DOI:10.3969/j.issn.1003-5060.2024.04.010

基于二阶边缘检测算法的随机架构设计

刘志雨, 解光军
(合肥工业大学 微电子学院,安徽 合肥 230601)

摘 要:数字图像边缘是具有明显亮度变化的像素集合,边缘检测是识别图像边缘的最佳方法。其中,二阶边缘检测算法具有很强的边缘定位能力,但在硬件实现上需要消耗大量资源,且易受到电路的内部噪声影响。文章提出拉普拉斯(Laplace)和高斯拉普拉斯(Laplacian of Gaussian,LoG) 2种常见二阶边缘检测算法的随机电路结构,并控制输入比特流的相关性来优化电路,进一步提高运行效率。实验结果表明,相比于传统的加权二进制实现,该电路消耗更少的功耗和电路面积,同时拥有更高的容错性。

关键词:二阶边缘检测算法;容错;低成本设计;随机计算;低功耗设计
中图分类号:TN402 **文献标志码**:A **文章编号**:1003-5060(2024)04-0496-06

Stochastic architecture design for second-order edge detection algorithm

LIU Zhiyu, XIE Guangjun
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: An edge in a digital image is a collection of pixels with significant brightness variations, and edge detection is the best method to identify the edges of an image. Among them, the second-order edge detection algorithm has a strong edge localization capability, but it consumes a lot of resources in hardware implementation and is vulnerable to internal noise in the circuit. This paper proposes the stochastic circuit structure for two common second-order edge detection algorithms, Laplace and Laplacian of Gaussian(LoG), and controls the correlation of the input bitstream to optimize the circuit and further improve the operational efficiency. Experimental results show that the circuit structure consumes less power and circuit area while possessing higher fault tolerance than the conventional weighted binary implementation.

Key words: second-order edge detection algorithm; fault tolerance; low-cost design; stochastic computing(SC); low-power design

边缘检测能够高效地识别图像边缘,是检测不连续灰度图像的最佳方法,在图像预处理过程中起着重要作用,广泛应用于目标识别和图像分割等计算机视觉领域。目前关于边缘检测的算法不断被开发,根据对像素点求导阶数的不同,这些算法主要可以分为基于一阶微分的基本算法和基于二阶微分的高级算法。相比于一阶算法,二阶算法强调的是图像突变而不是灰度缓慢变化的区域,对边缘的定位能力更强。

尽管边缘检测算子能够高效地识别图像边缘,但实现时需要大量的硬件资源,并且对噪声较为敏感,这在一定程度上限制了它的应用范围。文献[1]基于现场可编程门阵列(field programmable gate array,FPGA)开发Canny算法的硬件架构,尽管利用了中值滤波进行降噪,但仍然对软错误敏感。另外,随着器件制造工艺的快速发展,

收稿日期:2022-01-10;**修回日期**:2022-04-13
基金项目:安徽省科技重大专项资助项目(16030901007);中央高校基本科研业务费专项资金资助项目(JZ2020HGTA0085;JZ2020HGQA0162)
作者简介:刘志雨(1996—),男,安徽蚌埠人,合肥工业大学硕士生;
解光军(1970—),男,安徽合肥人,博士,合肥工业大学教授,博士生导师,通信作者,E-mail:gjxie8005@hfut.edu.cn.

现代计算机的硬件实现受到应用程序的严格限制，如极小的体积、低功耗和高可靠性[2-4]。由于在制造过程的物理错误和软错误不可避免，传统的二进制方法往往会过度设计系统以确保无误的结果，其代价是消耗大量硬件资源和更大的功耗。

随机计算（stochastic computing，SC）作为一种非传统的计算范式[5]早在 1969 年就被 Gains 提出，但在当时没有引起重视。随着现代计算机技术的发展，SC 以其低成本和高容错的特性成为传统加权二进制结构的潜在替代方案之一。近年来，SC 已经成功应用于一些对精度相对宽松的领域，如图像处理[6]、神经网络[7]、LDPC[8]等。文献[9]从现代数字电路设计的角度，系统分析了 SC 的优势和发展前景。

本文提出基于拉普拉斯（Laplace）和高斯拉普拉斯（Laplacian of Gaussian，LoG）算法的随机实现。所提随机电路结构优势如下：

1）整体电路设计只使用一个共享随机数源生成比特流，极大地降低了由发生器带来的硬件成本和能耗。

2）所提设计通过合理利用比特流的相关性，避免后级电路运算中存在的相关性损失，提高了整体输出的精度。

1　相关背景

1.1　二阶边缘检测算法

边缘检测算法通常对数字灰度图像的一阶或二阶导数进行操作。一阶微分边缘算子也称为梯度边缘算子，它利用图像在边缘处的阶跃性，即图像梯度在边缘取得极大值的特性进行检测，图像 $I(x,y)$ 在点 (x,y) 处梯度的计算公式如下：

$$\nabla I=(I_x,I_y)=\left(\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}\right) \tag{1}$$

实际运算中一般利用有限差分进行梯度近似计算，梯度模值的计算公式可表示为：

$$|\nabla I|=\sqrt{I_x^2+I_y^2}\approx|I_x|+|I_y| \tag{2}$$

与一阶微分算法不同，二阶微分边缘检测算法利用图像在边缘处灰度的剧烈变化导致二阶微分出现零值这一特性进行检测，更加强调图像灰度的突变的检测。

Laplace 算子本质上是一种二阶导数算子，其定义为两个方向上一阶导数的内积，即

$$\nabla^2 I=\frac{\partial^2 I}{\partial x^2}+\frac{\partial^2 I}{\partial y^2} \tag{3}$$

对图像使用二阶差分代替二阶导数，则 Laplace 算子的近似表达式如下：

$$\nabla^2 I=I(i,j+1)+I(i,j-1)+I(i-1,j)+I(i+1,j)-4I(i,j) \tag{4}$$

对应的二阶边缘检测算法的卷积核如图 1a 所示。Laplace 算子是一种高通滤波模型，对于出现的独立噪声非常敏感，为了减少噪声对边缘的影响，首先对图像低通滤波，LoG 算子用高斯函数作为低通滤波器，然后再进行二阶微分运算。运算过程可以转换为先对高斯函数进行二阶微分，再利用高斯函数的二阶微分结果对图像进行卷积运算，具体过程如下：

$$\nabla^2[G(x,y)*I(x,y)]=\nabla^2 G(x,y)*I(x,y) \tag{5}$$

LoG 算子的数学表达式如下：

$$\nabla^2 G(x,y)=\frac{\partial^2}{\partial^2 x}G(x,y)+\frac{\partial^2}{\partial^2 y}G(x,y)=\frac{x^2+y^2-\sigma^2}{\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{6}$$

根据式(6)推导的一种常用的卷积核如图 1b 所示。

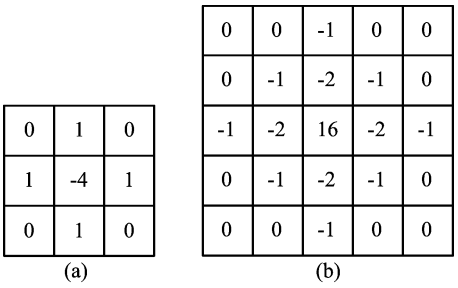


图 1　二阶边缘检测算法的卷积核

1.2　随机计算

随机计算将数据通过随机数发生器（stochastic number generator，SNG）编码成一串 0 和 1 组成的二进制比特流的形式进行概率计算。在随机计算中，比特流常用单极性和双极性 2 种格式编码，在固定比特流长度下，单极性编码的精度是双极性的 2 倍，因此本文使用比特流的单极性表示。另外，随机比特流的每一位具有相同权重，在较长位流中单个比特的数值翻转只会导致数据微小的变化，因此随机计算结构固有极强的容错能力[10]。

随机结构主要由 3 部分组成：① 随机化单元，它将给定的数值 B 与随机数发生器每个周期生成的一个整数 R 进行比较来产生随机比特流，一般由线性反馈移位寄存器（linear feedback shift register，LFSR）组成；② 算术处理单元，以

比特流的形式运行算法;③ 去随机化单元,用于将随机比特流转换回以二进制格式的输出值,通常是由计数器组成[11]。在随机计算中,可以通过简单的组合逻辑实现算术运算[4]。如与门可以实现乘法,多路复用器(multiplexer,MUX)可以实现加法。比特流之间的相关性对计算结果有非常关键的影响[12],通常用随机计算相关性(stochastic computing correlation,SCC)进行定量计算[13]。若 SCC 为 1 则表明位流间具有最大正相关性,通过异或门(XOR)可以实现减法运算,此时与门可以实现取最小值操作而不是乘法运算。因此,合理地利用相关性可以为随机电路的设计提供便利。

2 二阶边缘检测算法的硬件实现

本文旨在为 Laplace 和 LoG 二阶边缘检测算法提供一种通用的随机电路设计思路,对于不同的卷积核都可以基于下文中提出的电路架构实现。为了方便描述,本文使用如图 1 所示的 2 个卷积核作为模板进行分析。

2.1 传统二进制电路实现

Laplace 与 LoG 2 种二阶边缘检测算法的传统电路实现如图 2 所示。

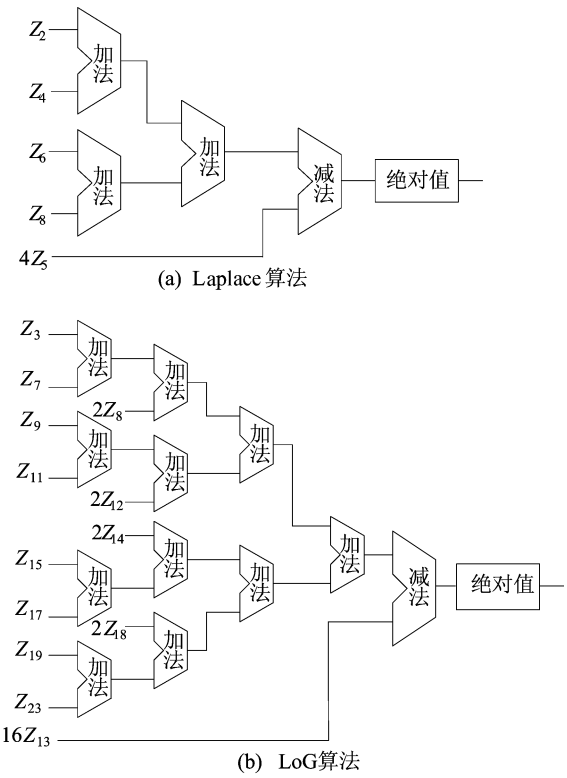


图 2 传统电路实现

对于 Laplace 算法,用图 1a 所示的 3×3 窗口,输入像素 $Z_1 \sim Z_9$ 用于计算窗口中心位置的像素输出 S_5 的值。同理,LoG 算法使用图 1b 所示的 5×5 窗口,输入像素 $Z_1 \sim Z_{25}$ 用于计算处理后窗口中心位置的像素输出 S_{13} 的值。在传统实现中,Laplace 和 LoG 算法依次执行多级加法、减法和绝对值运算,本文使用常用于图像处理操作的 8 位无符号格式数字来表示像素大小,Laplace 和 LoG 算法的传统电路实现分别使用了 3 个和 11 个加法器。

2.2 随机电路实现

由于单极性随机电路中的所有数据均在[0,1]区间内进行计算,则对应的 Laplace 和 LoG 算法随机表达式可改写为:

$$S_5 = \left|\frac{1}{4}(Z_2+Z_4+Z_6+Z_8)-Z_5\right| \tag{7}$$

$$S_{13} = \left|\frac{1}{16}(Z_3+Z_7+2Z_8+Z_9+Z_{11}+2Z_{12}+2Z_{14}+Z_{15}+Z_{17}+2Z_{18}+Z_{19}+Z_{23})-Z_{13}\right| \tag{8}$$

上述 2 种算法都需要对输入位流取平均值后执行绝对值减法运算。通过使用 2^k 选 1 的 MUX 计算 2^k 个输入像素的平均值,其中 k 表示 MUX 选择端的个数,对应连接到编码值为 0.5 的 k 条不相关的比特流。同时,MUX 的数据输入端具有相关不敏感性。因此,本文采用共享随机数源的结构生成具有最大正相关性的像素比特流,由于选择端给定独立的比特流,经过均值后的输出位流与输入位流依然具有最大的相关性,这是其他随机结构不具有的优势。接着通过 XOR 门实现绝对值减法,可以实现这 2 种边缘检测算法的算术单元电路,如图 3 所示。

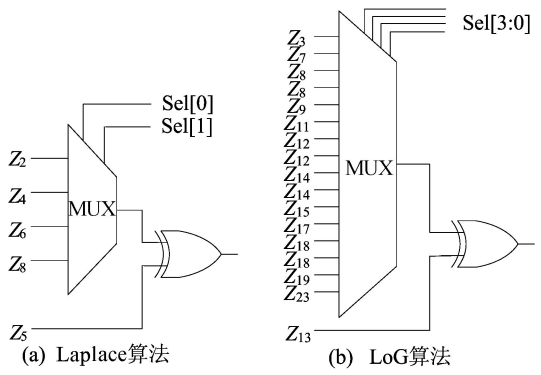


图 3 算术处理电路的实现

相比于精简的算术计算单元,随机数转换电路占据绝大部分的硬件开销。若使用独立随机数

源的设计，Laplace 和 LoG 算法的电路实现则分别需要 5 和 17 个随机数源来生成像素点对应的输入位流，同时因为失去了序列间的最大正相关的性质，所以还需要设计复杂的随机时序逻辑单元去实现绝对值减法运算。而共享随机数源的设计仅需要 1 个随机数源生成像素序列，而且整个算术处理单元结构非常简单，仅由 MUX 和 XOR 门组成，因此在硬件消耗占有优势。

算术处理单元的选择端需要连接到编码值为 0.5 的不相关比特流来将数值限制到[0，1]区间内，因而要实现本文的 2 种二阶算法就对应 2 个和 4 个选择位，需要对应数量的 SNG 产生不相关的比特流。基于共享随机数源的电路结构如图 4 所示。尽管像素输入只消耗一个共享的 LFSR，但 MUX 的选择端依然需要使用独立的数源，这极大地增大了硬件成本，算术运算单元的低功耗优势相对被过多的 LFSR 单元所抵消。

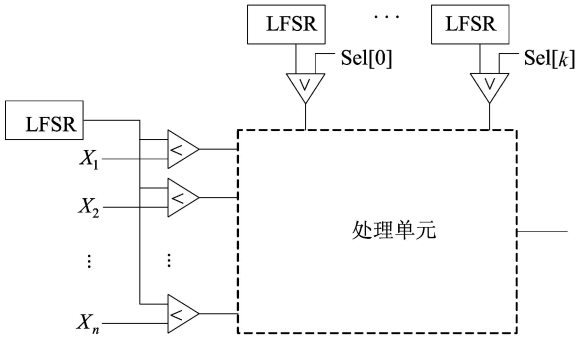


图 4　基于共享 LFSR 的随机数发生器

本文提出的优化设计如图 5 所示。由于组成 LFSR 的任一 D 触发器的输出均为 0.5 的比特流，考虑将这些比特流直接与 MUX 的选择端相连。优化后的发生器电路只使用一个 LFSR 产生像素点和选择端的随机比特流，比在选择端使用独立 SNG 的电路设计消耗更少的硬件资源。

上文论述了所提设计的低成本优势，在计算误差方面也需测试。为了衡量本文提出的优化设计与选择端使用独立 SNG 结构的精度，使用原始图像作为像素输入，计算各个选择比特流之间、选择比特流和像素输入位流之间的平均绝对 SCC，并执行 1 000 次蒙特卡洛模拟，2 种算法的测试结果见表 1 和表 2 所列。由表 1、表 2 可知，本文的优化设计在不同位流长度下实现 2 种算法时均拥有较低的平均绝对 SCC 数值，即意味着拥有更高的精度。整体上看，针对 Laplace 和 LoG 算法的随机电路实现，在算术运算单元相同的情况下，本文所提出的优化设计不管在硬件成本还是在计算精度方面，相比于图 4 设计均具有优势。

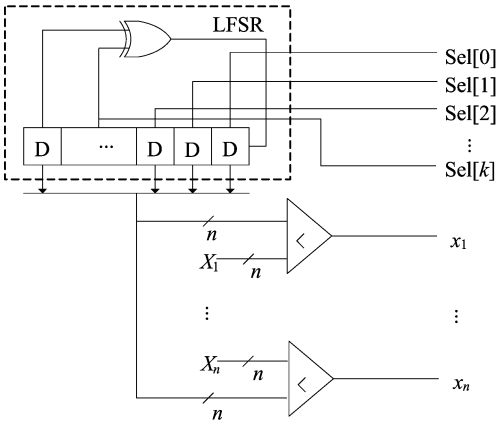


图 5　共享 LFSR 随机发生器的优化设计

表 1　Laplace 算法 2 种随机设计的平均绝对 SCC

比特流长度/位	8	16	32	64	128
独立 SNG	0.362	0.191	0.133	0.070	0.033
本文设计	0.332	0.183	0.068	0.039	0.019

表 2　LoG 算法 2 种随机设计的平均绝对 SCC

比特流长度/位	16	32	64	128	256
独立 SNG	0.114	0.062	0.029	0.012	0.008
本文设计	0.093	0.059	0.027	0.012	0.007

3　实验和结果

本节对 2 种二阶边缘检测算法的常规二进制电路与随机电路进行比较。通常，随机实现的准确性与所使用的比特流长度成正比。更长的比特流意味着更长的运行时间，虽然通过提高系统的工作频率可以缓解，但这将导致更高的功率消耗，因此实际应用中需要在硬件成本，准确性和运行时间之间进行权衡。本文选择一个原始图像作为实验样本如图 6 所示。

图 6　原始图像

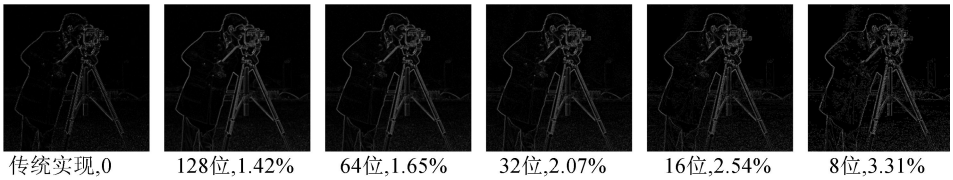
在通过 MATLAB 验证功能性后，使用 Modelsim 对传统实现和不同位流长度的随机实现进行实验仿真；然后利用 Synopsys 设计编译器和 TSMC 45 nm 库对相关电路进行综合；最后从仿

真结果、硬件成本和容错性的角度比较不同电路的性能。

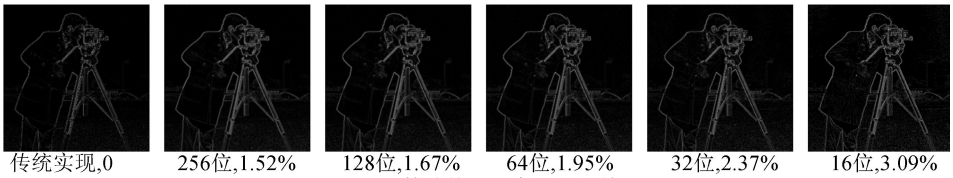
3.1 仿真结果

上述二阶边缘检测算法的常规和不同位流长度下随机实现的仿真结果如图7所示,并在图片下方标注对应的错误率。

从图7可以看出,在最差的情况下,Laplace算法使用8位比特流的随机实现的错误率为3.31%,LoG算法在使用16位比特流条件下的随机实现的误差率为3.09%。在16位比特流条件下,LFSR的D触发器数量已经和复用器选择端口数量持平,若再降低比特流长度,则需增加LFSR的个数。实际上大多数数字图像处理算法在误差率低于5%是可以接受的[11],因此对于Laplace和LoG算法,使用8位和16位比特流的随机实现是可行的。



(a) Laplace算法的不同实现及误差



(b) LoG算法的不同实现及误差

图7 二阶算法传统实现和随机实现的仿真结果

3.2 硬件资源与能耗对比

从Synopsys Design Compiler工具的综合报告可以直观地比较所有方法所需的硬件资源,随机实现的硬件评估包含转换电路在内,比较结果见表3所列。

由表3可知,随机实现比传统实现使用了更少的硬件成本。Laplace算法的基于8位比特流的随机电路实现面积占用仅为传统实现的22.4%,在功率方面仅为传统实现的19.45%。使用16位比特流的LoG算法随机实现的面积占用比仅为传统电路的15.23%,在功率方面为传统实现的15.01%。

表3 Laplace和LoG算法电路的硬件性能对比

算法	实现方式	面积/μm^2	功率/nW	关键路径延迟/ns	延迟/ns	PDP/aJ	算法	实现方式	面积/μm^2	功率/nW	关键路径延迟/ns	延迟/ns	PDP/aJ
Laplace	传统实现	285.68	4.01	3.37	3.37	13.52	LoG	传统实现	701.97	8.86	4.43	4.43	39.26
	128位随机	170.51	2.08	1.82	1.82×2^7	483.44		256位随机	336.76	4.38	2.33	2.33×2^8	2 610.14
	64位随机	148.69	1.81	1.62	1.62×2^6	187.92		128位随机	280.63	3.55	1.83	1.83×2^7	830.78
	32位随机	121.30	1.48	1.46	1.46×2^5	69.05		64位随机	217.06	2.83	1.67	1.67×2^6	302.47
	16位随机	87.78	1.09	1.03	1.03×2^4	17.88		32位随机	164.65	2.11	1.61	1.61×2^5	108.70
	8位随机	64.64	0.78	0.97	0.97×2^3	6.05		16位随机	106.93	1.33	1.21	1.21×2^4	25.72

尽管随机电路单个时钟周期内消耗更少的资源,但是随机计算存在长延迟这一固有缺点,导致总能耗可能会比常规实现更大。为了更加公平地比较能耗,本文使用功率延迟乘积(power-delay-product,PDP)来评估能耗,其中延迟是指电路的关键路径延迟和比特流长度的乘积。从表3可以看出:Laplace算法的传统电路所消耗的能量是使用8位随机电路的2.2倍;LoG算法的传统电路所消耗能量是使用16位随机电路的1.5倍。

3.3 容错对比

本文使用文献[11]中的方法在内部电路中随机注入软错误,加入3%的错误率意味着占总量30%的信号比特数被随机选择并翻转,然后测量传统实现和不同位流长度下随机实现所对应的平均输出错误来评估容错性。假设待处理图像的高为H,宽度为W,则错误率用平均绝对误差E来

衡量，计算公式为：

$$E=\frac{\sum_{i=1}^{H}\sum_{j=1}^{W}|S_{i,j}-C_{i,j}|}{255HW} \tag{9}$$

其中：$C_{i,j}$ 为无噪声输入的常规实现电路的输出图像；$S_{i,j}$ 为同一算法下注入噪声后随机或常规实现电路的输出图像。Laplace 和 LoG 算法各种实现的错误率与平均绝对误差的关系如图 8 所示。

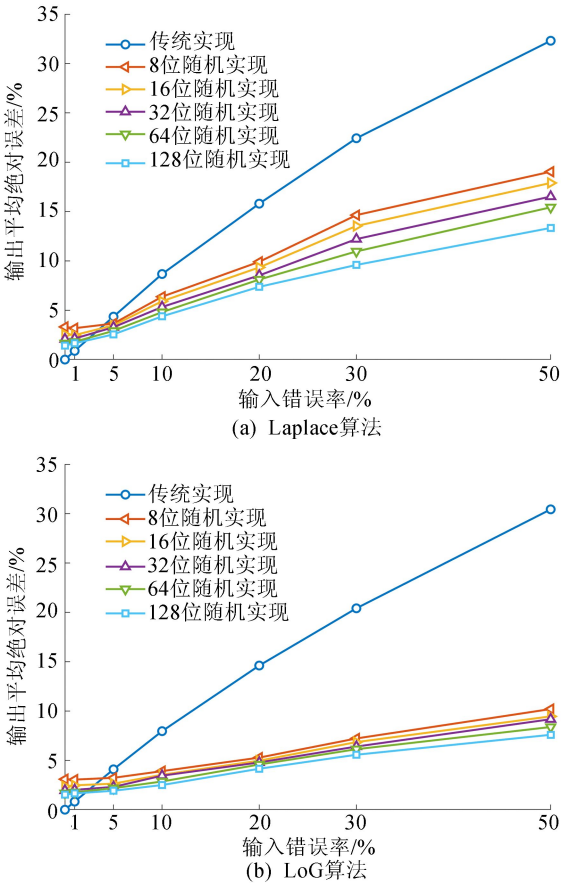


图 8　传统实现和随机实现容错曲线

由图 8 可知，随机实现的精度与比特流长度成正比，当模拟输入的误差率大于 5%时，Laplace 算法的 8 位随机实现、LoG 算法的 16 位随机实现在容错性上都优于对应的传统电路实现。

4　结　　论

本文提出了基于随机计算的 2 种二阶边缘检测算法的容错结构。随机结构通过共享随机源，并利用相关性来降低硬件开销。另外，随机实现可以根据要求灵活地调整比特流长度达到精度和能耗的权衡。实验结果表明，本文提出的随机实现比传统加权二进制电路实现拥有更高的容错性，且消耗更少的面积和功耗。

［参　考　文　献］

[1] LI X, JIANG J, FAN Q. An improved real-time hardware architecture for Canny edge detection based on FPGA[C]//The Third International Conference on Intelligent Control and Information Processing. [S. l.]: IEEE, 2012: 445-449.

[2] HAN J, CHEN H, LIANG J, et al. A stochastic computational approach for accurate and efficient reliability evaluation[J]. IEEE Transactions on Computers, 2014, 63(6): 1336-1350.

[3] ALAGHI A, QIAN W, HAYES J P. The promise and challenge of stochastic computing[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(8): 1515-1531.

[4] WOO K L, GUTHAUS M R. Fault-tolerant synthesis using non-uniform redundancy[C]//The 27th IEEE International Conference on Computer Design. [S. l.]: IEEE, 2009: 213-218.

[5] GAINES B R. Stochastic computing systems[M]. Berlin Heidelberg: Springer, 1969: 37-172.

[6] LI P, LILJA D J, QIAN W, et al. Computation on stochastic bit streams digital image processing case studies[J]. IEEE Transactions on Very Large Scale Integration Systems, 2013, 22(3): 449-462.

[7] LIU S, JIANG H, LIU L, et al. Gradient descent using stochastic circuits for efficient training of learning machines [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(11): 2530-2541.

[8] HUANG K. Efficient Algorithms for stochastic decoding of LDPC codes[D]. Ann Arbor: Northeastern University, 2016.

[9] ALAGHI A, HAYES J P. Survey of stochastic computing [J]. ACM Transactions on Embedded Computing Systems, 2013, 12(2): 1-19.

[10] NAJAFI M H, SALEHI M. A fast fault-tolerant architecture for sauvola local image thresholding algorithm using stochastic computing [J]. IEEE Transactions on Very Large Scale Integration Systems, 2016, 24(2): 808-812.

[11] QIAN W, LI X, RIEDEL M D, et al. An architecture for fault-tolerant computation with stochastic logic[J]. IEEE Transactions on Computers, 2010, 60(1): 93-105.

[12] LEE V T, ALAGHI A, CEZE L. Correlation manipulating circuits for stochastic computing[C]//Automation & Test in Europe Conference & Exhibition. [S. l.: s. n.], 2018: 1417-1422.

[13] ALAGHI A, HAYES J P. Exploiting correlation in stochastic circuit design[C]//IEEE 31st International Conference on Computer Design. [S. l.]: IEEE, 2013: 39-46.

（责任编辑　李　凯）