基于对称及特征的NPN布尔匹配算法

张菊玲[1,3]，杨国武[1]，吴尽昭[2]，郭文强[3]

(1. 电子科技大学计算机科学与工程学院大数据研究中心 成都 611731；

2. 广西民族大学广西混杂计算与集成电路设计分析重点实验室 南宁 530006；

3. 新疆财经大学计算机科学与工程学院 乌鲁木齐 830012)

【摘要】该文提出了一种基于对称及特征的成对比较的NPN布尔匹配算法，利用变量对称、1阶特征向量及香农分解设计完成了NPN布尔匹配。算法利用具有相同1阶特征向量是两个布尔函数NP等价的必要条件，和具有相同1阶特征是两个变量具有映射关系的必要条件搜索两个布尔函数之间的候选变换并进行验证。对称及特征的使用降低了候选变换搜索的空间，提高了NPN等价匹配的速度。

关 键 词 NPN等价； 香农分解； 工艺映射； 变量映射； 变量对称

中图分类号 TP302.2 **文献标志码** A doi:10.3969/j.issn.1001-0548.2018.06.012

NPN Boolean Matching Algorithm Based on Symmetry and Signature

ZHANG Ju-ling[1,3], YANG Guo-wu[1], WU Jin-zhao[2], and GUO Wen-qiang[3]

(1. Big Data Research Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731;

2 Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities Nanning 530006;

3 School of Computer Science and Engineering, Xinjiang University of Finance and Economics Urumqi 830012)

Abstract The paper proposes a pairwise NPN (Input Negation and/or Input Permutation and/or Output Negation) Boolean matching algorithm based on symmetry and signature. The algorithm utilizes variable symmetry, the first order signature vector and Shannon decomposition to design and implement NPN Boolean matching. The candidate NP transformations between two functions are searched and verified through two necessary conditions: 1) two NP equivalent Boolean functions must have the same the first order signature vector and 2) two variables having a mapping relation must have the same the first order signature. The use of symmetry and signature reduces the search space of the candidate NP transformations and speeds up NPN Boolean matching process.

Key words NPN equivalent; Shannon decomposition; technology mapping; variable mapping; variable symmetry

在集成电路设计中，判定两个布尔函数(NPN)等价问题具有重要的意义。NPN布尔匹配在电路设计的工艺映射中是一个关键步骤。对于一个给定的电路，工艺映射需要从标准电路库中找到一个与指定电路NPN等价的最优逻辑门组合电路[1-3]。如果布尔函数 f 与 g 是NPN等价的，那么就可以使用函数 f 的电路来实现 g 的电路。

当前，主要有3种NPN布尔匹配方法。1) 成对比较匹配算法。该方法对于给定的两个布尔函 f 数与 g，通过变量的特征寻找两个函数的变量之间的关系，从而找到可以将 f 转换为 $g/\overline{g}$ 的变换[4-7]。2) 基于正规式的算法。该方法从每个NPN等价类中找到具有特殊值的某个布尔函数作为该等价类的正规式。文献[8-13]中的作者研究了基于正规式的布尔匹配算法。正规式通常具有最大或最小真值，或是具有最大特征向量的函数。给定两个待匹配的布尔函数 f 和 g，该类算法分别计算 f 的正规式 F 与 g 的正规式 G。如果满足 $F=G$，那么 f 与 g 是NPN等价的。3) 基于SAT(boolean satisfiability)的匹配算法，该类算法将布尔匹配问题转换为SAT问题进行求解[14-15]。

1 基本概念及问题陈述

$f(x_1,x_2,\cdots,x_n)$ 是n个输入1个输出的布尔函数，

收稿日期：2017－12－26；修回日期：2018－06－15

基金项目：国家自然科学基金(61572109，11371003，61163066)

作者简介：张菊玲(1977－)，女，博士生，主要从事逻辑综合方面的研究.

$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 表示函数 f 的n维向量，文献[7]将 f 的最小项个数记为$|f|$。布尔函数可以用真值表、BDD(binary decision diagram)或01串表示。因为采用BDD可以快速的计算布尔函数的香农余子式特征，所以本文采用BDD表示布尔函数。

定义 1　NP变换：NP变换T是对布尔函数输入的非运算和置换运算，$T\boldsymbol{X}=(x_{\pi(1)}^{\varphi(1)},x_{\pi(2)}^{\varphi(2)},\cdots,x_{\pi(n)}^{\varphi(n)})$，其中 $\pi(i)\in\{1,2,\cdots,n\}$ 是对 x_i 的置换。

如 $\pi(1)=3$ 表示变量 x_1 被置换为 x_3；$\varphi(i)\in\{0,1\}$ 是对 x_i 的非运算，当 $\varphi(i)=1$ 时表示对变量 x_i 做非运算，否则不做非运算。

定义 2　NPN等价：函数 $f(\boldsymbol{X})$ 和 $g(\boldsymbol{X})$ 是NPN等价的，当且仅当存在一个NP变换T满足条件 $f(T\boldsymbol{X})=g(\boldsymbol{X})$ 或者 $f(T\boldsymbol{X})=\overline{g(\boldsymbol{X})}$。

例1：设有布尔函数 $f(\boldsymbol{X})=x_1\overline{x_2}+x_1x_2x_3$，$g(\boldsymbol{X})=x_2x_3+x_1x_2\overline{x_3}$ 和 $h(\boldsymbol{X})=x_3+\overline{x_1}\,\overline{x_3}+\overline{x_2}\,\overline{x_3}$。

因为存在一个 $T=(x_2,\overline{x_3},x_1)$ 使得条件 $f(T\boldsymbol{X})=g(\boldsymbol{X})$ 和 $f(T\boldsymbol{X})=\overline{h(\boldsymbol{X})}$ 成立，所以 $f(\boldsymbol{X})$ 与 $g(\boldsymbol{X})$ 是NP等价，与 $h(\boldsymbol{X})$ 是NPN等价的。

定义 3　变量映射：NP变换T也可表示为在向量$\boldsymbol{X}$上的一组映射 $T=\{x_1\to x_{\pi(1)}^{\varphi(1)},x_2\to x_{\pi(2)}^{\varphi(2)},\cdots,x_n\to x_{\pi(n)}^{\varphi(n)}\}$。若布尔函数 f 和 g 在NP变换T的作用下是NP等价的，那么函数 f 中的变量 x_i 与函数 g 中的变量 $x_{\pi(i)}$ 之间有映射 $\alpha_i:x_i\to x_{\pi(\mathrm{i})}^{\varphi(\mathrm{i})}$。

变量 x_i 和 x_j 之间存在的映射分为两种情况：1) 同相映射 $x_i\to x_j$（$\overline{x_i}\to\overline{x_j}$）；2) 异相映射 $x_i\to\overline{x_j}$（$\overline{x_i}\to x_j$）。例1中的NP变换T可以表示为：$T=\{x_1\to x_2,x_2\to\overline{x_3},x_3\to x_1\}$。

定义 4　变量对称：布尔函数 f 中的变量 x_i 与 $x_j/\overline{x_j}$ 是对称的，当且仅当交换 x_i 与 $x_j/\overline{x_j}$ 之后函数 f 的值不变。

定义 5　香农余子式：布尔函数 f 关于变量 $x_i/\overline{x_i}$ 的香农余子式记为 $f_{x_i}/f_{\overline{x_i}}$，其中 $f_{x_i}=f[x_i\leftarrow 1]$，$f_{\overline{x_i}}=f[x_i\leftarrow 0]$。

当变量 x_i 和 x_j 满足条件 $f_{x_i\overline{x_j}}=f_{\overline{x_i}x_j}$ 时，变量 x_i 与变量 x_j 对称；当变量 x_i 和 x_j 满足条件 $f_{x_ix_j}=f_{\overline{x_i}\,\overline{x_j}}$ 时，x_i 变量$\overline{x_j}$对称。

定义 6　1阶特征：布尔函数 f 关于变量 $x_i/\overline{x_i}$ 的1阶特征为 $(|f_{x_i}|,|f_{\overline{x_i}}|)$，$|f_{x_i}|/|f_{\overline{x_i}}|$ 是函数 $f_{x_i}/f_{\overline{x_i}}$ 的最小项的个数。

定义 7　1阶特征向量：具有n个输入的布尔函数 f 有n个变量，n个变量的1阶特征集合称为函数 f 的特征向量，记为 $\boldsymbol{V}^f=\{(|f_{x_1}|,|f_{\overline{x_1}}|),(|f_{x_2}|,|f_{\overline{x_2}}|),\cdots,(|f_{x_n}|,|f_{\overline{x_n}}|)\}$。

两个NP等价的布尔函数具有相同的1阶特征向量；若函数 f 与 g 是NP等价的，并且变量 x_i 和 x_j 之间有映射关系，x_i 和 x_j 一定具有相同的1阶特征[7]。

定义 8　变量映射集：布尔匹配中，布尔函数 f 的变量 x_i 的所有变量映射构成 x_i 的变量映射集，记为

$$\chi_i=\{\alpha\,|\,x_i\to x_{\pi(i)}^{\varphi(i)}\},\pi(i)\in\{1,2,\cdots,n\},\varphi(i)\in\{0,1\}$$

布尔函数 f 与 g 的匹配中，f 的变量 x_i 可能有零个或多个变量映射，$|\chi_i|$ 记为变量 x_i 的变量映射集的阶，即变量 x_i 具有的可能映射的个数。

因为NP变换不会改变变量的对称性，所以对称变量和非对称变量之间不存在映射。

定义 9　对称类：函数 f 有变量集合 $S_i=\{x_i,x_{i1},x_{i2},\cdots,x_{i(k-1)}\}$，$S_i$ 中所有变量相互对称，集合 S_i 是函数 f 的一个对称类。

若函数 f 有对称类 $S_i=\{x_i,x_{i1},x_{i2},\cdots,x_{i(k-1)}\}$，函数 g 有对称类 $S_j=\{x_j,x_{j1},x_{j2},\cdots,x_{j(k-1)}\}$。如果对称类 S_i 和 S_j 中的变量具有相同的1阶特征，那么对称类 S_i 和 S_j 之间存在映射 $\beta_i:S_i\to S_j$。对称类 S_i 中的任意一个变量可以映射到对称类 S_j 中的任一变量。若对称变量相位都是确定的，可在 S_i 和 S_j 之间产生 $k!$ 种映射关系。因为对称变量交换后函数的不变性，本算法只需检测集合 S_i 和 S_j 产生的一种变量映射关系。

香农引理：布尔函数 f 有等式 $f=x_if_{x_i}+\overline{x_i}f_{\overline{x_i}}$。

香农引理也称为香农分解，可以将n变量输入的布尔函数 f 分解为两个n–1变量输入的布尔函数。若函数 f 和函数 g 关于 $T=\{x_1\to x_{\pi(1)}^{\varphi(1)},x_2\to x_{\pi(2)}^{\varphi(2)},\cdots,x_n\to x_{\pi(n)}^{\varphi(n)}\}$ 是NP等价的，那么利用T中任意变量映射中的两个变量分解后得的两组函数一定也是NP等价的。假设T中有映射 $x_i\to x_j$，f 按 x_i 分解得到 $f=f_1+f_0=x_if_{x_i}+\overline{x_i}f_{\overline{x_i}}$，$g$ 按 x_j 分解得到 $g=g_1+g_0=x_jg_{x_j}+\overline{x_j}g_{\overline{x_j}}$，$f_1$ 与 g_1 是NP等价的，f_2 与 g_2 是NP等价的。这里的变量 x_i 和 x_j 被称为分解变量。

本文NPN布尔匹配的基本思想是：对于给定的两个待匹配的布尔函数 f 和 g，通过变量的1阶特征和香农分解搜索两个函数的变量之间的映射关系T，检测是否满足 $f(T\boldsymbol{X})=g(\boldsymbol{X})$ 或者 $f(T\boldsymbol{X})=\overline{g(\boldsymbol{X})}$。

2 提出的NPN布尔匹配算法

2.1 相位分配

给定两个布尔函数 f 和 g，NPN匹配需要确定对函数的输入非、输入置换和输出非：

1) 输出非：NPN等价的两个布尔函数具有相同或互补的最小项个数。若 $|f|=|g|$ 且 $|f|\neq|\overline{g}|$，f 无非运算，f 和 g 同为正相；若 $|f|=|\overline{g}|$ 且 $|f|\neq|g|$，f 有非运算，f 为正相g为反相；若 $|f|=|\overline{g}|$ 且 $|f|=|g|$，可能有输出非，可能没有输出非，f 为正相 g 的相位不定。

2) 输入非：若变量 x_i 和 x_j 之间有映射关系，并且它们是同相位的，x_i 无非运算；如果它们是相反相位的，x_i 有非运算。若 $|f_{x_i}|>|f_{\overline{x_i}}|$，$x_i$ 的相位为正；若 $|f_{x_i}|<|f_{\overline{x_i}}|$，$x_i$ 的相位为反；若 $|f_{x_i}|=|f_{\overline{x_i}}|$，$x_i$ 的相位不能确定。

3) 输入置换：若变量 x_i 和 x_j 具有相同的1阶特征，那么它们之间可能存在映射。

$$(|f_{x_i}|,|f_{\overline{x_i}}|)=(|g_{x_j}|,|g_{\overline{x_j}}|) \tag{1}$$

$$(|f_{x_i}|,|f_{\overline{x_i}}|)=(|g_{\overline{x_j}}|,|g_{x_j}|) \tag{2}$$

满足式(1)时，两个变量是同相；满足式(2)时，两个变量是反相；同时满足式(1)和式(2)，两个变量的相位不能确定，相位需要后续判定或同时检测同相和反相两种情况。

2.2 候选变换搜索

给定要匹配的布尔函数 f 和 g，本算法的目的是找到一个NP变换T，该变换能够将 f 转换为 g 或者 $\overline{g}$。对于任意n变量输入的布尔函数 f，有 $n!2^{n+1}$ 个NPN变换，但是能够将 f 转换为 $g/\overline{g}$ 的是少数。本算法通过对称和1阶特征向量搜索在 f 和 $g/\overline{g}$ 之间可能存在的NP变换，这些变换称为候选变换。每找到一个候选变换就验证该候选变换是否能将 f 转换为 $g/\overline{g}$。

每个候选变换由n个变量映射组成，本算法根据函数 f 和 g 的1阶特征向量及香农分解来搜索它们之间可能存在的NP变换。假设当前搜索函数 f 的变量 x_i 的映射，有以下几种情况：

1) 函数 g 中只有一个变量 x_j 与其具有相同的1阶特征，并且变量 x_i 的相位确定。那么 $|\chi_i|=1$，x_i 的映射唯一。

2) 函数 g 中只有一个变量 x_j 与其具有相同的1阶特征，并且变量 x_i 的相位不确定。那么 $|\chi_i|=2$，算法生成映射 $x_i\to x_j$ 和 $x_i\to\overline{x_j}$。算法先处理 $x_i\to x_j$，如果该映射所生成的NP变换不能将 f 转换为 $g/\overline{g}$，然后再处理 $x_i\to\overline{x_j}$。

3) 变量 x_i 是非对称变量且相位确定，g 有变量 $x_{j1},x_{j2},\cdots,x_{jk}$ 与其有相同的1阶特征。那么 $|\chi_i|=k$，本算法按顺序先生成 x_i 和 x_{j1} 之间的映射并处理由此映射所产生的NP变换。如果该NP变换不能将 f 转换为 $g/\overline{g}$，那么算法选取下一个即 x_i 和 x_{j2} 之间的映射处理，直到有一个NP变换可以将 f 转换为 $g/\overline{g}$ 或 χ_i 中所有的映射处理完毕。

4) 变量 x_i 是非对称变量并且相位不确定，g 中有变量 $x_{j1},x_{j2},\cdots,x_{jk}$ 与其有相同的1阶特征。那么 $|\chi_i|=2k$，算法检测映射集 $\{x_i\to x_{j1},x_i\to\overline{x_{j1}},x_i\to x_{j2},x_i\to\overline{x_{j2}},\cdots,x_i\to x_{jk},x_i\to\overline{x_{jk}}\}$，处理方式同步骤3)。

5) 变量 x_i 是对称变量且相位确定，所在对称类是 $S_i=\{x_i,x_{i1},x_{i2},\cdots,x_{i(k-1)}\}$，$g$ 中只有一个对称类 $S_j=\{x_j,x_{j1},x_{j2},\cdots,x_{j(k-1)}\}$ 与 S_i 具有相同的变量个数和相同的1阶特征。那么 $|\beta_i|=1$，算法只需检测由 S_i 和 S_j 之间产生的一个种映射关系即可。

6) 变量 x_i 是对称变量且相位不确定，所在对称类为 $S_i=\{x_i,x_{i1},x_{i2},\cdots,x_{i(k-1)}\}$，函数 g 中只有一个对称类 $S_j=\{x_j,x_{j1},x_{j2},\cdots,x_{j(k-1)}\}$ 与 S_i 具有相同的变量个数和相同的1阶特征。令 x_i 和 x_j 分别为同相和异相两种情况，在 S_i 与 S_j 之间可以产生2组不同映射关系，即 $|\beta_i|=2$。算法先处理同相，若同相所产生的NP变换不能将 f 转换为 $g/\overline{g}$，再处理异相。

7) 变量 x_i 是对称变量且相位确定，所在对称类为 $S_i=\{x_i,x_{i1},x_{i2},\cdots,x_{i(k-1)}\}$，$g$ 有对称类 $S_{j1},S_{j2},\cdots,S_{jk}$ 均与 S_i 具有相同的变量个数和相同的1阶特征。那么 $\beta_i=\{S_i\to S_{j1},S_i\to S_{j2},\cdots,S_i\to S_{jk}\}$，对称类 S_i 有k组对称映射关系，即 $|\beta_i|=k$，算法依次处理每一种对称映射关系，直到找到一个NP变换将 f 转换为 $g/\overline{g}$ 或所有的对称映射关系处理完毕。

8) 变量 x_i 是对称变量且相位不确定，所在对称类为 $S_i=\{x_i,x_{i1},x_{i2},\cdots,x_{i(k-1)}\}$，$g$ 有对称类 $S_{j1},S_{J2},\cdots,S_{jk}$ 均与 S_i 具有相同的变量个数和相同的1阶特征。那么 $\beta_i=\{S_i\to S_{j1},S_i\to S_{j2},\cdots,S_i\to S_{jk}\}$，由于这些对称类中变量的相位都是不确定的，算法在处理每个对称映射关系时需要考虑正相和反相两种情况，因此对称类 S_i 有$2k$组对称映射关系，即

$|\beta_i|=2k$。对这2k组对称映射关系的处理方式与情况7相同。

本算法采用树来存储每个变量可能的映射，树中的每个节点是一个变量的一个映射，树共有n层。若第k层只有一个节点，那么该层的变量只有一个映射；如果有多个节点，那么该层的变量有多个可能的映射。这颗树称之为候选变换树，该树的每个分支为一个候选NP变换。本算法采用深度优先搜索方式，只要找到一个满足条件的候选变换，算法就终止。用伪代码描述的候选变换搜索如下：

```
Procedure 1 Search Transformation
Input data_f, data_g, f, g, map_tree
Output 0 or 1
Function Search(data_f, data_g, f, g, map_tree)
Int min_num=32 768, min;
If D1
  Generate a candidate transformation
  Return Verify(f, g, map_tree)
  Else if compute_vector(f, g, data_f, data_g)=0 then
      Return 0
   Else
      For each x_i∈f(X)
        |χ_i/β_i|=num(x_i, f, g)
        If |χ_i|=1 then
          α_i→map_tree
        Else if |β_i|=1
          β_i→map_tree
        Else if |β_i|=k_1
            If (k_1<min_num)
                min_num= k_1
                min=i
            End if
        Else        // |χ_i|=k_2
            If(k_2<min_num)
                min_num= k_2
                min=i
            End if
        End if
    End for
    If |χ_i|=1 || |β_i|=1
      Update data_f, data_g
      Search(data_f, data_g, f, g, map_tree)
    Else
      For each α∈χ_min or β∈β_min
        α/β→map_tree
        Update data_f, data_g
        Search(data_f, data_g, f, g, map_tree)
      End for
    End if
    Return 0
End function
```

Procedure 1中的条件D1表示map_tree是否产生一个完整分支，每产生一个完整分支即生成一个候选变换T，Procedure 1调用verify()验证T。data_f和data_g是由分解变量构成的两个BDD表达式。data_f和data_g的初始值为常量bddtrue，函数Search()每调用一次需要更新data_f和data_g。若第k次调用时的分解变量分别是x_l和x_p，那么data_f=data_f$\wedge x_l$, data_g=data_g$\wedge x_p$。然后，Procedure 1调用compute_vector()更新变量的1阶特征并判断两个1阶特征向量是否相等，如果不相等则返回0。

2.3 匹配算法

NPN布尔匹配可描述如下：给定两个布尔函数$f(\boldsymbol{X})$和$g(\boldsymbol{X})$，是否存在一个NP变换T使得条件$f(T\boldsymbol{X})=g(\boldsymbol{X})$或$f(T\boldsymbol{X})=\overline{g(\boldsymbol{X})}$成立，如果存在这样的$T$，那么$f(\boldsymbol{X})$与$g(\boldsymbol{X})$NPN等价，否则，$f(\boldsymbol{X})$与$g(\boldsymbol{X})$不NPN等价。

匹配算法首先确定函数f和g的相位，如果相位能够确定，则计算f和$g/\overline{g}$的1阶特征向量，同时计算变量的相位。然后，比较两个1阶特征向量是否相等。如果不相等，则f和g不NPN等价；如果相等，则调用Search()函数。如果Search()返回1，则f和g是NPN等价的，否则，不NPN等价。

如果$|f|=|\overline{g}|$且$|f|=|g|$，则g的相位不确定。首先为函数g分配正相位，然后检测f和g是否NP等价，如果NP等价，那么f和g是NPN等价；如果f和g是不NP等价，就检测f和$\overline{g}$是否NP等价，如果等价，那么f和g是NPN等价，否则，f和g不NPN等价。

例2：给定布尔函数$f(\boldsymbol{X})=\overline{x_1x_2x_3}x_4+\overline{x_1}x_2x_3(\overline{x_4}x_5+x_4\overline{x_5})+\overline{x_1}x_2(\overline{x_3}x_4\overline{x_5}+x_3\overline{x_4})+x_1\overline{x_2x_3x_4}+x_1\overline{x_2}x_3(\overline{x_4x_5}+x_4x_5)+x_1x_2(\overline{x_3x_4}+\overline{x_3}x_4x_5+x_3x_4)$，$g(\boldsymbol{X})=\overline{x_1}x_2(\overline{x_3}x_5+x_3\overline{x_5})+\overline{x_1}x_2(\overline{x_3}x_5+x_3\overline{x_4}x_5+x_3x_4\overline{x_5})+x_1\overline{x_2}(\overline{x_3x_4x_5}+\overline{x_3}x_4x_5+x_3\overline{x_4}x_5+x_3x_4\overline{x_5})+x_1x_2(\overline{x_3x_5}+x_3x_5)$，检测$f$和$g$是否NPN等价，其过

程如下：

1) data_f=bddtrue，data_g=bddtrue，建立 f 和 g 的BDD，计算1阶特征向量，结果为$\boldsymbol{V}^f$={(9,7), (8,8), (8,8), (8,8), (8,8)}，$\boldsymbol{V}^g$={(8,8), (8,8), (8,8), (8,8), (9,7)}。经过对称检测得到函数 f 有对称类 $S_2=\{x_2,x_5\}$，函数 g 有对称类 $S_2=\{x_2,x_4\}$。根据1阶特征向量可知 f 的变量 x_1 相位为正，g 的变量 x_5 相位为正，其他变量相位不能确定。计算 f 中每个变量的变量映射集，可得 $\chi_1=\{x_1\to x_5\}$ 是第一个要处理的映射集，data_f更新为 x_1 ,data_g更新为 x_5。

2) 更新1阶特征向量，结果为：$\boldsymbol{V}^f$={(0,0), (5,4), (4,5), (4,5), (5,4)}，$\boldsymbol{V}^g$={(4,5), (5,4), (4,5), (4,5), (0,0)}，已经确定了的映射关系中的变量的1阶特征更新为(0,0)。从该结果可以看出，所有变量的相位已经确定，并且可以得到 $\chi_2=\{S_2\to S_2\}$ 是当前要处理的映射集。算法搜索到一种对称映射关系 $\beta_2=\{x_2\to x_2,x_5\to\overline{x_4}\}$。更新data_f为 x_1x_2 、data_g为 x_5x_2。

3) 更新1阶特征向量，结果为$\boldsymbol{V}^f$={(0,0), (0,0), (2,3), (3,2), (0,0)},$\boldsymbol{V}^g$={(2,3), (0,0), (3,2), (0,0), (0,0)}。搜索到要处理的映射集为 $\chi_3=\{\overline{x_3}\to\overline{x_1},\overline{x_3}\to x_3\}$。算法选择第一个变量映射处理并更新data_f和data_g，data_f= $x_1x_2\overline{x_3}$ 并且data_f= $x_5x_2\overline{x_1}$。

4) 更新1阶特征向量，结果为$\boldsymbol{V}^f$={(0,0), (0,0), (0,0), (1,2), (0,0)}, $\boldsymbol{V}^g$={(0,0), (0,0), (1,2), (0,0), (0,0)}。搜索到变量映射集 $\chi_4=\{\overline{x_4}\to\overline{x_3}\}$。至此，找到一个候选变换 $T=\{x_1\to x_5,x_2\to x_2,x_5\to\overline{x_4},\overline{x_3}\to\overline{x_1},\overline{x_4}\to\overline{x_3}\}$，通过验证 $f(T\boldsymbol{X})=g(\boldsymbol{X})$，所以 f 和 g 是NPN等价的。

例2的候选变换树如图1所示。

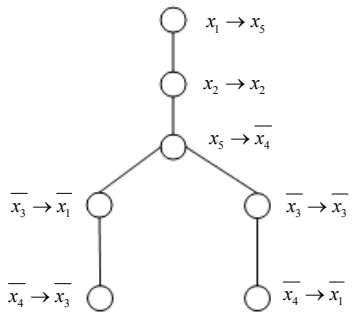

图1 例1的候选变换树

5变量输入布尔函数有NP变换 $5!2^5$ 个，例1中的候选变换树总共可能的NP变换为2个，匹配中验证第一个NP变换后算法就终止了。如果采用文献[7]中的算法对例1的两个函数进行匹配，在上述的第2步中要处理的变量映射集是 $\chi_2=\{x_2\to x_2,x_2\to\overline{x_3},x_2\to\overline{x_4}\}$，那么其候选变换树中的第二层将会产生4个分支。从例1可以看出，本算法相对文献[7]的搜索空间减小了很多，其主要原因是对称变量的使用。

3 实验结果

使用本算法在大量的电路函数中进行测试，并与文献[7]的结果进行对比。测试中检测了等价和非等价电路函数的匹配速度，测试电路包括随机生成的电路函数和部分MCNC标准电路库中的函数。实验设备的配置为3.3 GHz CPU、4 GB RAM。

表1～表3列出了对等价和非等价电路匹配测试的结果，包括匹配的最短时间、最长时间和平均时间，运行时间以秒为单位。表1中列出了对MCNC标准电路库的等价电路函数的匹配运行结果，表2中列出了对随机产生的等价电路函数的匹配运行结果。

表1 等价的MCNC标准电路函数匹配测试结果 s

变量个数	最短时间	最长时间	平均时间	文献[7]平均时间
7	0.000 03	0.000 44	0.000 13	0.000 62
8	0.000 24	0.000 64	0.000 40	0.001 03
9	0.000 05	0.001 12	0.000 59	0.003 91
10	0.000 10	0.003 34	0.000 63	0.010 17
11	0.000 10	0.002 18	0.000 84	0.024 12
12	0.000 09	0.006 73	0.001 47	0.029 52
13	0.000 34	0.021 35	0.003 94	0.032 87
14	0.000 40	0.018 36	0.007 61	0.052 19
15	0.000 27	0.042 48	0.012 81	0.104 51
16	0.000 54	0.062 11	0.031 34	0.381 33
17	0.000 91	0.629 30	0.216 52	1.541 29
18	0.001 75	1.869 12	0.338 04	3.125 62
19	0.003 51	3.229 40	1.836 66	8.842 31
20	0.006 95	5.472 99	2.088 19	14.431 70

表2 等价随机电路函数匹配测试结果 s

变量个数	最短时间	最长时间	平均时间	文献[7]平均时间
7	0.000 01	0.001 08	0.000 19	0.000 73
8	0.000 07	0.001 31	0.000 35	0.001 83
9	0.000 12	0.002 32	0.000 57	0.004 52
10	0.000 18	0.002 14	0.000 94	0.009 13
11	0.000 03	0.003 76	0.001 87	0.025 37
12	0.003 01	0.009 11	0.004 41	0.030 14
13	0.000 96	0.014 37	0.008 72	0.047 81
14	0.015 75	0.027 12	0.019 59	0.077 16
15	0.034 87	0.086 55	0.044 41	0.196 23
16	0.083 65	0.105 83	0.095 57	0.402 23
17	0.194 27	0.247 79	0.217 98	1.884 26
18	0.457 87	0.533 29	0.504 68	3.971 53
19	1.081 47	1.299 24	1.179 28	8.119 62
20	2.452 43	5.568 71	2.787 01	15.781 41

从等价的测试中可以看出，就平均匹配速度而言，在MCNC等价电路匹配测试中，本算法相比文献[7]的算法提高了86%；在随机等价电路匹配中相

比文献[7]的算法提高了83%。本算法对非等价电路函数匹配结果如表3所示。

表3　非等价电路函数匹配测试结果　　s

变量个数	最短时间	最长时间	平均时间	文献[7]平均时间
7	0.000 007	0.000 086	0.000 033	0.000 093
8	0.000 007	0.000 117	0.000 051	0.000 151
9	0.000 008	0.000 209	0.000 076	0.000 216
10	0.000 012	0.000 339	0.000 110	0.000 310
11	0.000 018	0.000 645	0.000 187	0.000 719
12	0.000 039	0.000 711	0.000 293	0.000 929
13	0.000 111	0.002 481	0.000 501	0.002 501
14	0.000 158	0.001 935	0.000 647	0.004 647
15	0.000 495	0.004 565	0.000 849	0.006 835
16	0.000 294	0.007 301	0.000 934	0.009 934
17	0.000 268	0.011 201	0.001 626	0.014 260
18	0.001 481	0.023 069	0.002 835	0.032 817
19	0.001 248	0.045 970	0.004 034	0.044 034
20	0.002 841	0.082 492	0.005 202	0.952 021

从非等价电路匹配测试结果中可以看出，本算法相对文献[7]中的算法，平均匹配速度提高了79%。

整体来说，本算法实现了快速的NPN布尔匹配，是一种非常有效的NPN布尔匹配算法。

4　结 束 语

本文提出了一种基于对称及特征的NPN布尔匹配算法，通过1阶特征向量和香农分解建立两个布尔函数之间的变量映射，采用深度优先搜索查找使得一个布尔函数转换为另外一个布尔函数的NP变换。利用变量的对称特性进一步减少了候选NP变换的搜索空间，从而提高了NPN布尔匹配的速度。通过实验证明了本算法的有效性，能够应用于电路设计的工艺映射中。未来的研究方向是将本算法扩展到多输出布尔函数及含有无关项的布尔函数的NPN等价匹配中。

参 考 文 献

[1] KAPOOR B. Improved technology mapping using a new approach to Boolean matching[C]//Proceedings the European Design and Test Conference European Design and Test Conference. Paris : IEEE Press, 1995, 3: 86-90.

[2] MAILHOT F, MICHELI G DE. Technology mapping using Boolean matching and don't care sets[C]//Proceedings of the European Design Automation Conference. Glasgow: IEEE Press, 1990, 3: 212-216.

[3] DAMIANI M, SELCHENKO A Y. Boolean technology mapping based on logic decomposition[C]//The 16th Symposium on Integrated Circuits and Systems Design. Sao Paulo: IEEE Press , 2003, 9: 35-40.

[4] CHEN K C, YANG C Y. Boolean matching algorithms [C]//International Symposium on VLSI Technology, Systems, and Applications. Taipei: China, IEEE Press , 1993, 5: 44-48.

[5] ABDOLLAHI A. Signature based Boolean matching in the presence of don't cares[C]//The 45th ACM/IEEE Conference on Design Automation Conference. Anaheim, USA: IEEE Press, 2008, 6: 642-647.

[6] LAI Y T, SASTR S, PEDRAM M. Boolean matching using binary decision diagrams with applications to logic synthesis and verification[C]//IEEE International Conference on Computer Design: VLSI in Computers and Processors. Cambridge, USA : IEEE Press, 1992, 10: 452-458.

[7] ZHANG J, YANG G, WU J, et al. Cofactor-based NPN Boolean matching algorithm[C]//International Conference on Information Management. Chengdu: IEEE Press, 2017.

[8] ADBOLLAHI A, PEDRAM M. Symmetry detection and Boolean matching utilizing a signature-based canonical form of Boolean functions[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(6): 1128-1137.

[9] DEBNATH D, SASAO T. Efficient computation of canonical form for Boolean matching in large libraries[C]// Asia and South Pacific Design Automation Conference. Yohohama: IEEE Computer Society, 2004.

[10] AGOSTA G, BRUSCHI F, PELOSI G, et al. A transform-parametric approach to Boolean matching[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2009, 28(6): 805-817.

[11] CHAI D, KUEHLMANN A. Building a better Boolean matcher and symmetry detector[C]//International Conference on Design, Automation and Test in Europe. Munich: IEEE Press, 2006.

[12] HUANG Z, WANG L, NASIKOVSKIY Y. Fast Boolean matching based on NPN classification[C]//International Conference on Field-Programmable Technology. Kyoto: IEEE Press, 2013.

[13] PETKOVSKA A, SOEKEN M, MICHELI G D, et al. Fast hierarchical NPN classification[C]//International Conference on Field Programmable Logic and Applications. Lausanne: IEEE Press, 2016.

[14] MATSUNAGA Y. Accelerating SAT-based Boolean matching for heterogeneous FPGAs using one-hot encoding and cegar technique[C]//The 20th Asia and South Pacific Conference on Design Automation. Chiba: IEEE Press, 2015.

[15] KATEBI K, MARKOV L. Large-scale Boolean matching [C]//Advanced Techniques in Logic Synthesis Optimizations & Applications. Dresden: IEEE Press, 2010.

编　辑　刘飞阳