

基于近似计算技术的 FPRM 逻辑功耗优化

王一琛, 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: 提出了一种基于近似计算技术的 Fixed Polarity Reed-Muller(FPRM)逻辑功耗优化算法, 该算法包括基于信号概率和跳变密度的固定极性 Reed-Muller(RM)函数动态功耗模型, 基于遗传算法的以功耗优化为导向的 RM 逻辑极性搜索方法, 以及利用双锐积运算的 RM 逻辑错误率计算方法. 在错误率的约束下, 通过有选择性地删减部分乘积项, 实现功耗优化. 提出的算法用 C 语言实现, 并用 MCNC Benchmark 电路测试. 结果表明: 与原始 FPRM 电路功耗相比, 在平均错误率为 3.21% 时, 电路动态功耗平均减少了 22.77%.

关键词: 近似计算; 错误率; 极性搜索; 功耗优化

中图分类号: TN47; TP391

文献标志码: A

文章编号: 1001-5132 (2020) 01-0045-06

逻辑功能正确一直是传统数字逻辑在设计时的首要条件, 且为保证功能正确不惜牺牲电路的面积、功耗、时延等性能指标. 但在有些应用中, 如图像、音频处理, 待处理的数据具有一定的容错性, 在某一范围内的计算错误不会影响电路的实际应用, 待处理数据的这种容错特性意味着可以适当降低计算精度, 进而实现电路其他性能指标的提升, 这种非精确计算称之为近似计算. 利用近似计算技术实现电路的逻辑优化称为近似优化. 目前, 近似计算已成为数字集成电路设计中一个新的策略.

逻辑函数通常采用 2 种基本形式, 既可采用基于“与/或/非”运算的传统布尔(Traditional Boolean, TB)逻辑, 也可采用基于“与/异或”运算的 Reed-Muller(RM)逻辑. 目前, 在 TB 逻辑上开展的研究主要包括基于近似计算的二级逻辑电路优化^[1-3]、多级逻辑电路优化^[4-5]等. 多数优化算法以乘积项为基础实现近似函数的构造和近似度计算, 但是也有基于伪乘积项^[6]或借助二元决策图(Binary Decision Diagram, BDD)来实现^[7].

除 TB 逻辑外, RM 逻辑正吸引越来越多研究

者的兴趣, 其原因包括: (1)随着集成电路技术的发展, 构成 RM 逻辑的“异或”门无论是速度、功耗还是面积都获得极大的改进, 使之更加适合实际应用. (2)与 TB 逻辑相比, RM 逻辑在算术及通信电路、可逆逻辑设计^[8]、可测试方面等应用具有优势. (3)近一半的电路, 如采用 RM 逻辑进行综合可获得更加简单的结果. (4)有些新的器件具有“异或”特性, 相比互补金属氧化物半导体(Complementary Metal Oxide Semiconductor, CMOS)电路, 新器件在“异或”运算实现将变得极为简单.

目前, 已有不少 RM 逻辑功耗优化方法, 包括 RM 函数极性搜索方法、多输入 XOR 门阵列分解方法; 搜索方法包括遗传算法^[9]、粒子群算法^[10]、Min-Huffman 算法^[11]等. 上述优化算法均没有涉及错误引入对功耗优化的影响.

本文主要研究近似计算技术在 RM 逻辑电路功耗优化中的应用, 提出了一种利用近似计算技术实现 FPRM 逻辑电路功耗优化的方法, 即建立以信号概率和跳变密度为参数的 RM 逻辑电路动态功耗数学模型; 并在错误率约束下, 利用提出的优化策略删减部分乘积项, 实现 FPRM 逻辑电路

收稿日期: 2019-05-28.

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

基金项目: 国家自然科学基金(U1709218, 61871242); 浙江省自然科学基金(LY19F040004); 宁波市自然科学基金(2019A610077).

第一作者: 王一琛(1993-), 女, 浙江舟山人, 在读硕士研究生, 主要研究方向: 逻辑综合与优化. E-mail: 543746091@qq.com

*通信作者: 王伦耀(1972-), 男, 浙江宁波人, 教授, 主要研究方向: 逻辑综合与优化. E-mail: wanglunyao@nbu.edu.cn

动态功耗的优化.

1 RM 逻辑电路的动态功耗数学模型

逻辑电路的功耗包括静态功耗和动态功耗, 动态功耗是功耗的主要部分, 由电路中所有节点电容充放电引起, 可用式(1)表示:

$$P_{dp} = \frac{1}{2} V_{dd}^2 \sum_{i=0}^{N-1} c_i w_i, \quad (1)$$

式中: P_{dp} 为动态功耗; N 为电路总节点数; V_{dd} 为电源电压; c_i 为第 i 个节点的等效电容; w_i 为第 i 个节点的开关活动率, 其大小等于第 i 个节点单位时间内平均充放电次数.

从式(1)可知, 只要减小任何一个参数都可以实现电路的动态功耗优化. 本研究主要通过降低电路开关活动率 w_i 来实现电路的功耗优化.

对于第 i 个节点 w_i , 其大小与加在该节点上的信号 x 的概率 $P(x)$ 和跳变密度 $D(x)$ 有关. 其中 $P(x)$ 等于信号 x 在时间 T 内取值为逻辑 1 的比例.

$$P(x) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t) dt. \quad (2)$$

信号 x 的跳变密度 $D(x)$ 等于 x 在单位时间内的跳变次数, 即在时间 T 内的跳变次数 $n_x(T)$ 与 T 比值.

$$D(x) = \lim_{T \rightarrow \infty} n_x(T) / T. \quad (3)$$

从式(3)跳变密度定义可看出, 信号的跳变密度和开关活动性本质一致, 因此本研究用信号跳变密度来表示开关活动性.

RM 逻辑是由“与/异或”运算构成的二级逻辑, 因此 RM 逻辑电路功耗估算可以通过对与门阵列和异或门阵列功耗估算来实现. 考虑到多输入逻辑门均可以分解成二输入逻辑门的组合, 因此本研究通过对二输入与门和异或门功耗的估算来实现 RM 逻辑电路的功耗估算.

对于输入为 x_1 和 x_2 , 输出为 y_2 的二输入与门, 设 x_1 和 x_2 的信号概率分别为 $P(x_1)$ 和 $P(x_2)$, 跳变密度分别为 $D(x_1)$ 和 $D(x_2)$. 考虑到二输入与门中只有一个输入信号为高电平时, 另一个信号的跳变才能传输到输出端, 因此输出端 y_2 的跳变密度 $D(y_2)$ 可以表示为:

$$D(y_2) = P(x_1)D(x_2) + P(x_2)D(x_1). \quad (4)$$

通过式(4)可以推导出 K 个输入变量的与门的

输出端 y_K 的跳变密度 $D(y_K)$ 可以表示为:

$$D(y_K) = \sum_{i=1}^K [(\prod_{j=1}^K P(x_j)) D(x_i)], \quad (5)$$

式中: $j \in [1, 2, \dots, K], i \neq j$.

对于有 K 个输入变量的异或门, 由于输入端的任何跳变都会引起输出端的变化, 因此输出端的跳变密度 $D(y_K)$ 就等于各输入信号跳变密度之和, 即:

$$D(y_K) = \sum_{i=1}^K D(x_i). \quad (6)$$

利用式(5)和式(6)可以计算出由 W 个与门, V 个异或门构成的 RM 逻辑电路所有内部节点的总的跳变密度 D_{RM} , 即:

$$D_{RM} = \sum_{i=1}^W D_{AND}(i) + \sum_{j=1}^V D_{XOR}(j), \quad (7)$$

式中: $D_{AND}(i)$ 表示第 i 个与门输出跳变密度; $D_{XOR}(j)$ 表示第 j 个异或门输出跳变密度.

在 RM 逻辑电路的功耗优化中, 极性变换是实现 RM 逻辑电路功耗优化的一种有效手段. 以式(8)为例:

$$\begin{aligned} f_m(x_1, x_2, x_3) &= \overline{x_1 x_2 x_3} \oplus \overline{x_1 x_2} \oplus x_1 x_3 = \\ &= [\overline{x_1 x_2} (1 \oplus x_3)] \oplus \overline{x_1 x_2} \oplus [(1 \oplus \overline{x_1}) x_3] = \\ &= \overline{x_1 x_2 x_3} \oplus x_3 \oplus \overline{x_1 x_3} : (P = (110)_2 = 6) = \\ &= 1 \oplus x_1 \oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3 : \\ &= (P = (000)_2 = 0). \end{aligned} \quad (8)$$

假设 3 个输入变量的概率与跳变密度分别为 $P(x_1, x_2, x_3) = (0.7, 0.8, 0.3)$, $D(x_1, x_2, x_3) = (0.2, 0.4, 0.1)$, 则由式(5)~(7)可以得到: 当极性分别为 6 和 0 时, 式(8)的 RM 函数对应总的信号跳变密度分别为 0.244 和 0.878. 因此, 可以通过极性变换实现 RM 逻辑电路的功耗优化^[12].

2 近似 RM 逻辑的错误率计算方法

基于近似计算的 RM 逻辑函数优化可以通过增减原函数的乘积项来实现 RM 逻辑函数面积优化, 减少电路节点数, 从而实现 RM 逻辑函数功耗优化.

图 1(a)为式(8)对应的卡诺图, 若修改图 1(a)中某些最小项的取值, 如将最小项 $\overline{x_1 x_2 x_3}$ 的取值由“1”变为“0”, 即改变图 1(b)中标有“F”的最

小项取值, 则由图 1(b)可得式(8)的近似表达式为:

$$f_{ap} = \overline{x_1} \oplus x_3. \quad (9)$$

显然式(9)要比式(8)简单. 从图 1 可知, 通过引入特定的错误可实现 RM 逻辑函数的简化. 图 1(b)中可能的输入有 $2^3=8$ 种, 只有输入 $(x_1x_2x_3)=(011)$ 时, 输出错误, 因此错误率为 $1/8$.

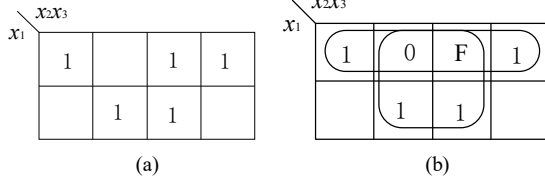


图 1 近似 RM 逻辑函数的卡诺图表示

近似逻辑优化是在近似度约束条件下的逻辑优化, 其中近似度表示近似逻辑函数与原逻辑函数的近似程度, 而近似程度可用错误率(Error Rate, ER)^[1]以及错误偏差(Error Distance, ED)^[13]来描述, 也可以用错误频率(Error Frequency)和错误幅度(Error Magnitude)^[14]来描述. 本文用 ER 来描述近似逻辑函数与原函数之间的近似程度. 近似逻辑函数 $\hat{f}(X)$ 与原函数 $f(X)$ 的错误率定义为引起 $f(X)$ 错误输出的所有输入组合之和占总输入组合的百分比, 可表示为:

$$ER = \frac{\sum (\hat{f}(X) \neq f(X))}{2^n}, \quad (10)$$

式中: ER 表示错误率; X 表示输入的最小项的集合; n 为输入的变量数; $\sum (\hat{f}(X) \neq f(X))$ 表示引起 $f(X)$ 错误输出的所有输入组合之和.

近似函数的错误率计算方法与逻辑函数采用的形式有关. 以图 2 卡诺图对应的逻辑函数为例, 在图 2(a)中, 取值为“1”的最小项组成的逻辑覆盖对应的逻辑函数为 $f(x_1, x_2, x_3) = \overline{x_1}x_3 + x_1x_2\overline{x_3}$; 如果将图 2(b)中标有“F”的最小项 $x_1x_2x_3$ 作为错误引入, 将其取值由“0”变为“1”, 可以将函数进一步简化为: $f_{app_TB}(x_1, x_2, x_3) = \overline{x_1}x_3 + x_1x_3$. f_{app_TB} 即为 f 的一个近似 TB 逻辑函数. 在图 2(b)中, 引起错误输出的最小项只有 $x_1x_2x_3$, 因此, 可以计算得到错误率为 $1/8$.

在 RM 逻辑中, “异或”运算是基本运算, 由于存在 $1 \oplus 1 = 0$, 因此在 RM 函数中添加或删除偶数个相同的乘积项后, 其逻辑功能将保持不变. 在

图 2(c)中, 标有“T”的最小项被偶数次覆盖, 考虑到 $1 \oplus 1 = 0$, 因此用 RM 逻辑表示图 2(c)的表达式为 $f_{app_RM}(x_1, x_2, x_3) = x_1 \oplus x_3$. 在图 2(c)中, 可以计算得到错误率为 $1/8$.

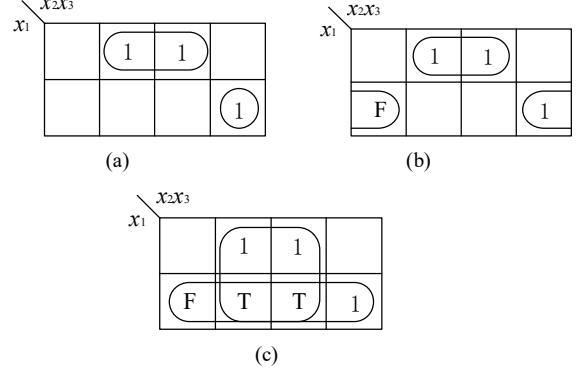


图 2 原函数及其近似逻辑函数的卡诺图表示

从图 2 的例子中不难发现, RM 逻辑与 TB 逻辑近似优化不同, 主要表现在: (1)引入取值为“0”的某些乘积项, 可简化 RM 表达式, 但不增加错误输出. (2)在错误率计算方面, 对于 TB 逻辑而言, 可以直接通过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 RM 逻辑, 近似函数的乘积项集合与原函数的乘积项集合的差异不一定是引起错误输出的乘积项的集合, 因为差异部分可能包含了偶数次逻辑覆盖, 即 RM 逻辑与 TB 逻辑的错误率计算方法不同. 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.

对于一个由 W 个乘积项组成的 RM 逻辑函数, 总可以写成式(11)形式:

$$f_{RM}(X) = \oplus_{i=0}^{W-1} p_i = f_{app_RM}(X) \oplus (\oplus_{j=0}^{v-1} p_j) = f_{app_RM}(X) \oplus f_{er}(X), \quad (11)$$

式中: f_{RM} 为原始逻辑函数; f_{app_RM} 为 f_{RM} 的近似逻辑函数; X 为输入变量集合; $\oplus_{j=0}^{v-1} p_j$ 为引起 f_{RM} 与 f_{app_RM} 输出不同的乘积项集合, 称其为错误项集合, 对应的逻辑函数为 $f_{er}(X)$, 称为错误函数.

利用式(10)求解 RM 逻辑错误率过程, 实际上是求解 $f_{er}(X) = \oplus_{j=0}^{v-1} p_j$ 包含最小项数量的过程.

比较图 2(b)和图 2(c)可知, RM 逻辑和 TB 逻辑最大的不同就在 RM 逻辑中, 被偶数次覆盖的部分

对应的是逻辑“0”,因此,必须将 $\oplus \sum_{j=0}^{v-1} p_j$ 中被偶数次覆盖部分除去后剩下的最小项就是符合式(10)中 $\sum (\hat{f}(X) \neq f(X))$ 输入组合.去除 $\oplus \sum_{j=0}^{v-1} p_j$ 中偶数次覆盖的最小项可以通过乘积项双锐积运算实现^[15].

算法1为利用双锐积运算实现逻辑覆盖中去除偶数次覆盖的伪代码,其中 C_{er} 为 $f_{er}(X) = \oplus \sum_{j=0}^{v-1} p_j$ 的乘积项集合,符号“ $\otimes$ ”表示乘积项不相交锐积运算^[16].

算法1 double_disj_sharp(C_{er}, C_{out})

{For any two RM products p_i, p_j in C_{er} ,

do { $C_1 = p_i \otimes p_j$;

$C_2 = p_j \otimes p_i$;

$C_{out} = C_1 \cup C_2 \cup \{C_{RM} - \{p_i, p_j\}\}$;

} while ($p_i \cap p_j \neq \emptyset$);}

以图3(a)为例,假设 $p_1 = x_2 x_3$, $p_2 = x_1$,由算法1可得: $C_1 = p_1 \otimes p_2 = x_1 x_2 x_3$, $C_2 = p_2 \otimes p_1 = \{x_1 x_2, x_1 x_3\}$, $C_{out} = C_1 \cup C_2 = \{x_1 x_2 x_3, x_1 x_2, x_1 x_3\}$. C_{out} 对应的卡诺图如图3(b)所示,即被 p_1 和 p_2 偶数次覆盖的最小项 $x_1 x_2 x_3$ 已经被去除.

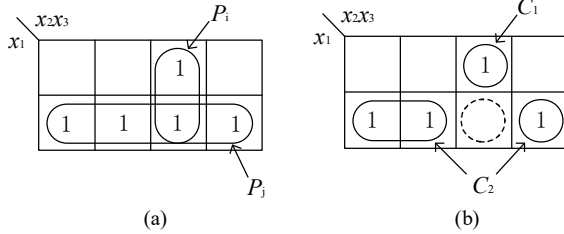


图3 利用双锐积运算去除 RM 逻辑中的偶数次逻辑覆盖

3 基于近似计算技术的 FPRM 逻辑电路功耗优化算法

基于近似计算的 FPRM 逻辑电路功耗优化算法大致可分为 2 个部分:(1)是利用遗传算法实现以功耗优化为导向的 RM 逻辑的极性搜索,得到功耗优化下的 FPRM 表达式;(2)引入近似计算技术,在错误率约束下,通过删除 FPRM 函数中特定的乘积项实现功耗的进一步优化.

在利用遗传算法实现有利于 RM 逻辑电路功耗优化的极性搜索中,染色体长度为变量个数,染色体中取值为“1”的表示对应的变量取反变量形式,“0”的表示取原变量.每个电路的输入信号的概率和跳变密度均为随机产生一个(0,1)之间的数.

在给定信号概率和跳变密度下,利用式(5)~(7)计算某个染色体 G_i 对应极性的 FPRM 函数的功耗 $P_{Power}(G_i)$,将其作为适应度函数,即

$$F_{fitness} = 1 / P_{Power}(G_i). \quad (12)$$

算法2为基于近似计算技术的 FPRM 逻辑电路功耗优化代码,其中为了方便实现 TB 逻辑向 FPRM 逻辑转化,先将测试电路表示为不相交乘积项之“或”形式,然后利用基于不相交乘积项列表技术^[17]实现 TB 逻辑向 FPRM 逻辑转化.

算法2 FPRM_pwr_opt_ap(C_{dis})

{ $C_{RM_pwr} = GA_pwr(C_{dis})$; //step 1

sort_pwr_pdct(C_{RM_pwr}); //step 2

do { $C_{er} = remove_pdct(C_{RM_pwr})$; //step 3

double_disj_sharp(C_{er}, C_{out}); //step 4

$ER = M[C_{out}] / 2^n$; //step 5

} while ($ER \leq ER_0$); }

算法2中 step 1 用遗传算法实现以功耗优化为导向的 FPRM 函数极性搜索,得到 FPRM 函数的乘积项构成集合 C_{RM_pwr} ; step 2 用式(5)计算 C_{RM_pwr} 中各 RM 乘积项的跳变密度,并按照跳变密度进行降序排序; step 3 按照乘积项的跳变密度大小从大到小试探性地逐个删除,并将删除的乘积项放到 C_{er} ; step 4 利用双锐积运算去除 C_{er} 中乘积项之间偶数次覆盖,并将不存在偶数次覆盖的乘积项存储在 C_{out} ; step 5 中 $M[C_{out}]$ 表示乘积项集合 C_{out} 包含的最小项的数量.由于双锐积后的乘积项均为不相交乘积项,因此 $M[C_{out}]$ 大小等于每个不相交乘积项包含的最小项数量的和.另外, n 为函数的输入变量数.

在执行 step 3 到 step 5 过程中,如果移除某一乘积项 p_i 后,发现 $ER > ER_0$,则程序会放弃删除,转而试探性往下删除其他乘积项,并判断是否符合错误率约束条件.循环执行 step 3 到 step 5,直到无法满足 $ER \leq ER_0$, ER_0 为设定的错误率.

4 实验结果和分析

本文提出的 RM 逻辑功耗优化算法在 win 7 64 位操作系统 4 GB 内存环境下运行,用 C 语言编程实现,采用 MCNC Benchmark 电路进行验证.在多输出逻辑函数处理时,常用的处理方法是将多输出函数转化为单输出函数,然后按照单输出函数

进行处理. 本文只有输出相等的乘积项才进行比较处理. 文中遗传算法种群最大个体为 30, 最大迭代数为 50, 染色体交叉率为 55%, 变异率为 16%. 电路的功耗用电路内部节点总的跳变密度表示. 实验中每个电路输入信号的概率与跳变密度随机产生, 并将近似计算的错误率阈值设置为不大于 5%.

表 1 为实验结果, 其中 “i/o/p” 分别对应测试电路的输入数、输出数和乘积项数; P_{pr} 表示功耗减少百分比, 运算时间单位为秒:

$$P_{pr} = \frac{P_{Power}(No_AC) - P_{Power}(With_AC)}{P_{Power}(No_AC)} \times 100\%, \quad (13)$$

式中: $P_{Power}(No_AC)$ 和 $P_{Power}(With_AC)$ 分别表示在相同输入信号下原电路和经近似计算优化后电路总的信号跳变密度.

表 1 实验结果

电路名称	i/o/p	错误率/%	功耗减少/%	时间/s
5xp1	7/10/80	3.13	13.96	1.18
f51m	8/8/80	3.91	25.97	1.57
newtag	8/1/14	3.13	22.15	0.16
clip	9/5/212	3.91	9.26	11.20
life	9/1/140	2.73	14.15	7.41
sym10	10/1/837	2.44	56.29	25.15
alu2	10/6/257	3.61	22.78	17.28
cu	14/11/22	3.13	24.91	0.67
pm1	16/13/37	3.13	7.48	0.59
pcle	19/9/45	1.56	50.31	0.69
cc	21/20/53	3.91	6.63	0.93
mux	21/1/18	3.91	17.81	0.52
examples	85/66/369	3.22	24.33	48.01
平均		3.21	22.77	8.87

从表 1 可知, 在错误率阈值 5% 的前提下, RM 逻辑的功耗优化明显. 如电路 sym10 和 pcle 等电路功耗少了 56.29% 和 50.31%, 优化效果明显. 但也有部分电路, 如 pm1 和 cc 电路功耗减少不明显, 分别减少了 7.48% 和 6.63%, 其主要原因是在采用 FPRM 形式下, pm1 和 cc 的乘积项之间相互交叠比较复杂, 删除少量的乘积项就会带来较大的错误率, 导致在误差阈值内优化的功耗有限. 反之, 在 FPRM 形式下, 电路 sym10 和 pcle 存在部分乘积项,

这些乘积项与其他乘积项交叠程度不大, 而对应的信号密度比较大. 因此, 通过删除这些乘积项, 在减少总的信号跳变密度的同时又能较好地控制错误率的增加, 从而得到较好的优化效果.

本文提出的算法可以处理输入为 85 个变量的大电路, 其关键原因在于无论是 RM 逻辑的极性搜索还是 RM 函数的错误率求解, 均基于乘积项, 而不是最小项. 因此, 待处理电路的乘积项的数量会影响算法的运行时间. 此外, 考虑到逻辑函数的乘积项数量与输入变量的数量无关, 所以本文算法可以实现大函数的 FPRM 的功耗优化.

5 结论

本文提出了一种基于近似计算技术的 RM 逻辑功耗优化算法, 该算法利用遗传算法, 结合提出的以信号概率和跳变密度为参数的 RM 函数动态功耗数学模型和以功耗优化为导向的极性搜索, 得到功耗优化后的 FPRM 表达式, 然后利用提出的基于双锐积运算的 RM 函数的错误率计算方法, 通过有选择性地删除某些 FPRM 乘积项, 实现在错误率约束下的 FPRM 逻辑的功耗优化. 提出的优化算法用 MCNC 电路进行了测试, 在平均错误率约束为 3.21% 时, 测试电路功耗的平均功耗减少了 22.77%.

参考文献:

- [1] Shin D, Gupta S K. Approximate logic synthesis for error tolerant applications[C]//Design, Automation & Test in Europe Conference & Exhibition, 2010:957-960.
- [2] Zou C, Qian W, Han J. DPALS: A dynamic programming-based algorithm for two-level approximate logic synthesis[C]//IEEE 11th International Conference on ASIC, 2015:1-4.
- [3] Ichihara H, Inaoka T, Iwagaki T, et al. Logic simplification by minterm complement for error tolerant application[C]//33rd IEEE International Conference on Computer Design, 2015:94-100.
- [4] Miao J, Gerstlauer A, Orshansky M. Multi-level approximate logic synthesis under general error constraints[C]//IEEE ACM International Conference on Computer-Aided Design, 2014:504-510.
- [5] Wu Y, Qian W K. An efficient method for multi-level approximate logic synthesis under error rate constraint

- [C]//Proceedings of the 53rd Annual Design Automation Conference. New York, USA: ACM Press, 2016:1-6.
- [6] Bernasconi A, Ciriani V. 2-spp approximate synthesis for error tolerant applications[C]//17th Euromicro Conference on Digital System Design, 2014:411-418.
- [7] Soeken M, Große D, Chandrasekharan A, et al. BDD minimization for approximate computing[C]//21st Asia and South Pacific Design Automation Conference, 2016: 474-479.
- [8] Mozammel H A K. Design of reversible synchronous sequential circuits using pseudo Reed-Muller expressions [J]. IEEE Transactions on Very Large Scale Integration Systems, 2014, 22(11):2278-2286.
- [9] 尹浩凯, 马雪娇, 夏银水. 基于近似逻辑的不完全指定 FPRM 函数的功耗优化[J]. 宁波大学学报(理工版), 2019, 32(3):28-34.
- [10] 俞海珍, 汪鹏君, 张会红, 等. 基于三值多样性粒子群算法的 MPRM 电路综合优化[J]. 电子学报, 2017, 45(7): 1601-1607.
- [11] Wang X, Lu Y, Zhang Y, et al. Probabilistic modeling during power estimation for mixed polarity Reed-Muller logic circuits[C]//Green Computing & Communications, IEEE, 2013:1414-1418.
- [12] 瞿婷, 王伦耀, 罗文强, 等. 大函数 ISFPRM 面积优化方法[J]. 电子学报, 2018, 46(5):1101-1106.
- [13] 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.
- [14] Miao J, Gerstlauer A, Orshansky M. Approximate logic synthesis under general error magnitude and frequency constraints[C]//Proceedings of the International Conference on Computer-Aided Design, 2013:779-786.
- [15] 王伦耀, 夏银水, 陈偕雄. 逻辑函数的双逻辑综合与优化[J]. 计算机辅助设计与图形学学报, 2012, 24(7): 961-967.
- [16] 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.
- [17] 王玉花, 王伦耀, 夏银水. 基于不相交项并行列表技术的 FPRM 实现[J]. 电子与信息学报, 2014, 36(9):2258-2264.

Power optimization for FPRM logic using approximate computing technique

WANG Yichen, WANG Lunyao*, XIA Yinshui, CHU Zhufei

(Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China)

Abstract: A power optimization algorithm based on approximate computing technique is proposed for FPRM logic circuits. The algorithm includes FPRM logic circuits dynamic power estimation model based on signal probability and transition density, a genetic algorithm for RM logic power optimization using polarity searching, and the error rate calculation for RM logic using double sharp product operation. Under the constraint of error rate, some product terms are selectively deleted to reduce power consumption. The proposed algorithm is implemented in C programming language and tested under MCNC benchmarks. The experimental results show that using the approximate computing technique presented in this work, the average dynamic power can be reduced by 22.77% with the average error rate of 3.21%.

Key words: approximate computation; error rate; polarity searching; power consumption optimization

(责任编辑 史小丽)