサイクルにおいて値が同じであることを指す。これらの手法では、まず、初期値が同じ全ての状態変数を同一の等価クラスに入れる。そして、シミュレーション結果や次状態関数によって、等価でない状態変数が同じ等価クラス内にあることが分かった場合、等価クラスを分割する。これを等価クラスの分割が起こらなくなるまで繰り返し、(潜在的に)等価である状態変数を知ることができる。本研究では、これと同様の手法によって、等価になる信号の時間属性の振り分けを行う。

2.3 タグシミュレーション

タグシミュレーション[4]とは、ある信号・変数に生じた変化(タグ)が、どのように伝播するかを調べる手法である。例えば、入力信号にタグが付加された場合、それがある出力信号まで伝播すれば、その出力信号はタグが付けられた入力信号に依存していることが分かる。文献[5]では、Cプログラムの特定の位置で挿入されたタグが、プログラムの出力に伝播されるか

どうかを調べることにより、可観測性の測定を行っている。タグの伝播は、基本的には、タグが付加されている変数・信号を含む演算結果が代入された変数・信号にタグを付加することによって実現できる。この伝播の際に、$\delta-\delta=0$ や $\delta\times 0=0$ のようなタグが伝搬しない場合を特別に考慮することにより、より精度の高いタグ伝播が可能である。本研究では、どのタイミングで与えられた入力信号が、どのタイミングの出力信号に影響を与え得るかを調べるためにタグシミュレーションを用いている。

3. 順序回路に対するレイテンシとスループットによる等価性の自動推定

3.1 対象とする等価性

本研究で対象とする等価性は、2.1 節で紹介したレイテンシとスループットによる指定方法とする。また、一つの設計のレイテンシとスループットの値は入力によらず固定値とする。さらに、スループットは全ての入力および出力信号において共通の値であるとする。一般的な信号処理や単純なパイプライン設計などは、これらの条件を満たす。また、[2] でも同様の等価性を対象としている。

ここで、提案手法において用いる、信号の時間属性について説明する。信号の時間属性とは、ある信号に対する有効なレイテンシとスループットを表したものであり、以下のように定義される。

$$< i,l,t \mid i\in I, l\in Z, t\in Z, l\geqq 0, t\geqq 1 >\in A_i$$

$$< o,l,t \mid o\in O, l\in Z, t\in Z, l\geqq 0, t\geqq 1 >\in A_o$$

ここで、I は入力信号の集合、O は出力信号の集合、Z は整数の集合、l はレイテンシ、t はスループットである。また、A_i は入力信号の時間属性の集合、A_o は出力信号の時間属性の集合である。例えば、図 1 に表されている信号 output の時間属性は、レイテンシが 1、スループットが 3 であるため、$< output, 1, 3 >$ となる。

ある信号の時間属性について、その信号値が有効なサイクルの集合 V は、

$$V=\langle l+j\times t | j\in Z, j\geqq 0\rangle$$

で表される。例えば、図 1 の信号 output の値が有効となるサイクルは $1+3j$ $(j=0,1,2,3,...)$、つまり、1, 4, 7, ... サイクル目となる。

レイテンシとスループットによる等価性指定は、対応する信号の時間属性のペアの集合を指定することにより行う。今、入力信号の時間属性のペアの集合 AC_i と出力信号の時間属性のペアの集合 AC_o によって設計 1 と設計 2 の等価性が指定されているとする。つまり、

$$AC_i=\{< a_{i1,1}, a_{i2,1} >, < a_{i1,2}, a_{i2,2} >, ...\}$$

$$AC_o=\{< a_{o1,1}, a_{o2,1} >, < a_{o1,2}, a_{o2,2} >, ...\}$$

ただし、$a_{i1,1}, a_{i1,2}, ...$ は設計 1 の入力信号の信号属性、$a_{i2,1}, a_{i2,2}, ...$ は設計 2 の入力信号の信号属性、$a_{o1,1}, a_{o1,2}, ...$ は設計 1 の出力信号の信号属性、$a_{o2,1}, a_{o2,2}, ...$ は設計 2 の出力信号の信号属性である。このとき、この等価性指定に対して設計 1 と設計 2 が等価であるとは以下が成り立つことである。

$$\forall < a_{i1,k},\ a_{i2,k} >=$$
$$<< i1_k, l_{i1_k}, t_{i1_k} >, < i2_k, l_{i2_k}, t_{i2_k} >>\in AC_i$$
$$(i1_k[l_{i1_k}+\ j\times t_{i1_k}] = i2_k[l_{i2_k}+j\times t_{i2_k}](j=0,1,2,...))$$
$$\Longrightarrow$$
$$\forall < a_{o1,k},\ a_{o2,k} >=$$
$$<< o1_k, l_{o1_k}, t_{o1_k} >, < o2_k, l_{o2_k}, t_{o2_k} >>\in AC_o$$
$$(o1_k[l_{o1_k}+\ j\times t_{o1_k}] = o2_k[l_{o2_k}+j\times t_{o2_k}](j=0,1,2,...))$$

ただし、$v[t]$ は信号 v の t サイクル目の値である。本研究における等価性の指定は、この等価性の定義を満たすような入出力信号のペアの集合 AC_i と AC_o をそれぞれ選ぶことであると言い換えることができる。また、このとき、各 $a_{i1,k}$ と $a_{i2,k}$、そして、各 $a_{o1,k}$ と $a_{o2,k}$ は等価な信号の時間属性であると呼ぶ。

3.2 提案する等価性指定の自動推定フロー

提案する等価性指定の自動推定フローを図 2 に示す。

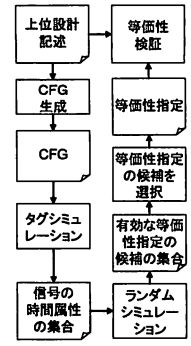

図 2 等価性指定の自動推定フロー

フローの入力は、上位設計記述である。ここでは、主に C 言語ベースの動作レベル設計記述を対象としているが、提案手法は特定の設計記述言語に依存するものではない。

まず、タグシミュレーションを行うためのプリプロセスとして、コントロールフローグラフ (CFG:Control Flow Graph) の生成を行う。次に、生成された CFG に対して静的なタグシミュレーションを適用し、有効である可能性がある信号の時間属性の集合を生成する。ここで、タグシミュレーションを用いずに一定の範囲のレイテンシとスループットについて、信号の時間属性を網羅的に生成することもできるが、次工程のランダムシミュレーションの計算量は、生成した信号の時間属性の数に対して急速に増大するため、タグシミュレーションを用いて生成する数を小さく抑えることが重要である。

続いて、信号の時間属性同士の対応関係を、ランダムシミュレーションを行い、対応する信号の時間属性のペアの集合を得る。ここで、対応する信号の時間属性とは、等価な信号の時間属性である可能性のある信号の時間属性である。最後に、出力

```
int A, B1, B2; //Inputs
int C, D //Outputs
main(){
  int x, y, z;
  if(A > B1) x = A - B1;
  else x = B1 - A;
  y = x + A;
  C = 2 * y;
  if(B2 > 0) D = B2;
  else D = -B2;
}
```

```
int A, B; //Inputs
int C, D //Outputs
main(){
  int x, y, z;
  while(1){
    if(A > B)x = A - B;
    else x = B - A;
    waitfor(1);
    y = x + A;
    C = 2 * y;
    if(B > 0){waitfor(1); z = B;}
    else{z = -B; waitfor(1);}
    waitfor(1);
  }
}
```

図 3 タグシミュレーションを適用する設計例

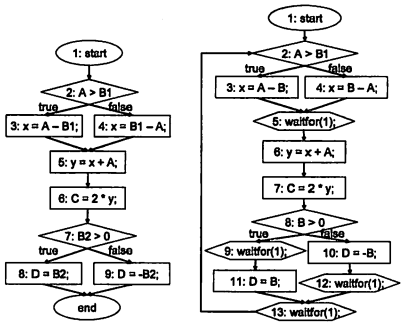

図 4 図 3 の CFG

された対応する信号の時間属性のペアの集合の中から、一定の基準で信憑性が高い集合を選び、それを等価性指定として出力する。これは、レイテンシとスループットによる等価性指定そのものであるため、SLEC などの等価性検証ツールでの検証に直接利用することができる。

3.3 タグシミュレーション

本節では、静的なタグシミュレーションにより、有効である可能性がある信号の時間属性の集合を生成する手法について述べる。

タグシミュレーションの入力は CFG である。CFG とは設計の制御の流れを表した図である。例として、図 3 の C 言語記述の CFG を図 4 に示す。図 3 の左側は通常の C 言語による動作記述記述、右側はスケジューリング後の等価な動作記述である。右側の記述中の $waitfor(1)$ は 1 サイクルの時間経過を表す文である。左側の記述では、main 関数が 1 度実行されることをもって 1 サイクルと数えることにする。図 4 の CFG において、条件分岐は一つのノードからのエッジの分岐となっていることが分かる。

提案するタグシミュレーションは以下の手順で行う。

（1） 現在時刻を 0 として処理をスタートし、エッジを辿る

（2） 辿った先のノードにより、以下の処理を行う

- 代入文: 左辺の変数もしくは信号に対して、右辺に存在する各変数もしくは入力信号から以下の情報をタグとして伝播させる
  - 入力信号: 信号名と現在のサイクルのペア
  - 変数: その変数に伝播しているタグ情報
- 条件分岐: 合流するまでに到達する全ての代入文ノードの左辺に対して、条件文中の変数もしくは入力信号について上の場合と同様にタグが伝播する。
- waitfor 文: 現在時刻を 1 進める

（3） 現在のノードからのエッジを辿り、処理を継続する。条件分岐ノードの場合はエッジが複数出ているため、それぞれを独立のプロセスとして辿り処理を継続する。複数のエッジが合流しているノードでは、全てのエッジの処理が完了するまで処理を停止する。全てのエッジの処理が完了した場合、伝播しているタグ情報を統合し、単一のプロセスとして以降の処理を継続する。

図 4 の右側の CFG に対してタグシミュレーションを適用する例を示す。まず、イタレーションの最初のノードであるノード 2 から処理を開始する。この時の時刻は 0 である。ノード 2 は条件分岐であるため、条件文中に存在する二つの入力信号 A, B について生成されたタグ $(A,0),(B,0)$ が、合流までに到達する代入文の左辺に伝播する。ノード 2 からは複数のエッジが存在するため、true 側と false 側について別々に処理を行う。true 側では、次にノード 3 に遷移する。代入文であるため、左辺の変数 x に対して右辺のタグが伝播する。右辺には二つの入力信号 A, B が存在するため、x に伝播するタグは $(A,0),(B,0)$ となる。これは、ノード 2 の条件分岐のタグと同一である。同様に false 側ではノード 4 について処理を行い、x に $(A,0),(B,0)$ が伝播する。ノード 5 で二つのエッジが合流するため、タグの統合を行う。ここでは、x に伝播しているタグは同一なので、タグの統合による変化は無い。ノード 5 は waitfor 文なので、ここで時刻が 1 になる。ノード 6 の代入文では、右辺の変数 x および入力信号 A についてのタグが伝播する。特に、入力信号 A についてのタグは時刻が 1 であるため、$(A,1)$ となる。すなわち、x のタグと合わせて、左辺の変数 y に伝播するタグは $(A,0),(B,0),(A,1)$ となる。以降、同様に処理を進めると、時刻 1 の時点で出力信号 C に伝播するタグは $(A,0),(B,0),(A,1)$、時刻 1 の時点で出力信号 D に伝播するタグは $(B,1)$ 時刻 2 の時点で出力信号 D に伝播するタグは $(B,1),(B,2)$ である。タグシミュレーションは、ノード 13 において時刻 3 で終了する。すなわち、1 イタレーションにかかる時間は 3 ということになる。主力信号の値は再代入されない限り変化しないと考えると、タグシミュレーションの結果は以下のようになる。

```
(C, 1) : (A, 0), (B, 0), (A, 1)
(C, 2) : (A, 0), (B, 0), (A, 1)
(C, 3) : (A, 0), (B, 0), (A, 1)
(D, 1) : (B, 1)
(D, 2) : (B, 1), (B, 2)
(D, 3) : (B, 1), (B, 2)
```

次に、タグシミュレーションの結果から、有効である可能性がある信号の時間属性の集合を生成する手法について説明する。スループットは設計が同じ状態に戻るまでにかかる時間であるので、タグシミュレーションの終了時刻と一致する。また、その時刻の倍数の時刻についても動作の周期性が保証されるため、スループットである可能性がある。レイテンシについては、タグシミュレーションの結果に含まれる、その信号についての全

表 1　ランダムシミュレーションの結果の例

cycle	$< out1, 0, 1 >$	$< out2, 1, 2 >$	$< out3, 0, 1 >$	$< out4, 1, 1 >$
0	0	8	0	19
1	1	0	2	0
2	2	12	4	1
3	3	3	8	2
4	4	14	10	3
5	5	6	12	4

てのペアの時刻が該当する。なぜならば、タグシミュレーションの結果に含まれるペアの時刻は、出力信号については、その出力信号に入力信号に依存した意味のある値が代入された時刻を表し、入力信号については、その時刻の値が出力に伝播していることを表しているからである。逆に、タグシミュレーションに含まれない時刻の値は、他の信号と無関係であるため、有効な値とは成り得ない。

したがって、スループットの値を 5 以下に限定した場合、先の図 4 右の CFG に対するタグシミュレーションからは、以下の信号の時間属性の集合を得ることができる。

```
<A, 0, 3>, <A, 1, 3>, <B, 0, 3>, <B, 1, 3>, <B, 2, 3>, <C, 1, 3>,
<C, 2, 3>, <C, 3, 3>, <D, 1, 3>, <D, 2, 3>, <D, 3, 3>
```

タグシミュレーションの結果を用いずに、スループットおよびレイテンシが 5 以下の全ての信号の時間属性を生成した場合は、$6 \times 5 \times 4 = 120$ 個の信号の時間属性が生成されるため、タグシミュレーションにより生成する信号の時間属性の数を大幅に削減できていることが分かる。

3.4　ランダムシミュレーションによる信号の時間属性の対応付け

本節では、ランダムシミュレーションにより、前節で生成した信号の時間属性を対応付ける手法について述べる。

まず、入力信号に関する等価性指定が正しく、その等価性指定に適合した入力パタンが入力された場合、シミュレーションの結果、出力信号の値がどのようになるかを考える。この場合、3.1 節で述べた等価性の定義によると、対応する信号の時間属性に対して、有効なサイクルの値の列が等しくなる。例えば、表 1 のランダムシミュレーションの結果の例では、$< out1, 0, 1 >$ の有効なサイクルの値の列は $< 0, 1, 2, 3, 4, 5$ であり、$< out4, 1, 1 >$ の有効なサイクルの値の列は $0, 1, 2, 3, 4$ である。二つの列は完全に一致していないが、これはシミュレーションのサイクルが 5 サイクル目までしか行われていないためであり、二つの信号の時間属性が対応する可能性が残っている。すなわち、二つの信号の時間属性に対して、両者の有効なサイクルの値の列が包含関係にあれば、それらの信号の時間属性は等価になる可能性があるといえる。

本研究では、2.2 節で紹介したレジスタの対応を推定する手法を、この対応の可能性の定義に適合するように拡張し、シミュレーション結果から信号の時間属性の対応を自動推定する手法を提案する。

提案手法では、等価クラスという概念を用いる。等価クラスは信号の時間属性の集合で、所属する信号の時間属性は対応する可能性があることを表す。また、等価クラスは、所属する信号の時間属性の有効な値の列のうち、一番長いものをラベルとして持つ。なぜならば、ある信号の時間属性がその等価クラスに所属できるかどうかは、その信号の時間属性の有効な値の列とそのラベルの列が包含関係にあるかどうかを調べれば判定できるからである。一番長い列は、所属する全ての信号の時間属性の列を含んでいるため、全ての列と比較を行う必要がない。

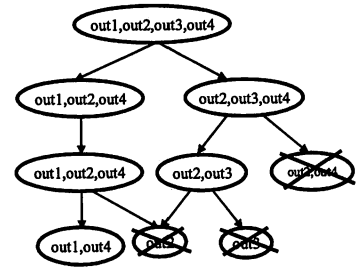

図 5　time-attribute 振り分け図

提案する手法の手順は以下の様である。

（1）　全ての信号の時間属性をを一つの等価クラスに入れる

（2）　各サイクルごとに

（a）　シミュレーション結果を読み込む

（b）　各信号の時間属性について、有効な値が追加されたかを確認する

（c）　各等価クラスについて、所属する信号の時間属性の値が追加されたかどうかを確認する。追加された場合は

- その信号の時間属性がその等価クラスに所属できるかを判定する
- 所属できる場合は、その等価クラスのラベルを更新する (その信号の時間属性の列がその等価クラスの中で一番長い場合)
- 所属できない場合は、その等価クラスから取り除く。取り除いた結果、所属する等価クラスが一つもなくなった場合は、新しい等価クラスを作り、そこに追加する。その際は、元の等価クラスに所属する信号の時間属性のうち、新しい等価クラスに所属し得るもの全てを追加する。
- 所属する要素の数が 1 つ以下になった等価クラスや、所属する要素が全て同一の信号に対するものであるような等価クラスは削除する。

表 1 の例について上記の手法を適用すると、図 5 に示したような等価クラスの分割が行われる。

最終的に生成された等価クラスの集合が等価性指定の候補となるが、その中には冗長なものが含まれる可能性がある。最終的に生成された等価クラスの数が多い場合には、その中からラベルの有効な値の列が長いものを選択するのが良い。なぜならば、有効地の列が長ければ長いほど、偶然同じ等価クラスに入る確率は減少するからである。また、著者が実験した限りでは、入力の等価性指定が誤っていたり、二つの設計が不等価である場合は、最終的に等価クラスが一つも残らない場合がほとんどであり、残った場合も等価性指定においては不十分な数であった。

本節で述べた手法は、入力の等価性指定が正しいことを仮定

表 2　設計の入出力数とタグシミュレーションの結果

設計	入力数	出力数	設計誤り	入力の時間属性数		出力の時間属性数	
				削減前	削減後	削減前	削減後
MAX3							
m1	3	1	無し	100	1	100	1
m2	1	1	無し	100	3	100	3
m3	3	1	無し	100	1	100	1
m4	3	1	有り	100	1	100	1
m5	1	1	無し	100	3	100	2
m6	1	1	無し	100	3	100	2
IDCT							
i1	64	64	無し	10000	1	10000	1
i2	64	64	無し	10000	1	10000	1
i3	64	64	無し	10000	1	10000	1
i3	64	64	無し	10000	1	10000	3

していたが、入力の等価性指定の候補が複数存在する場合には、全ての場合について提案手法を適用する。その際、試行した入力の等価性指定が誤っているかどうかは、残った等価クラスの数から判定できる。したがって、最も多くの等価クラスが残った入力の等価性指定が、正しいものである可能性が高い。

4. 実　　験

本節では、前節で提案した手法を実装して実験を行った結果を示す。今回例題として用いた設計は 2 種類があり、3 つの入力のうち最大のものを返す小さな設計 (MAX3) と、逆離散コサイン変換 (IDCT:Inverse Discrete Cosine Transform) 設計で、SpecC [6] で記述されている。

実験では、設計の詳細度や設計誤りの有無により、MAX3 では 6 種類、IDCT では 4 種類の設計をそれぞれ用意した。MAX3 の行数いずれも 30 行程度、IDCT はいずれも 500 行程度であった。それぞれの設計の入出力数および設計誤りの有無を表 2 の 2, 3, 4 列目に示す。

タグシミュレーションの結果を、表 2 の 5〜8 列目に示す。それぞれの値は、入力/出力信号一つあたりについて生成された信号の時間属性の数である。なお、削減前の値は、一定範囲のレイテンシ・スループットについて網羅的に生成した結果である。指定した範囲は、MAX3 については最大レイテンシは 9・最大スループット 10、IDCT については最大レイテンシ 99・最大スループット 100 である。タグシミュレーションは人手で行った。表の結果から、タグシミュレーションにより大幅に信号の時間属性が削減できていることが分かる。

続いて、ランダムシミュレーションの結果を表 3 に示す。このステップは C++で実装したプログラムを用いて実験を行った。実験には 3.0GHz のプロセッサと 6GB のメモリを持つマシンを使用した。IDCT の例題については、ポート数が多いため、タグシミュレーションによる信号の時間属性の削減を行わない場合は、数時間経過しても処理が終了しなかった。実験結果より、信号の時間属性の削減を行うことにより、このステップにかかる時間を大幅に短縮できていることが分かる。また、設計に設計誤りが含まれる場合や、等価性の指定が誤っている場合は、等価クラスが全て消滅し、信号の時間属性のペアが出力されないことが分かる。

これらの実験結果から分かる、手法の実用性についてについて検討する。ランダムシミュレーションの実験では、入力の等価性指定が正しい場合と誤っている場合を別々に実験したが、

表 3　ランダムシミュレーションの結果

検証対象	入力の指定の正しさ	実行時間 (秒)		生成された出力の等価性指定の有無
		時間属性削減前	時間属性削減後	
m1 vs m3	正しい	9.6	0.02	有り
m1 vs m2	正しい	2.4	0.03	有り
m3 vs m5	正しい	1.4	0.01	有り
m2 vs m6	正しい	3.8	0.02	有り
m3 vs m6	正しい	2.2	0.02	有り
m4 vs m6	正しい	1.4	0.01	無し
i1 vs i2	正しい	-	1.7	有り
i1 vs i3	正しい	-	8.5	有り
i3 vs i4	正しい	-	3.4	有り
i3 vs i4	誤り	-	2.7	無し

実際はどの入力の指定が正しいかは不明であるため、全ての入力の等価性指定について試行する必要がある。誤っている場合はランダムシミュレーションにより生成される出力の等価性指定の数が小さくなるため、正しい場合との区別は容易である。試行回数については、"入力信号数$^{\text{信号一つあたりの時間属性数}}$" となる。タグシミュレーションにより信号の時間属性を削減した結果、実験に使用した例題については数回に抑えられているが、設計によっては、試行回数が膨大となる可能性がある。

5. まとめと今後の課題

本稿では、レイテンシとスループットによる等価性の指定を設計のシミュレーション結果から自動的に推定する手法を提案した。まず、タグシミュレーションを適用することにより、信号の時間属性の数を絞り、ランダムシミュレーションによりそれらの対応関係を解析する。実験を通して、IDCT などのポート数が多く、人手による等価性指定が困難な設計についても、提案手法により正しい等価性指定が推定できることを確認した。

今後の研究課題としては、以下の項目が挙げられる。

- ハンドシェークにより通信するシステムや、コントロールシステムなど、レイテンシやスループットでは等価性が定義できない設計についても対応できるように研究を進める。
- 多重ループを含むなど、静的なタグシミュレーションが適用できない設計を扱えるような手法を考案する。
- 3.4 節の手法はアルゴリズムの改良により更なる高速化が可能だと考えられる。

文　　献

[1] M. Fujita, "Equivalence Checking Between Behavioral and RTL Descriptions with Virtual Controllers and Datapaths," *ACM Transactions on Design Automation of Electronic Systems Science and Technology*, Vol.10, No.4, , pp.610–626, Oct. 2005.

[2] SLEC System
http://www.calypto.com/slecsystem.php

[3] A. Kuehlmann and C. van Eijk, "Combinational and Sequential Equivalence Checking," *in LOGIC SYNTHESIS AND VERIFICATION*, pp.348–350, 2002.

[4] F. Fallah, S. Devadas, and K. Keutzer, "OCCOM: Efficient computation of observability-based code coverage metrics for functional simulation," *Proc. Design Automation Conf.*, pp.152–157, June 1998.

[5] F. Fallah, I. Ghosh, and M. Fujita, "Event-Driven Observability Enhanced Coverage Analysis of C Programs for Functional Validation," *Proc. of Asia South-Pacific Design Automation Conference*, pp.43–48, 2003.

[6] *SpecC: Specification Language and Methodology*, Kluwer Academic Publisher, March 2000.