基于近似计算技术的FPRM电路面积优化

王伦耀,夏银水,储著飞
(宁波大学信息科学与工程学院,浙江宁波 315211)

摘 要: 近似计算技术通过降低电路输出精度实现电路功耗、面积、速度等方面的优化.本文针对RM(Reed-Muller)逻辑中"异或"运算特点,提出了基于近似计算技术的适合FPRM逻辑的电路面积优化算法,包括基于不相交运算的RM逻辑错误率计算方法,及在错误率约束下,有利于面积优化的近似FPRM函数搜索方法等.优化算法用MCNC(Microelectronics Center of North Carolina)电路进行测试.实验结果表明,提出的算法可以处理输入变量个数为199个的大电路,在平均错误率为5.7%下,平均电路面积减少62.0%,并在实现面积优化的同时有利于实现电路的动态功耗的优化且对电路时延影响不大.

关键词: 近似计算;RM函数;固定极性;逻辑优化
中图分类号: TP391.41 **文献标识码:** A **文章编号:** 0372-2112 (2019)09-1868-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.09.008

Area Optimization of FPRM Circuits Using Approximate Computing

WANG Lun-yao, XIA Yin-shui, CHU Zhu-fei
(*Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, Zhejiang* 315211, *China*)

Abstract: Approximate computing is a novel way in logic circuit design which offers the savings of the power, area and delay at cost of reduced accuracy. This paper focused on the fixed-polarity Reed-Muller(RM) functions area optimization by using approximate computing technique which is different from those used in traditional Boolean functions optimization in term of the characteristic of "XOR" in RM functions. The proposed algorithm mainly consists of the method of the error rate computing of RM functions using disjointed products and the approach of the approximate FPRM functions searching for less area under the given error rate constraint. The proposed algorithm is tested under MCNC(Microelectronics Center of North Carolina) benchmarks. The experimental results show that it can deal with the large function with 199 inputs. And by using the approximate computing technique, the average area can be reduced by 62.0% with the average error rate of 5.7%. The proposed approximate computing technique based algorithm is also beneficial for dynamic power saving and has little effect on the delay while optimizing the area of a circuit.

Key words: approximate computing; Reed-Muller functions; fixed-polarity; logic optimization

1 引言

近似计算(approximate computing)指计算结果与正确结果存在差异.近似计算技术体现在逻辑电路设计上表现为在给定的输入下,电路的实际输出与正确的结果有偏差.这种输出有偏差电路也称近似电路(approximate circuits),相应的逻辑优化技术称为近似优化技术.近似电路的出现一方面是由于有些电路本身具有容错特性;另一方面是有些电路的应用场合允许电路输出存在偏差.如在图像处理上,即使图像中一些像素的计算存在偏差,人眼也发现不了.正是由于电路在输出结果上允许近似,使得电路在设计时可以平衡电路输出精度与电路复杂性的关系,从而实现电路功耗、面积、速度等方面的优化,也使得近似计算成为目前集成电路设计中一个新的、重要的策略[1~3],并用于电路设计中[4,5].

近似计算中的近似程度可以用错误率(Error Rate, ER)、错误偏量(Error Distance, ED),均方差等指标来衡量[6~8].基于近似计算的逻辑优化技术的核心思想是在满足一定近似程度约束下实现电路功耗、面积、时延等

收稿日期:2018-04-09;修回日期:2019-03-28;责任编辑:梅志强
基金项目:国家自然科学基金(No.61471211, No.U1709218, No.61871242);浙江省自然科学基金(No.LY19F040004)

优化.在逻辑综合上,已开展的研究包括以二级逻辑为基础,通过添加或删除某些最小项实现面积优化[9];在多级逻辑综合上,通过对公共子电路进行近似优化[10]实现电路面积优化.目前基于近似计算技术的逻辑优化研究基本上针对以"与/或/非"为基本运算的传统布尔逻辑(Traditional Boolean,TB)开展的,而对于以"与/异或"为基本运算的 RM 逻辑电路基于近似计算技术优化的研究却鲜有报道.

RM 逻辑广泛应用于算术电路、通信电路,以及可逆逻辑电路[11,12]等设计中.研究表明,近一半的逻辑电路用 RM 逻辑实现,电路结构可以得到进一步的优化.由于"异或"运算是构成 RM 逻辑的基本运算,且满足 $1\oplus 1=0$,这意味着将偶数个相同的乘积项添加到 RM 函数中,不改变 RM 函数的正确性;同样,删除 RM 函数覆盖中偶数次重复的部分也不改变 RM 函数的正确性."异或"运算的这种特点,使得原来用于传统布尔逻辑电路的近似优化技术在 RM 函数近似优化过程中的应用遇到挑战.

本文主要讨论利用近似计算技术实现固定极性下 RM(Fixed Polarity Reed-Muller,FPRM)逻辑电路的面积优化.主要包括近似 FPRM 函数构建和近似 FPRM 函数的错误率 ER 计算.其中近似 FPRM 函数构造借助谱系数技术和利用遗传算法对错误率约束下的近似 FPRM 函数搜索实现.近似 FPRM 函数的错误率通过比较近似 FPRM 函数与原函数逻辑覆盖的差异得到.实验结果显示,在平均错误率为 5.7% 情况下,电路面积优化可以达到 62.0%;并且本文提出的算法可以实现输入 199 的大 FPRM 函数的近似逻辑优化.

2　RM 函数近似计算中错误率统计及算法实现

RM 逻辑以"与/异或"作为基本运算,当 RM 逻辑表达式中每一个变量被限定只能以原变量或反变量的其中一种形式出现时,这种 RM 表达式称为固定极性 RM(Fixed Polarity Reed-Muller,FPRM)表达式.反之,没有上述限定的 RM 表达式称为混合极性 RM(Mixed Polarity Reed-Muller,MPRM)表达式.由于 RM 逻辑表达式的复杂程度与变量的取值情况有关,因此极性变换是实现 RM 逻辑优化的一个重要途径[13,14].

对于变量 x,在 TB 逻辑中存在 $1+x=1$,但对于 RM 逻辑中的"异或"运算存在 $1\oplus x=\bar{x}$,即满足 $1\oplus 1=0$,这使得 RM 逻辑在基于近似计算优化方面具有新的特点.以图 1 为例,图 1(a)和图 1(b)卡诺图中取值为逻辑"1"的最小项一样.其对应的函数表示为 $f(x_1,x_2,x_3)=\overline{x_1}x_3+x_1x_2\,\overline{x_3}$.

如果将最小项 $x_1\,\overline{x_2x_3}$ 作为错误引入,$x_1\,\overline{\overline{x_2x_3}}$ 对应图 1 中标有"F"的方格.利用图 1(a)的卡诺图,就得到

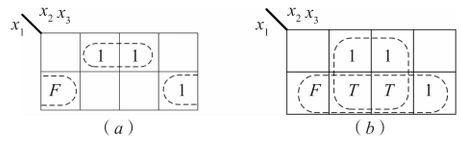

图1　RM函数中的最小项偶数次覆盖

$f(x_1,x_2,x_3)$ 的近似 TB 表达式为 $f_{\text{app\_TB}}(x_1,x_2,x_3)=\overline{x_1}x_3+x_1\,\overline{x_3}$;但如用 RM 逻辑来表示,可以得到图 1($b$)的卡诺图,对应的近似 RM 表达式为 $f_{\text{app\_RM}}(x_1,x_2,x_3)=x_1\oplus x_3$.在图 1($b$)中,标"$T$"的最小项被偶数次覆盖,考虑到 $1\oplus 1=0$,因此图 1(b)和图 1(a)逻辑功能一致,错误项均为 $x_1\,\overline{x_2x_3}$.

从图 1 的例子中不难发现 RM 逻辑和 TB 逻辑近似优化的不同:(1)对特定取值为"0"的乘积项的引入,可以简化 RM 表达式,但不增加错误输出;(2)对于 TB 逻辑,可以通过直接比较近似函数的乘积项集合与原函数的乘积项集合的差异得到引起错误输出的乘积项的集合,进而计算出错误率;但对于 RM 逻辑而言,则无法通过直接比较二个函数的乘积项集合得到引起错误输出的乘积项的集合,即 RM 逻辑与 TB 逻辑的错误率计算方法是不一样的.TB 逻辑和 RM 逻辑的上述差异使得原来适用 TB 逻辑近似优化的方法不能直接用于 RM 逻辑的近似优化.

在近似计算中,对于一个输入为 n 个变量的不含无关项的逻辑函数 f,如果用 C_{app} 表示近似函数包含的最小项集合,C_{org} 表示原函数 f 包含的最小项的集合,则错误率(ER)可以用式(1)表示:

$$\text{ER}=\frac{\Delta M[C_{\text{app}},C_{\text{org}}]}{2^n}*100\% \tag{1}$$

式(1)中,符号 $\Delta M[C_{\text{app}},C_{\text{org}}]$ 表示 C_{app} 和 C_{org} 包含的互不相同的最小项数量.$\Delta M[C_{\text{app}},C_{\text{org}}]$ 可用式(2)来表示.

$$\Delta M[C_{\text{app}},C_{\text{org}}]=M[(C_{\text{app}}\cup C_{\text{org}})-C_{\text{app}}\cap C_{\text{org}}] \tag{2}$$

式(2)中,符号 $M[C]$ 表示集合 C 包含的最小项的数量.

考虑到集合运算 $(C_{\text{app}}\cup C_{\text{org}})-C_{\text{app}}\cap C_{\text{org}}=(C_{\text{app}}-C_{\text{app}}\cap C_{\text{org}})\cup(C_{\text{org}}-C_{\text{app}}\cap C_{\text{org}})$,且属于集合 $(C_{\text{app}}-C_{\text{app}}\cap C_{\text{org}})$ 的最小项与属于集合 $(C_{\text{org}}-C_{\text{org}}\cap C_{\text{app}})$ 的最小项不重复,因此式(2)可以用式(3)来表示.

$$\begin{aligned}\Delta M[C_{\text{app}},C_{\text{org}}]&=M[C_{\text{app}}-C_{\text{app}}\cap C_{\text{org}}]\\&+M[C_{\text{org}}-C_{\text{app}}\cap C_{\text{org}}]\end{aligned} \tag{3}$$

式(3)中,$(C_{\text{app}}-C_{\text{app}}\cap C_{\text{org}})$ 包含的最小项属于近似函数但不属于原始函数;$(C_{\text{org}}-C_{\text{org}}\cap C_{\text{app}})$ 包含的最小项属于原始函数但不属于近似函数.

对 RM 函数而言,由于存在 $1\oplus 1=0$,使得适合 TB 逻辑函数的错误率计算方法不能直接用于 RM 逻辑的错误率计算.如 TB 逻辑函数 $f_{\text{TB}}(x_1,x_2,x_3)=x_1x_2+x_2x_3$ 和 RM 逻辑函数 $f_{\text{RM}}(x_1,x_2,x_3)=x_1x_2\oplus x_2x_3$,虽然构成

二个函数的乘积项是一样的,但是它们对应的最小项却不一样的,其中f_{RM}中不包含最小项$x_1x_2x_3$. 其原因是最小项$x_1x_2x_3$在f_{RM}中出现了2次而被删除. 因此在统计构成f_{RM}的任意二个乘积项p_i和p_j,$i\neq j$,所包含的最小项数量时,必须将p_i和p_j相交的部分分别从p_i和p_j中去除,剩下的部分才是$p_i\oplus p_j$对应的最小项. 为删除RM逻辑中被偶数次覆盖的部分,本文提出如算法1所示的乘积项双不相交锐积算法. 该算法通过构成RM逻辑的任意二个乘积项的相交部分,实现RM逻辑中被偶数次覆盖的部分的删除,得到与该RM函数等效的TB逻辑函数对应的乘积项的组合,进而实现错误率的计算.

算法1 double_disj_sharp(C_{RM})

{for any two RM products p_i,p_j in C_{RM},
do {$C_1=p_i\otimes p_j$;
$C_2=p_j\otimes p_i$;
$C_{RM}=C_1\cup C_2\cup\{C_{RM}-\{p_i,p_j\}\}$;
}while($p_i\cap p_j\neq\varnothing$);}

算法1中,RM函数的原始乘积项寄存在集合C_{RM},集合C_1和C_2分别用于存储二个乘积项不相交锐积的结果. 符号"$\otimes$"表示乘积项之间的不相交锐积运算,且有$p_i\otimes p_j=p_i-p_i\cap p_j$[15]. 以$f_{RM}(x_1,x_2,x_3)=x_1x_2\oplus x_2x_3$为例,$(x_1x_2)\otimes(x_2x_3)$等于在$x_1x_2$中删除$x_1x_2$与$x_2x_3$的相交部分,得到$(x_1x_2)\otimes(x_2x_3)=x_1x_2\overline{x_3}$;同理$(x_2x_3)\otimes(x_1x_2)=\overline{x_1}x_2x_3$. 通过双不相交锐积运算,将原本分别属于$x_1x_2$与$x_2x_3$的乘积项$x_1x_2x_3$删除了,得到与$f_{RM}(x_1,x_2,x_3)=x_1x_2\oplus x_2x_3$对应的TB逻辑乘积项组合为$\{x_1x_2\overline{x_3},\overline{x_1}x_2x_3\}$.

考虑到算法1是利用不相交锐积实现二个RM乘积项中相同部分的去除,因此,原始RM乘积项集合经过算法1后得到的是不相交乘积项的集合. 算法1这种特性使得由不相交乘积项构成的乘积项集合C包含的最小项的数量$M[C]$的计算变得十分的方便. 假设C中有K个不相交乘积项构成,则

$$M[C]=\sum_{i=0}^{K-1}(2^v|_{p_i\in C}) \tag{4}$$

式(4)中,$(2^v|_{p_i\in C})$表示属于集合C的乘积项p_i中有v个变量没有出现,则p_i包含的最小项数量为2^v. 利用式(4)可以避免为统计最小项的数量而对乘积项进行最小项展开,同时由于逻辑函数的不相交乘积项的数量远远小于最小项数量,且与逻辑函数的输入变量数无关,因此基于不相交乘积项的方法适合处理大逻辑函数. 此外,当v比较大时,2^v对应的十进制数也变得很大,式(4)的求和运算也变得不方便. 在本文中,用一个n位一维数组M_ARY来表示有n个变量的逻辑函数的$M[C]$,其中$M_ARY[v]=h$表示v个变量没有出现的乘积项数量一共有h个.

算法2 two_function_disj(C_1,C_2)

{for any product p_i in C_1 and any product p_j in C_2;
do {$C=p_i\otimes p_j$;
$C_1=(C_1-\{p_i\})\cup C$;
}while($p_i\cap p_j\neq\varnothing$);}

算法2中,集合C_1对应函数f_1乘积项集合,C_2对应函数f_2乘积项集合. 式(3)中$(C_{app}-C_{app}\cap C_{org})$可以用two_function_disj($C_{app}$,$C_{org}$)实现. 同理$(C_{org}-C_{org}\cap C_{app})$可以two_function_disj($C_{org}$,$C_{app}$)实现. 再利用式(4)就可以求得式(3)包含的最小项的数量. 进而可以得到式(1)的ER值.

3 纯RM近似函数的搜索

在利用近似计算技术实现电路面积优化时,电路面积经常用逻辑函数表达式中包含的字母数来衡量[6~10]. 在RM函数中,包含字母数最少的一种表达式称为纯RM(pure Reed-Muller)表达式. 该表达式为输入变量的"异或". 如$f_{PRM}(x_1,x_2,x_3,x_4)=x_1\oplus x_2\oplus\overline{x_3}\oplus x_4$就是一个纯RM表达式. TB逻辑表达式是否可以转换为一个纯RM表达式或与一个纯RM表达式近似,可以通过分析TB逻辑表达式的谱系数来实现[16].

对于一个n变量的TB函数f_{TB},其谱系数有2^n个,且任何一个谱系数r_i,$0\leqslant i\leqslant 2^n-1$,其取值范围为:$-2^n\leqslant r_i\leqslant 2^n$. 从谱系数$r_i$的取值情况可以得到:(1)$|r_i|$的大小反映了由$r_i$推导出的$f_{PRM}$与$f_{TB}$是否近似. $|r_i|$越大,f_{PRM}与f_{TB}越近似. 当$|r_i|=2^n$时,$f_{TB}=f_{PRM}$;(2)当$r_i<0$时,$\overline{f_{PRM}}$与f_{TB}近似;(3)f_{PRM}的表达式可以由r_i的下标i对应的二进制展开得到,f_{PRM}中只包含与i的二进制表示中取值为1的位对应的变量.

如TB函数$f_{TB}=x_1\overline{x_2x_3}+x_2x_3$的谱系数为$\boldsymbol{R}=\{r_0,r_1,r_2,\cdots,r_7\}=\{2,2,2,-6,2,2,2,2\}$,显然$\max(\boldsymbol{R})=r_3=-6$. 由于$3=(011)_2$,且$r_3<0$,所以$f_{PRM}=\overline{x_2\oplus x_3}=\overline{x_2}\oplus x_3$. 图2为$f_{TB}$与$f_{PRM}$的卡诺图表示.

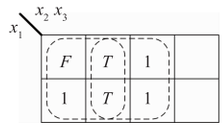

图2 近似函数f_{PRM}的卡诺图

图2中标"1"的部分对应f_{TB},二个虚线框分别对应f_{PRM}中的$\overline{x_2}$和x_3. 图2中标"T"的部分被二个虚线框偶

数次覆盖,考虑到 $1 \oplus 1 = 0$,因此,标"T"的部分实际逻辑值为"0". 而只有标"F"的位置是 $f_{TB}=0$ 但 $f_{PRM}=1$ 的地方,所以只有 x_1, x_2, x_3 均为"0"时,$f_{PRM} \neq f_{TB}$,其他情况下均相等. 但从字母数上比较,f_{TB} 有 5 个,而 f_{PRM} 只有 2 个,优化效果明显.

在本文中,通过下面方法由谱系数得到对应的 f_{PRM}.

(1)用快速算法得到 TB 函数逻辑函数的谱系数 $\boldsymbol{R}$[16];

(2)在 $\boldsymbol{R}$ 中寻找 $\max(|r_i|), r_i \in \boldsymbol{R}$,如果 $|r_i| \geq 2^{n-1}$,则 f_{PRM} 的表达式就是 i 的二进制表示中取值为"1"的变量的"异或". 当 $r_i < 0$ 时,取反其中任意一个变量. 当存在多个相同的 $\max(|r_i|), r_i \in \boldsymbol{R}$,且 $|r_i| \geq 2^{n-1}$,取表达式最简单的那一个作为 f_{PRM}.

(3)在本文中,只有 $\max(|r_i|) \geq 2^{n-1}$ 时才将 r_i 对应的纯 RM 逻辑函数作为构成 f_{TB} 近似函数的一部分加以提取,其中 n 为 f_{TB} 的变量数.

4　基于近似计算技术的 FPRM 面积优化的算法实现

基于近似计算技术的 FPRM 面积优化的核心是在引入一定错误输出时尽可能地减少逻辑表达式中包含的字母数. 有二个基本实现途径. 第一种途径是通过在 TB 逻辑函数中添加或删除一些特定的乘积项,使得修改后的 TB 函数转化为 FPRM 表达式后具有较少的字母数,从而实现电路面积优化;第二途径是先将 TB 逻辑函数转化为 FPRM 表达式,然后通过选择性的删除 FPRM 表达式中的部分乘积项,使得到的近似 FPRM 表达式字母数减少的同时满足错误率约束. 在第一种方法中,错误率计算比较容易,但要得到含字母数少的近似 FPRM 表达式比较难. 其主要原因是 TB 表达式中乘积项的增减与 FPRM 表达式中的字母减少没有必然联系. 较少乘积项的 TB 函数对应的 FPRM 表达式有时候更加复杂;而第二种方法则是容易得到含字母数少的近似 FPRM 表达式,但错误率计算过程复杂. 其主要原因在于在 TB 逻辑中,被乘积项覆盖到的部分都是逻辑"1",而在 RM 逻辑中,只有那些被乘积项奇数次覆盖到的部分才是逻辑"1".

本文的基于近似计算技术的 FPRM 面积优化是在第二种途径基础上,经过适当调整实现的. 算法 3 为优化算法的伪代码.

算法 3　FPRM_Area_Appr(C_{PLA}, C_{XOR_app})

```
{ C_dis = prdct_disjnt(C_PLA);//step1
  C_PRM = spectrum_analysis(C_dis);//step2
  if(C_PRM ≠ ∅){
      C_PRM_B = double_disj_sharp(C_PRM);//step3
      C_ex_PRM 1 = two_function_disj(C_dis, C_PRM_B);//step4-1
      C_ex_PRM 2 = two_function_disj(C_PRM_B, C_dis);//step4-2
      C_ex_PRM = C_ex_PRM 1 ∪ C_ex_PRM 2;//step4-3
      C_FPRM = FPRM_using_disprdct(C_ex_PRM);//step5
      C_XOR_app = GA_APP(C_FPRM);//step6}
else { C_FPRM = FPRM_using_disprdct(C_dis);//step7
      C_XOR_app = GA_APP(C_FPRM);//step8}
}
```

算法 3 中,待测试的 TB 逻辑函数采用 PLA 格式,C_{PLA} 表示 TB 逻辑函数包含的乘积项的集合,C_{XOR_app} 表示优化后的近似 FPRM 包含的 RM 乘积项集合.

在 step1 中,将 C_{PLA} 中的乘积项转化为不相交乘积项[15],有利于实现对大函数的 FPRM 的转化.

在 step2 中,对待优化函数进行谱系数分析,执行第一次近似计算;由于谱系数的计算是基于最小项展开的,因此当函数变量数 n 比较大时,会因最小项数过于庞大而不适合计算函数的谱系数. 在本文中只对输入变量不超过 20 的函数进行谱系数分析,大于 20 的函数直接令 $C_{PRM}=\varnothing$.

在 step3 中,利用算法 1 的双不相交锐积运算,删除 C_{PRM} 乘积项之间偶数次相交部分,将 C_{PRM} 转化为对应的不相交 TB 乘积项集合 C_{PRM_B}.

在 step4-1 到 step4-3 中利用二个逻辑函数相交部分删除算法完成 C_{PRM_B} 与 C_{dis} 相交部分的删除,得到 $C_{ex\_PRM} = (C_{PRM\_B} \cup C_{dis}) - (C_{PRM\_B} \cap C_{dis})$. 在 C_{ex_PRM} 中,各个乘积项是不相交的,且它们之间是逻辑"或"关系.

在 step5 中,利用基于不相交乘积项的列表技术[17],实现 TB 逻辑向 FPRM 逻辑的转化. 在本文中取 FPRM 的极性为 0.

在 step6 中,用遗传算法来实现 FPRM 乘积项的选择性删除. 具体做法如下:

(1)以 step5 中得到的 FPRM 乘积项数作为染色体的长度,对于某一个染色体 chrm_i,其染色体位为"1"表示该乘积项删除;

(2)用算法 1 将 FPRM 乘积项转化为不相交 TB 乘积项,利用式(1)计算误差值 ER_i;

(3)统计 chrm_i 中取值为"0"的位对应的 RM 乘积项包含的字母数,并加上 f_{PRM} 的字母数得到近似 FPRM 表达式中的字母数总和 Lts_i;

(4)考虑到不同染色体,如 $\text{chrm}_i, \text{chrm}_j, i \neq j$,可能存在 $\text{ER}(\text{chem}_i) > \text{ER}(\text{chem}_j)$ 而 $\text{Lt}(\text{chem}_i) < \text{Lt}(\text{chem}_j)$,或者倒过来这种情况,为了判别哪一个解更优,本文借助衡量电路性能的功耗延迟积的概念,引入字母错误率积的概念,用于衡量近似 FPRM 函数的优化效果,并以式(5)作为染色体 chrm_i 的代价函数.

$$\mathrm{cost}(\mathrm{chrm}_i) = \mathrm{ER}_i * \mathrm{Dec}(\mathrm{Lts\_app}_i) \quad (5)$$

式(5)中 $\mathrm{Dec}(\mathrm{Lts\_app}_i) = \mathrm{Lts\_app}_i/\mathrm{Lts\_org}$. 其中 $\mathrm{Lts\_org}$ 和 $\mathrm{Lts\_app}_i$ 分别表示构成 FPRM 函数的乘积项包含的原始字母数和运用近似计算技术后,染色体 chrm_i 对应的乘积项包含的字母数.

5 实验结果及分析

本文基于近似计算的 FPRM 面积优化算法用 C 语言实现,并在操作系统为 Windows10,CPU 时钟频率为 2.7Ghz,内存为 8GB 的 PC 机上实现. 表 1 为用近似计算方法优化后 RM 函数字母数的减少情况. 表 1 中的测试电路来自 MCNC Benchmark. 其中"*i*/*o*"表示测试电路的输入和输出变量数. Lts_org 表示待测试逻辑函数的在零极性下的 FPRM 表达式包含的字母数,Lts_app 表示利用近似计算技术优化后 FPRM 表达式包含的字母数;ER 为错误率,用式(1)方法求得;算法运行单位时间为秒. 考虑到初始种群对遗传算法的结果和运算时间影响比较大,本文给出的是 10 次运行的平均值. 遗传算法的参数设置:交叉率 0.5,变异率 0.15,种群数与输入的变量数 n 有关,当 $2^{n-1}<30$ 时,种群数为 2^{n-1},否则为 30,代数为 30. 表 1 中为了控制错误率 ER 在某一个数值 α 附近,适应度函数需要在式(5)基础上进行了适当的调整,改为

$$\mathrm{fitness}(\mathrm{chrm}_i) = \frac{1}{\left(\frac{\mathrm{ER}_i}{\alpha}\right)^{\beta} * \mathrm{Dec}(\mathrm{Lts\_app}_i)} \quad (6)$$

表 1 基于近似计算的 FPRM 优化结果

电路名称	*i*/*o*	Lts_org	Lts_app	ER(%)	面积优化程度	时间(s)
5xp1	7/10	365	150	11.9	58.9	0.17
alu2	10/6	1681	359	11.6	78.6	68.74
apex6	135/99	133633	52671	4.5	60.6	62.63
b12	15/9	1221	100	3.1	91.8	0.12
c8	28/28	4003	301	12	92.5	0.13
cc	21/20	240	158	1.5	34.2	0.06
clip	9/5	2206	1605	10.4	27.2	3.82
cu	14/11	1912	712	0.2	62.8	0.05
ex4	128/28	87582	17061	<0.1	80.5	86.67
examples	85/66	6979	1633	5.1	76.6	1.77
i1	25/13	7486	1216	<0.1	83.8	0.14
i6	138/67	1149	191	8.7	83.4	1.21
i7	199/67	1129	214	11.8	81.0	2.56
i8	133/81	434457	99592	7.4	77.1	782.91
parity	16/1	16	16	0	0	0.02
pm1	16/13	184	167	<0.1	9.2	0.05
s641	55/43	192916	42042	2.9	78.2	546.78
x3	135/99	133633	24846	3.8	81.4	295.34
x4	94/71	29848	5549	3.1	81.4	22.03
xor5	5/1	5	5	0	0	<0.01
平均				5.7	62.0	

在式(6)中,当 $\beta>1$ 时,某一个染色体 chrm_i 对应的错误率 $\mathrm{ER}_i>\alpha$ 或 $\mathrm{ER}_i<\alpha$ 时,$\mathrm{fitness}(\mathrm{chrm}_i)$ 会有明显的减少或增加,进而增加 $\mathrm{ER}_i<\alpha$ 的染色体被选中的概率,实现错误率的控制. 在本文中,取 $\beta=2$,$\alpha=10\%$. 由于 RM 逻辑电路的面积与 RM 函数包含的字母数密切相关,因此可以通过比较逻辑函数包含的字母数来衡量电路面积的大小. $\mathrm{Area_opt} = (\mathrm{Lts_org}-\mathrm{Lts_app})/\mathrm{Lts_org}$用来表示面积优化程度.

从表 1 可以看出,通过引入错误,测试电路的面积优化效果明显. 在平均错误率为 5.7% 情况下,字母数平均减少了 62.0%. 表 1 中电路 parity 和 xor5 可以由谱系数分析得到它们对应一个纯 RM 表达式,其表达式包含的字母数已经是最少了,如果再利用近似计算技术化简,只会增加字母数并会引入额外的错误率. 因此,本文一旦发现逻辑函数可以表示成一个纯 RM 表达式,本文就不再用近似计算技术简化. 反映在表 1 中,这类电路的错误率为 0,电路优化程度也为 0.

表 1 中电路近似优化后,通过式(6)的适应度函数将最大错误率控制 10% 左右. 除适应度函数外,也可以通过调节 FPRM 乘积项的删除概率来进一步控制错误率. 由于染色体中位的取值表示 FPRM 乘积项的取舍情况,当某一位染色体取值为"1"时,表示该位对应的乘积项删除. 考虑到 FPRM 近似函数的错误率和面积优化情况均与乘积项取舍情况有关,因此可以通过调节染色体上每一位取值为"1"的概率来影响近似函数的错误率和电路面积优化效果. 在本文中染色体各位取值为"1"的概率为 0.2.

本文的算法速度与待处理的逻辑函数的输入与输出的变量数大小没有必然的联系. 影响本文算法的速度的是嵌入在遗传算法中用于错误率计算的子程序有关,分别对应算法 1 和算法 2 的逻辑函数双锐积运算和逻辑函数相交部分删除运算. 当 FPRM 函数中被删除的乘积项数量越多,以及它们之间包含关系的越复杂,错误率计算越耗时.

在本文算法基本上以不相交乘积项为基础. 一般情况下,函数的不相交乘积项的数量远远小于函数包含的最小项的个数,且不相交乘积项的数量与输入变量的个数无关,从而使本文的算法适合处理输入变量很多的函数. 在表 1 中,最大的输入变量的数量为 199 个. 此外不相交乘积项的引入可以直接利用式(4)来计算最小项数量而不需要将乘积项展开为最小项.

表 1 中虽然只给出了利用近似计算技术在实现 FPRM 表达式的字母数减少的结果,但考虑到电路的动态功耗与电路包含的节点数量和节点充放电情况有关,而 FPRM 表达式中字母数的减少将有助于电路节点的减少,进而有助于电路动态功耗的减少. 在电路时延

方面,由于优化前后得到的 FPRM 函数对应的电路均为"与/异或"逻辑运算构成二级电路,因此,采用近似计算技术实现电路面积优化后,对电路时延影响不大.

6　结论

RM 逻辑以"与/异或"为基本逻辑运算,且"异或"运算具有 $1 \oplus 1 = 0$ 的特点,因此,RM 逻辑近似优化方法与 TB 逻辑的优化方法存在很大差别. 本文提出的基于近似计算技术实现 FPRM 面积优化主要包括下面三方面内容:(1)在乘积项不相交锐积运算的基础上提出了乘积项双不相交锐积算法和不同逻辑覆盖相交部分去除算法,进而实现 RM 逻辑函数近似优化后错误率的计算. (2)利用逻辑函数谱分析技术,实现与 TB 逻辑函数对应的纯 RM 近似逻辑表达式的搜索. (3)利用遗传算法,构造以错误率约束和面积优化为导向的适应度函数,实现 FPRM 函数的进一步优化. 实验结果显示,本文优化算法可以实现输入变量为 199 的逻辑函数近似优化,在平均错误率为 5.7% 情况下,FPRM 函数的字母数平均减少了 62.0%. 在实现面积优化的同时有利于实现电路的动态功耗的优化且对电路时延影响不大.

参考文献

[1] XU Q, MYTKOWICZ T, KIM N S. Approximate computing: a survey [J]. IEEE Design & Test, 2016, 33(1): 8 – 22.

[2] DAYA S K, BABAK Z, MEHRZAD S, et al. Quality control for approximate accelerators by error prediction [J]. IEEE Design & Test, 2016, 33(1): 43 – 50.

[3] DUCKHWAN K, JAEHA K, SAIBAL M. A power-aware digital multilayer perceptron accelerator with on-Chip training based on approximate computing [J]. IEEE Transactions on Emerging Topics in Computing, 2017, 5(2): 164 – 178.

[4] GUPTA V, MOHAPATRA D, RAGHUNATHAN A, et al. Low-power digital signal processing using approximate adders [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32(1): 124 – 137.

[5] VASILEIOS L, GEORGIOS Z, DIMITRIOS S, et al. Approximate hybrid high radix encoding for energy-efficient inexact multipliers [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018, 26(3): 421 – 430.

[6] SHIN D, GUPTA S K. Approximate logic synthesis for error tolerant applications [A]. 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010) [C]. Piscataway: IEEE, 2010. 957 – 960.

[7] LIANG J, HAN J, LOMBARDI F. New metrics for the reliability of approximate and probabilistic adders [J]. IEEE Transactions on Computers, 2013, 62(9): 1760 – 1771.

[8] MOMENI A, Han J, Montuschi P, et al. Design and analysis of approximate compressors for multiplication [J]. IEEE Transactions on Computers, 2015, 64(4): 984 – 994.

[9] ICHIHARA H, INAOKA T, IWAGAKI T, et al. Logic simplification by minterm complement for error tolerant application [A]. Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD) [C]. Piscataway: IEEE, 2015. 94 – 100.

[10] WU Y, QIAN W. An efficient method for multi-level approximate logic synthesis under error rate constraint [A]. Proceedings of the 2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC) [C]. Piscataway: IEEE, 2016. 1 – 6.

[11] MOZAMMEL H A K. Design of reversible synchronous sequential circuits using pseudo Reed-Muller expressions [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2014, 22(11): 2278 – 2286.

[12] 王友仁,沈先坤,周影辉. 基于 KFDD 的可逆逻辑电路综合设计方法[J]. 电子学报,2014,42(5):1025 – 1029.
WANG You-ren, SHEN Xian-kun, Zhou Ying-hui. Synthesis design method of reversible logic circuit based on kronecker functional diagram [J]. Acta Electronica Sinica, 2014, 42(5): 1025 – 1029. (in Chinese)

[13] He Z, XIAO L, ZHANG L, et al. EMA-FPRMs: An efficient minimization algorithm for fixed polarity Reed-Muller expressions [A]. Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT) [C]. Piscataway: IEEE, 2016. 253 – 256.

[14] 汪鹏君,汪迪生,蒋志迪,等. 基于 PSGA 算法的 ISFPRM 电路面积与功耗优化[J]. 电子学报,2013,41(8):1542 – 1548.
WANG Peng-jun, WANG Di-sheng, JIANG Zhi-di, et al. Area and power optimization of ISFPRM circuits based on PSGA algorithm [J]. Acta Electronica Sinica, 2013, 41(8): 1542 – 1548. (in Chinese)

[15] 王伦耀,夏银水,陈偕雄,等. 基于不相交乘积项的逻辑探测和拆分算法[J]. 电子学报,2012,40(10):2091 – 2096.
WANG Lun-yao, XIA Yin-shui, CHEN Xie-xiong, et al. Logic detection and decomposition algorithm based on disjointed cubes [J]. Acta Electronica Sinica, 2012, 40(10): 2091 – 2096. (in Chinese)

[16] 陈偕雄,沈继忠. 近代数字理论[M]. 杭州:浙江大学出版社,2001. 85 – 102.
CHEN Xie-xiong, SHEN Ji-zhong. Modern Digital Theory [M]. Hangzhou: Zhejiang University Press, 2001. 85 – 102. (in Chinese)

[17] 王玉花,王伦耀,夏银水. 大电路固定极性 Reed-Muller

逻辑快速转换算法[J]. 计算机辅助设计与图形学学报,2014,26(11):2091 -2098.
WANG Yu-hua, WANG Lun-yao, XIA Yin-shui. A fast Reed-Muller fixed polarity conversion algorithm for large circuits[J]. Journal of Computer-Aided Design & Computer Graphics,2014,26(11):2091 -2098. (in Chinese)

作者简介

王伦耀 男,1972 年生于浙江宁波. 宁波大学信息科学与工程学院教授. 研究方向为低功耗集成设计、集成电路设计自动化等.
E-mail:wanglunyao@ nbu. edu. cn

夏银水 男,1963 年生于浙江余姚. 宁波大学信息科学与工程学院教授. 研究方向为低功耗集成设计、物联网芯片设计、集成电路设计自动化等.

储著飞 男,1986 年生于安徽潜山. 宁波大学信息科学与工程学院副教授. 研究方向为集成电路设计自动化、低功耗集成电路设计等.