第14卷 第4期 电子科学学刊 Vol. 14, No. 4
1992 年 7 月 JOURNAL OF ELECTRONICS July, 1992

多输出逻辑函数最小覆盖算法原理的改进

陈 苏 许道荣
(清华大学,北京 100084)

摘要 本文对求逻辑函数最小覆盖的析取 (extraction) 法在多输出情况下的应用进行了研究,弥补了原有算法在多输出极值项处理上的欠缺,并在此基础上定义了 3 种不同情况下的劣势项. 对劣势项的定义做了 3 种推广,可以更有效地求出项数最少输出无冗余的覆盖.给出了 MOSE 算法.

关键词 多输出函数最小化;可编程逻辑阵列化简;最小"与或"表达式

一、引 言

求多输出逻辑函数的最小覆盖是逻辑综合中的一个基本问题. 随着可编程逻辑阵列 (PLA) 的应用,对逻辑化简又有了更多的研究. 由于逻辑化简的复杂性, 大部分算法都采用启发性的规则,寻求无冗余或近似最优的结果. 例如文献 [1～4] 等的算法. 对算法的评价,最好的办法是将结果与最简结果进行比较.但是因为目前尚无求最小覆盖的较好的算法,所以只能在算法之间进行相对比较. 对于启发式算法,同一问题要用几种方法计算,以取得较好的结果. 另外,随着计算机内存的增加,速度的提高和价格的下降,过去受这些因素限制而不宜使用的最简算法,现在对于 20 个甚至更多个变量的中规模问题是可行的了[5]. 而函数的简化还直接对应着 PLA 中项数和点数的减少. 这对减少测试向量,改进折叠效果,从而对减小芯片面积,提高速度都有直接作用. 因此, 对求最小覆盖的研究一直具有重要意义. 这方面的算法主要有文献 [5～7] 等等.

求最小覆盖的原理有两种,都以质蕴含项集合为基础. 一种是 Petrick 函数方法[8]. 通过展开定义在覆盖表上的覆盖函数来找出最小覆盖, 如文献 [7,9] 等. 另一种方法采用析取 (Extraction) 原理,如文献 [5,6,10] 等. 前者原理简单,但计算量很大, 应用于多输出情况还缺少较有效的方法. 后者的原理是在质蕴含项中反复进行极值项提取, 直到函数被覆盖[11]. 文献 [6] 把析取 (Extraction) 法推广到多输出函数. 文献 [5] 在这一原理基础上采用新的数据结构方法, 大大提高了算法的速度. 基于这两种原理的上述算法,除文献 [6] 之外,都是只求项数最小的覆盖, 输出连接 (PLA 或阵列中的点)只要

1991. 06. 13 收到, 1992. 03. 03 定稿.

求无冗余.

本文将研究析取 (extraction) 原理在多输出函数中应用的问题,目的是求出项数最少,同时连接数也最少的最小覆盖.

二、符号和基本概念

函数 $F=\{f_i(x_1, \cdots, x_m)\mid i=1, \cdots, n\}$ 可以表示为映射 $F:\{(0, 1, x)^m\}\to\{(0, 1, d)^n\}$. 在算法中多用输出立方体集合 $F=\{c\}$ 来表示,其中 $c=c_\alpha\circ c_\Omega$, $c_\alpha=c_{\alpha 1}\cdots c_{\alpha m}$ 称为输入或 α 立方体, $c_\Omega=c_{\Omega 1}\cdots c_{\Omega n}$ 称为输出或 Ω 立方体.

$D(c_\alpha)$ 或 $D_\alpha(c)$ 表示 c_α 中 "x" 的个数; $D(c_\Omega)$ 或 $D_\Omega(c)$ 表示 c_Ω 中 "1" 的个数. $D_\alpha()=0$ 的立方体称为顶点; $D_\alpha()>0$ 的立方体看作是顶点的集合.

定义 1 设 M 是实现 F 的立方体集合. 如果 M 按顺序满足下列条件:

(1) $|M|=\min$

(2) $\sum\limits_{c\in M}(m-D_\alpha(c))=\min$

(3) $\sum\limits_{c\in M}D_\Omega(c)=\min$

则称 M 是 F 的最小覆盖,记作 $M(F)$.

这 3 个条件描述的是: 实现函数的总项数, 各项中的因子之和以及每个函数的项数都最少,即对应着最简的"与或"表达式.

关于多输出情况下的极值项和劣势项的定义请见文献 [6]. 当立方体 π 是 π' 的劣势项时,用 $\pi<\pi'$ 表示. 用 "#" 表示立方体的锐积,其定义请见文献 [6]. MOPI(F) 表示 F 的多输出质蕴含项集合.

三、求最小覆盖的算法

求 $M(F)$ 的根据是

定理 1[6] 若 $\pi\in M(F)$, 则 $\pi\subseteq\pi'\in\mathrm{MOPI}(F)$.

如果用 EXT() 和 DOM() 分别表示提取极值项和删除劣势项的过程,并且不考虑循环发生时,文献 [6] 中的多输出析取 (extraction) 过程可以描述为

```
{C = ON(F); Z = MOPI(F); M' = φ;
 while (C != φ)
   {E = EXT( );                    /*step1*/
    E → H ∪ G;                     /*step2*/
    M' = M' ∪ H;                   /*step3*/
    C = C # H;                     /*step4*/
    Z = Z ⊢ G;                     /*step5*/
    DOM( );                        /*step6*/
```

}

$M(F) = M'$;

}

其中，$E = \{e = e_\alpha \circ e_\Omega\}$ 是产生的极值项．一般地，e_α 不是 e_Ω 中所有函数的极值项．令 $e_\Omega = h_\Omega + g_\Omega$，使 e_α 是 h_Ω 中而不是 g_Ω 中函数的极值项 (step2)．$H = \{h = e_\alpha \circ h_\Omega\}$ 在这里称为真极值项，将 $G = \{g = e_\alpha \circ g_\Omega\}$ 称为假极值项．对于孤立的一次提取极值项过程，有

定理 2[6]　$h \subseteq \pi'$，$\pi' \in M(F)$

因而，算法认为有

定理 3[6]　$M(F) = \bigcup_k H_k$，$k = 1$，$\cdots$

H_k 为各次提取出的极值项集合．

算法将 g 送回 Z(step5)，如果以后再次成为极值项，则可以与 M' 中对应的 h 合并，而不增加 M' 中的项数；同时，覆盖每个函数的项都是极值项，保证了每个函数的项也最少．这里称送回 Z 中的 g 为已选项，其余未被选过的为未选项；C 中由已选项所覆盖的点称为准覆盖点，其余的为未覆盖点．但是可以发现，被送回的 g 在以后的 DOM() 过程中，可能成为某个未选项的劣势项而被删去，使所覆盖的准覆盖点又成为未覆盖点．为了覆盖这些点又要引入新的项，使 M' 中项数增加，使结果不一定最简．由此可见，算法未能使定理 3 成立．

在 McBoole 算法中，不对 E 进行分解，求出的 $M(F)$ 只满足定义 1 中的条件 (1)．

四、MOSE 算法

1. 对多输出析取 (Extraction) 算法的改进

为了避免已选项 g 成为未选项的劣势项，必须对其加以区别．这里利用 $D_\alpha()$ 函数来标志已选项．当 g 送回 Z 时，令 $D_\alpha(g) = \text{const} > m$，使其不会成为任何未选项的劣势项．这种方法将 Z 划分成已选项 Z_p 和未选项 Z_q 两个集合；对于 $\pi \in Z_p$，$D_\alpha(\pi) > m$；而对于 $\pi \in Z_q$，$D_\alpha(\pi) < m$．

不过，改进后由于 Z_p 的保留，使循环的可能性大为增加．对此，利用 Z_p 和准覆盖点的概念，可以将劣势项根据不同情况重新定义，尽量找出 Z 中项对 C 构成冗余覆盖的充分条件，并删除冗余覆盖，使循环的可能性增加很少或不增加．

2. 劣势项的重新定义

下面用 DM_0 表示原来的劣势项，用 DM_i 表示将要定义的第 i 种劣势项．与 Z 的划分对应，C 也划分为准覆盖点集合 C_p 和未覆盖点集合 C_q．

定义 2　DM_1：若 $\pi \in Z_q$，并且 $\pi \cap C_q = \phi$，则称 π 为第 1 种劣势项．

定义 3　DM_2：若 $\pi \in Z_p$，且 $(\pi \cap C_p) \subseteq \pi' \in (Z_p - \pi)$，则称 π 为第 2 种劣势项．

定义 4　DM_3：若 $\pi \in Z_q$，$A = \pi \cap C_q$，如果 $A \subseteq \pi' \in (Z_q - \pi)$，且 $D_\alpha(\pi') \geqslant D_\alpha(\pi)$，则称 π 为第 3 种劣势项．

定理 4 删除 $DM_{1\sim3}$，不影响对 $M(F)$ 的提取.

证明 如果 π 属于 DM_1，从定义可以看出，π 中不含未覆盖点，肯定不会成为极值项被提取. 如果 π 属于 DM_2，只含准覆盖点，而这些点被另一已选项 π' 包含，肯定不会成为极值项. 如果 π 属于 DM_3，它所含的未覆盖点均被另一 $\pi' \in Z_q$ 包含，且维数小于 π'，也没有成为极值项的可能. 因此，$DM_{1\sim3}$ 的存在已成为冗余，删除它们不会影响求最小覆盖 $M(F)$.

从定义 2～4 可看出，DM_3 定义在未选项之间，DM_2 定义在已选项之间，而 DM_1 定义的是成为已选项集合的劣势项的未选项.

3. 劣势项定义的推广

如果将定义 1 中的条件 (3) 放宽到接近最小的无冗余覆盖，那么，劣势项的定义还可以推广. 这里用 ED_i 表示第 i 种推广定义的劣势项. 设 $A_\Omega = \bigcup_i a_{i\Omega}$，$a_i \in A$，$A$ 是立方体阵列.

定义 5 ED_1：若 $\pi \in Z_q$，π 不属于 DM_3，但是 $A_\Omega \fallingdotseq \pi_\Omega$，则称 $\pi_\alpha \circ (\pi_\Omega - A_\Omega)$ 为第 1 种广义劣势项.

定义 6 ED_2：若 $\pi \in Z_p$，π 不属于 DM_2，但是 $A - (\pi \cap C_p) \# (Z_p - \pi) = \phi$，则称 π 为第 2 种广义劣势项.

定义 7 ED_3：若 $\pi \in Z_p$，π 不属于 ED_2，但是 $A_\Omega \fallingdotseq \pi_\Omega$，则称 $\pi_\alpha \circ (\pi_\Omega - A_\Omega)$ 为第 3 种广义劣势项.

定理 5 删除 $ED_{1\sim3}$，不影响 $M(F)$ 中的项数.

证明 如果某项 π 被选入 $M(F)$，一定是由于其所含的未覆盖点成为极值点. 如 π 属于 ED_1，则在 $\pi_\Omega - A_\Omega$ 的那些函数上已不含未覆盖点，因而，$\pi_\alpha \circ (\pi_\Omega - A_\Omega)$ 已成为冗余. 删除后，影响的只是 π_α 在 $\pi_\Omega - A_\Omega$ 那些函数上所覆盖的准覆盖点将由哪些项来覆盖. 但所涉及的项都是已选项，因此对结果中的总项数没有影响. 至于 ED_2 和 ED_3，它们都是已选项，根据推广的定义，删去后，所覆盖的点仍是准覆盖点，不会成为未选项的极值点，而使未选项被选中，引起项数增加.

下面是删除 $DM_{1\sim3}$ 和 $ED_{1\sim3}$ 的过程.

```
DOMIN( )
 {for(i = 1; i<=|Z|;i++)
   if(πi ∈ Zq)
     {if((A - πi∩Cq)==φ)Z = Z - πi;      /*πi 是 DM1*/
       else if(A⊆π' ∈ (Zq - πi))πiΩ = πiΩ - AΩ;      /*πi 是 ED1*/
            else if(Dα(π') ≥ Dα(πi))Z = Z - πi;      /*πi 是 DM3*/
     }
     else
       if((A = π∩Cp)⊆π' ∈ (Zp - π))Z = Z - πi;      /*πi 是 DM2*/
     for(i = 1;i<=|Zp|;i++)
       if((A = πi∩Cp)⊆(Zp - πi))Z = Z - πi;      /*πi 是 ED2*/
```

```
        else π_iΩ = π_iΩ − A_Ω;                    /*π_i 是 ED_3*/
  }
```

4. MOSE 算法

```
MOSE( )
  {C = ON(F); Z = MOPI(F); M' = φ;
    while(C != φ)
      {E = EXT( );                                  /*step1*/
       if(E==φ){BRANCH( ); Continue;}               /*step1'*/
       E → H∪G;                                     /*step2*/
       M' = M'∪H;                                   /*step3*/
       C = C#H;                                     /*step4*/
       Z_p = Z_p + G;                               /*step5*/
       D = DOMIN( );                                /*step6*/
       if(D==φ){BRANCH();continue;}                 /*step6'*/
      }
    M(F) = M';
  }
```

其中，BRAHCN( ) 是分支过程，将在后面说明. EXT( ) 可以不变. D 是被删去的劣势项. step1～6 与前面的析取（Extraction）算法中的 step1～6 分别对应. 这里的 step5 显式说明将 G 送入 Z_p，意味着 G 与 Z 中其他项相区别.

5. 循环和分支方法

析取（Extraction）法的原理，就是不断删去质蕴含项对函数顶点的冗余覆盖，反复提取极值项. 劣势项是一种冗余覆盖，不断将其删去以产生极值项，使算法进行下去. 但是，有时会进入一种既无劣势项可删，又无极值项可提取的情况，即循环状态. 这表明劣势项只是冗余覆盖的充分条件，而不是必要条件，还存在其它形式的冗余覆盖. 在无法进一步确定可删除项的充分条件时，对循环只有采取分支处理.

绝对寻优的方法是进行分支限界搜索. 当 Z 较大时，要花费大量时间，甚至无法完成. 通常采用的是引入启发性策略挑选或删除一项的方法来打破循环，这里称为简单分支方法.

在 MOSE 中采用如下策略的简单分支方法.

分路策略　删去 Z_q 中含未覆盖点最少的项中 $D(\pi_\Omega)$ 最大的一项 π；如果 $Z_q=\phi$，则删去 Z_p 中含准覆盖点最少的项中 $D(\pi_\Omega)$最大的一项 π.

分支要尽量不影响最终结果的最优性，同时尽量有效地打破循环，首先考虑前者.

五、讨　论

1. 算法的实现

MOSE 算法已用 C 语言实现，程序约 32K，可处理 $m=16$，$n=16$ 的函数，必要

时，可将 m 和 n 扩大到 32。算法实现中，为了减少计算量，在删劣势项过程中，仅在与刚选出的极值项相交的项中寻找劣势项；而提取极值时，仅在与刚删除的劣势项相交的项中寻找。这与文献 [5] 中的方法类似。

2. 存在的问题和改进

质蕴含项的数目是影响化简效率的根本原因。对于需要先生成 MOPI(F) 的求最小覆盖的算法更是这样。因此，要使 MOSE 算法能适用于更大规模的函数，就要设法少生成不属于 $M(F)$ 中的项。

分支是影响求最小覆盖的因素。McBoole 算法将 Z 划分为不相交子集后再分别进行分支，减少了分支深度。但它需要划分存在，其次对各个子集仍有一个分支策略的问题。

3. 算法的效果

与启发性算法不同，求最小覆盖的算法的正确性是由算法每一步骤的正确性保证的。但是由于分支过程引入了启发性因素，使这种保证受到一些影响。表 1 中给出 4 个例子的数据。其中 McB. 是 McBoole 的等效算法，但由于实现技术不同和分支方法的差别，表中的运行时间和结果仅供参考。

六、结 论

本文在对多输出析取 (Extraction) 算法中已选项与未选项加以区别的基础上，建立了准覆盖点的概念，并根据不同情况重新定义了劣势项，使得求最小覆盖的基本原理推广到多输出情况后得到了相应的完善。此外还提出了 3 种广义劣势项，将其删除，可以得到输出无冗余的最小覆盖。在这些改进的基础上，形成了求多输出逻辑函数最简两级表达式的 MOSE 算法，并用 C 语言实现，达到了预期的结果。

表 1

例子	输入/输出	算法	结果			分支次数	时间 (s)
			项数	"与"阵列点数	"或"阵列点数		
1	8/8	MOSE	71	426	201	12	15.83
		McB.	71	439	250	0	12.46
2	12/15	MOSE	61	325	164	18	185.83
		McB.	61	336	217	16	51.60
3	4/3	MOSE	6	14	11	0	手工
		Extraction	8	19	12	0	计算
4	3/7	MOSE	4	12	21	0	手工
		Extraction	6	15	13	0	计算

参 考 文 献

[1] R. K. Brayton et al., Logic Minimization Algorihms for VLSI Synthesis, Kluwer Academic Publishers, Boston, (1984), Char. 4.

[2] S. J. Hong, P. G. Cain, D. L. Ostapko, *IBM J. Res. Develop.*, 18(1974)5, 443—458.

[3] P. Agrawal, V. D. Agrawal, N. N. Biswas, Multiple Output Minimization, 22nd Design Automation Con-

ference, Las Vegas. Nevada, (1985), 674—680.
[4] D. W. Brown, A State-Machine Synthesizer-SMS, 18th Design Automation Conference, Nashville, Tenn. (1981), 301—305.
[5] M. R. Dagenais, V. K. Agarwal, N. C Rumin, *IEEE Trans. on CAD*, CAD-5(1986)1, 229—238.
[6] J. P. Roth, Computer Logic, Testing and Verification, Computer Science Press, Inc., Potomac, Maryland, (1980), 1—32.
[7] S. J. Hong, S. Muroga, *IEEE Trans. on C*, C-40(1991)1, 53—65.
[8] S. R. Petrick, A Direct Determination of the Irredundant Forms of a Boolean Function From the Set of Prime Implicants, Air Force Cambridge Res. Center, Betford, MA, Tech, Rep. No. AFCRC-TR-56-110, (1956).
[9] 刘明业，李建一，计算机学报，6(1983)2，136—145.
[10] J. P. Roth, Algebraic topological methods in synthesis, Annals of Computation Lab. of Harvard Univ Vol. 29, (1959), 53—73.
[11] 刘明业，计算机辅助逻辑设计理论，第一版，科学出版社，北京，1985年，第三章.

IMPROVEMENT OF THE ALGORITHM PRINCIPLE FOR MINIMIZATION OF MULTI-OUTPUT FUNCTIONS

Chen Su　Xu Daorong

(Tsinghua University, Beijing 100084)

Abstract　The application of extraction principle for logic function minimization to multiple-output cases is studied. The defect in original algorithm in dealing with multi-output extremals is complemented, and three kinds of less-than terms in different conditions are defined. In addition, three kinds of generations for the definition of less-than terms are given, so as to more efficiently find out the coverage with minimal number of terms and irredundant function outputs. Finally, an algorithm MOSE based on the work is presented.

Key words　Multi-output function minimization; Reduction of programmable logic array (PLA); Minimal "AND-OR" expression