

基于性能相似度的 Buffer 库压缩算法

苏晓东, 王静秋, 马 鸿

(中国科学院自动化研究所 国家专用集成电路设计技术工程研究中心, 北京 100190)

摘 要: 以原库与压缩库的性能相似度为出发点, 提出了一种 buffer 库压缩算法, 并建立了虚单元、实单元和概率加权距离的概念. 用环境参数对原库进行筛选, 其结果构成虚单元库; 对虚单元进行聚类, 将中心点映射到实单元得到压缩库. 将单元对环境的适应性量化, 作为先验知识, 为聚类中心的选择提供优先级. 采用 3 种方案实现该算法思想, 经实验证明, 所得压缩库与原库的性能相似度高, 误差平方和 (SSE) 仅为已有算法的 9.6%、10.4% 和 6%.

关键词: buffer 库压缩; 性能相似度; 虚单元; 聚类; 加权距离

中图分类号: TP391.72

文献标识码: A

文章编号: 1000-7180(2011)02-0037-06

Buffer Library Compression Based on Performance Similarity

SU Xiao-dong, WANG Jing-qi, MA Hong

(National ASIC Design Engineering Center, Institute of Automation,
Chinese Academy of Science, Beijing 100190, China)

Abstract: Taking performance similarity as a point of departure, this paper proposes an algorithm for buffer library compression. Concepts of virtual cell, real cell and probability weighted distance are established. Buffers with best performance under certain environment points are identified as cells of virtual library. Virtual cells are clustered in the second step. Center of each cluster is finally mapped to compressed library. Environmental adaptability of each buffer is quantized to give priority to cluster center identification. Above idea is achieved by using three programs. Result shows that three implementations achieve 9.6%, 10.4% and 6% of SSE respectively compared with existed algorithm, which represents better performance similarities between compressed library and the unpruned one.

Key words: buffer library compression; performance similarity; virtual cell; cluster; weighted distance

1 引言

随着 IC 芯片集成度的提高, 全局互连线的电容值和延时不断增加, 互连线的优化变得日益重要. 导线的延时与线长的平方成正比, 若将长导线分割成若干段较短的导线, 则能够有效地降低导线的总延时. 在各种技术中, 插入 buffer 常被认为是减少长导线传播延时最有效的方法^[1-2]. 它将互连线缩短使传播延时以平方关系减小, 这在导线足够长时足以弥补插入 buffer 带来的额外延时^[3]. 因此, 许多 EDA 工具都将 buffer 插入作为重要功能之一.

文献[4]提出了 buffer 插入的动态规划算法,

以降低树形 RC 网络的 Elmore 延时, 文献[5]提出导线的分段算法, 能够在根导线上插入多个 buffer. 文献[6]将可选 buffer 的种类扩展到 buffer 库, 文献[7]用聚类的方法对 Buffer 库进行压缩. 研究该算法, 我们发现压缩结果严重向大单元倾斜, 在对长导线进行解耦时 (此时, 小尺寸 buffer 更常用), 相对于原库, 压缩库具有较大的性能损失. 分析发现, 原算法还存在一些其他问题.

本文从不同的角度出发, 建立了虚单元、实单元和概率加权距离的概念, 并提出了新的 buffer 库压缩算法. 实验证明, 得到的压缩库在各种环境下, 表现出与原库更加相似的性能.

2 基本思想

本文基于如下定义、假设与基本思想:

环境:单元延时除了取决于自身参数外,还受负载和前级驱动的影响.将前级驱动电路等效为驱动电阻 R_{driver} ,将负载等效为负载电容 C_{load} .定义 $(C_{\text{load}}, R_{\text{driver}})$ 为单元工作的“环境”.

延时模型:本文使用的等效模型如图 1 所示^[7].Alpert 给出了导线中插入 buffer 的延时模型^[5],我们考虑了以该模型代替图 1,更精确地模拟 buffer 的实际工作环境.研究发现,延时的闭合表达式包含较多不确定参数,使问题产生不必要的复杂化.以单位长度导线电容为例,随着集成度的提高,它不仅取决于导线的线宽、所在金属层,还更多地受到与相邻导线间耦合电容的影响.尝试采用 Synopsys 的线性公式计算图 1 所示模型的传播延时,但在当前工艺下精确度较低.我们通过调用当前工艺的 SPICE 模型,可以迅速得到精确的延时.

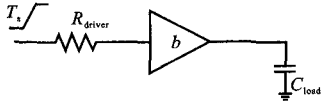


图 1 单元的等效延时模型

虚(实)单元(库):设单元 b 在环境 $(C_{\text{load}}, R_{\text{driver}})$ 下具有延时 delay ,定义单元 b 为“实单元”,实单元的集合称为“实单元库”;定义四元组 $(b, C_{\text{load}}, R_{\text{driver}}, \text{delay})$ 为“虚单元”,虚单元的集合称为“虚单元库”.

设在环境 $E_1 = (C_1, R_1)$ 下,原库 A 中某单元 b_1 相对 A 中其他单元具有最小延时 d_1 ,定义 d_1 为 A 在 E_1 环境下的最小延时.设在 E_1 附近 $E_2 = (C_2, R_2)$ 处,压缩库 B 中某单元 b_2 (且 $b_2 \in A$) 相对 A 中其他单元具有最小延时 d_2 ,则我们说 B 在 E_2 环境下可得到最小延时.假设单元的延时是环境 (C, R) 的连续函数,则认为在 E_1 环境下 B 可得到近似最小延时,此时用 B 代替 A 产生的误差可由虚单元 $c_1 = (b_1, C_1, R_1, d_1)$ 和 $c_2 = (b_2, C_2, R_2, d_2)$ 间的欧氏距离表示,如式(1)所示.

$$\text{dist}(i, j) =$$

$$\sqrt{\left(\frac{C_j - C_i}{C_{\max}}\right)^2 + \left(\frac{R_j - R_i}{R_{\max}}\right)^2 + \left(\frac{D_j - D_i}{D_{\max}}\right)^2} \quad (1)$$

式中引入 $C_{\max}$ 、 $R_{\max}$ 和 $D_{\max}$ 进行归一化,保证每个分量都落在 $[0, 1]$ 范围内,避免由于某个分量数值过大而影响其他分量特性的表达.

理想的压缩库在各种环境下,都可得到最小延

时.这是不现实的,更加合理的期望是:在各种环境下,压缩库可以得到最小或近似的最小延时,且 B 代替 A 所产生的误差平方和 (SSE) 最小. SSE 反映压缩库在整个环境区域上与原库的性能相似度, SSE 越小相似度越高,若 SSE 等于 0,则表示压缩库在整个环境区域上能得到与原库同样的性能. SSE 的定义如式(2)所示,其中 $F(A)$ 为原库在各种环境下取到最小延时的虚单元集合, $F(B)$ 为压缩库可以取到最小延时的虚单元集合.

$$\text{SSE} =$$

$$\sum_{i=1}^{|F(A)|} \left(\min_{j=1}^{|F(B)|} \{ \text{dist}(c_i, b_j) \mid b_j \in F(B), c_i \in F(A) \} \right)^2 \quad (2)$$

压缩带来性能损失, B 在某些环境下无法得到最小延时,但如果 $F(B)$ 能够均匀分布在环境范围,则 $F(A) - F(B)$ 中的虚单元可用 $F(B)$ 中与它距离最近的虚单元近似代替,由此产生的误差较小.如图 2 为例予以说明.

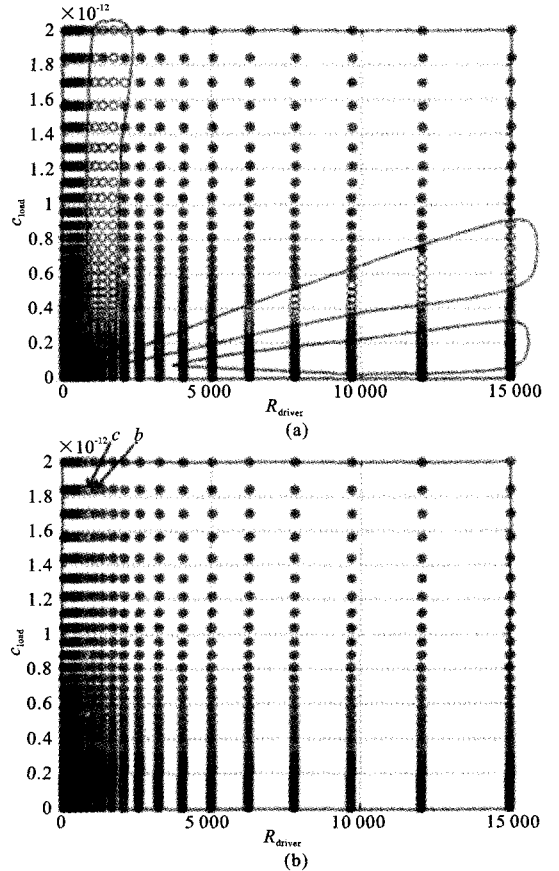


图 2 压缩库对环境的覆盖度

图 2 中,“o”为要考察的环境点,原库在此处具有最小延时,“*”表示压缩库在此处可取得最小延时.图 2(a)中,原库由 52 个具有反相输出的 buffer

构成,使用 Alpert 的 buffer 库选择算法将其压缩到 17 个 buffers. 可以看到,至少在 3 个环境区域中,压缩库无法得到最小延时. 使用压缩库代替原库,将在这 3 个区域产生较大的误差. 图 2(b)中,使用改进算法将原库压缩到 17 个 buffer, $F(B)$ 均匀地分布在环境范围内. 虽然在个别环境点处(如箭头所指 c 点)压缩库无法得到最小延时,但这些点并未集中于同一区域,在附近总能找到属于 $F(B)$ 中的点(如箭头所指 b 点),因此压缩库与原库的 SSE 较小,性能相似度较高.

我们还注意到,部分单元对环境的适应性强,在许多环境点处都具有最小延时. 聚类时,这部分单元应当被优先选作类的中心. 我们用概率加权距离来实现这种优先关系,加权距离由式(3)求得.

$$P_{\text{dist}}(i, j) = -\log\{P[f(c_j)]\} \times \text{dist}(i, j) \quad (3)$$
 式中, $P[f(c_j)]$ 表示 c_j 对应的实单元 $f(c_j)$ 在初筛过程中的命中率. 假设待考察的环境点有 n 个,若 $f(c_j)$ 在其中 m 个点处具有最小延时,则 $P[f(c_j)] = 100\% \times m/n$. 注意,由于 $P_{\text{dist}}(i, j) \neq P_{\text{dist}}(j, i)$, 概率加权距离只能被用来确定类的中心,而不能用来确定类的划分.

如图 3 所示说明概率加权距离的作用. 用启发式 K-Center 算法^[7](不使用概率加权距离)对 15 个虚单元聚类,得到结果如图 3(a)所示,样本点被分为 3 类,聚类中心映射为 3 个实单元(b_1, b_2, b_3).

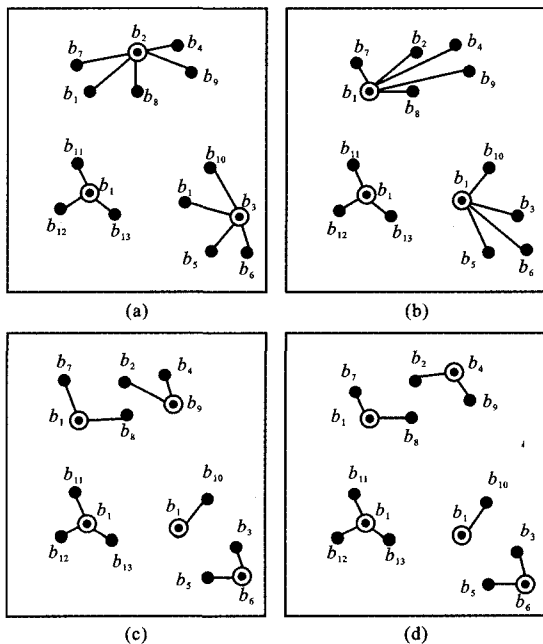


图 3 概率加权距离示意图

引入概率加权距离后,用同样的算法对这 15 个样本点进行聚类,如图 3(b)、(d)所示. 在图 3(b)中 b_1

与最远点 b_9 距离较大,但 b_1 被命中了 3 次,因此概率加权距离 P_{dist} 仅为欧氏距离 dist 的 0.7 倍;而 b_2 仅命中 1 次, P_{dist} 为 dist 的 1.18 倍. 比较 P_{dist} 后发现,选 b_1 作为类的中心误差更小. 图 3(c)在图 3(b)的基础上进行两次迭代,半径较大的两个类被分拆. 使用 P_{dist} 重新确定图 3(c)中每个类的中心,得到结果如图 3(d)所示. 在图 3(d)中,虚单元分为 5 类,聚类中心映射为 3 个实单元(b_1, b_4, b_6)(实单元 b_1 对应 3 个虚单元),明显比图 3(a)具有更小的 SSE.

3 基于性能相似度的 Buffer 库压缩算法

基于以上思想,本节提出 buffer 库压缩算法:首先遍历各种环境,找出当前环境下原库 A 的最小延时与对应单元,构成虚单元库 S ; 然后对 S 聚类,并将聚类中心映射到实单元库 B .

3.1 初筛选算法

初筛选算法如下:

Step 1: $S_0 = S_1 = \emptyset$; Hit=0;

Step 2: For $C_{\text{load}} = C_{\text{min}} \cdots C_{\text{max}}$ do

For $R_{\text{driver}} = R_{\text{min}} \cdots R_{\text{max}}$ do

Step3: $[b, \text{delay}] = \text{FindBufferWithMinDelay}(A, C_{\text{load}}, R_{\text{driver}}, 0)$; (反相 buffer)

$S_0 = S_0 \cup \{(b, C_{\text{load}}, R_{\text{driver}}, \text{delay})\}$;

Hit[b]=Hit[b]+1;

$[b, \text{delay}] = \text{FindBufferWithMinDelay}(A, C_{\text{load}}, R_{\text{driver}}, 1)$; (正相 buffer)

$S_1 = S_1 \cup \{(b, C_{\text{load}}, R_{\text{driver}}, \text{delay})\}$

Hit[b]=Hit[b]+1;

Step4: 返回 S_0, S_1, Hit

Step2~Step3, 遍历各种环境. ($C_{\text{load}}, R_{\text{driver}}$) 的范围可根据工艺确定,例如,在 65 nm 工艺下我们取 C_{load} 范围为 0.55 fF~2 pF, R_{driver} 范围为 80 Ω ~15 k Ω . 分别将 C_{load} 和 R_{driver} 的取值范围离散化为 100 级和 25 级的等比数列,构成 2 500 个环境点. Step3, 找出当前环境点下 A 中具有最小延时的单元 b , 将虚单元 $c = (b, C_{\text{load}}, R_{\text{driver}}, \text{delay})$ 加入 S 中,并更新单元 b 的命中次数 Hit[b]. 注意,我们将具有正相输出的单元与反相输出的单元分开处理. Step4, 完成迭代过程后,返回 S (由 S_0 和 S_1 组成) 和 Hit. S_0 和 S_1 中各有 2 500 个虚单元,它们记录 2 500 个环境点下 A 的最小延时及对应单元.

3.2 聚类压缩算法

改进启发式 K-Center 算法,使其适用于虚实单元相互映射的情况,如下:

Step1: 找出第 1 个聚类中心 c_1 , 它与 S 中离它最远的

样本点距离最小. 所有样本点以 $\text{CurCntr}(1) = c_1$ 为中心构成 1 个类; $\text{Flag} = 0$;

Step2: While($\text{Flag} \leq \text{Convits}$) {

Step2.1 If($|f(\text{CurCntr})| \leq K$) {

Center = CurCntr ;

$B = f(\text{CurCntr})$; $\text{Flag} = 0$;

} Else { $\text{Flag} = \text{Flag} + 1$; }

Step2.2 在 $S\text{-CurCntr}$ 中找出新的聚类中心 c_i , 它与原有最近的中心点距离最大, $\text{CurCntr} = \text{CurCntr} \cup \{c_i\}$;

Step2.3 为 CurCntr 中每个点 c_i 建立一个类 $C_i = \{c_i\}$, 将 $S\text{-CurCntr}$ 中的样本各自加到最近的类中

Step2.4 重新确定每个类的中心, 找到 $c_j \in C_i$ 使 $\max\{P_{\text{dist}}(c, c_j) | c \in C_i\}$ 最小;

$\text{CurCntr} =$ 重新确定的中心

}

Step3: 返回 B

该算法仅是基本思想的一种实现方法, 后面将会看到其它算法也可实现该思想.

Step1, 根据最大距离最小原则选择第 1 个中心样本. 使用最大最小原则消除了原算法中随机初始点对聚类结果的影响. 我们尝试随机选择初始中心, 在 4000 次运行过程中没有得到更小的 SSE.

Step2, 进行若干次迭代. Step2.2, 每次迭代都使用最小距离最大原则, 从非中心样本中找出新的聚类中心. Step2.3, 将非中心样本加到最近的类中. Step2.4, 使用概率加权距离, 根据最大距离最小原则, 重新确定每个类的中心. 注意, 在重新确定聚类中心时, 可能出现压缩库 $f(\text{CurCntr})$ 规模小幅振荡的现象. 例如, 100 个聚类中心映射为 10 个实单元; 经过一次迭代后, 所有样本被划分为 101 个类, 并重新确定每个类的中心, 可能出现 101 个聚类中心映射为 9 个实单元的现象. Step2.1, 若在连续 Convits 次迭代(我们取 20)中 $|f(\text{CurCntr})|$ 均超过 K , 则认为压缩库的规模不会再振荡到 K 以下, 压缩结果收敛到最后一次 $|f(\text{CurCntr})| \leq K$ 时状态. 我们试图用局部 SSE 最小代替最大距离最小原则, 实验证明前者将产生更大的全局 SSE.

4 实验分析与比较

本节对一个由反相输出单元构成的 buffer 库进行实验分析. 原库由 INV 和 BUF3 组成, 共 52 个单元, 其中 BUF3 是三级反相器链. 根据 Rabaey 的分析^[1], 反相器链在驱动大负载时具有相对最小延时, 而单级反相器在驱动小负载和解耦长导线时具有相对最小延时. 因此优秀的压缩结果应该包含

INV 中的小单元和 BUF3 中的大单元, 且与原库性能相似度较高. 经过初筛选, 虚单元库 S 可映射为 36 个实单元.

我们实现了上文提出的算法. 作为比较, 也实现了 Alpert 的算法^[7], 并使用 APC 聚类算法^[8]实现本文的基本思想. 由于 APC 可以对具有非对称相似度的样本进行聚类, 我们分别构造对称相似度矩阵 S_{dist} 和非对称相似度矩阵 AS_{dist} , 如式(4)、(5)所示.

$$S_{\text{dist}}(i, j) = -\text{dist}(i, j) \quad (4)$$

$$AS_{\text{dist}}(i, j) = \max\{S_{\text{dist}}\{i, F[f(c_j)]\}\} \quad (5)$$

$S_{\text{dist}}(i, j)$ 构成对称阵, 其值为负的欧氏距离. $AS_{\text{dist}}(i, j) \neq AS_{\text{dist}}(j, i)$, 它构成非对称阵. 式(5)中, $f(c_j)$ 是虚单元 c_j 所对应的实单元, $F[f(c_j)]$ 为该实单元在虚单元库 S 中的所有映射. 可以理解为, $F[f(c_j)]$ 是 S 中与四元组 c_j 第 1 个分量相同的虚单元的集合. $AS_{\text{dist}}(i, j)$ 等于 c_i 与 $F[f(c_j)]$ 的最大相似度, 这表示 $f(c_j)$ 在 S 中的所有映射点到 c_i 的最近距离. 对角线元素 $S_{\text{dist}}(k, k)$ 和 $AS_{\text{dist}}(k, k)$ 表示对样本 k 的偏好, 数值大的优先选作聚类中心, 同时也被用来控制压缩库的规模. 我们构造对角线元素如式(6)所示.

$$S_{\text{dist}}(k, k) = AS_{\text{dist}}(k, k) \quad (6)$$

式中, $P[f(c_k)]$ 表示 c_k 对应的实单元在初筛选时被命中的概率. P_{max} 和 P_{min} 分别为取值上下界, 用于控制压缩库的规模. 调整取值的分布, 使其尽量覆盖整个 $[P_{\text{min}}, P_{\text{max}}]$ 的范围. 例如, 选取 $P_{\text{min}} = -0.75$ (对称阵为 -0.7), $P_{\text{max}} = 0$, $\alpha = 6$, 我们得到 $K = 17$ 的压缩库.

比较各类算法的结果如表 1 所示. 其中, Alpert 算法从 buffers 间特征参数的相似度出发对 buffer 库进行压缩, 而其他 3 种是本文基本思想的 3 种实

表 1 各算法压缩结果比较

	K	SSE	选中的单元
Alpert 算法	17	1.365 1	INV: 1. 7/3. 7/6. 7/10. 6/13. 8/18/23/
			30/39/51
			BUF3: 8. 2/13. 8/23/30/39/51/51_S
本算法	17	0.130 7	INV: 1/1. 3/1. 7/2. 9/3. 7/6. 3/8. 2/
			10. 6/13. 8/51
			BUF3: 10. 6/18/23/30/39/51/51_S
APC-S (S_{dist})	17	0.142 4	INV: 1/1. 7/2. 2/2. 9/3. 7/4. 8/6. 3/
			10. 6/13. 8/51
			BUF3: 13. 8/18/23/30/39/51/51_S
APC-AS (AS_{dist})	17	0.081 2	INV: 1/1. 7/2. 9/3. 7/6. 3/10. 6/13. 8/
			18/30/51
			BUF3: 10. 6/18/23/30/39/51/51_S

现. 可以看到, Alpert 算法的 SSE 最大, INV 中尺寸小于 10 单元有 9 个, 但仅有 3 个被选中, 大尺寸的 INV 几乎全部被选中, 与本节开始时的预期不符, 在低负载区产生了较大误差, 如图 4(a) 所示. APC-S 使用对称相似度矩阵 S_{dist} , SSE 等于 0.142 4, 仅为 Alpert 算法的 10.4%, INV 中有 7 个小单元被选中, 符合预期, 误差分布如图 4(b) 所示. APC-AS 使用非对称相似度矩阵 AS_{dist} , 在 4 种算法中 SSE 最小, 仅为 Alpert 算法

的 6%, 但 INV 中仅 5 个小单元被选中, 误差分布结果如图 4(c) 所示. 产生该现象的原因是, 初筛选时环境参数 R_{driver} 为等比数列, 取值更集中于低电阻区, 此时大单元具有更好的性能, 导致小单元被舍弃, 读者可对实际芯片中 R_{driver} 的分布情况进行统计, 使结果更加合理. 本算法的 SSE 为 0.130 7, 介于 APC-S 和 APC-AS 之间, 是 Alpert 算法的 9.6%, INV 中有 7 个小单元被选中, 符合预期.

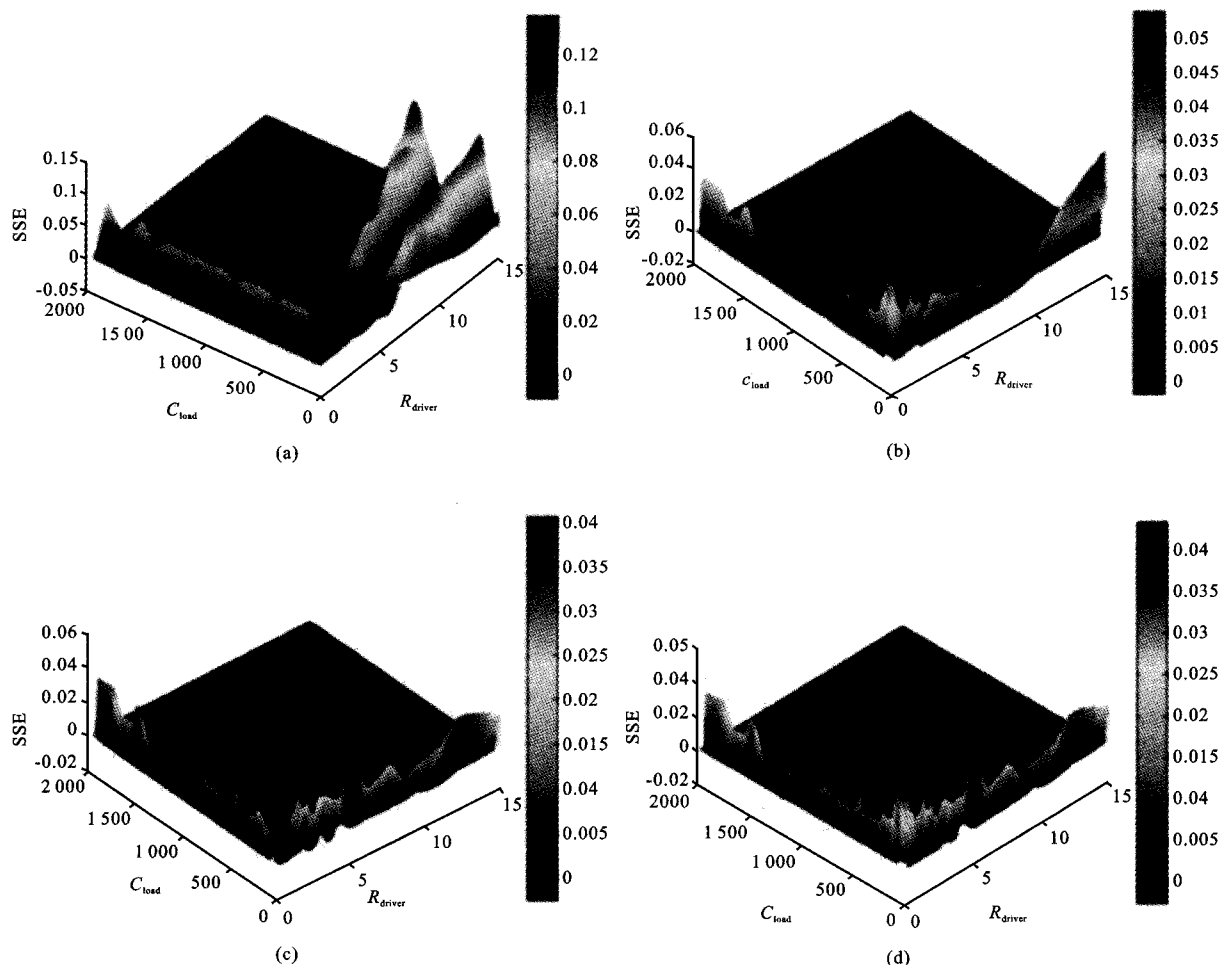


图4 各类算法误差分布

5 结束语

本文提出了 buffer 库压缩的基本思想. 我们分别用 K-Center 的改进算法、APC-S 算法和 APC-AS 算法来实现基本思想, 在相同压缩率下, 压缩库与原库的误差平方和仅为 Alpert 算法的 9.6%、10.4% 和 6%, 同时消除了原算法中初始中心的选择对压缩结果的影响. 在三种实现中, APC-AS 的压缩结果与原库具有最高的性能相似度, 但 APC 算法无法指定压缩率, 需要反复调整 $P_{\min}$ 、 $P_{\max}$ 和参数才能得到所需规模的压缩库; 此外, APC-AS 对环境参数敏

感, 需要大量关于环境的统计数据, 才能得到合理的压缩结果. 改进型 K-Center 算法能够快速 (通常在 1 分钟内) 产生指定规模的压缩库, 压缩结果合理, 与原库的性能相似度介于 APC-S 和 APC-AS 之间. 实验证明, 基于本文的基本思想, 在高性能的应用中使用 APC-AS 算法, 而在普通应用中使用改进型 K-Center 算法可以得到与原库具有相似性能的压缩库.

参考文献:

- [1] 孔昕, 吴武臣, 侯立刚, 等. VLSI 互联线的延时优化研

- 究[J]. 微电子学与计算机, 2010, 27(4): 66—74.
- [2] 刘颖, 翁健杰, 戎蒙恬. 用改进激活集合法优化 VLSI 互连线[J]. 微电子学与计算机, 2004, 21(12): 203—207.
- [3] Cong J, Pan D Z. Interconnect estimation and planning for deep submicron designs[C]// Proceedings of the 36th ACM/IEEE Design Automation Conference. New Orleans, LA: ACM/IEEE, 1999: 507—510.
- [4] L P P P van Ginneken. Buffer placement in distributed RC-tree networks for minimal elmore delay[C]// IEEE International Symposium on Circuits and Systems. New Orleans: IEEE, 1990: 865—868.
- [5] Alpert C J, Devgan A. Wire segmenting for improved buffer insertion[C]// Proceedings of the 34th IEEE/ACM Design Automation Conference, New York: IEEE, 1997: 588—593.
- [6] Lillis J, Cheng C K, Lin T T Y. Simultaneous routing and buffer insertion for low power and a generalized delay model[J]. IEEE Journal of Solid-State Circuits, 1996, 31(3): 437—447.
- [7] Alpert C J, Gandham R G, Neves J L, et al. Buffer library selection[C]// Proceedings of the 18th IEEE International Conference on Computer Design. Austin: IEEE, 2000: 221—226.
- [8] Brendan J Frey, Delbert Dueck. Clustering by passing messages between data points[J]. Science, 2007, 315(5814): 972—976.

作者简介:

苏晓东 男, (1984—), 硕士研究生. 研究方向为高性能标准单元库设计.

王静秋 女, (1979—), 硕士, 助理研究员. 研究方向为单元库设计.

马 鸿 女, (1981—), 博士. 研究方向为 VLSI 设计自动化.

(上接第 36 页)

- [3] 李楠, 喻明艳. 16×16 快速乘法器的设计与实现[J]. 微电子学与计算机, 2008, 25(4): 156—159.
- [4] 李军强, 李东生, 李奕磊, 等. 32×32 高速乘法器的设计与实现[J]. 微电子学与计算机, 2009, 26(12): 23—26.
- [5] 陈博亚. 大邻域图像处理系统的研制[D]. 北京: 清华大学, 2006.
- [6] Lyons M J, Budynek J, Akamatsu S. Automatic classification of single facial images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999, 21(12): 1357—1362.
- [7] Yan Xiang, Su Guangda. Multi-parts and multifeature fusion in face verification[C]// Proceeding of IEEE CVPR Workshop on Biometrics. Anchorage: IEEE, 2008: 1—6.
- [8] 苏光大. 图像并行处理技术[M]. 北京: 清华大学出版社, 2002.

作者简介:

丁 亮 男, (1985—), 硕士研究生. 研究方向为数字集成电路设计.

苏光大 男, (1948—), 教授. 研究方向为图像处理系统人脸识别.