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

2006－ＳＬＤＭ－126（19）
2006／10／27

効率的なブーリアンマッチングのための論理関数の新しいシグネチャについて

松永　裕介†

† 九州大学システムLSI研究センター
〒814-0001 福岡市早良区百道浜 3-8-33
E-mail: †matsunaga@slrc.kyushu-u.ac.jp

あらまし　論理合成の一処理であるテクノロジマッピングにおいて，マッピング対象の部分回路と論理的な等価なセルを見つけるためにNPN同値類の判定を行うブーリアンマッチング処理が用いられる．ブーリアンマッチングでは1つの論理関数と，複数の論理関数の間のマッチを調べる必要があり，処理の効率化のため，各々の論理関数に対して，自身が属するNPN同値類のなかの代表関数を求め，代表関数の一致・不一致によってNPN同値類の判定を行う手法が提案されている．本稿では，NPN同値類の代表関数を求める際に探索すべき候補の数を削減するための新たな指標として重み別Walshシグネチャを提案する．重み別Walshシグネチャは，従来の手法では候補を絞ることが難しかった，多数の極性変換候補を持つ論理関数に対して効果的である．

キーワード　ブーリアンマッチング，NPN同値類，Walsh変換，テクノロジマッピング

New signatures for efficient Boolean matching

Yusuke MATSUNAGA†

† System LSI Research Center, Kyushu University
Momochihama 3-8-33, Sawara-ku, Fukuoka, 814-0001 Japan
E-mail: †matsunaga@slrc.kyushu-u.ac.jp

Abstract　In technology mapping of logic synthesis, Boolean matching that detects NPN equivalence relations is used so to find a library cell logically equivalent to a given sub-circuit to be mapped. Boolean matching requires matching with one logic function against many logic functions coresponding to library cells. To make the matching faster, a method that calculates the representative function of NPN equivalence for each logic function and compares those representative functions to find a match, has been proposed. This paper presents a novel index, called the weighted Walsh signatures, to reduce the number of candidates for searching the representative functions. The weighted Walsh signatures are effective especially for logic functions having many polarity variations, which are not effectively reduced with the previous techniques.

Key words　Boolean matching, NPN equivalence, Walsh transform, technology mapping

1. はじめに

論理関数 f の入力や出力の極性反転と入力順序の入れ替えを行った結果，論理関数 g に変換されたとき，2つの論理関数 f と g はNPN同値類であるという．このNPN同値類の判定を行うことをブーリアンマッチングと呼ぶ．ブーリアンマッチングの良く知られた応用例として，論理合成のテクノロジマッピング中のマッチング処理がある．

典型的なテクノロジマッピング処理は大まかに次の3つの処理からなる[1].

（1）　対象回路の2入力ゲートのみからなるネットワーク(サブジェクトグラフ)への分解

（2）　サブジェクトグラフの部分グラフに対するマッチの列挙

（3）　マッチによる最適被覆(DAG covering)

2番目のマッチの列挙とは，サブジェクトグラフの部分グラフと何らかの意味で「マッチする」をセルライブラリ中のセルを見つける処理である．このマッチの基準としてはグラフの構造に基づくものとグラフの表す論理関数に基づくものが提案されている．後者のマッチの場合，部分グラフの表す論理関数が

セルの論理関数に一致すればその部分グラフにセルが「マッチする」と見なすものである．この論理関数的なマッチを一般化すると，2つの論理関数が一致する場合以外にも，入力順序の入れ替えや入力・出力の極性反転も考慮する必要がある．そのように一般化されたマッチング問題は上記のNPN同値類判定問題，すなわちブーリアンマッチング問題と定式化される．

2つの論理関数 f と g がNPN同値類かどうかを判定する単純な方法として，どちらか一方の論理関数に対して，すべての入力および出力の極性反転の組み合わせと入力順序の組み合わせを試して，他方の論理関数に一致するかどうか確かめる方法が考えられる．しかし，論理関数の入力数を n とすると試すべき組み合わせの数は $2^{(n+1)} \times n!$ であり，n の増大に伴って急激に増えるのでこのような単純なアルゴリズムを使うことはできない．そこで，2つの論理関数が同一のNPN同値類に属するかどうかを判定する様々なブーリアンマッチングアルゴリズムが提案されてきた[2]〜[5].

一方，テクノロジマッピングにおける実際のブーリアンマッチングの問題は，ある部分グラフの表す論理関数 f に対して，f とNPN同値な論理関数を持つセルライブラリ中のセルを求める問題である．もちろん，上記の，セルライブラリ中の各々のセルの論理関数 g に対して，それが f と同一のNPN同値類であるかどうかを判定するアルゴリズムを繰り返し適用すれば所望のセルを見つけることができる(もしくはそのようなセルが存在しないことがわかる)．しかし，この方法ではセルライブラリの規模が大きくなるにつれてブーリアンマッチングにかかる計算時間が線形に増加する傾向にあり効率的とは言えない(注1)．そこで，テクノロジマッピング向きのブーリアンマッチングの手法として，与えられた論理関数に対するNPN同値類の代表関数を求める方法が考案されている[6]〜[11]．これらの手法はテクノロジマッピングの本処理に先立って，セルライブラリ中の各々のセルの論理関数 g_i に対する代表関数 g_i^r を求める前処理を行う．テクノロジマッピングの最中では部分グラフの表す論理関数に対してその代表関数を求め，その代表関数と同一の代表関数を持つセルを探索する処理を行う．すると，テクノロジマッピング中のブーリアンマッチングに必要な計算時間はセルライブラリ中のセル数に独立となり，特に多数のセルをもつセルライブラリに対して有利となる．

代表関数として満たさなければならない性質は，同一のNPN同値類に属する論理関数に対してはユニークな代表関数が選ばれなければならない，ということである．逆に，この性質を満たしていれば上記の方法によって1対多のブーリアンマッチングを行うことができる．そこで，問題はいかに高速に計算できるような代表関数の定義を見つけるか，ということになる．代表関数の計算を効率よく行うための手法として，シグネチャ(signature)と呼ばれる指標を用いて入力順序や入力・出力の極性に関する制約を与えて探索範囲を狭める手法が多く提案されており，そのようなシグネチャでは コファクターシグネチャ/Walsh シグネチャと呼ばれるものが広く用いられている(注2)．しかし，論理関数によってはこのコファクターシグネチャでは全く正規化を行うことができず，代表関数を求めるために多くの変換された論理関数を生成しなければならないものが存在する．本稿では，そのような論理関数の正規化に有効と思われる，重み別 Walsh シグネチャの提案を行う．実験によれば多くの論理関数に対して，この重み別 Walsh シグネチャによる正規化が有効なことが示されている．

以下，2. 章では必要な概念・用語の定義を行い，既存手法について簡単に説明する．次に 3. 章で提案する重み別 Walsh シグネチャの定義およびそれを用いた探索範囲の絞り込みアルゴリズムについて説明する．4. 章では実験によって提案手法が効率よく，効果的に探索範囲を絞り込んでいることを示す．

2. 準備と既存手法の概観

2.1 NPN同値類

[定義 1] n 入力論理関数 $f(x_1, x_2, \ldots, x_n)$ に対して，入力 x_i および出力の極性反転を行う変換を極性変換と呼び，$\pi^N = (p_1, p_2, \ldots, p_n, p_o)$ で表す．ここで $p_i (1 \leq i \leq n)$ および p_o は0または1で，0の時は該当の入力または出力の極性判定がなく，1の時は極性反転があることを示す． □

[定義 2] n 入力論理関数 $f(x_1, x_2, \ldots, x_n)$ に対して，入力 x_i の順序の入れ替えを行う変換を順序変換と呼び，$\pi^P = (q_1, q_2, \ldots, q_n)$ で表す．ここで，$q_i (q \leq i \leq n)$ は，入力 x_i の変換先の位置を示す．たとえば，$q_1 = 3$ の時，もとの関数の1番目の入力 x_1 が変換先の関数の3番目の入力になっていることを表している． □

[定義 3] n 入力論理関数 $f(x_1, x_2, \ldots, x_n)$ に対して，極性変換 π^N を適用し，引き続き順序変換 π^P を適用した変換をNPN変換と呼び，$\pi = (\pi^N, \pi^P)$ で表す．また，論理関数 f がNPN変換 π によって論理関数 g に変換されるとき，$\pi(f) = g$ と表す． □

[例題 1] 論理関数 $f = (\overline{x_1} + \overline{x_2}) \cdot \overline{x_3}$ に対して，NPN変換 $\pi = (\pi^N, \pi^P)$(ただし，$\pi^N = (0,0,1,0)$，$\pi^P = (2,3,1)$)を適用することを考える．まず，極性変換 π^N によって，$f' = \pi^N(f) = (\overline{x_1} + \overline{x_2}) \cdot x_3$ と変換される．次に，順序変換 π^P によって，$g = \pi(f) = \pi^P(f') = x_1 + (\overline{x_2} \cdot \overline{x_3})$ と変換される． **— 例題終わり**

[定義 4] 論理関数 f に対してあるNPN変換を適用した結果の関数を g となるとき，f は g とNPN同値であるといい，$f \equiv g$ と表す． □

[定理 1] NPN同値は同値関係である．すなわち，NPN同値の関係は，

- 反射律 $f \equiv f$
- 対称律 $f \equiv g$ なら $g \equiv f$

(注1)：多くの場合，簡単な手間で2つの関数が同一のNPN同値類ではないと判定できるので，実際にはライブラリサイズに比例することはない．

(注2)：コファクターシグネチャ と Walsh シグネチャは異なるものだが後述するように本質的には等価なシグネチャである．

- 推移律 $f \equiv g$ かつ $g \equiv h$ なら $f \equiv h$

を満たす.

[定義 5] 定理 1 より NPN 同値の関係に基づいて，論理関数の同値類集合を定義することができる．すなわち，その要素のいかなるペアも NPN 同値となるような関数の集合を定義することができる．そのような集合を NPN 同値類と呼ぶ． □

2 つの論理関数に対するブーリアンマッチング問題とは，与えられた 2 つの論理関数 f と g が NPN 同値であるかを判定する問題であり，複数の論理関数に対するブーリアンマッチング問題とは与えられた 1 つの関数 f および，複数の論理関数 $G = \{g_1, g_2, \ldots\}$ に対して，f と NPN 同値な論理関数 g_i を見つける問題と見なすことができる．後者のブーリアンマッチング問題でよく用いられている手法は NPN 同値類に対する代表関数を求めるものである．何らかの方法により，すべての NPN 同値類に対して各々1 つずつの代表関数を定めておき，任意の論理関数 g からその関数が属している NPN 同値類の代表関数 g^r を求める手段を定めておくものとする．すると，もしも 2 つの関数 f と g が同じ NPN 同値類に属しているとすると，それらの代表関数は等しくなる．すなわち $f^r = g^r$ が成り立つことになる．テクノロジマッピングにおけるブーリアンマッチングの場合，一方の関数 f はテクノロジマッピング処理の最中にはじめて生成されるため，その場で代表関数を求める必要があるが，もう一方の関数群 $G = \{g_1, g_2, \ldots\}$ はセルライブラリが決まった時点で生成されており，それらの関数に対する代表関数群 $G^r = \{g_1^r, g_2^r, \ldots\}$ はテクノロジマッピング処理を始める前にあらかじめ計算しておくことができる．一般に，テクノロジマッピングで用いられるセルの入力数はほとんどが 10 入力以下であり，多くても 20 入力を越えることはない．そのため，f の代表関数 f^r に一致する関数が $G^r = \{g_1^r, g_2^r, \ldots\}$ に含まれているかの検索はハッシュ関数を併用することでほとんど定数時間で行うことができる．つまり，1 回のブーリアンマッチングにかかる計算時間は f の代表関数 f^r を求める処理のみに依存し，セルライブラリ中のセル数には依存しないことになる．

2.2 コファクタシグネチャと Walsh シグネチャ

代表関数の選びかたは唯一の方法があるわけではなく，さまざまなものが提案されている．文献 [9], [11] で用いられている方法は，n 入力論理関数を真理値表で表したときのビットベクタを 2^n ビットの 2 進数とみなした時に，もっとも大きな数となる関数を代表関数とするものである．単純なやり方では $2^{(n+1)} \times n!$ 通りのすべての関数をいったん生成して，そのなかから最大のビットベクタを選ぶ方法が考えられるが効率が悪い．そこで，文献 [9] では前もって計算しておいたビットベクタと変数順の表を用いて，最大とならない変換候補を枝刈りしている．文献 [11] ではさらに考慮すべき関数の変換候補を減らすためにいくつかのテクニックを用いている．

[定義 6] 論理関数 f を 1 にする入力値の要素数を f の真理値数と呼び $|f|$ で表す． □

文献 [11] では他の多くの手法と同様に，f の真理値数と $\overline{f}$ の真理値数を比較し，大きい方の出力極性のみを探索候補として残すというヒューリスティックを用いている．これは真理値数が入力極性の反転および入力順序の入れ換えに対して不変であるという性質を用いたもので，同じ NPN 同値類の関数であれば必ず同じ出力極性が選ばれる．ただし，$|f| = |\overline{f}|$ のときには注意が必要で，この場合には真理値数を用いて出力の極性を定めることはできない．

[定義 7] 論理関数 f に対して $|\overline{f}| - |f|$ を f の 0 次の Walsh シグネチャと呼び，$W^0(f)$ と表す． □

定義 7 により，次式が成り立つ．

$$W^0(f) = (2^n - |f|) - |f| \quad (1)$$
$$= 2^n - 2 \times |f| \quad (2)$$
$$W^0(\overline{f}) = 2^n - 2 \times |\overline{f}| \quad (3)$$
$$= 2^n - 2 \times (2^n - |f|) \quad (4)$$
$$= -2^n + 2 \times |f| \quad (5)$$
$$= -W^0(f) \quad (6)$$

式 (2) より真理値数と 0 次の Walsh シグネチャが互いに変換可能なことがわかる．つまり，0 次の Walsh シグネチャを用いても真理値数と同様に出力極性の絞りこみを行うことができる．さらに，式 (6) より，論理関数の出力の極性反転を行った場合に 0 次の Walsh シグネチャも正負の符号が反転することがわかる．極性反転がそのまま符号反転となるので説明が単純となるため，以降の説明では 0 次の Walsh シグネチャを用いることとする．

[定義 8] 論理関数 f および，その入力 x_i があたえられたとき，x_i に 1(または 0) を代入した時の関数を f の x_i に対するコファクター (cofactor) と呼び，$f_{x_i=1}$(または $f_{x_i=0}$) と表す． □

[定義 9] 論理関数 f および，その入力 x_i があたえられたとき，$|f_{x_i=1}|$ を f の x_i に対するコファクターシグネチャと呼び，$C^1(f, x_i)$ と表す． □

コファクターシグネチャは x_i 以外の入力の極性反転や x_i 以外の入力の順序の入れ換えに対して不変であるという性質を持つ．そこで，文献 [11] などでは $C^1(f, x_i)$ と $C^1(f, \overline{x_i})$ の大小を比較して大きい方の値を取る入力の極性を選ぶヒューリスティックを用いている．$C^1(f, x_i) = C^1(f, \overline{x_i})$ となる場合を除いて，同一の NPN 同値類に属する関数に対しては一様に同じ極性を選ぶことができる．

[定義 10] 論理関数 f および，その入力 x_i があたえられたとき，$W^0(f \oplus x_i)$ を f の x_i に対する 1 次の Walsh シグネチャと呼び，$W^1(f, x_i)$ と表す． □

0 次の Walsh シグネチャの場合と同様に 1 次の Walsh シグネチャはコファクタシグネチャ(および真理値数) から計算することが可能であり，また，$W^1(f, x_i) = -W^1(f, \overline{x_i})$ という性質を満たす．以降の説明では，コファクタシグネチャの代わりに 1 次の Walsh シグネチャを用いる．

ある関数の 0 次の Walsh シグネチャと 1 次の Walsh シグネ

チャを計算したところ，それら全てのシグネチャが非0の場合には，代表関数を得るための極性変換が一意に定まったことになる．さらに，全ての1次のWalshシグネチャの値が異なっていたら，その値を降べき(あるいは昇べき)の順に並べることで，一意に入力順序を決めることができ，このような場合には，以降の処理は不要となる．

2.3 対称変数

多くの場合，上記のようなWalshシグネチャが得られることは稀で，0値のシグネチャにより極性が定まらない場合や，同一の値を持つ複数の1次のWalshシグネチャがあるために入力順序が定まらない場合がある．しかし，複数の入力の1次のWalshシグネチャが同一の場合には，それらが対称変数である場合が考えられる．

[**定義** 11] 論理関数 f および入力 x_i，$x_j(i<j)$ に対して，x_i と x_j の順序を入れ替えた順序変換 $\pi^P=(1,2,\ldots,i-1,j,i+1,\ldots,j-1,i,j+1,\ldots,n)$ を考える．$\pi^P(f)=f$ の時，すなわち，f 中の x_i と x_j を取り替えても論理関数が変化しない時，x_i と x_j は対称であるといい，x_i SYM x_j と表す(注3)． □

定義により，対称変数に関する順序変換はもとの論理関数を変えないので，区別する必要はない．そこで，対称な変数の集合を求めて，それらが連続した位置に現れる入力順序のみを考慮することで，探索すべき入力順序を大幅に減らすことができる．例えば，x_1,x_2,x_3,x_4 の4つの入力の順列は全部で24通り考えられるが，もしも x_1 と x_3 が対称とわかれば，考慮するのは以下の6通りとなる．

(x_1,x_3,x_2,x_4) (x_1,x_3,x_4,x_2)

(x_2,x_1,x_3,x_4) (x_2,x_4,x_1,x_3)

(x_4,x_1,x_3,x_2) (x_4,x_2,x_1,x_3)

変数の対称性に関してはもう少し複雑な場合が存在するが，本稿の主題から離れるため，ここでは詳細は省略する．

1次のWalshシグネチャを用いても組み合わせの数を減らせない場合の処理として，高次の(複数の変数を用いた)Walshシグネチャを用いた正規化処理が提案されている[10](注4)．この手法は入力変数順を確定させるためには有効な場合が多いが，入力の極性が確定していない場合(つまり，$W^1(f,x_i)=0$ の時)には有益な情報が得られないことが多い．また，探索アルゴリズムが再帰呼び出しの形で記述されており処理が複雑になっている．

3. 重み別 Walsh シグネチャ

前述の様に，最近のほとんどすべてのブーリアンマッチングアルゴリズムは0次のWalshシグネチャ，1次のWalshシグネチャおよび対称変数の情報を用いて，探索すべき変換候補を絞り込んでいる．極性変換に関しては，0次のWalshシグネチャの正負で出力極性が制限され，1次のWalshシグネチャの正負および対称性によって入力の極性が制限されるが，これらの制限は入力順序に依存しない．また，順序変換は1次のWalshシグネチャの絶対値の大小および対称性の情報によって制限されるが，こちらは入力・出力の極性には依存しない．つまり，0次のWalshシグネチャ，1次のWalshシグネチャおよび対称変数の情報を用いて絞り込んだ変換の集合 $\Pi=\{\pi_i\}$ は，極性変換の集合 $\Pi^N=\{\pi_j^N\}$ と順序変換の集合 $\Pi^P=\{\pi_k^P\}$ の直積集合 $\Pi^N\times\Pi^P=\{(\pi_j^N,\pi_k^P)\}$ で表される．

この直積集合を展開する前にどちらかの変換集合をさらに絞り込むことができれば探索すべき組合わせの数を大幅に減らすことが期待できる．そこで，極性変換の集合を絞り込むための新しいシグネチャとして重み別Walshシグネチャと呼ぶ指標を提案する．

[**定義** 12] n 次元のビットベクタ $m=(m_1,m_2,\ldots,m_n)$ (ただし $m_i\in\{0,1\}$) に対して，m 中の1の数を m の重みと呼び，$w(m)$ と表す．つまり，$w(m)=|\{m_i|m_i=1\}|$ である． □

[**定義** 13] n 入力論理関数 f に対して $f(m)=1$ となるような入力 m のうち，重みが k である物の数を重み k の真理値数と呼び $C(f,k)$ で表す．つまり，$C(f,k)=|\{m|m\in\{0,1\}^n,f(m)=1,w(m)=k\}|$ である． □

[**定義** 14] 重みが k であるような入力のみを考えた0次のWalshシグネチャを重み k の0次のWalshシグネチャと呼び，$WW^0(f,k)$ で表す．$WW^0(f,k)=C(\overline{f},k)-C(f,k)$ である． □

[**定義** 15] n 入力論理関数 f に対して，重み0から n までの0次Walshシグネチャを並べたものを重み別0次のWalshシグネチャと呼び $\mathcal{W}^0(f)$ で表す．$\mathcal{W}^0(f)=(WW^0(f,0),WW^0(f,1),\ldots,WW^0(f,n))$ である． □

この重み別の0次のWalshシグネチャは入力の極性に依存し，入力の順序には依存しない．そこで，この重み別の0次のWalshシグネチャを用いて極性変換の絞り込みを行う(図1)．

```
wwalsh_0(Π^N) {
    max_vector ← (−∞, −∞, ..., −∞)
    Π^N_new ← Φ
    foreach π^N ∈ Π^N {
        U ← W^0(π^N(f))
        if U > max_vector {
            max_vector ← U
            Π^N_new ← {π^N}
        }
        else if U = max_vector
            Π^N_new ← Π^N_new ∪ {π^N}
    }
    return Π^N_new
}
```

図1 重み別0次のWalshシグネチャを用いた絞り込み

(注3)：この対称性を non-equivalence symmetry と呼び，x_i NE x_j と表している文献もある．

(注4)：この論文中ではコファクターシグネチャと呼んでいる

上記のアルゴリズム中で重み別 0 次の Walsh シグネチャ ($n+1$ 次元ベクタ) 同士の比較は辞書式順序で行うこととする．つまり，2 つのベクタ $(u_0, u_1, \ldots, u_n)$ と $(v_0, v_1, \ldots, v_n)$ を比較するとき，まず最初に u_0 と v_0 を比較する．もしも同値の場合には u_1 と v_1 を比較する，という風に順に要素を比較する．

0 次の Walsh シグネチャと同様に 1 次の Walsh シグネチャに関しても重みを考慮して $n+1$ 次元のベクタ化することが可能である．

[定義 16] n 入力論理関数 f とその入力 x_i に対して $\mathcal{W}^0(f \oplus x_i)$ を重み別 1 次の Walsh シグネチャと呼び，$\mathcal{W}^1(f, x_i)$ で表す．□

重み別 1 次の Walsh シグネチャは入力の極性に依存し，入力の順序には依存しない (x_i の順序を不変とした場合)．そこで，図 1 と同様のアルゴリズムでさらに極性変換の候補を絞ることが可能である．また，もしも極性変換が唯一に定まっている場合には，対称ではないが 1 次の Walsh シグネチャが等しい 2 つの入力 x_i, x_j の重み別 1 次の Walsh シグネチャを比較することで，順序変換の候補を絞り込める場合がある．

このように，0 次および 1 次の Walsh シグネチャを拡張することで，従来よりも多くの情報を得ることができ，結果として代表関数を求めるために探索しなければならない変換の数を減らす効果が期待できる．

4. 実験結果

上記のシグネチャの効果を調べるために，入力数 4 から 15 までの論理関数をそれぞれ 10000 個，ランダムに生成し，NPN 同値類の代表関数を求める実験を行った．変換候補を絞り込む処理は以下の 3 段階で行った．

baseline: 0 次の Walsh シグネチャ，1 次の Walsh シグネチャおよび対称変数の情報を用いて変換候補の絞り込みを行う．

w0: baseline の結果に対して重み別 0 次の Walsh シグネチャを用いた絞り込みを行う．

w1: w0 の結果に対して重み別 1 次の Walsh シグネチャを用いた絞り込みを行う．

各々の処理の後で，以下の計測を行う．

#undet: 変換候補が唯一に定まらなかった関数の数

#pundet: 極性変換候補が唯一に定まらなかった関数の数

#cand: 延べの変換候補数 (唯一に定まる場合を除く)

結果を表 1 に示す．一見して明らかなように，baseline の絞り込みはあまり有効ではなく，多くの変換候補が残っている．一方，重み別 0 次の Walsh シグネチャを用いると，多くの極性変換候補を削除することができている．この傾向は入力数が多くなるにつれて顕著になる．ただし，重み別 0 次の Walsh シグネチャは順序変換に関してはなにもしないので延べの変換候補数はさほど減少していない．最後に，重み別 1 次の Walsh シグネチャを用いると 7 入力以上のほとんどの関数に対して代表関数を求める唯一の変換候補を得ることができている．入力数が少ない場合には重み別のベクタの次元も低いのであまり効果的な情報が得られないためと思われる．今回の実験ではランダムに生成した 7 入力以上の関数に対しては，提案する重み別 Walsh シグネチャが非常に効果的であることが確認された．

表 1 重み別 Walsh シグネチャの効果

入力数	計測項目	baseline	w0	w1
4	#undet	7706	6160	3697
	#pundet	4536	1895	1895
	#cand	75906	38568	6145
5	#undet	9860	9721	3899
	#pundet	4379	648	98
	#cand	293064	116287	5641
6	#undet	9997	9990	1421
	#pundet	3951	150	0
	#cand	373929	175416	1615
7	#undet	9987	9969	236
	#pundet	3497	16	0
	#cand	389858	202623	238
8	#undet	9974	9949	16
	#pundet	3071	0	0
	#cand	357324	206995	16
9	#undet	9939	9899	2
	#pundet	2600	0	0
	#cand	272895	184301	2
10	#undet	9852	9777	0
	#pundet	2189	0	0
	#cand	220472	135065	0
11	#undet	9684	9581	0
	#pundet	1710	0	0
	#cand	126494	99257	0
12	#undet	9374	9245	0
	#pundet	1360	0	0
	#cand	89318	76749	0
13	#undet	8976	8795	0
	#pundet	1091	0	0
	#cand	64478	55517	0
14	#undet	8412	8227	0
	#pundet	842	0	0
	#cand	45664	39315	0
15	#undet	7598	7415	0
	#pundet	645	0	0
	#cand	25756	25217	0

次に，計算時間に関する実験結果を示す．他の文献にも示されている通り，今日の標準的な計算機を用いると，このようなブーリアンマッチングは数マイクロ秒で実行可能である．ところが，マイクロ秒単位の時間計測は誤差が大きいため，今回の実験では同一の関数に対する処理を 100 回繰り返し，総実行時間を 100 で割ることで，近似的に一回の処理時間を計測している．使用計算機は Pentium 4(2.8GHz) である．表 2 にその実験結果を示す．

baseline と記された列は baseline 処理で候補の絞り込みを行った後で残った変換候補を全列挙して代表関数を求めたときの処理時間 (単位はマイクロ秒) である．括弧内は baseline 処理のみにかかった時間である．w1 と記された列は baseline 処

表 2 処理時間

入力数	baseline		w1	
4	6.2	(0.6)	3.4	(2.1)
5	14.7	(0.8)	3.2	(2.3)
6	17.9	(0.9)	4.7	(4.4)
7	24.8	(1.1)	5.0	(4.9)
8	38.1	(1.3)	5.3	(5.3)
9	65.5	(1.7)	5.9	(5.9)
10	113.8	(2.2)	6.5	(6.5)
11	186.1	(3.5)	7.8	(7.8)
12	298.0	(5.7)	9.7	(9.6)
13	491.0	(10.2)	13.6	(13.6)
14	804.1	(20.5)	22.1	(21.2)
15	1281.8	(40.8)	45.1	(41.2)

理の後で 0 次および 1 次の重み別 Walsh シグネチャを用いた絞り込みを行い，残った変換候補から代表関数を求めたときの処理時間と，baseline 処理および重み別 Walsh シグネチャを用いた絞り込みのみにかかった時間 (括弧内) である．表 1 の結果から推測できるように baseline のみの場合探索すべき変換候補が減っていないので後半の代表関数の探索に多くの時間を費やしている．一方，重み別 Walsh シグネチャを用いた場合には探索にはほとんど時間を費やしておらず，ほとんどの処理時間は (通常の)0 次および 1 次の Walsh シグネチャを計算する処理で費やされていることがわかる．後半の探索アルゴリズムは単純なものを用いているので高速化の余地はあるが，重み別 Walsh シグネチャを用いた場合には探索にかかる時間は無視できるので特に文献 [10] のような複雑なアルゴリズムを実装する必要性はないと思われる．今回の実験では論理関数を真理値ベクタ (真理値表の列をベクタと見なしたもの) の形で実装しているので，Walsh シグネチャの計算などの論理関数処理は入力数に対して指数的に時間がかかるようになっている．特に入力数が多い場合には実装に検討が必要な項目である．逆に入力数の多い論理関数に対しては提案している重み別 Walsh シグネチャが少ないオーバーヘッドで大きな効果をあげている．これは，重み別 Walsh シグネチャの場合，一部の重みのシグネチャを求めただけで処理が終わる場合が多いので，通常の Walsh シグネチャに比べて入力数が多いときのオーバーヘッドが少ないと考えられる．

絶対的な処理時間を他の文献と比較することは用いている論理関数の集合が異なることや実験の条件が異なるため難しいが，10 入力の論理関数に対する NPN 同値類の代表関数を 10 マイクロ秒以内で見つけている例はほとんどなく，今のところ，知られている手法の中では最高速の部類に入るものと思われる．

5. おわりに

本稿では，ブーリアンマッチングで用いられる，NPN 同値類の代表関数の探索時に候補となる変換を絞り込むための新たな手法として重み別 Walsh シグネチャを提案し，実験によりその有効性を確認した．今後は実際のテクノロジマッパの中で用いられる場合に適した論理関数処理などの実装に関して検討を行っていく予定である．

謝辞 本研究の一部は文部科学省「知的クラスタ創成事業」補助金による．

文 献

[1] R. Rudell: "Logic Synthesis for VLSI Design", PhD thesis, U. C. Berkeley (1989).

[2] F. Mailhot and G. D. Micheli: "Technology Mapping Using Boolean Matching", Proceedings of the European Conference on Design Automation, pp. 180–185 (1990).

[3] J. Mohnke and S. Malik: "Permutation and phase independent Boolean comparison", Proceedings of IEEE European Conference on Design Automation, pp. 86–92 (1993).

[4] Y. Matsunaga: "A New Algorithm for Boolean Matching Utilizing Structural Information", IEICE Trans. Inf. & Syst., **E78-D**, no. 3, pp. 219–223 (1995).

[5] L. Benini and G. D. Micheli: "A survey of Boolean matching techniques for library binding", ACM transaction on Design Automation of Electronic Systems, **vol. 2**, no. 3, pp. 193–226 (1997).

[6] J. B. Burch and D. E. Long: "Efficient boolean function matching", Proceedings of IEEE International Conference on Computer Aided Design, pp. 408–411 (1992).

[7] U. Hinsberger and R. Kolla: "Boolean matching for large libraries", Proceedings of IEEE/ACM Design Automation Conference, pp. 206–211 (1998).

[8] J. Circ and C. Sechen: "Efficient canonical form for Boolean matching of complex functions in large libraries", IEEE Trans. Comput.-Aided Design Integrated Circuits, **vol. 22**, no. 5, pp. 535–544 (2003).

[9] D. Debnath and T. Sasao: "Efficient computation of canonical form for Boolean matching in large libraries", Proceedings of Asia and South Pacific Design Automation Conference, pp. 591–596 (2004).

[10] A. Abdollahi and M. Pedram: "A new canonical form for fast Boolean matching in logic synthesis and verification", Proceedings of IEEE/ACM Design Automation Conference, pp. 379–384 (2005).

[11] D. Chai and A. Kuehlmann: "Building a Better Boolean Matcher and Symmetry Detector", Proceedings of Design Automation and Test in Europe, pp. 391–398 (2006).