

基于虚拟单元映射的电路面积优化算法

罗文强, 王伦耀*, 岑旭梦, 夏银水

(宁波大学 信息科学与工程学院, 浙江 宁波 315211)

摘要: 相比较于标准单元库映射, library-free 映射中采用的动态生成的虚拟单元, 具有更大的灵活性, 因而在电路面积优化上往往可以取得更好的效果. 本文提出的基于虚拟单元的面积优化算法包括面积估算和覆盖策略两部分. 在面积估算上提出了基于 n 元树和逻辑努力的虚拟单元的 CMOS 电路实现及面积估算方法. 在覆盖策略上提出了一种带有局部寻优策略的离散粒子群优化算法. MCNC 测试电路的实验结果表明, 相比较于标准单元库映射, 对于绝大多数电路, 本文给出的算法可以使电路面积得到进一步的优化.

关键词: 逻辑优化; library-free; 映射算法; 逻辑努力

中图分类号: TP391.41

文献标志码: A

文章编号: 1001-5132 (2018) 01-0052-07

Library-free 映射相对于标准单元库映射, 它在电路的面积、延时等性能优化上更具优势^[1], 因此它具有很大的研究价值. 目前人们对 library-free 映射的研究主要有两方面: (1)虚拟单元电路优化和性能评估^[2-4]; (2)覆盖策略^[5-7]. 对于覆盖策略, 文献[5]提出了一种类似蚕食的方法, 该方法将晶体管的栈长作为约束项, 迭代地将满足约束的树从原网络中分离, 进而得到映射后的网络. 文献[6]提出了一种称为波阵面映射的方法, 该方法先将电路由输入到输出分成多个层级, 然后根据信号流向, 从电路的输入端到输出端逐层级依次求取局部最优覆盖, 进而实现电路全局优化. 文献[7]提出了一种递归树遍历的方法, 该方法将整个电路的最优覆盖视作多个子电路的最优覆盖之和, 该方法的效率取决于子电路的划分方式.

本文提出一种基于虚拟单元映射的电路面积优化算法, 主要包含面积估算和覆盖策略两方面. 其中在面积估算上利用提出的虚拟单元的 n 元树 (n -ary tree) 结构得到对应的 CMOS 电路结构, 然后结合逻辑努力^[8]来实现 CMOS 电路的面积估算. 在覆盖策略上采用基于离散粒子群优化 (Discrete Particle Swarm Optimization, DPSO) 算法^[9]对电路

网表实现基于虚拟单元映射的覆盖. 实验结果表明, DPSO 算法能有效地缓解由于解空间的急剧增大所带来的求解困难, 且求得的面积最优解相较于基于标准单元库的面积最优映射有明显的改善.

1 虚拟单元的面积估算

在 library-free 映射中, 所使用到的门单元都是动态生成的虚拟单元, 因此, 虚拟单元的面积评估方法将直接影响整个电路面积的优化结果. 本节将给出一种基于逻辑努力的 CMOS 电路面积估算模型, 以及虚拟单元的 n 元树结构面积估算的流程, 并以此作为本文的面积最优映射算法的基础.

1.1 基于逻辑努力的面积估算模型

由文献[8]可知, 一个逻辑门的逻辑努力定义为这个门的输入电容与一个能提供与该门相同输出电流的反相器的输入电容之比. 在 CMOS 电路中, 考虑到逻辑门的输入电容与输入端相连接的 MOS 管的栅极面积成正比, 故可以用逻辑努力来估算逻辑门 G 的面积 S_G ^[7]. 即

$$S_G = \sum_i g_i, \quad (1)$$

式中 g_i 表示第 i 个输入的逻辑努力.

收稿日期: 2017-05-18.

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

基金项目: 国家自然科学基金 (61471211, 61571248, 61501268).

第一作者: 罗文强 (1991-), 男, 福建浦城人, 在读硕士研究生, 主要研究方向: 逻辑综合与优化. E-mail: iroche@126.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

1.2 n 元树的虚拟单元面积估算

Library-free 映射过程中, 子电路都是动态生成. 每一个子电路的功能都可以表示成由“与/或/非”基本逻辑运算组合而成的树状结构, 本文称之为 n 元树. 由 n 元树可以得到相应逻辑功能的虚拟单元的 CMOS 电路结构以及对应的输入电容, 并由此得到虚拟单元电路的面积.

假设图 1(a)是 library-free 映射算法过程中动态生成的一个 4 输入子电路对应的 n 元树 T . 在 T 中, 方块表示 T 的根, 对应于子电路的输出; 三角形为 T 的叶子, 对应于子电路的输入; 圆圈为 T 的内部结点, 对应于子电路的逻辑运算; 虚边表示互补, 即包含一个反相器. 由布尔函数的德·摩根定律可知, “与/或”形式的逻辑函数的取反等于输入变量取反, “与”运算同“或”运算互换. 因此, T 的补就是将 T 中的“与”结点变成“或”结点, “或”结点变成“与”结点, 同时, 叶子的输出取反. 图 1(b)是图 1(a)的补网络 T' , 即 $F' = \overline{F}$.

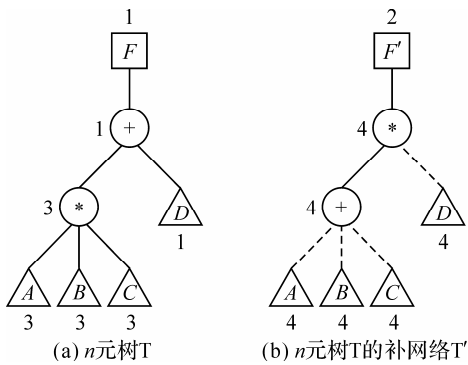


图1 n 元树计算输入电容

CMOS 电路由 pMOS 管构成的上拉网络和由 nMOS 管构成的下拉网络连接而成. 上拉网络导通时 CMOS 电路输出高电平, 而下拉网络导通时输出为低电平. n 元树 T 的结构与实现它所代表的逻辑功能的 CMOS 电路结构存在着对应关系. 其对应规则如下:

(1)若 T 对应 CMOS 电路的上拉网络, 则它的补网络 T' 对应下拉网络, 反之也成立;

(2) T 中的每个叶子对应 CMOS 电路中都连接一个 MOS 管, 其中上拉网络为 pMOS, 下拉网络为 nMOS. 同“与”结点的各叶子连接的 MOS 管为串联关系, 同“或”结点的各叶子连接的 MOS 管为并联关系. 同时 T' 也满足该规则;

(3)下拉网络的所有输入作为整个电路的输入, 且整个 CMOS 电路对应的逻辑功能与下拉网络对应的 n 元树的功能互补.

以图 1(a)为例, 若它对应上拉网络, 则图 1(b)对应的是下拉网络. 由上面的对应规则可知, 上拉网络中与输入 A, B, C 相连的 3 个 pMOS 管串联, 然后再与输入 D 相连的 pMOS 管并联. 同理, 由图 1(b)可得, 下拉网络中与输入 $\overline{A}, \overline{B}, \overline{C}$ 相连的 3 个 nMOS 管并联, 然后再与输入 $\overline{D}$ 相连的 nMOS 管串联. 由于下拉网络的输入均取反, 则最终得到图 2(a)所示电路中的所有输入均连接了反相器. 由于图 2(a)的下拉网络为图 1(b), 所以图 2(a)的 CMOS 电路的逻辑功能为 $F = \overline{F'} = (\overline{A} + \overline{B} + \overline{C})\overline{D}$, 即为 4 输入“或-与-非”门, 简称 OAI31 复合门. 同理, 若图 1(b)对应上拉网络, 则图 1(a)对应下拉网络, 对应的 CMOS 电路如图 2(b)所示, 它实现了 AOI31 复合门功能.

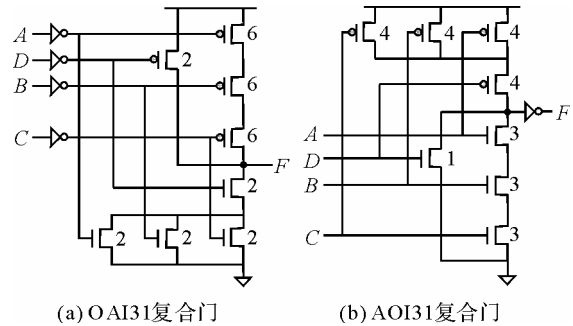


图2 n 元树生成的 CMOS 电路

由上述对应规则可知, n 元树的叶子都连接了一个 MOS 管, 因此, 图 1 中叶子上的数字表示与叶子相连的 MOS 管的输入电容. 在本文中, 由 n 元树得到的 CMOS 电路中 MOS 管宽度设置上要求新生成的电路可以提供与反相器一样的输出电流, 即电路的面积可以由式(1)来描述. 图 1 中 n 元树从根到叶子上的输入电容的计算方法如下:

(1)对于下拉网络, n 元树的根结点初始值为 1; 而上拉网络的根结点初始值为 2.

(2)由根结点向叶子逐级递推计算, 对于 n 元树中的“与”结点, 若它输入个数为 k , 且它的上一级结点的输入电容为 γ' , 则该“与”结点的输入电容 $\gamma = k\gamma'$; 对于 n 元树中的“或”结点, 它的输入电容等于上一级结点的输入电容.

(3)叶子的输入电容等于与该叶子相连的结点

的输入电容.

从上面的计算方法可知, 非叶子上的数字是一个中间计算量, 并不反映电路的面积.

图 1(a)给出了按照上述计算方法得到的构成下拉网络的各 nMOS 管的输入电容, 图 1(b)给出了构成上拉网络的各 pMOS 管的输入电容, 对应的 CMOS 电路及其 MOS 管的输入电容如图 2(b)所示.

利用 n 元树从根到叶子上的输入电容的计算方法, 可以得到与 n 元树对应的 CMOS 电路的面积, 其计算方法如下: 设 n 元树 T 有 α 个叶子, 其中有 $\beta, \beta \leq \alpha$ 个叶子的输出边为虚边; γ_i 表示 T 中第 i 个叶子的输入电容, T' 为 T 的补网络, γ'_j 为 T' 中第 j 个叶子的输入电容; 用 C_n 表示 T 作为下拉网络, T' 作为上拉网络构成的 CMOS 电路的输入电容之和; 而用 C_p 表示 T' 作为下拉网络, T 作为上拉网络构成的 CMOS 电路的输入电容之和, 则 C_n 和 C_p 的计算公式如下:

$$C_n = \sum_{i=1}^{\alpha} \gamma_i + \sum_{j=1}^{\alpha} \gamma'_j + 3(\beta + 1), \quad (2)$$

$$C_p = 2 \sum_{i=1}^{\alpha} \gamma_i + \frac{1}{2} \sum_{j=1}^{\alpha} \gamma'_j + 3(\alpha - \beta), \quad (3)$$

$$\Delta C = C_n - C_p = \frac{1}{2} \sum_{j=1}^{\alpha} \gamma'_j - \sum_{i=1}^{\alpha} \gamma_i + 3(2\beta - \alpha + 1). \quad (4)$$

利用式(2)、式(3)和图 1 可以计算出图 2(a)和图 2(b)中电路的输入电容之和分别为 40 和 29 个单位, 这与直接按照图 2(a)和图 2(b)中的电路计算出来的结果一致, 其中反相器为 3 个单位输入电容^[8]. 另外, 图 2(a)和图 2(b)所示的电路功能一样, 但从式(4)可知, 二者输入电容之和相差 11 个单位, 由此可知选择合适的上拉或下拉网络, 可以实现电路面积的进一步优化.

Algorithm 1 为由 n 元树估算对应虚拟单元面积的伪代码, 其中子函数 CalcLeafCapa()按照本小节提出的方法计算 n 元树的每个叶子的输入电容 γ , 子函数 CalcByEqua4()按照式(4)计算 ΔC .

Algorithm 1 Calculate the area of virtual cell

Input: N -ary tree (T_n)

Output: Area of virtual cell (A)

Function: CalcCellArea

```

 $\gamma_n \leftarrow \text{CalcLeafCapa}(T_n)$ 
 $\Delta C \leftarrow \text{CalcByEqua4}(\gamma_n)$ 
if  $\Delta C \leq 0$  then
     $C \leftarrow \text{CalcByEqua2}(\gamma_n)$ 
else
     $C \leftarrow \text{CalcByEqua3}(\gamma_n)$ 
end if
 $A \leftarrow \text{CalcByEqua1}(C)$ 

```

2 基于 DPSO 算法的覆盖策略

基于虚拟单元的映射, 其最大的优势是不受单元库的约束、解空间更大. 但伴随着解空间的增加, 算法求解时间也快速增加. 在大空间搜索上遗传算法是一种常用的方法, 相对于遗传算法, DPSO 算法的参数更少、实现更容易, 收敛速度较快且往往具有较好的结果^[9]. DPSO 算法常应用于超大规模集成电路的划分^[9], 神经网络训练^[10]等. 本节将给出一种基于 DPSO 算法的覆盖策略, 用于基于虚拟单元映射的电路面积优化.

2.1 问题描述

一个由组合逻辑门和时序器件组成的电路网表可以用有向无环图(Directed Acyclic Graph, DAG)来表示. 定义 $G(V, E)$ 为一个 DAG, 其中结点集合 V 包含电路的原始输入输出、内部组合逻辑门和时序器件; 有向边集合 E 包含所有结点之间的连线. 而面积最优覆盖就是将 V 中全部结点划分到 m 个子集中, 记作 $V_1, V_2, \dots, V_m$; 其划分目标就是全部子集的面积之和最少, 并且 V 中的每个结点有且仅被划分到一个子集中和全部子集均满足无输出锥集(Free Fanout Cone, FFC)^[11]的约束. 即

$$\min S(V) = \sum_{i=1}^m S(V_i), \quad (5)$$

$$\text{s.t.} \begin{cases} \bigcup_{i=1}^m V_i = V \ (V_i \in C_{\text{FFC}}) \\ V_i \cap V_j = \emptyset \ (i \neq j) \end{cases}, \quad (6)$$

式(5)中 $S(V)$ 表示集合 V 的面积; 式(6)中符号“ $\cup$ ”表示并集, $V_i \in C_{\text{FFC}}$ 表示集合 V_i 满足 FFC 约束.

2.2 编码方式

事先按照深度优先搜索(Depth First Search, DFS)的顺序对电路网表 DAG 中的每个结点进行编号, 如图 3 所示; 而满足约束条件的每一个粒子即

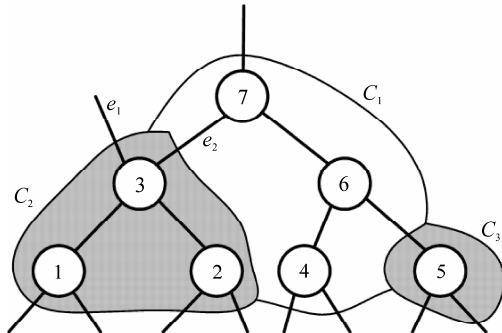
代表面积最优覆盖的一个可行解。

采用 0-1 二进制的编码方式, 将粒子 i 在 t 时刻的位置 X_i^t 表示成一个 w 维的 0-1 排列串, $X_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{iw}^t)$. 其中 x_{iw}^t 表示粒子 i 在 t 时刻第 w 个结点位的取值, $x_{iw}^t = 1$ 代表以结点 w 为根结点实现当前编码条件下的最大无输出锥集(Maximum Free Fanout Cone, MFFC)^[11].

图 3(a)是粒子位置的 7 维 0-1 二进制编码, 图 3(b)是粒子 1 生成的覆盖. 其中粒子 1 的位置编码结点位 3,5,7 分别为 1, 相应地分别生成以结点 3,5,7 为根结点的 3 个 MFFC, 即包含结点 1,2,3 的 C_2 , 包含结点 4,6,7 的 C_1 以及包含结点 5 的 C_3 . 粒子 2 的位置编码只有结点位 7 为 1, 因此, 对应一个以结点 7 为根结点, 包含其他所有结点的一个覆盖. 但从图 3(b)可知, 结点 3 有 e_1, e_2 二条输出边, 导致由粒子 2 生成的覆盖有 2 个输出, 分别是 e_1 和结点 7 的输出, 从而违反 FFC 约束. 而粒子 3 对应的覆盖不包含结点 7, 违反了全部结点有且仅被覆盖一次的约束. 从上面例子可以看出, 为保证每个粒子均不会违反约束, 至少多输出结点和原始输出结点的位编码应该为 1. 因此, 事先生成一个位值预置编码 X_c , X_c 在多输出结点和原始输出结点的位编码为 1, 其他位编码为 0, 然后对粒子的位置 X 与 X_c 进行位“或”运算即可保证新生成的位置 X 满足约束条件.

DAG结点	1	2	3	4	5	6	7
粒子1	0	0	1	0	1	0	1
粒子2	0	0	0	0	0	0	1
粒子3	0	0	1	0	0	1	0

(a)粒子的0-1二进制编码



(b)粒子1生成的覆盖

图 3 粒子的位置编码

2.3 适应值函数

本文的求解目标是面积最小覆盖, 因此将覆盖的面积作为适应值, 而适应值评估函数的定义如下所示:

$$f = S(X_i^t) = \sum_{j=1}^z s_j^t, \quad (7)$$

式中 $S(X_i^t)$ 表示位置 X_i 在 t 时刻生成覆盖的面积; s_j^t 表示在 t 时刻第 j 个 MFFC 的面积, 它按照第 1.2 小节的方法估算, 共有 z 个满足约束条件的 MFFC; 评估函数 f 越小, 覆盖面积也就越小. 对于违反约束的位置 X , 对它的惩罚措施是将其的适应值设为一个较大的值.

2.4 局部寻优策略

为了克服 DPSO 算法局部搜索能力不足和早熟收敛的缺陷, 本文引入了动态局部寻优策略. 即在 DPSO 算法运行中发现种群历史最优位置连续 N 代不更新, 则对种群历史最优位置做 DFS 局部搜索, 动态地更新种群历史最优位置, 克服由于位置固定和局部搜索能力不足带来的早熟收敛缺陷, 使算法能得到一个更优质的解.

其中 DFS 局部搜索是一种试探法算法思想, 它首先对种群历史最优位置 X' 的第 1 个结点位 x_1 取反, 得到新的位置 $X(\bar{x}_1)$, 若新位置 $X(\bar{x}_1)$ 的适应值更优则令 $X' = X(\bar{x}_1)$, 否则不更新 X' ; 之后依次对 X' 中的结点位 $x_2, x_3, \dots, x_w$ 做同样的操作. 例如图 3(a)中的粒子 1, 假设它是种群历史最优位置 X' , 对其结点位 x_1 取反可得新位置 $X(\bar{x}_1) = 1010101$, 假设新位置 $X(\bar{x}_1)$ 的适应值更优, 则令 $X' = X(\bar{x}_1) = 1010101$; 之后对 X' 的结点位 x_2 取反得到新位置 $X(\bar{x}_2) = 1110101$, 假设新位置 $X(\bar{x}_2)$ 的适应值不比 X' 优则继续操作 X' 的下一个结点位, 以此类推直至遍历所有结点位.

2.5 算法实现

Algorithm 2 为基于 DPSO 算法思想的面积最优覆盖算法的伪代码, 子函数 ReadAndOptiCirc() 依次读取、优化输入电路并输出 DAG 电路网表; 子函数 MovInvToLeaf() 按照德·摩根定律, 尽可能地将 DAG 中的虚边自根结点推至叶子^[12]; 子函数 DfsLocalOpti() 按照第 2.4 小节提出的方法进行局部寻优; 子函数 InitPop() 和 UpdtPop() 分别初始化种群和更新种群, 二者均调用 Algorithm 1 中的函

数 CalcCellArea() 计算适应值.

经过子函数 MovInvToLeaf() 操作, 得到的 DAG 只可能存在多输出结点和叶子的输出边为虚边, 这可以避免大部分相邻 MFFC 之间的反相器归属问题, 一定程度上降低了算法的复杂度; 对于多输出结点和叶子的虚边, 本文的处理方式是将反相器划归给上一级 MFFC 的输入, 例如图 3(b), 假设结点 5 的输出边为虚边, 则将该虚边的反相器划归给 C_1 中的结点 6.

Algorithm 2 Covering algorithm using DPSO

Input: Input circuit (IC)

Output: Global best position ($gBest$)

Function: CoveAlgoByDpso

$DAG \leftarrow \text{ReadAndOptiCirc}(IC)$

$DAG \leftarrow \text{MovInvToLeaf}(DAG)$

for $i = 1$ to Ng **then**

$\{V_i, X_i\} \leftarrow \text{InitPop}(DAG, \text{CalcCellArea})$

$pBest_i \leftarrow X_i$

end for

$gBest = \min\{pBest_i\}$

while $j \leq N + K$ **then**

for $i = 1$ to Ng **then**

$\{V_i, X_i\} \leftarrow$

$\text{UpdPop}(V_i, X_i, DAG, \text{CalcCellArea})$

if $\text{fitness}(X_i) < \text{fitness}(pBest_i)$ **then**

$pBest_i \leftarrow X_i$

if $\text{fitness}(pBest_i) < \text{fitness}(gBest)$ **then**

$gBest \leftarrow pBest_i$

$j \leftarrow 0$

end if

end if

end for

$j \leftarrow j + 1$

if $j \geq N$ **then**

$gBest \leftarrow \text{DfsLocalOpti}(gBest)$

end if

end while

3 实验结果与分析

本文的算法在伯克利大学的逻辑综合工具 ABC^[13] 的开源代码上进行 C 语言编程实现, ABC

的版本号 1.01; 相关软硬件环境为 Win7 x64/CPU i5@3.30 GHz/RAM 8.00 GB. 算法的相关参数设置如下: 惯性权重为 0.9, 学习因子 c_1 和 c_2 均为 0.6; 种群规模 Ng 为 20; 种群历史最优位置保持代数 N 为 10, K 为 5.

表 1 给出的是利用 MCNC 标准电路进行测试得到的数据. 表 1 中 $i/o/w/v$ 依次是电路的原始输入、原始输出、内部结点和自由结点的数量, 自由结点是指位值预置编码 X_c 中位编码为 0 的结点, 如图 3(b) 中的结点 1,2,4,6; ABC/SCs 是指运行 ABC 中的标准单元库映射“map -a”命令得到的结果. 为了使 ABC/SCs 得到的电路面积的结果与本文算法得到的电路面积的结果具有可比性, 在实验中, 均采用逻辑努力的方法对电路面积进行估算; LFs 是指本文提出的基于虚拟单元映射的电路面积优化算法得到的结果. 由于实验结果与算法的粒子初始位置有关, 故本文采用多次平均的方法来衡量所提出算法的性能, 即表 1 中的面积 A_2 和求解时间 T_2 均为程序运行百次得到的平均值; 求解时间 “<1” 表示它小于 1 ms; 面积优化是指 LFs 比 ABC/SCs 得到的面积减少的百分比. 即

$$\text{面积优化} = \frac{A_1 - A_2}{A_1} \times 100\%. \quad (8)$$

从表 1 中的数据可以看出, LFs 的求解时间 T_2 与自由结点数 v 相关, 而与电路的原始输入、输出没有直接的关联; 相比较于 ABC/SCs, LFs 的求解时间在大电路 *cht*、*soar* 和 *x3* 上会多些, 这主要是因为解的搜索空间更大, 所以需要花费更多的时间. 由本文 2.2 小节的编码方式可知, 解空间的上限为 2^w , 满足约束条件的可行解有 2^v 种; 但表 1 中的大部分求解时间 T_2 均为 “<1”, 这也验证了本文所构造的算法的高效性.

同时从表 1 中可以看出, 相比较于 ABC/SCs, 本文所构造的 LFs 对大部分电路的解都有一定的改善, 这源于 LFs 的虚拟单元是按需所生而不受限制于标准单元库, 它能选取更合适的虚拟单元来降低电路面积. 即使某些电路的面积没有得到优化, 但两种方法的求解时间也相当.

对于电路 *dec*, 它的自由结点数 $v=0$, 所以只有一种可行解. 同时 ABC/SCs 和典型的基于 LUT 的 FPGA 映射算法很相似^[13], 它和 LFs 一样都得满足

表 1 测试数据

电路	$i/o/w/v$	ABC/SCs		LFs		面积优化/%
		面积 A_1	求解时间 $T1/ms$	面积 A_2	求解时间 $T2/ms$	
<i>al2</i>	16/47/98/16	288.33	<1	259.21	<1	10.10
<i>alcom</i>	15/38/78/11	232.33	<1	207.00	<1	10.90
<i>alu1</i>	12/8/30/22	83.33	<1	77.85	<1	6.58
<i>b11</i>	8/31/69/22	200.67	<1	181.48	<1	9.56
<i>cc</i>	21/20/42/25	123.33	<1	109.72	<1	11.04
<i>cht</i>	47/36/148/111	442.67	<1	384.80	15	13.07
<i>dec</i>	8/256/304/0	810.67	<1	810.67	<1	0.00
<i>decoder</i>	4/7/34/20	92.33	<1	89.07	<1	3.53
<i>dk27</i>	9/9/57/32	158.67	<1	148.20	<1	6.60
<i>i1</i>	25/16/32/16	90.33	<1	83.54	<1	7.52
<i>i5</i>	133/66/132/66	352.00	<1	352.00	<1	0.00
<i>m1</i>	6/12/70/44	189.33	<1	182.82	<1	3.44
<i>misex2</i>	25/18/91/53	250.33	<1	234.95	<1	6.14
<i>newapla1</i>	12/7/21/7	60.33	<1	54.67	<1	9.38
<i>newapla2</i>	6/7/16/3	47.00	<1	41.67	<1	11.34
<i>risc</i>	8/31/70/21	204.00	<1	184.33	<1	9.64
<i>soar</i>	83/94/506/282	1 389.33	16	1 326.42	70	4.53
<i>wim</i>	4/7/26/11	74.00	<1	69.00	<1	6.76
<i>x3</i>	135/99/603/453	1 725.00	31	1 562.30	120	9.43

FFC 的约束, 因此两种方法得到的面积值相等; 对于电路 *i5*, 它的自由结点数 $v=66$, 但面积优化 0%, 这是因为 *i5* 的自由结点分布比较松散, 生成不了较大的 MFCC, 所以发挥不出虚拟单元的优势。

在图 4 的图例中, DPSO 表示基本离散粒子群优化算法, LDPSO 表示本文中带有局部寻优策略的算法。从图 4 中可以看出 LDPSO 算法的收敛速度和精度明显好于 DPSO 算法, 从而验证了本文所

构造的带有局部寻优策略的电路面积最优映射算法的有效性。

4 结论

本文主要包含两部分内容, 即面积估算和覆盖策略。在面积估算上, 提出了基于 n 元树的虚拟单元逻辑功能的 CMOS 电路实现方法, 并进一步提出结合逻辑努力, 用 n 元树实现虚拟单元的面积

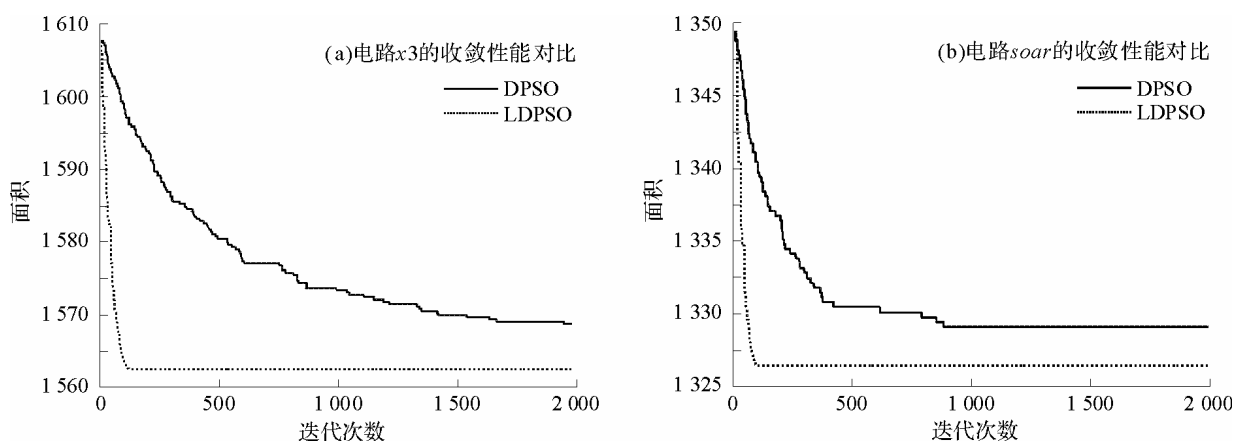


图 4 算法收敛性

估算方法. 由于逻辑函数转换为对应的 n 元树和 n 元树求补过程都很容易实现, 故有利于算法效率的提高. 在覆盖策略上, 提出了一种带有局部寻优策略的 DPSO 算法, 该算法在解空间增大的情况下仍有较高的求解效率. 在本文中, 由于虚拟单元是在面积优化下, 按照映射需要而生成相应的单元电路, 因此, 相比较于标准单元库映射, 本文的算法在多数情况下电路面积均获得了进一步优化.

参考文献:

- [1] 岑旭梦, 王伦耀, 夏银水. 基于逻辑复合门映射的电路面积优化[J]. 宁波大学学报(理工版), 2016, 29(4):38-43.
- [2] El-Masry H, Al-Khalili D. Cell stack length using an enhanced logical effort model for a library-free paradigm [EB/OL]. [2016-04-05]. <http://ieeexplore.ieee.org/document/6122371/>.
- [3] Possani V N, Callegaro V, Reis A I, et al. Graph-based transistor network generation method for supergate design[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016, 24(2):692-705.
- [4] Conceicao C, Posser G, Reis R. Reducing the number of transistors with gate clustering[EB/OL]. [2016-12-28]. <http://ieeexplore.ieee.org/document/7451035/>.
- [5] Reis A I. Covering strategies for library free technology mapping[EB/OL]. [2015-09-16]. <http://ieeexplore.ieee.org/document/803115/>.
- [6] Marques F S, Rosa Jr L S, Ribas R P, et al. DAG based library-free technology mapping[EB/OL]. [2016-08-21]. <https://dl.acm.org/citation.cfm?doid=1228784.1228857>.
- [7] Pullerits M, Kabbani A. Area minimization for library-free synthesis[EB/OL]. [2016-02-07]. <http://ieeexplore.ieee.org/document/5393800/>.
- [8] Sutherland I, Sproull B, Harris D. Logical Effort: Designing Fast CMOS Circuits[M]. Burlington: Morgan Kaufmann Publishers Inc., 1999.
- [9] 郭文忠, 陈国龙, 彭少君, 等. 求解 VLSI 电路划分问题的混合粒子群优化算法[J]. 软件学报, 2011, 22(5): 833-842.
- [10] 刘红, 曹颖, 隆腾舞. 基于改进离散粒子群优化算法的容差电路故障特征提取[J]. 兵工学报, 2015, 36(8): 1494-1501.
- [11] Vemuri N, Kalla P, Tessier R. BDD-based logic synthesis for LUT-based FPGAs[J]. ACM Transactions on Design Automation of Electronic Systems, 2002, 7(4): 501-525.
- [12] Jain A, Bryant R E. Inverter minimization in multi-level logic networks[EB/OL]. [2015-10-19]. <http://ieeexplore.ieee.org/document/580098/>.
- [13] Berkeley Logic Synthesis and Verification Group. ABC: System for sequential logic synthesis and formal verification[EB/OL]. [2016-10-08]. <https://bitbucket.org/alanmi/abc>.

Area optimization using virtual cells mapping

LUO Wen-qiang, WANG Lun-yao*, CEN Xu-meng, XIA Yin-shui

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

Abstract: Compared to the standard cells mapping, the cell in library-free mapping is virtual, and generated dynamically. The flexibility of virtual cell offers a way for area optimization in the course of mapping. The area optimization in this paper includes two parts: area estimation and covering algorithm. The area estimation consists of the n -ary tree based virtual cell generation and the logic effort based cell area estimation. In covering algorithm, a discrete particle swarm optimization algorithm with local search strategy is presented for global area optimization. The proposed area optimization algorithm is tested under MCNC (Microelectronics Center of North Carolina) benchmarks. The experimental results suggest that compared with the standard cells mapping, for the most of circuits, the proposed algorithm can further optimize the circuit area.

Key words: logical optimization; library-free; mapping algorithms; logical effort

(责任编辑 韩 超)