

一种改进的混合极性列表技术

李新文*, 夏银水*, 王伦耀

(宁波大学 电路与系统研究所, 浙江 宁波 315211)

摘要: 针对基于多输出混合极性 Reed-Muller 逻辑表达式极性转换问题, 提出了基于不相交项的列表极性转换方法, 并通过极性搜索实现函数的最小化. 结果表明: 与传统列表极性转换法相比, 提出的新方法能有效降低时间复杂度.

关键词: 列表法; 混合极性 RM 逻辑; 极性转换; 不相交项

中图分类号: TP391

文献标志码: A

文章编号: 1001-5132 (2015) 01-0042-05

目前的数字系统往往是基于传统布尔逻辑来实现, 事实上逻辑函数也可以用 XOR/AND 为基础的 Reed-Muller(RM)逻辑来实现^[1], 而且与 Boolean 逻辑实现的电路相比, 用 RM 逻辑实现的电路在面积和可测试性等方面有一定的优势^[2-3]. RM 逻辑电路的发展主要受到工艺方面的制约^[4], 而随着工艺的进步, FPGA 技术的出现为 RM 电路提供了有效的硬件实现平台^[5]. 因此, RM 逻辑电路综合与优化技术的研究显得更加重要.

RM 逻辑函数按极性可分为固定极性(FPRM)和混合极性(MPRM)两种形式. 对于 n 输入变量的逻辑函数分别有 3^n 个混合极性和 2^n 个固定极性, 从而对应有 3^n 个混合极性表达式和 2^n 个固定极性表达式, 且所有固定极性表达式都包含在混合极性表达式之中. 因此, 优化 MPRM 逻辑函数比优化 FPRM 逻辑函数有更大的空间, 但相应的优化过程也更复杂. 基于极性变换的优化是 RM 逻辑函数优化的常用方法之一, 需要有快速的 RM 逻辑函数的极性转换技术的支持.

笔者针对最小项 MPRM 逻辑表达式列表转换技术的缺点^[6-7], 提出基于不相交项^[8-9]的列表极性转换方法, 且所提出的新方法避免了将原始逻辑函数转换成最小项, 从而大大减少了极性转换过程中的乘积项个数. 且以逻辑表达式乘积项个数为成本函数, 通过极性搜索得到给定函数的最佳

极性, 实现逻辑函数的优化, 最后通过实验验证算法的有效性.

1 理论基础

任意 n 个输入变量和 m 个输出变量的逻辑函数 $\{f_k : B^n \rightarrow B \mid 0 \leq k \leq m-1\}$ 可用最小项表达式表示为:

$$f_k(x_{n-1}, x_{n-2}, \dots, x_0) = \sum_{i=0}^{2^n-1} a_{i,k} m_i, \quad (1)$$

其中, $\sum$ 为“或”运算; f_k 为第 k 个输出变量, $k=0, 1, \dots, m-1$; 下标 i 可用二进制表示为 $i=(i_{n-1}i_{n-2}\dots i_0)$; $a_{i,k}$ 为最小项系数, 取值 1 或 0, 表示 m_i 是否在第 k 个最小项表示式中出现; 最小项 m_i 可用符号表示为 $m_i = \dot{x}_{n-1}\dot{x}_{n-2}\dots\dot{x}_0$, 其中,

$$\dot{x}_j = \begin{cases} \bar{x}_j, & i_j = 0, \\ x_j, & i_j = 1, \end{cases} \quad 0 \leq j \leq n-1. \quad (2)$$

运用 Shannon 公式^[10], 1 个 n 变量 RM 逻辑函数 $f(x_{n-1}, x_{n-2}, \dots, x_i, \dots, x_0)$ 可对任意变量 x_i 展开为:

$$f(x_{n-1}, x_{n-2}, \dots, x_i, \dots, x_0) = \bar{x}_i f(x_{n-1}, x_{n-2}, \dots, 0, \dots, x_0) \oplus x_i f(x_{n-1}, x_{n-2}, \dots, 1, \dots, x_0), \quad (3)$$

代入 $\bar{x}_i = 1 \oplus x_i$ 与 $x_i = 1 \oplus \bar{x}_i$ 中, 可得到另外 2 种 RM 表达式:

$$f(x_{n-1}, x_{n-2}, \dots, x_i, \dots, x_0) = f(x_{n-1}, x_{n-2}, \dots, 0, \dots, x_0) \oplus x_i \{f(x_{n-1}, x_{n-2}, \dots, 0, \dots, x_0) \oplus$$

收稿日期: 2013-04-06.

宁波大学学报(理工版)网址: <http://journal.nbu.edu.cn/>

基金项目: 国家自然科学基金重点项目(61131001); 浙江省自然科学基金重点项目(Z1090622); 教育部博士点基金(20113305110001).

第一作者: 李新文(1989-), 男, 内蒙古赤峰人, 在读硕士研究生, 主要研究方向: 集成电路设计. E-mail: 663098673@163.com

*通信作者: 夏银水(1963-), 男, 浙江余姚人, 博士/研究员, 主要研究方向: 高信息密度集成电路设计. E-mail: xiayinshui@nbu.edu.cn

$$\begin{aligned}
& f(x_{n-1}, x_{n-2}, \dots, 1, \dots, x_0) \}, \\
& f(x_{n-1}, x_{n-2}, \dots, x_i, \dots, x_0) = f(x_{n-1}, x_{n-2}, \dots, 1, \dots, \\
& x_0) \oplus \bar{x}_i \{ f(x_{n-1}, x_{n-2}, \dots, 0, \dots, x_0) \oplus \\
& f(x_{n-1}, x_{n-2}, \dots, 1, \dots, x_0) \}. \quad (5)
\end{aligned}$$

因此, 每个变量 x_i 都有 3 种形式. 极性 P 可用三进制表示 $P = P_{n-1}P_{n-2} \dots P_0$, 其中, P_k 为变量 x_k 的极性: 若 $P_k = 2$, 变量 x_k 将以原变量或反变量的形式出现; 若 $P_k = 1$, 变量 x_k 将只以反变量的形式出现; 若 $P_k = 0$, 变量 x_k 将只以原变量的形式出现. n 变量 MPRM 表达式可表示为:

$$f(x_{n-1}, x_{n-2}, \dots, x_0) = \sum_{j=0}^{3^n-1} b_j \pi_j, \quad (6)$$

其中, 乘积项 π_j 可表示为 $\pi_j = (\dot{x}_{n-1} \dot{x}_{n-2} \dots \dot{x}_0)$, 下标 j 表示为 $j = (j_{n-1} j_{n-2} \dots j_0)$. $\dot{x}_i$ 、 $\bar{\pi}_j$ 、 j_i 和 P_i 有如下关系:

$$\dot{x}_j = \begin{cases} \bar{x}_j, & j_i = 0, P_i = 2, \\ x_j, & j_i = 1, P_i = 2, \end{cases} \quad (7a)$$

$$\dot{x}_j = \begin{cases} 1, & j_i = 0, P_i = 0, \\ x_j, & j_i = 1, P_i = 0, \end{cases} \quad (7b)$$

$$\dot{x}_j = \begin{cases} 1, & j_i = 0, P_i = 1, \\ \bar{x}_j, & j_i = 1, P_i = 1. \end{cases} \quad (7c)$$

2 基于最小项的 MPRM 列表极性转换法

逻辑函数的列表表示法不仅可以表示传统的 Boolean 逻辑, 也可以表示 MPRM 逻辑. 对于单输出逻辑函数, 可以用 $(i_{n-1} i_{n-2} \dots i_0)$ 表示逻辑表达式中乘积项下标的二进制表示; 对于多输出逻辑函数, 用 $(c_{m-1} c_{m-2} \dots c_0)$ 来表示 m 个输出, 每位 c_j 的取值分别表示对应乘积项 $(i_{n-1} i_{n-2} \dots i_0)$ 是否出现在该输出函数中: 若 c_j 取 0, 表示 $(i_{n-1} i_{n-2} \dots i_0)$ 不出现; 若 c_j 取 1, 表示 $(i_{n-1} i_{n-2} \dots i_0)$ 出现.

不同极性下 MPRM 逻辑函数有不同的表示形式, 采用极性变换优化 MPRM 逻辑函数成为有效优化方法之一. MPRM 逻辑每位都有 3 种极性, 所以每位的极性转换共有 6 种情况, 下面以极性 2 转换到极性 0、1 为例进行说明^[6].

(1) 当初始极性 $P_k = 2$, 目标极性 $P'_k = 0$ 时, 表达式中应只出现原变量 x_k , 若出现 $(\dot{x}_{n-1} \dot{x}_{n-2} \dots \bar{x}_k \dots \dot{x}_0)$, 列表法表示为 $(i_{n-1} i_{n-2} \dots i_{k+1} 0 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$, 根据公式:

$$\begin{aligned}
& \dot{x}_{n-1} \dot{x}_{n-2} \dots \bar{x}_k \dots \dot{x}_0 = \dot{x}_{n-1} \dot{x}_{n-2} \dots x_k \dots \dot{x}_0 \oplus \\
& \dot{x}_{n-1} \dot{x}_{n-2} \dots \dot{x}_{k+1} \dot{x}_{k-1} \dots \dot{x}_0, \quad (8)
\end{aligned}$$

其中, $\dot{x}_{n-1} \dot{x}_{n-2} \dots \dot{x}_{k+1} \dot{x}_{k-1} \dots \dot{x}_0$ 在 $P'_k = 0$ 极性下的列表法表示式与 $\dot{x}_{n-1} \dot{x}_{n-2} \dots \bar{x}_k \dots \dot{x}_0$ 在 $P_k = 2$ 极性下的列表法表示式相同, 保留此行列列表项, 再向列表中添加新项 $(i_{n-1} i_{n-2} \dots i_{k+1} 1 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$ 即可.

(2) 当初始极性 $P_k = 2$, 目标极性 $P'_k = 1$ 时, 表达式中应只出现反变量 $\bar{x}_k$, 若出现 $(\dot{x}_{n-1} \dot{x}_{n-2} \dots x_k \dots \dot{x}_0)$, 列表法表示为 $(i_{n-1} i_{n-2} \dots i_{k+1} 1 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$, 根据公式:

$$\begin{aligned}
& \dot{x}_{n-1} \dot{x}_{n-2} \dots x_k \dots \dot{x}_0 = \dot{x}_{n-1} \dot{x}_{n-2} \dots \bar{x}_k \dots \dot{x}_0 \oplus \\
& \dot{x}_{n-1} \dot{x}_{n-2} \dots \dot{x}_{k+1} \dot{x}_{k-1} \dots \dot{x}_0, \quad (9)
\end{aligned}$$

其中, $\dot{x}_{n-1} \dot{x}_{n-2} \dots \bar{x}_k \dots \dot{x}_0$ 在 $P'_k = 1$ 极性下的列表表示式与 $\dot{x}_{n-1} \dot{x}_{n-2} \dots x_k \dots \dot{x}_0$ 在 $P_k = 2$ 极性下列表表示式相同, 保留此行列列表项, 再向列表中添加新项 $(i_{n-1} i_{n-2} \dots i_{k+1} 0 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$, 再将所有列表行的 i_k 位取反.

其余 4 种转换规则均可按照上述方法得到. 所有 6 种转换规则可归结为:

(1) 初始极性 P_k 与目标极性 P'_k 若满足 $(P_k = 0, P'_k = 1)$, $(P_k = 1, P'_k = 0)$, $(P_k = 2, P'_k = 1)$ 中之一, 且原列表中出现 $(i_{n-1} i_{n-2} \dots i_{k+1} 1 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$, 则插入新行 $(i_{n-1} i_{n-2} \dots i_{k+1} 0 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$;

(2) 初始极性 P_k 与目标极性 P'_k 若满足 $(P_k = 0, P'_k = 2)$, $(P_k = 2, P'_k = 0)$, $(P_k = 1, P'_k = 2)$ 中之一, 且在原列表中出现 $(i_{n-1} i_{n-2} \dots i_{k+1} 0 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$, 则插入新行 $(i_{n-1} i_{n-2} \dots i_{k+1} 1 i_{k-1} \dots i_0, c_{m-1} c_{m-2} \dots c_0)$.

(3) 初始极性 P_k 与目标极性 P'_k 若满足 $(P_k = 2, P'_k = 1)$ 或 $(P_k = 1, P'_k = 2)$ 则将列表行 $(i_{n-1} i_{n-2} \dots i_k \dots i_0, c_{m-1} c_{m-2} \dots c_0)$ 中 i_k 取反.

3 基于不相交项的列表极性转换算法

笔者提出了一种基于不相交项的混合极性列表转换算法, 即首先将逻辑函数表达式转换为乘积项两两不相交的形式, 则:

$$f = \bigcup_{i=1}^k p_i = \bigoplus_{i=1}^k p_i, \quad (10)$$

其中, $\bigoplus$ 代表异或运算. 不相交项与最小项都是两两互不相交的, 但是 1 个不相交项通常包含多个最小项, 因此使用不相交项表示 1 个函数要比最小项简洁得多.

3.1 不相交项的生成

定义 1 对于任意 2 个变量 $x_i, x_j \in \{0, 1, -\}$ (其中 “-” 表示任意值), 表达式 $x_i \cap x_j$ 代表 x_i 与 x_j 的相交运算, 运算规则为: $0 \cap 0 = 0, 1 \cap 1 = 1, 0 \cap - = 0, 1 \cap - = 1, - \cap - = -, 1 \cap 0 = \emptyset$.

定义 2 对于任意 2 个乘积项 p_w, p_v 和 x_{wi}, x_{vi} 分别代表乘积项的第 i 位, 如果存在某位的按位异或操作满足 $x_{wi} \wedge x_{vi} = 1$, 则称乘积项 p_w, p_v 互不相交, 记为 $p_w \cap p_v = \emptyset$.

定义 3 对于任意 2 个乘积项 p_w, p_v , 若存在 $p_w \cap p_v = \{(x_{w(n-1)} \cap x_{v(n-1)}), \dots, (x_{w0} \cap x_{v0})\} = p_v$, 则称 p_w 包含 p_v , 记为 $p_v \subseteq p_w$.

设定有个逻辑函数表达式共有 n 个乘积项, 变量 a, b 为指向逻辑函数中乘积项的指针, ($a = i, b = j$) 表示 a, b 分别指向逻辑函数的第 i 个与第 j 个乘积项. a 的取值范围在 $1 \sim n$, b 的取值范围在 $2 \sim n$, 且在运算过程中 b 值要始终大于 a 值. 根据不相交锐积运算性质可知^[8-9], 运算结果可能会产生多个乘积项, 我们可用 d 来表示运算结果中包含的不相交乘积项个数.

算法步骤如下:

(1) 设置算法中变量初始值. $a = 1$ 表示 a 指向第 1 个乘积项; $b = n$ 表示指向最后 1 个乘积项; $d = 0$; $m = n$ 表示 m 记录上次运算循环结束时逻辑函数中的乘积项个数.

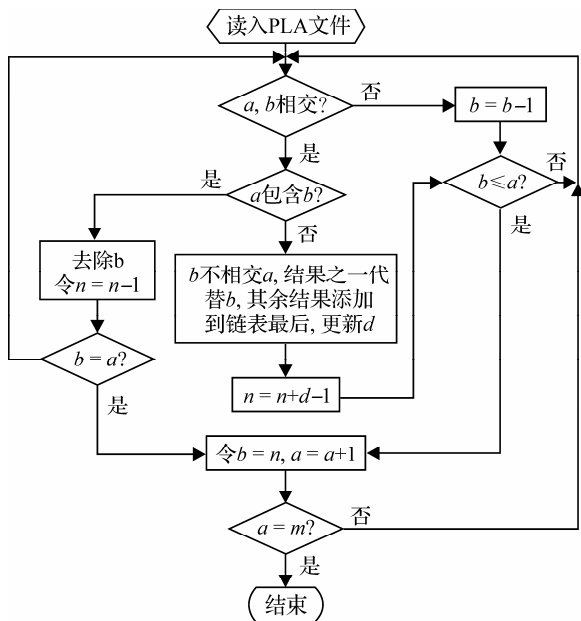


图 1 生成不相交项的算法流程

(2) 判断 a 和 b 的乘积项是否相交. 若相交, 则算法进入步骤 3; 若不相交, 进入步骤 5.

(3) 判断乘积项 a 是否包含乘积项 b . 若包含, 则在逻辑函数中去掉 b , 设置 $n = n - 1$, 如果 $b = a$, 进入步骤 6, 否则返回步骤 2 继续执行.

(4) 若不包含, 执行不相交锐积运算 $b \otimes a$, 更新变量 d , 并用运算结果的 1 个乘积项代替指针 b 当前指向的乘积项, 其余乘积项添加到函数链表最后面, 设置 $n = n + d - 1$, 如果 $b \leq a$ 进入步骤 6, 否则返回步骤 2.

(5) 设置 $b = b - 1$, 如果 $b \leq a$ 进入步骤 6, 否则返回步骤 2.

(6) 设置 $b = n$, $a = a + 1$. 如果 $a = m$, 则算法结束; 否则设 $m = n$, 返回步骤 2 继续执行.

相应算法流程如图 1 所示.

3.2 基于不相交项的列表转换算法

在最小项表达式中, 乘积项的各位只出现 “0” “1”, 而在不相交项中会出现 “-”, 因此必须要找出 “-” 在列表转换中的转换规则, 以下举例说明.

情况 1 原列表行中 $i_k = -$.

在极性 $P = (2222)_3$ 下, 某个乘积项为 $\{0-00\}$, 将其转换为极性 $P' = (2022)_3$. 利用最小项列表转换规则可得结果为 $\{0000\}$, 极性 $P'_2 = 0$ 下 $i_2 = 0$ 代表 $x_2 = 1$, 与极性 $P_2 = 2$ 下 $i_2 = -$ 意义相同, 因此定义: 在 0、1、2 三种极性下都以 $i_k = -$ 代表 $x_k = 1$. 这样在极性 $P_k = 2$ 下, $i_k \in \{0, 1, -\}$, $P_k = 0$ 或 1 下 $i_k \in \{1, -\}$, 上述转换结果为 $\{0-00\}$, 没有变化. 将 6 种极性都进行转换后发现, 若原列表中存在 $i_k = -$, 则极性转换时此位不用发生变化.

情况 2 原列表行中 $i_k = 1$.

当初始极性 P_k 与目标极性 P'_k 满足 $(P_k = 0, P'_k = 2)$ 或 $(P_k = 2, P'_k = 0)$ 时, $i_k = 1$ 都表示 $x_k = 1$, 列表不变. 当 $(P_k = 0, P'_k = 1)$ 或 $(P_k = 1, P'_k = 0)$ 或 $(P_k = 2, P'_k = 1)$ 时, 由 Shannon 公式可知, 要向原列表行添加新行 $(i_{n-1}i_{n-2}\dots i_{k+1}-i_{k-1}\dots i_0, c_{m-1}c_{m-2}\dots c_0)$, 其中, $(P_k = 2, P'_k = 1)$ 下的 $i_k = 0$ 同时变为 $i_k = 1$. 当 $(P_k = 1, P'_k = 2)$ 时, 将 $i_k = 1$ 变为 $i_k = 0$.

情况 3 原列表中 $i_k = 0$.

由情况 1 可知, 初始极性 $P_k = 0, 1$ 下不可能出现 $i_k = 0$. 当 $(P_k = 2, P'_k = 1)$ 时, $i_k = 0$ 变为 $i_k = 1$.

当 $(P_k = 2, P'_k = 0)$ 时, 添加新行 $(i_{n-1}i_{n-2} \cdots i_{k+1}1i_{k-1} \cdots i_0, c_{m-1}c_{m-2} \cdots c_0)$, 并将 $i_k = 0$ 变为 $i_k = -$.

将以上 3 种情况总结为:

(1) 当极性转换属于 $(P_k = 0, P'_k = 1)$ 、 $(P_k = 1, P'_k = 0)$ 、 $(P_k = 2, P'_k = 1)$ 中之一时, 若有 $i_k = 1$, 则添加新行 $(i_{n-1}i_{n-2} \cdots i_{k+1} - i_{k-1} \cdots i_0, c_{m-1}c_{m-2} \cdots c_0)$.

(2) 当极性转换属于 $(P_k = 2, P'_k = 0)$ 时, 若有 $i_k = 0$, 则添加新行 $(i_{n-1}i_{n-2} \cdots i_{k+1}1i_{k-1} \cdots i_0, c_{m-1}c_{m-2} \cdots c_0)$, 并将 $i_k = 0$ 变为 $i_k = -$.

(3) 当极性转换属于 $(P_k = 2, P'_k = 1)$, 若有 $i_k = 0$, 则将其变为 $i_k = 1$; 当属于 $(P_k = 1, P'_k = 2)$, 若有 $i_k = 1$, 则将其变为 $i_k = 0$.

最后, 检查列表中是否有 $i_{n-1}i_{n-2} \cdots i_0$ 项相同的两行. 若有, 则删除这两行的 $c_{m-1}c_{m-2} \cdots c_0$ 项按位异或运算. 若异或结果为 0, 则删除这两行; 否则将异或结果覆盖其中一行的 $c_{m-1}c_{m-2} \cdots c_0$ 项, 删除另一行.

例如给定 1 个逻辑函数如下:

$$\begin{cases} f_1 = \bar{x}_1\bar{x}_0 + x_0, \\ f_0 = \bar{x}_1\bar{x}_0. \end{cases} \quad (11)$$

现将此逻辑表达式转换为极性 $P = (210)_3$ 的表达式. 转换过程如图 2 所示.

$$\begin{array}{ccc} P = (222)_3 & P = (212)_3 & P = (210)_3 \\ \hat{x}_2 \hat{x}_1 \hat{x}_0 f_1 f_0 & \hat{x}_2 \hat{x}_1 \hat{x}_0 f_1 f_0 & \hat{x}_2 \hat{x}_1 \hat{x}_0 f_1 f_0 \\ - 0 0 1 1 & - 1 0 1 1 & - 1 - 1 1 \\ - - 1 1 0 & - - 1 1 0 & - - 1 1 0 \\ & & - 1 1 1 1 \end{array}$$

图 2 列表转换过程

最后, 我们可验证一下算法的有效性:

$$f_1 = \bar{x}_1 \oplus x_0 \oplus (\bar{x}_1 x_0) = \bar{x}_1(1 \oplus x_0) \oplus x_0 = \bar{x}_1\bar{x}_0 \oplus x_0 = \bar{x}_1\bar{x}_0 + x_0, \quad (12a)$$

$$f_0 = \bar{x}_1 \oplus (\bar{x}_1 x_0) = \bar{x}_1(1 \oplus x_0) = \bar{x}_1\bar{x}_0, \quad (12b)$$

由此可证提出的算法是正确的.

列表法的时间复杂度是 3^n , 其中 n 为列表的列数. 所以减少所需转换列数可以极大地减少算法时间复杂度. 笔者所提出的算法中选取某个判断量 $\alpha\%$, 若某列中的 “-” 的个数大于逻辑函数表达式不相交项个数的 $\alpha\%$, 则此列不进行转换. 对大量 PLA 测试文件进行统计分析得出, $\alpha\%$ 取值为 90% 时, 所需转换列数大都在 15 列以下, 因此可将 90% 设定为判断标准. 算法伪代码如下:

Step 1 读入 PLA 格式测试电路.

Step 2 根据 3.1 节描述算法将测试电路的乘积项转换为两两不相交的形式.

Step 3 计算列表中每列 “-” 的个数, 并将个数大于不相交项个数 $\alpha\%$ 的列所在数组的下标存储在数组 No_Conversion 中.

Step 4 初始化成本函数 Cost 与最优解 Min_Cost 为不相交项个数, 初始极性 $P[n] = \{2, 2, \dots, 2\}$, 其中, $n =$ 输入变量个数 $- R_{\text{No_Conversion}}$ 元素个数.

Step 5 根据 3 进制的运算规则得出目标极性 $R_{\text{New_P}}[n] = P[n] - 1$, 由 3.2 节所述转换规则进行极性转换, 其中 No_Conversion 所对应的列不需转换. 合并新行与原行中 $i_{n-1}i_{n-2} \cdots i_0$ 项相同的两行.

Step 6 更新 R_{Cost} 值为新极性下乘积项个数.

Step 7 若 $R_{\text{Cost}} < R_{\text{Min_Cost}}$, 则 $R_{\text{Min_Cost}} = R_{\text{Cost}}$.

Step 8 $P[n] = R_{\text{New_P}}[n]$.

Step 9 若重复次数小于 3^n , 重复 Step 5~Step 8; 否则输出 $R_{\text{Min_Cost}}$.

4 实验结果及分析

以逻辑表达式中所含乘积项数作为电路面积估算模型, 提出算法用 C 语言实现, 通过 VC++6.0 编译, 硬件环境为 Pentium(R) Dual-Core 3.0 GHz CPU 2 GB RAM; 实验采用的测试电路是 MCNC 和 ISCAS 基准电路, 所有实验电路均取值 $\alpha\%$ 为 90%.

实验结果见表 1. 其中, 第 1、2 栏分别是测试电路名称与输入、输出变量个数, 第 3、4 栏与 5、6 栏分别是文献[6]与本文的实验结果, time 与 terms 是算法所需 CPU 时间与所得的乘积项数目. 由表可见, 对于输入个数较少的电路来说, 本文算法与文献[6]算法相比优化结果相当. 而当输入个数 > 14 时, 文献[6]的算法几乎无法运行出结果, 而由于本文算法时间复杂度并不随输入变量个数呈指数倍增长, 所以计算速度具有明显优势. 同时, 文献[6]算法中变量出现情况只有 3 种——只出现原变量、只出现反变量、原反变量同时出现, 而本文算法只要满足不相交就可以, 例如 $\bar{x}_2x_1x_0$ 、 x_2x_1 、 $\bar{x}_1x_0$ 在文献[6]算法转换过程中不可能出现, 但它们是两两不相交的, 本文算法可直接进行转换, 这可导致两种算法解空间的不同, 从而影响优化结果.

表1 实验结果

电路名称	输入/ 输出	文献[6]		本文	
		time/s	terms	time/s	terms
newtag_d	8/1	0.14	6	0.13	6
rd84	8/4	1.47	107	0.73	107
sqrt8	8/4	0.45	26	0.44	26
misex1	8/7	0.25	13	0.34	17
apex4	9/19	18.16	444	28.25	451
alu2	10/6	65.81	225	40.70	225
cu	14/11	-	-	18.83	22
b12	15/9	-	-	69.42	76
pm1	16/13	-	-	71.83	27
tcon	17/16	-	-	0.00	31
pcl8	19/9	-	-	35.55	32
cc	21/20	-	-	4.89	53
cm150a	21/1	-	-	0.02	17
pcler8	27/17	-	-	309.56	56
c8	28/18	-	-	226.77	82

5 结论

提出了基于不相交项的列表转换算法,新算法较基于最小项的列表转换方法需处理的数据量大为减少.同时通过给定判断量 $\alpha\%$,进一步减少了算法的时间复杂度.最后以逻辑表达式乘积项个数为成本函数,通过极性搜索,实现给定逻辑函数的优化.将本文算法用于15个MCNC和ISCAS基准电路的优化,实验结果表明,与文献[6]的算法相比,在处理大电路时,本文算法计算速度具有明显优势.

参考文献:

- [1] Katz R H, Borriello G. Contemporary logic design[M]. 2nd ed. Upper Saddle River: Pearson Prentice Hall, 2004:38-66.
- [2] Damarla T, Karpovsky M G. Reed-Muller spectral technique for fault detection[J]. IEEE Transactions on Computers, 1989, 38(6):788-797.
- [3] Reddy S M. Easily testable realization for logic functions [J]. IEEE Transactions on Computers, 1972, 21(11):1183-1188.
- [4] Wang L, Almaini A E A. Optimisation of Reed-Muller PLA implementations[J]. Circuits, Devices and Systems, 2002, 149(2):119-128.
- [5] Ko S B. Area minimization of exclusive-OR intensive circuits in FPGAs[J]. Journal of Electronic Testing: Theory and Applications, 2004, 20(6):661-665.
- [6] 李辉, 汪鹏君, 王振海. 混合极性列表技术及其在MPRM 电路面积优化中的应用[J]. 计算机辅助设计与图形学学报, 2011, 23(3):527-533.
- [7] Almaini A E A, McKenzie L. Tabular techniques for generating Kronecker expansions[J]. Computers and Digital Techniques, 1996, 143(4):205-212.
- [8] 边计年, 薛宏熙, 苏明, 等. 数字系统设计自动化[M]. 北京: 清华大学出版社, 2005:209-216.
- [9] Falkowski B J, Schafer I, Perkowski M A. A fast computer algorithm for the generation of disjoint cubes for completely and incompletely specified boolean function[C]//Proceedings of the 33rd Midwest Symposium on Circuits and Systems, Piscataway, 1991:1119-1122.
- [10] 陈偕雄, 沈继忠. 近代数字理论[M]. 杭州: 浙江大学出版社, 2001:2-5.

An Improved Tabular-technique for Mix-polarity

LI Xin-wen, XIA Yin-shui*, WANG Lun-yao

(Institute of Circuits and Systems, Ningbo University, Ningbo 315211, China)

Abstract: Regarding the issues of the multi-output mixed polarity with Reed-Muller (MPRM) polarity conversion, a disjointed cube based tabular technique is proposed by searching the best polarity for function minimization. The experimental results show that the time complexity is notably reduced compared to that of traditional tabular technique.

Key words: tabular-techniques; mixed-polarity Reed-Muller; polarity conversion; disjointed cubes

(责任编辑 章践立)