2B－5

周辺回路を考慮した論理検証のための特徴関数の生成について

田治米 純二† 新井 浩志‡ 深澤 良彰†

†早稲田大学理工学部 ‡千葉工業大学工学部

1 はじめに

ディジタルハードウェアの形式的検証に要する時間は、その回路規模に対して指数関数的に増大するため、従来の検証手法を大規模な回路に適用する事は困難である．一般に、n 入力論理回路は，2^n のすべての論理値組合せを受け付けるとは限らない。我々は、検証対象回路の周辺回路が出力する論理値集合を簡単に求めることができれば、検証すべき入力項目を削減することによって検証を効率化することができると考えている。周辺回路が出力する論理値集合は、特徴関数を用いて表現することができる。本稿では二分決定グラフ (以下 BDD) を用いて特徴関数を表現する。従来、関数ベクトル $F=[f_1...f_n]$ に対して特徴関数として、論理積 $(\epsilon_1 f_1)\wedge...\wedge(\epsilon_n f_n)$(ここで $\epsilon_k f_k$ は f_k、もしくは f_k の否定を表わす) をとる方法 [1] が示されている。この方法を用いると論理積が恒偽でない場合は、出力関数 $\epsilon_k f_k$(k=1,..,n) に対応する解が存在することになる。これにより出力関数の組合せに等価な集合から特徴関数を生成することができる。しかし、この方法では解が存在しない場合にも論理積を計算してしまうため、効率が悪い。本報告では、BDD を用いて特徴関数を効率的に求める方法を提案する。

2 特徴関数の生成

本手法では、特徴関数を入力の組合せから求める。そして、複数の入力組合せをまとめて計算し、出力の組合せを満たす入力組合せが存在しない場合には探索を打ち切る。これにより効率的に特徴関数を求めることができる。

2.1 特徴関数

$B=\{0,1\}$ をブール値の集合とする。関数ベクトル $F=[f_1...f_n]$ は B^m から B^n への関数である。ここで m は関数ベクトル F の変数の数を表わす。この関数ベクトルによって表わされる B^n の部分集合 S の特徴関数 $\chi_F: B^n \rightarrow B$ は、$f_1,..,f_n$ に対応する論理変数 $y_1,..,y_n$ を用いることによって、

$\chi_F(y_1,..,y_n)=1$
　iff
$[y_1...y_n]\in S$ を満足する n 変数論理関数

として定義される。集合を特徴関数で表現した場合、和集合、積集合、補集合などの集合操作を、特徴関数に対する論理和、論理積、論理否定演算に置き換えることによって、集合上の操作を効率的に行なうことができる。

2.2 特徴関数生成アルゴリズム

以下では、関数ベクトル $F=[f_1...f_n]$ に対して特徴関数を表わす BDD を生成する方法について述べる。まず、f_k に対して、入力集合を表わす BDD との論理積をとる。これにより f_k を満たす入力集合を求めることができる。この計算結果は、f_k を満たす入力しかない場合、$\overline{f_k}$ を満たす入力しかない場合、f_k と $\overline{f_k}$ を満たす入力がある場合の 3 通りである。この結果から、入力変数とは異なる f_k に対応する論理変数 y_k を用いて BDD を生成する。そして、それぞれの入力集合と f_{k+1} に対して同じ操作を繰り返す。この処理を $k=1$ から始め、最後の関数 f_n との論理積をとったところで、全ての計算が終わる。その結果から論理演算により、特徴関数を生成していく。具体的なアルゴリズムを図 1に示す。ここで、vtcf は特徴関数生成の再帰的な呼出しを表わし、*input* は入力集合の BDD 表現を表わす。

図 2に関数ベクトル $F=[f_1\ f_2\ f_3]$ (ここでは $f_1=x_1+x_2$, $f_2=x_1\oplus x_2$, $f_3=x_1\cdot x_2$ とする) から特徴関数 χ_F を求める場合の例を示す。まず、全入力の組合せから f_1 もしくは $\overline{f}_1$ を満たすものを調べる。この場合、両方を満たす入力組合せが存在する。そのため特徴関数は $\chi_F=\overline{y_1}\cdot$vtcf($f_2,\overline{f_1}$を満たす入力組合せ)$+y_1\cdot$vtcf($f_2,f_1$を満たす入力組合せ) で求めることができる。これを f_3 まで調べると vtcf の関数呼出しは終了となり値を返す。これにより特徴関数 $\chi_F=\overline{y_1}\cdot\overline{y_2}\cdot\overline{y_3}+y_1\cdot\overline{y_2}\cdot y_3+y_1\cdot y_2\cdot\overline{y_3}$ が得られる。この例では $\{01,10\}$ の出力の値が同じであり、解をまとめて扱うことで計算を効率的に行なっている。

A Method to Generate Characteristic Functions Considering Adjoining Logic Circuits.

Junji Tajime †, Hiroshi Arai ‡ and Yoshiaki Fukazawa †.

† School of Science & Engineering, Waseda University.

‡ Faculity of Engineering, Chiba Institute of Technology.

```
g ← f_k · input
  k = n のとき、
    if(g = false) return(ȳ_n);
    else if(g = input) return(y_n);
    else return(true);
  k < n のとき
    if(g = false)
        h ← vtcf(f_{k+1}, input);
        return(ȳ_k · h);
    else if(g = input)
        h ← vtcf(f_{k+1}, g);
        return(y_k · h);
    else h_1 ← vtcf(f_{k+1}, g);
        g' ← f̄_k · input;
        h_0 ← vtcf(f_{k+1}, g');
        return(y_k · h_1 + ȳ_k · h_0);
```

図 1: 特徴関数生成アルゴリズム

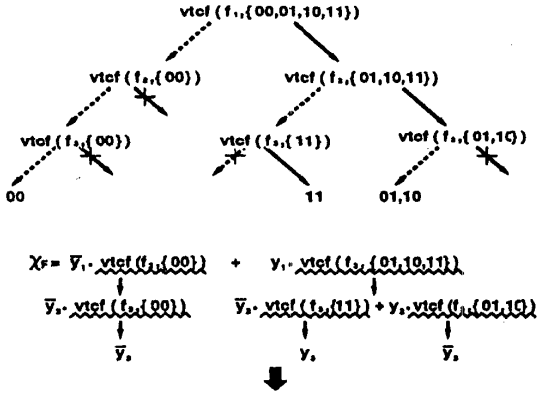

図 2: 特徴関数の計算例

2.3 出力関数の分割

本手法は出力集合が全ての出力の組合せよりも少なくなる場合に有効である。しかし、出力集合が全ての出力の組合せに近い場合は従来と同じように指数関数に比例した計算量が必要となる。本手法を用いて大規模な回路の特徴関数を生成するためには、出力関数を分割する必要がある。しかし、出力関数を分割して個別に特徴関数を求めると、真の出力集合よりも大きい集合が得られてしまう。そこで、真の出力集合に近づけるために出力関数同士の関連が強いものをまとめる必要がある。全く独立な入力を持つ出力関数はその出力値も独立であるため関連はないと考えられる。そのため出力関数の入力変数の重複が多いほど関連が強いと考える。共通の入力変数を持ち、重複しない入力変数が増えないため共通の入力変数を持ち、重複しない入力変数が増えないようにするため、次式 (1) の値が大きいものから出力関数を選んでいく。

(重複した入力変数の数) − (増える入力変数の数) (1)

実際の分割方法としては、一つの出力関数を選び（現段階では入力数の一番多いものか少ないもの）、それに対して (1) 式の値が大きいものから出力関数を選び、設定した出力数に達するまで出力関数をグループに含める。これを出力関数全てを分割するまで行なう。

3 評価とまとめ

本手法と従来手法を KUE-CHIP2[2] の回路で比較をしたものを表 1、2.3 節に述べた方法で回路を分割して生成した解と、真の解との比較を表 2に示す。使用したワークステーションは SUN SPARCstation20 である。表 1で "0.0" と示された部分は処理が 1(1/60 sec) 未満であることを表わしており、"−" は 24 時間以上計算しても処理が終らなかったことを示す。ここで、検証を目的として特徴関数を生成しているため、検査対象の入力と接続がある部分回路の結果を示してある。この結果を見ると、全ての場合に対して本手法が同じか速いことがわかる。2.3 による分割は、単純な分割に比べて多くの場合に良い結果が得られている。

本稿では、特徴関数を生成するためのアルゴリズムについて提案した。これからの課題として、より真の解に近づくような分割方法を考える必要がある。その一つとして単純に数で分けるのではなく、一度用いた出力関数でも関連の強いものは再び選ぶような重複を許した分割方法が有効であると考えている。

表 1: 従来方法との比較

検証対象	特徴関数抽出対象	入力数	出力数	ゲート数	特徴関数抽出処理時間*		
					従来法	全ての入力を試す	本手法
Control	Clockgen.	6	7	15	0.6	0.6	0.0
Control	IR,IDC	9	24	37	316810.0	20.4	10.0
Control	ALU	21	4	216	18.0	66892.0	12.4
Clockgen.	Control	21	2	14	0.0	13714.0	0.0
PC,MAR	Control	30	8	54	9.4	—	2.4
ALU	Control	35	29	102	—	—	537.0

* 単位 (1/60 second)

表 2: 分割での評価

検証対象	特徴関数抽出対象	分割数	特徴関数で展開される出力集合数			
			真の出力集合数	本手法	変更なし	全出力集合数
ALU	Control	10	56012	897120	1258560	536870912
Control	IR,IDC	10	352	7200	13312	16777216
		5	352	86016	75264	16777216
Control	Clockgen.	4	20	24	72	128
PC	Cont.	4	49	70	143	256

参考文献

[1] Coudert et. al. : Verification of Synchronous Sequential Machines Based on Symbolic Execution, *LNCS 407*, Springer-Verlag (1989).

[2] 越智, 澤田, 岡田, 上嶋, 神原, 濱口, 安浦 : "KUE-CHIP2 設計ドキュメント", 京都高度技術研究所 (1992) .