文章编号:1674 - 2869(2011)01 - 0100 - 04

增强型 Quine-McCluskey 算法及其实验

陈付龙[1],纪书国[2],朱朝霞[3]

(1.安徽师范大学计算机科学与技术系,安徽 芜湖 241000;2.滁州实验中学,安徽 滁州 239000;
3.长江大学计算机科学学院,湖北 荆州 434023)

摘　要:传统的 Quine-McCluskey 算法第一步多从只拥有最小项的真值表开始,而真值表中的最小项数目是算法时间复杂度的最重要因素,数目越大,算法完成需要的时间越多.采用增强 Quine-McCluskey 算法,对真值表大小进行控制,利用更多归约规则,从而减少算法执行时间.

关键词:Quine-McCluskey 算法;逻辑函数;逻辑化简

中图分类号:TP302.2　　**文献标识码**:A　　**doi**:10.3969/j.issn.1674-2869.2011.01.025

0 引　言

数字系统中的逻辑电路按其结构可分为组合逻辑电路和时序逻辑电路两大类型.组合逻辑电路是一种逻辑电路,它的任一时刻的稳态输出,仅仅与该时刻的输入变量的取值有关,而与该时刻以前的输入变量取值无关.在组合逻辑电路中,每个输出都是其输入变量的逻辑函数.从电路结构分析,组合电路由各种逻辑门组成,网络中无记忆元件,也无反馈线[1].时序逻辑电路是指电路任何时刻的稳态输出不仅取决于当前的输入,还与前一时刻输入形成的状态有关.这种电路跟组合逻辑电路相反,它的输出结果是依照目前的输入和先前的输入有关系,拥有储存元件(内存)来存储信息.在时序逻辑电路中,若采用 Mealy 型状态机描述,则每个输出和内部状态都是其输入变量和内部状态的逻辑函数;若采用 Moore 型状态机描述,则每个输出都是其内部状态的逻辑函数,每个内部状态都是其输入变量和内部状态的逻辑函数.可见,逻辑函数是描述数字电路功能的重要方法[2].

逻辑函数化简,是为了获得最简化的逻辑电路,从而提高电路速度、安全性等性能,减少电路资源消耗.逻辑函数化简,没有严格的原则,它一般是依以下几个方面进行:(1)逻辑电路所用的门最少;(2)各个门的输入端要少;(3)逻辑电路所用的级数要少;(4)逻辑电路要能可靠地工作.

常用的化简方法有公式化简法[7]、卡诺图化简法[3]、Quine-McCluskey 化简法[4-5,8]和 Espresso 启发式化简法[6,9]等.公式化简法适于做书面逻辑推导,在形式化逻辑演算中比较常用,由于规则较多,EDA 软件中较少采用.卡诺图化简法在简单的数字逻辑电路设计中常用,其优点是直观、方便、容易掌握,缺点是当变量的数目(大于 6)较大时,图形庞大复杂,并且不易编程实现.Quine-McCluskey 化简法在功能上等同于卡诺图,能对任意数目变量构成的布尔函数化简,其优点是采用了表格形式而不是图形方式,这使它更适于计算机实现,并且它还给出了检查逻辑函数是否达到了最小化形式的确定性方法,可以判断算法终止的时机,可获得最优结果.Espresso 化简法会对一个逻辑系统做一些单一化假设,所以它运行的非常快,甚至是大型系统也非常快,但不能保证获得最优结果.

1 Quine-McCluskey 算法

Quine-McCluskey 算法(简称 Q-M 算法)是 $A+A'=1$ 最小化逻辑函数的一种方法.

1.1 基本定义

定义 1(真值)　在数字逻辑中,真值、布尔值或逻辑值是指真(用 1 表示)、假(用 0 表示)、未知(或任意)(用 X 或 x 表示),简写为 v.

定义 2(蕴涵项)　一个蕴涵项(Implicant Term)就是一个由若干真值的有序序列构成一个真值向量,记为 V.

定义 3(最小项)　对蕴涵项 V,若每个变量的

收稿日期:2010 - 11 - 22

基金项目:国家自然科学基金(60773223);安徽省高校青年教师科研基金项目(2008JQ1057);安徽省高校自然科学研究重点项目(KJ2010A148)

作者简介:陈付龙(1978 -),男,安徽霍邱人,副教授,博士.研究方向:嵌入式系统.

真值必须而且只能以 0 或 1 的形式出现一次，则称其为最小项(Minterm).

定义 4(蕴含最小项)　对蕴含项 V，若 $V=v_1v_2\cdots x\cdots v_n$，则可将其分解为 $V_1=v_1v_2\cdots 0\cdots v_n$ 和 $V_2=v_1v_2\cdots 1\cdots v_n$，从而派生出两个新的蕴含项 V_1 和 V_2；如此重复，直到不可分解为止，即输入向量均为最小项，产生的全部最小项称为蕴含项 V 的蕴含最小项，记为 $[V]$.

定义 5(蕴含)　对于蕴涵项 V_1 和 V_2，若 $[V_1]\subseteq[V_2]$，则称 V_1 蕴含于 V_2，记为 $V_1\prec V_2$.

定义 6(真值表)　一个真值表是一个描述逻辑函数的输入向量和输出变量之间映射关系的二维表，由若干使输出变量取 1 值的蕴涵项组成，简写为 T.

定义 7(质蕴涵项)　对于 $V\in T$，若不存在 $V'\in T, V'\neq V$，使 $V\prec V'$，则称 V 为质蕴涵项(Prime Implicant)，简称为质项.

该定义表明质蕴含项蕴含的全部最小项不能由其他蕴涵项所包含，在逻辑函数表达式中表现为该“与项”描述的逻辑关系不能由其他与项所代表.

定义 8(必要质蕴涵项)　对于 $V\in T$，若对 $\forall V'\in T, V'\neq V$，且 $\exists V''\in[V]$ 使 $V''\notin[V']$，则称 V 为必要质蕴涵项(Essential Prime Implicant)，简称为必要质项.

该定义表明必要质蕴涵项在描述输入和输出之间的逻辑关系时是必不可少的，在逻辑函数中表现为一个质蕴涵项蕴含有不被函数的其他任何质蕴涵项所蕴含的最小项.

1.2　算法原理

Q-M 算法主要分两个步骤：

第一步　找到这个逻辑函数的所有质蕴涵项. 对某个蕴涵项，其匹配项是指当且仅当有一位与其相反的蕴涵项，也称逻辑相邻项. 找质蕴涵项的过程，就是对蕴涵项与其逻辑相邻项，利用 $A+A'=1$ 律进行归约的过程. 例如，$t=0100$，则其匹配项 t' 可能为 1100、0000、0110 或 0101. 其中，利用 $A+A'=1$ 律，0100 与 1100 归约为 $X100$.

第二步　在找到的所有质蕴涵项中继续找到所有的必要质蕴涵项，去掉非必要质蕴涵项.

1.3　算法评价

由于在 n 个输入变量的逻辑函数中，最小项数目最多可达 2^n，因此，搜索质蕴涵项是一个 NP-完全问题，Q-M 算法的时间复杂度依赖于搜索空间大小，随输入变量数目大小而呈指数增长.

2　增强 Quine-McCluskey 算法实现

传统的 Q-M 算法第一步多从只拥有最小项的真值表开始，而真值表中的最小项数目是算法时间复杂度的最重要因素，数目越大，算法完成需要的时间越多. 特别是与 Espresso 算法相比，Q-M 算法的执行效率较低. 但随着计算机性能的提升，在实际设计中，可对真值表大小进行控制，利用更多化简规则，从而减少 Q-M 算法执行时间，获得某些应用的最优化设计.

2.1　归约方法

在真值表中，可以用蕴涵项代替最小项，例如 $\{01,1X\}\Leftrightarrow\{01,10,11\}$. 但是，01 和 $1X$ 无法用传统的 $A+A'=1$ 归约. 为此，须再使用 $A'B+A=A+B$ 进行归约，从而获得质蕴涵项 $\{X1,1X\}$.

```
/* ……
算法：2.1 reduceTerm
功能：利用 A+A'=1 和 A'B+A=A+B 归约
输入：蕴涵项 Term1，Term2
输出：质蕴涵项表 */
Flag1←true;//未发现可用 A+A'=1 归约
Flag2←true;//未发现可用 A'B+A=A+B 归约
Term←Φ;
while (Term1≠Φ && Term2≠Φ){
//取向量第一个真值
V1←first(Term1);V2←first(Term2);
//取向量剩余真值
Term1 ← rest (Term1); Term2 ← rest (Term2);
switch (V1,V2){
case (1,1);case (0,0);case (X,X):
Term←Term∪{V1};
case (0,1);case (1,0): //发现 A+A'=1
if Flag1 then {Term←Term∪{X}; Flag1←false;}
case (1,X): //发现 A'B+A=A+B
if Flag2 then {Term←Term∪{1}; Flag2←false;}
}
}
return Term;//返回归约后的结果
/* …… */
```

2.2　查找质蕴涵项

```
/* ……
```

```
算法:2.2 find Prime Implicant
功能:找质蕴涵项
输入:单输出变量真值表 TruthTable
输出:质蕴涵项表 */
//从真值表提取蕴涵项
ImplicantTable ← extractImplicant(TruthTable);
PrimeTable←Φ;//初始化质蕴涵表
ActiveTable←Φ;//初始化活动表
while (ImplicantTable≠Φ){//当蕴涵表不空
//从蕴涵表中取蕴涵项
Term←fetchTerm(ImplicantTable);
ImplicantTable←ImplicantTable-{Term};//删除该蕴涵项
ActiveFlag←false;//置不活动标记
for each Term'∈ ImplicantTable do{//对每个蕴涵项
NewTerm←reduceTerm(Term, Term');//归约
if ||NewTerm||=||Term|| then {
//若逻辑相邻
//即归约后的向量长度与被归约向量长度一致
Flag←true;//置活动标记
//加入蕴涵表
ImplicantTable←ImplicantTable∪{NewTerm};
ActiveTable←ActiveTable∪{Term'};//加入活动表
}
}
if ActiveFlag then
ActiveTable←ActiveTable∪{Term}//若活动
else if Term ActiveTable then
PrimeTable←PrimeTable∪{Term};//加入质蕴涵表
}
return PrimeTable;//返回质蕴涵表
/* ..... */
```

2.3 查找必要质蕴涵项

```
/* .....
算法:2.3 find Essential Prime Implicant
功能:找必要质蕴涵项
输入:质蕴涵项表 PrimeTable
输出:必要质蕴涵项表 EssImpTable */
Minterms ← findMinterms(PrimeTable);//收集所有最小项
EssImpTable←Φ;//取第一项初始化必要质蕴涵表
while (Minterms≠Φ){//当最小项表不空
MinTimeTerms ← minTime (Minterms, PrimeTable);
//查找被 PrimeTable 中质蕴涵项蕴含次数最少的最小项
Term←maxTerm(MinTimeTerms,PrimeTable);
//查找蕴含的最小项与 MinTimeTerms 交集最多的质蕴涵项
EssImpTable← EssImpTable∪{Term};
Minterms←Minterms-MinTimeTerms;
PrimeTable←PrimeTable-{Term}
}
return EssImpTable;//返回必要质蕴涵表
/* ..... */
```

3 Quine-McCluskey 算法性能评测

Q-M 算法和增强 Q-M 算法性能比较见表 1.

表 1 Quine-McCluskey 算法性能对比

Table 1 Performance comparison of Quine-McCluskey algorithms

测试用例	Q-M 算法		增强 Q-M 算法	
	表长度	执行时间/ms	表长度	执行时间/ms
83 编码器	8	2.66	8	2.66
38 译码器	8	2.66	8	2.66
优先级 83 编码器	255	138 578.00	8	750.93
BCD 编码器	10	0.94	10	0.94
优先级 BCD 编码器	1 023	>1 000 000.00	10	6 725.15
BCD 译码器	10	3.91	10	3.91
BCD7 段 LED 显示译码器	10	2.96	10	2.96
74138 优先编码器	256	>1 000 000.00	10	2 466.40
74138 译码器	40	875	10	62.03
1 位全加器	8	0.94	8	0.94
1 位比较器	4	0.32	4	0.32
4 位比较器	1 171	>1 000 000.00	9	1 899.37
2 路多路器	8	0.47	4	0.31
4 路多路器	64	47.00	8	7.50
4 路分配器	8	0.31	8	0.31

比较发现，增强 Q-M 算法在某些应用设计中具有更加优异的性能.

4　结束语

通过压缩真值表的大小，利用更多化简规则，使 Q-M 算法性能得到显著提升，从而使其对大数量输入的数字电路最优化设计具有应用价值，如数码显示与译码电路[10-11]. 在《数字逻辑》课程教学中，对比传统的 Q-M 算法，讲授增强型 Q-M 算法，可加强对数字逻辑简化的理解，并在实际设计中进行运用.

参考文献：

[1] Predko Michael , Predko Myke. Digital electronics demystified[M]. New York: McGraw-Hill, 2004.

[2] 张天瑜. 数字电视译码电路中改进型欧几里德算法[J]. 武汉工程大学学报，2009，31(12)：73－78.

[3] Karnaugh Maurice. The map method for synthesis of combinational logic circuits [J]. Transactions of American Institute of Electrical Engineers part I, 1953，72(9)：593－599.

[4] McCluskey E J. Minimization of boolean tunctions [J]. The Bell System Technical Journal，1956：1417－1444.

[5] McCluskey E J. Introduction to the theory of switching circuits [M]. New York: McGraw-Hill, 1965.

[6] Popov Andrey, Filipova Krasimira. Genetic algorithms synthesis of finite state machines[C]//27th International Spring Seminar on Electronics Technology：Meeting the Challenges of Electronics Technology Progress. Bankya, Bulgaria，IEEE，2004，3：388－392.

[7] Fišer Petr. Minimization of boolean functions [D]. MS. Thesis of Czech Technical University，2002.

[8] Rudell Richard L. Multiple-valued minimization for PLA synthesis[J]. IEEE Trans. on CAD，1987，6(5)：725－750.

[9] 邱建林，王波，刘维富. 大变量多输出逻辑函数实质项识别算法[J]. 计算机工程，2007，33(17)：57－59.

[10] 胡国元，胡婧，朱雄伟，等. 显微数码互动实验室系统在微生物实验教学中的应用[J]. 武汉工程大学学报，2010，32(10)：107－110.

[11] 庄芸，李晖. 软件进程分解方法模式[J]. 武汉工程大学学报，2010，32(11)：91－93.

Enhanced Quine-McCluskey algorithm and experiment

CHEN Fu-long[1]，***JI Shu-guo***[2]，***ZHU Zhao-xia***[3]

(1. Department of Computer Science and Technology, Anhui Normal University, Wuhu 241000, China;
2. Chuzhou Experimental High School, Chuzhou 239000, China;
3. College of Computer Science, Yangtze University, Jingzhou 434023, China)

Abstract: In the traditional Quine-McCluskey algorithm, a truth table including only minterms is the input data. The number of minterms is the most important factor of the algorithm's time complexity. The greater the number, the more time the algorithm needs to complete. In this paper, an enhanced Quine-McCluskey algorithm takes a truth table in smaller size and makes use of more reduction rules to reduce the execution time.

Key words: Quine-McCluskey algorithm; logic function; logic optimization

本文编辑：邹礼平