

基于改进布谷鸟搜索的测试用例生成算法研究

何海鲜,冯 静

(武汉理工大学 计算机科学与技术学院,湖北 武汉 430070)

摘 要: 测试用例的自动生成是实现测试自动化技术的核心环节,为了提高自动化生成测试用例的效率,提出了一种基于布谷鸟搜索的改进算法. 该算法采用混沌立方映射初始化种群来增加种群的多样性,并对局部最优值引入高斯扰动增加种群变化活力,加快收敛速度. 另外,该算法还以分支函数插桩的方式构造适应度函数,以加快测试数据的优化. 通过三个不同的开源测试程序,和基本布谷鸟算法、基本遗传算法进行实验分析. 最终实验说明,该算法在自动化生成测试用例的效果和效率两方面均优于其他两种算法.

关键词: 软件测试;布谷鸟算法;混沌映射;高斯扰动;测试用例生成

中图分类号: TP311

文献标识码: A

文章编号: 1000—7180(2018)03—0004—05

Research on Test Data Generation Based on Improved Cuckoo Search

HE Hai-xian, FENG Jing

(Computer science and technology of Wuhan University of Technology, Wuhan 430070, China)

Abstract: Automatic test data generation is a key link in the process of test automatic technology. In order to improve the efficiency of testing automation, a new algorithm was proposed to improve the traditional cuckoo search. Chaotic sequence generated by cube map was used to initiate individual position, which strengthened the diversity of global searching. And a Gaussian disturbance would be given on the local optimum of each generation, which could increase the vitality of population change and accelerate the convergence rate. At the same time, the algorithm is also designed to fit the fitness function to accelerate the optimization process of the data. Through the triangle, mid and equal program, the basic cuckoo search and the basic genetic algorithm were compared. Experiments result shows that the new algorithm has obvious advantages in efficiency and effectiveness compared with the other two for test case generation.

Key words: software testing; cuckoo search; chaos mapping; gaussian disturbance; test case generation

1 引言

随着科技的突飞猛进的发展,软件系统早已成为社会的一部分,遍布在我们生活的方方面面. 到目前为止,保障软件系统质量和可靠性的最有效的方式仍然为软件测试^[1]. 其中,测试用例生成的效率决定了软件测试的整体效率,对整个软件测试工程的发展具有重要的影响^[2].

在软件测试的发展历程中,涌现出了很多的测

试用例生成方法,如数据流准则法^[3],状态模型法^[4],进化优化法^[5]. 在这些方法中,进化优化法的使用最频繁,它的核心是启发式搜索算法. 具体是将测试数据的优劣表达为其在搜索算法中的适应度,把数据生成问题转变为函数的最优值问题. Le 提出了一种基于遗传算法和模拟退火的 Simulink 模型测试数据自动生成方法^[6],该方法已经应用于一些不同的案例研究,得到了非常理想的结果. Sengupta 提出了一种基于粒子群算法的测试用例优化技

术^[7],并采用基于灵敏度的方法来纠正设计错误,显著降低计算复杂度的情况下,获得了更好的精度. Rodriguez将蚁群算法用于测试用例的生成问题^[8],使用贪心算法把用例生成问题转化成一种覆盖问题.这些算法为测试数据的生成问题提供了新的想法.

Cuckoo Search(CS)算法^[9]是受到布谷鸟寄生育雏行为和 Levy flights 机制的启发,提出的一种模仿生物繁殖的智能优化方法.由于其参数较少,便于扩展,同时算法的全局搜索能力强,一经提出便受到广大学者的欢迎,将其运用到工程领域的各个方面,并且取得了很好的效果.但同时算法也存在着迭代后期的收敛速度较慢、搜索精度不高等不足.因此本文引入混沌立方映射从数学角度初始化种群,并对局部最优值引入高斯扰动,提出了改进的布谷鸟算法,增加了种群变化活力,加快收敛速度.最后在具体的实验分析中证明了改进后的算法能够有效性并且高效性的应用于测试数据的自动生成过程中.

2 布谷鸟搜索算法

在自然界中,有些鸟类自己不筑巢、不孵卵、不育雏,而是依赖其他鸟类完成繁殖后代的任务,布谷鸟就是其中典型的代表,这种行为被称为巢寄生.布谷鸟每次会在别的鸟类的巢里产下一枚自己的蛋,让其帮助它抚育.巢主人有一定的概率发现这些外来鸟蛋,一旦发现,可以选择将其扔掉,或者离开自己的鸟巢在其他地方重建新巢.布谷鸟算法根据这种巢寄生策略,采用 Levy flights 更新鸟巢位置,同时设定以下三个规则^[9]:

(1)布谷鸟每一次只生下一枚自己的蛋,然后随便寻找一个寄生巢存放;

(2)在其随便寻找的寄生鸟巢中,最优秀的一组会保留至下一代;

(3)寄生鸟巢的多少不变的,并且一个外来鸟蛋被巢主人发现的几率为 Pa .

在以上三个规则的前提下,该算法采用 Levy flights 随机游走更新鸟巢位置,具体如公式(1)所示:

$$x_i^{(t+1)} = x_i^{(t)} + \partial \oplus L(\lambda); i = 1, 2, \dots, n \quad (1)$$

式中, x_i^t 代表第 i 个鸟巢在第 t 代的位置; $\oplus$ 表示点对点乘法; ∂ 表示步长控制量; $L(\lambda)$ 表示随机的搜索的路径,并且随机的步长符合 Levy 分布,即 $L \sim u = t^{-\lambda}$, ($1 < \lambda \leq 3$). 通过位置更新后,用随机数 $r \in$

$[0, 1]$ 与 Pa 进行对比,若 $r > Pa$, 则对 $x_i^{(t+1)}$ 进行随机的改变,反之保持不变.通过鸟巢的适应度值判断是否应该保留,最后剩下一组适应度值较高的仍记为 $x_i^{(t+1)}$.

3 改进的布谷鸟算法

目前,在基本的布谷鸟算法中,几乎全部采取随机初始化来产生鸟巢的初始位置,这种随机性可能造成鸟巢分布位置不均匀.针对混沌运动的规律性、随机性等特点,将混沌序列运用到初始化鸟巢位置的过程中,以达到增加鸟巢位置多样性,提高算法质量的目的.同时,对每次迭代得到的最优值根据高斯分布加入扰动算子,增加鸟巢位置的变化活力,避免搜索过程中出现局部最优.将混沌理论和高斯扰动引入到布谷鸟算法中,弥补在迭代后期出现收敛速度慢,精度低的缺点.

3.1 混沌初始化

混沌状态是自然界中大量存在的一种非线性状态,包括很多很好的特性,例如初始值敏感性、状态遍历性、混合和相似随机性等.正是由于这些优质特性,很多的学者将混沌映射用于一些算法的改进过程中.因此,本文为了增加鸟巢位置的多样性,在初始化阶段引入立方映射产生混沌序列,以得到更加均匀的鸟巢位置,为算法进行全局搜索打下良好的根基.在众多的混沌模型中,虽然目前使用率最高的是 logistic 映射,但在文献^[10]中已经证明,在均匀性方面, Cube 立方映射产生的混沌序列要比 logistic 映射产生的更好.因此,本文采用立方映射,具体如公式(2)如下:

$$\begin{aligned} X_{n+1} &= 4X_n^3 - 3X_n \\ -1 &\leq X_n \leq 1, n = 0, 1, 2, \dots \end{aligned} \quad (2)$$

式中,当立方映射的初始迭代值不为零时,系统便进入了混沌状态.

改进后布谷鸟算法的混沌初始化过程如下:

步骤一:设置最大的迭代次数为 M , 种群的规模为 N .

步骤二:随机产生 D 维向量 $X = (x_1, x_2, \dots, x_d)$, 其中 $x_i \in [-1, 1]$, $1 \leq i \leq d$, 且 $X \neq 0$ 作为初始化向量.

步骤三:将 X 通过式(2)迭代 M 次可以得到 M 个向量 $Y_i = (y_{i1}, y_{i2}, \dots, y_{id})$, $i = 1, 2, \dots, M$.

步骤四:将得到的混沌向量 Y 通过公式(3)映射到解空间:

$$x_{id} = L_d + (1 + y_{id}) \frac{U_d - L_d}{2} \quad (3)$$

式中, U_d 和 L_d 分别表示 D 维搜索空间的上下限, y_{id} 是通过步骤四得到的第 i 个鸟巢的第 d 维, x_{id} 为第 i 个鸟巢在第 d 维的坐标。

步骤五: 计算鸟巢的适应度值, 从中选出 N 个较高的作为算法的初始数据。

3.2 高斯扰动

为了避免 CS 算法在迭代过程中陷入局部最优值, 本文根据高斯分布对每一代最优个体的位置进行细微扰动, 增加鸟巢位置的变化活力。高斯扰动的表达式如(4)所示:

$$x'_t = x_t + a \oplus \varepsilon \quad (4)$$

式中, ε 是与 x_t 同阶的随机矩阵, $\varepsilon_{ij} \sim N(0, 1)$, $\oplus$ 表示点对点乘法, a 为常数, 本文取 $a = 1/3$ 来控制 ε 的搜索范围, 防止由于 ε 的随机取值范围较大, 而带动的鸟巢位置活动范围偏离过大。

对每次进化得到的最优值按式(5)进行更新:

$$x_{t+1} = \begin{cases} x'_t & f(y'_t) > f(y_t) \\ x_t & \text{其他} \end{cases} \quad (5)$$

式中, f 表示鸟巢的适应度值大小, 选择较高的一组进入下一次迭代, 算法通过高斯细微扰动可以帮助 x_t 跳出局部最优值区域, 增加布谷鸟鸟巢的变化活力。

4 测试用例生成模型

4.1 适应度函数

适应度函数是启发式搜索算法中评价解的优劣程度的标准, 在改进的 CS 搜索优化算法中是与测试用例生成问题的重要接口, 起着十分关键的作用。一般情况下, 用测试用例对程序各个分支的覆盖率作为其适应度的评价标准, 来表示该测试数据的优劣。

改进后的算法使用分支距. 离法^[11]来表达程序的适应度函数, 在被测程序的每个分支节点依次加上分支函数: $f(1), f(2), \dots, f(n)$, 形成插桩程序, 其中 n 表示被测程序的分支数, $f(i)$ 表示算法此时生成的测试数据在第 i 个分支节点的位置是否为真, 并通过公式(6)表达出两者之间的距离。分支函数的具体表达式如下:

$$f(i) = X_i - a \quad (6)$$

式中, X_i 表示此时的测试数据在第 i 个分支中的值, a 表示覆盖当前分支的值, 若 $f(i)$ 小于等于 0, 则表明测试数据覆盖了此分支, 将 $f(i)$ 设为 0, $f(i)$ 越小表示接近程度越大。这样, 在程序执行的过程中, 由分支函数的值就可以判断测试用例是否执行了所

选的目标路径。总的适应度函数 F 的表达式如下:

$$F = \frac{\sum_{i=1}^n \frac{M}{c + f(i)}}{n} \quad (7)$$

式中, $f(i)$ 为分支函数 n 表示总的分支数, M 表示适应度的上限, c 的取值为了保证 $f(i)$ 可以取到最小值。本文指定 $M = 1, c = 1$ 。

4.2 算法描述

改进的布谷鸟算法中鸟巢的选择、更新操作都是根据适应度高低决定的, 适应度高的鸟巢在迭代过程中表现出良好的优势, 而适应度较低的鸟巢经过 Levy flights 和高斯扰动等操作可以提高个体的适应度。每次迭代都会选择出适应度较高的鸟巢作为下一代的初始种群, 迭代次数通常标示着整个搜索过程的收敛状态。本文的算法以生成满足目标测试路径的测试用例为最优个体, 同时以达到迭代次数上限为算法的停止条件。

算法 1 测试用例生成算法。

begin

混沌初始化 N 个鸟巢的位置:

$X_i = (x_{i1}, x_{i2}, \dots, x_{id}), i = 1, 2, \dots, N;$

通过式(7)算出 N 个鸟巢地点的适应度值

$f(X_i), i = 1, 2, \dots, n;$

while (不满足结束条件)

通过 Levy flights 更新公式(1)产生一个新解 X_j ;

计算新解的适应度值 $f(X_j)$;

随机选择一个鸟巢 X_i

if($f(X_j) > f(X_i)$)

用新解 X_j 替代 X_i ;

end

以发现概率 Pa 去掉部分劣解并产生新解

通过式(4)和(5)对 N 个鸟巢的位置采用高斯扰动;

得到一组新的鸟巢位置

$X_i = (x_{i1}, x_{i2}, \dots, x_{id}), i = 1, 2, \dots, N;$

if($f(X_i) > f(X_i)$)

用新解 X_i 替代 X_i ;

end

对解进行排序并找到当前最佳解 X_{best} , 并记录适应度值 $f(X_{best})$;

end while

结果显示

end

5 实验分析

5.1 实验数据和参数设置

本文实验以开源程序为实验对象, 选取了 3 个

程序进行测试,来验证改进布谷鸟算法生成测试数据的优劣,并将其与基本布谷鸟算法(CS)、基本遗传算法(SGA)进行比较和分析. 开源程序的详细信息如表 1 所示.

表 1 测试程序				
程序编号	程序名称	变量个数	搜索范围	说明
1	Triangle	3	[0,100]	三角形类型判定
2	Min	4	[-10,10]	数组最小值
3	Equals	10	[-10,10]	两数组是否相等

程序 1 的目标路径为等边三角形的判定;程序 2 的目标路径数组的最小值;程序 3 的目标路径是两个数组是否相等,本实验将两个数组的长度均设为 5.

本文实验参数设置的详细信息如表 2.

表 2 实验参数设置			
参数	SGA	SC	改进的 SC
种群规模 N	10,20,30,40,50		
最大迭代次数 T	1 000	1 000	1 000
发现概率 P_a	—	0.25	0.25
交叉概率	0.85	—	—
变异概率	0.05	—	—
独立运行次数	20	20	20

为了保证实验的公平性,3 种方法在参数设置上,比如种群规模、最大迭代次数等参数值均设为相同,其他不同的参数均参考其他文献而设置. 本文使用算法的迭代次数为评价标准,每个程序均运行 20 次,若该程序在迭代次数上限 $T=1\ 000$ 之内覆盖到了目标路径,则将该程序此时的迭代次数记录下来;如果没有找到,则记录为 T . 评价标准即为这些记录的均值,即平均的迭代进化次数.

5.2 实验分析

本文选择三角形类型判定(Triangle)、数组最小值(Mid)和两数组是否相等(Equals)三个开源程序进行对比测试,比较基于改进布谷鸟算法、基本布谷鸟算法(CS)、基本遗传算法(SGA)下的测试用例生成情况.

本文的实验首先通过平均迭代次数(ANEG)进行评估,对三角形类型判定(Triangle)程序,设置种群规模为 30,分别记录搜索范围在[0,5]、[0,10]、[0,20]、[0,50]、[0,100]时达到目标路径时的迭代次数,由统计后的结果,计算出各个算法的平均的进化次数,比较三种搜索方法生成测试用例的性能. 本文通过 MATLAB 工具开展测试数据的研究,最终的效果如表 3 所示.

表 3 不同搜索范围下实验结果比较			
搜索范围	SGA	CS	改进的 CS
[0,5]	17.25	12.3	8.3
[0,10]	38.35	35.6	22.7
[0,20]	73.05	63.45	42.25
[0,50]	159.2	134.8	80.9
[0,100]	527.85	415.5	229.7

通过比较上表中三种算法在不同搜索范围下的实验结果,我们可以发现,随着搜索范围的逐步增大,三种算法生成最优测试数据的迭代次数均会逐步增加. 改进的 CS 算法迭代次数均少于基本遗传算法和 CS 算法,尤其是在搜索范围变大之后,这种优势表现的更加明显. 可以明显看出改进后的 CS 算法比其他两种基本算法在性能上提高了很多.

为了进一步验证改进算法的性能,本次分别记录三种算法在种群规模为 10~50 时,每次搜索到覆盖目标路径的测试数据,所需要的迭代次数,然后由统计的结果计算出运行 20 次的平均次数,其中三角形类型判定(Triangle)程序的搜索范围取[0,100]. 三种程序的实验效果分别如图 1 至图 3 所示.

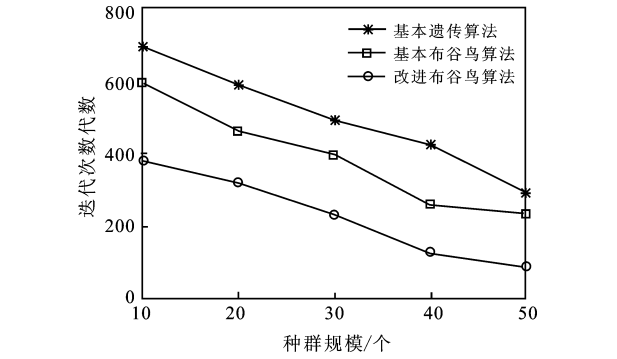


图 1 等边三角形最优个体迭代次数

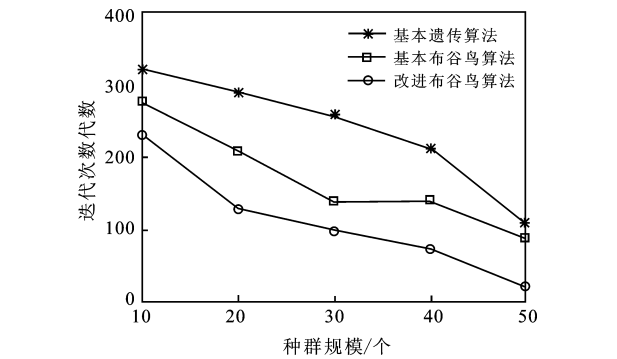


图 2 数组最小值最优个体迭代次数

通过以上三幅实验结果图可以看出,随着种群数量的逐渐增加,三种搜索算法得到最优测试用例的进化次数均会减少.

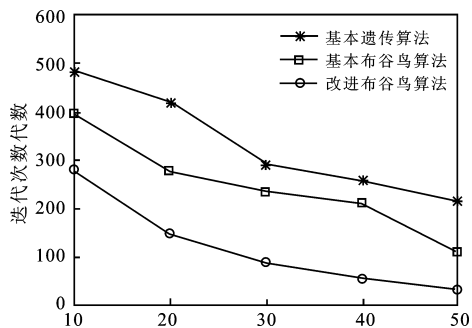


图3 两数组相等最优个体迭代次数

可以观察到,当种群规模相同时,基本 SC 算法所需要的进化次数少于基本 GA 算法,而改进后的 SC 算法明显低于另外的算法,同样生成覆盖目标路径的测试用例,经过改进后的 SC 算法在效率上要高出其他两种算法很多。当种群规模开始增大后,三种算法的迭代次数均呈下降状态,但可以看出改进后的 SC 算法的下降趋势普遍大于其他两种算法,很大程度上的增大了基本 SC 算法在迭代后期的收敛速度。除此之外,尽管种群规模在逐渐增大,但基本布谷鸟算法和基本遗传算法都或多或少的出现了,迭代次数不降反增或者变化不大的现象,即算法陷入了局部最优解,改进后的布谷鸟算法很好的避免了这一点,具有更优的迭代特性。

综上所述,改进后的布谷鸟算法从整体上加快了种群的进化过程,弥补了算法在进化中因陷入局部最优而导致的搜索停滞不前,收敛速度不高的缺点。能够快速找到覆盖源程序目标路径的最优测试数据,有效的提高了测试用例的生成效率,可以理想的运用到整个软件测试的流程中。

6 结束语

软件自动化测试的水平很大程度上取决于生成测试用例的效率,在基于搜索的用例生成技术中,即搜索算法的效率。

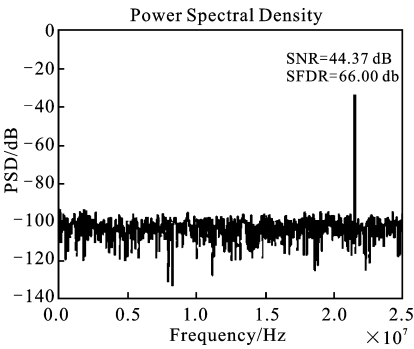
本文将布谷鸟算法用于测试用例的自动生成,并且针对传统算法中存在的迭代后期收敛速度较慢、搜索精度不高等问题,提出了一些改进算法,引入立方混沌映射对鸟巢位置进行初始化,增加了种群的多样性,并且对局部最优值引入高斯扰动增加种群变化活力,加快了收敛速度。另外,本文还对算法的适应度函数进行了有效的设计,针对测试用例生成的特点,用测试数据的分支覆盖率代表适应度值,以加快数据的优化过程。

改进后的布谷鸟算法提高了测试数据生成的效

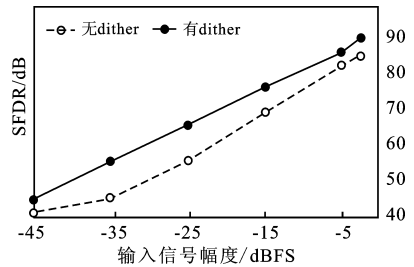
率和效果。本文利用三角形程序、最小值和数组相等程序,与基本布谷鸟算法、基本遗传算法进行了对比分析实验结果表明了算法的实用性及高效性,可以理想的运用到整个软件测试的流程中。

参考文献:

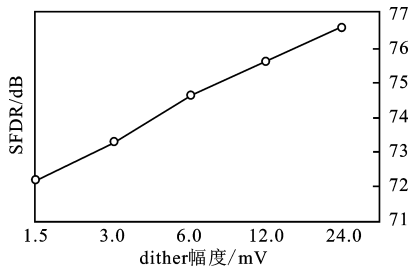
- [1] Zhitao H E, Chao L, Yan H H. Software testing evolution process model and growth of software testing quality[J]. Science China Information Sciences, 2015, 58(3):1-6.
- [2] Gay G, Staats M, Whalen M, et al. The risks of coverage-directed test case generation[J]. IEEE Transactions on Software Engineering, 2015, 41(8):803-819.
- [3] Denaro G, Pezze M, Vivanti M. Quantifying the complexity of dataflow testing[C]// International Workshop on Automation of Software Test. Stockholm, Sweden, IEEE, 2013:132-138.
- [4] 杨瑞. 基于 EFSM 的测试用例自动化生成关键技术研究[D]. 南京:南京大学, 2015.
- [5] Mahadik P P, Thakore D M. Search-based junit test case generation of code using object instances and genetic algorithm[J]. International Journal of Software Engineering & Its Applications, 2016, 10(5):95-108.
- [6] Le T M H, Tung K T, Binh N T. Mutation-based test data generation for simulink models using genetic algorithm and simulated annealing [J]. International Journal of Computer and Information Technology, 2014 (3):763-771.
- [7] Sengupta A, Mukhopadhyay S, Sinha A K. Automated verification of power system protection schemes-part ii: test case generation using swarm intelligence [J]. IEEE Transactions on Power Delivery, 2015, 30 (5):2087-2095.
- [8] Rodriguez G D, Rodríguez J. Automatic generation of GUI test cases using ant colony optimization and greedy algorithm[C]// XVIII Ibero-American Conference on Software Engineering. USA, Linna, 2015:209-221.
- [9] Yang X S, Deb S. Cuckoo search via levy flights[C]// Proceeding of World Congress on Nature & Biologically Inspired Computing, NaBIC 2009. Washington: IEEE Publications, 2009:210 - 214.
- [10] 周燕, 刘培玉, 赵静, 等. 基于自适应惯性权重的混沌粒子群算法[J]. 山东大学学报(理学版), 2012, 47 (3):27-32.
- [11] 姜淑娟, 王令赛, 薛猛, 等. 基于模式组合的粒子群优化测试用例生成方法[J]. 软件学报, 2016, 27(4): 785-801.



(c)加入失配加入 dither



(d)SFDR 与输入信号幅度的关系



(e)SFDR 与 dither 幅度的关系

图 7 ADC 频谱图

4 结束语

本文研究了宽带 dither 技术对 ADC 的 SFDR 性能的影响,从理论上分析了 dither 对 ADC 动态线性度的改善,并在 cadence 环境下进行了 12bit 流水线 ADC 的电路仿真,在输入信号中加入 dither,在 ADC 的输出端减去 dither 后,对其进行 FFT 频谱分析得到 ADC 的 SFDR。仿真结果表明,宽带

Dither 技术能够有效提高 ADC 的动态性能。

参考文献:

[1] Kikkert C J, Bigdeli A. Frequency shift dither for analogue to digital converters [C] // Fifth International Symposium on Signal Processing and its Applications, Seoul, Korea, 1999:729-732.

[2] 陈小猛,张羽中,艾宪芸,等. 数字多通道脉冲幅度分析器中数字 Dither 加入方法[J]. 核技术,2015,38(5): 47-52.

[3] Kikkert C J, Bigdeli A. Hardware additive dither for analogue to digital converters [C] // 14th Australian Microelectronics Conference, Australian,1997, 43(1): 156-161.

[4] Robert A Wannamaker, Stanley P Lipshitz, John Vanderkooy, et al. A theory of nonsubtractive dither [J]. Signal Processing IEEE Transactions on, 2000, 48 (2) :499-516.

[5] Quinn P J, Van Roermund A H M. Accuracy limitations of pipelined ADCs[J]. IEEE International Symposium on Circuits and Systems, 2005(3):1956-1959.

[6] 丁洋,王宗民,周亮,等. 一种用于 16 位流水线 ADC 的多比特子 DAC 电容失配校准方法[J]. 微电子学与计算机,2012,29(6):172-176.

[7] Yun Chiu, Paul R Gray, Borivojc Nikolic. A 14b 12MS/s CMOS pipeline ADC with over 100 dB SFDR [J]. IEEE Journal of Solid-State Circuits, 2004, 39 (12):2139-2151.

作者简介:

靳翔女,(1993-),硕士. 研究方向为流水线 ADC 设计.

闫肃男,(1978-),硕士,工程师. 研究方向为模拟集成电路.

赵洪明女,(1992-),硕士. 研究方向为流水线 ADC 设计.

王兴华(通信作者)女,(1983-),博士,讲师. 研究方向为模拟集成电路. E-mail: wxinghuad@126.com.

陈铖颖男,(1982-),博士,副研究员. 研究方向为模拟/混合信号集成电路设计.

(上接第 8 页)

作者简介:

何海鲜女,(1993-),硕士研究生. 研究方向为软件工程、软

件测试、人工智能. E-mail:931326653@qq.com.

冯静女,(1971-),博士,副教授,研究生导师. 研究方向为数据挖掘、嵌入式研发、人工智能.