一种高鲁棒性数字水印算法的FPGA实现

孙 峰 华云松 吴继伟
(上海理工大学光电信息与计算机工程学院 上海 200093)

摘 要：传统上数字水印算法使用软件方法实现。基于一种高鲁棒性的数字水印算法，研究了使用FPGA实现水印嵌入的方法。先通过提升小波变换将宿主图像进行3次分解，并用混沌置乱算法将水印图像进行置乱加密，然后将第3阶小波分解的LL部分进行SVD分解并得到对应的奇异值矩阵，然后将混沌置乱加密后的水印信息嵌入到该奇异值矩阵中。在该算法的FPGA实现中，主要研究了提升小波变换、混沌置乱加密、矩阵SVD分解的硬件实现方法。实验结果表明，该算法鲁棒性很高，嵌入水印的图像抗攻击性强，算法经过合理推导得以在FPGA上高效地实现。

关键词：数字水印；提升小波变换；混沌置乱加密；矩阵SVD分解；FPGA
中图分类号：TN911.73 **文献标识码**：A **国家标准学科分类代码**：510.4050

High robustness digital watermarking algorithm implemented based on FPGA

Sun Feng Hua Yunsong Wu Jiwei
(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: Traditionally digital watermarking algorithm is implemented by software method. This paper proposes a high robustness digital watermarking algorithm and an implementation based on FPGA. In this paper, the watermark embedding algorithm is mainly realized by LWT, CS and SVD, where lifting wavelet transform is to decompose the host image to 3 stages and chaotic scrambling algorithm is to encrypt the watermark image and then decompose the LL part of third LWT order using SVD to get the corresponding singular value matrix and last embed the encrypted watermark into the singular value matrix. In the FPGA implementation of this algorithm, the hardware implementation of lifting wavelet transform, chaotic scrambling encryption and matrix SVD decomposition are studied. The experimental results show that the proposed algorithm in this paper is robust, and the watermarked image has strong anti- attack ability. The algorithm can be efficiently implemented on FPGA. Finally, the experimental results have been verified, using FPGA to achieve the digital watermarking algorithm is stable and reliable, the watermark embedding rate is greatly improved.

Keywords: watermarking; lifting wavelet transform; chaotic scrambling encryption; SVD; FPGA

0 引 言

互联网+时代已经到来，网络信息传输、商品交易等越来越频繁，网络安全问题也日益突出，信息安全、知识产权保护成为所有互联网参与者共同关心的重要问题。数字水印技术是保障原始作者版权和信息认证来源及完整性的新型技术，关键信息以不可感知的形式嵌入到多媒体中，需通过专用检测方法才能提取出来。目前，普遍使用软件来实现图像数字水印，一旦信息量变大、速率变快，这种传统方式将不能满足实时处理要求。FPGA是一种硬件实现方式，具有高速的并行处理能力，且已经广泛应用于高性能应用场合。文献[1]在Contourlet域使用QR分解进行水印嵌入；文献[2]将水印插入拉普拉斯金字塔分解后的宿主图像中；文献[3]提出基于QR分解和小波变换的水印嵌入算法。利用FPGA实现数字水印已有些许文献提出，如文献[5]、[6]在DCT域使用硬件方式完成了水印嵌入，文献[7-9]讨论了在DWT域实现数字水印。但以上这些方式实现的数字水印都有抗攻击性差或实时性不高的问题[10]。针对这些不足，设计了一种适合于FPGA实现的数字水印算法，将水印信息进行混沌置乱加密，对宿主图像进行三阶小波变换，将加密后的水印嵌入到第3层 *LL* 部分的奇异值中，得到了非常高的抗攻击性，嵌入后的图像无法感知出水印信息，同时本设计实现过程中强调算法硬件结构及HDL编程的的合理性，以使算法得以高效实现。最后给出了仿真示例及抗攻击性测试的结果。

收稿日期：2017-02

1　算法原理

水印算法决定了嵌入水印后图像的抗攻击性。设计中使用的宿主图像 M_0 是大小为 $m\times m$(记为 N_M)、位宽为 8 位、颜色深度为 256 的灰度图,使用的水印信息图像 W_0 是大小为 $n\times n$(记为 N_W)的二值化图像。

将宿主图像进行 3 次二维小波变换,每一次变换都将图像分为 4 部分:LL_i、LH_i、HL_i、HH_i,其中 LL_i 是变换后各级的低频分量,LH_i 是各级的垂直方向高频分量,HL_i 是各级的水平方向高频分量,HH_i 是各级的对角方向高频分量($i=1,2,3$)如图 1 所示。根据小波变换的原理可知,图像经过小波变换之后,原图像的绝大多数能量主要被集中在变换后的低频子带 LL_i 中,如果嵌入水印后的图像受到攻击处理,低频子带的大部分信息仍然能被保留下来。因此本设计算法选择在低频子带 LL_3 中嵌入水印信号。

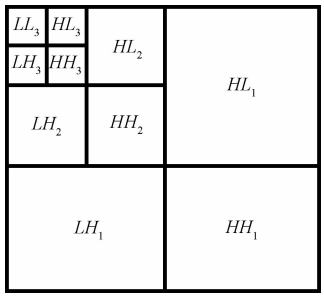

图 1　图像三级二维小波变换

普通的 DWT 算法硬件实现结构一般使用卷积结构,运算量大,逻辑资源消耗多,而基于提升结构的 DWT 在片内存储利用率和读取次数上都有优势。提升小波变换算法的基本思想是把有限长的高通和低通小波滤波器分解成一系列小型滤波器,进而分解为两个连续的上三角矩阵、下三角矩阵及一个对角矩阵,基本思想是用数据相关去除冗余。规范的提升小波算法分为 3 个步骤:分裂(Split)、预测(Predict)和更新(Update)。其变换原理如图 2 所示。

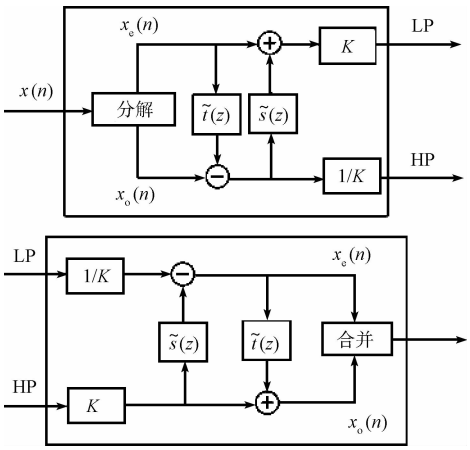

图 2　正反提升小波变换原理

为增加水印的安全性与抗攻击性,在将水印嵌入到宿主图像之前,采用图像置乱算法将二维二值水印图像 W_0 进行置乱变换,将 W_0 每个像素的原始坐标 (i,j) 映射到新坐标 (i',j'),再与 Logistic 序列做异或运算,最终得到基于混沌序列调制的加密水印序列 W_{ec}。其中 W_0 和 W_{ec} 的像素点的个数都为 N_W 个。

由 SVD 分解的性质知道,每个矩阵都可以通过 SVD 分解得到 3 个矩阵 $\boldsymbol{U}$、$\boldsymbol{S}$ 和 $\boldsymbol{V}$,$\boldsymbol{U}$、$\boldsymbol{V}$ 为左右酉矩阵,而矩阵 S 的对角线即为该矩阵的奇异值,每个不同的矩阵都可以得到独特的奇异值。将图像 M_0 经过 SVD 分解,得到的矩阵 S 的对角线为 σ_1、σ_2、σ_3、…、σ_r、σ_{r+1}、…、σ_k(即为奇异值),其中 r 为 M_0 的秩,k 为对角线元素个数,

σ_1 到 σ_r 不为 0,σ_{r+1} 到 σ_k 为 0,根据 SVD 奇异值的性质可知,图像的主要信息都集中在了 σ_1 到 σ_r 中,那么将加密后的水印信息嵌入到该奇异值序列中,则可大大提升嵌入水印后图像的抗攻击性。在本设计中,并不对图像做整体的 SVD 分解,而是先将宿主图像分解成 j 块 2×2 的矩阵,故 $j=(m/2)\times(m/2)$,再从这 j 个分块中随机选取 N_W 个 2×2 的矩阵,然后将加密后水印的 N_W 个像素点信息嵌入到 N_W 个奇异值中。这其中随机位置序列要存下来,以便于逆运算提取水印时使用。嵌入的方式是如果加密水印的对应像素是 1,将对应的左上角的奇异值 $S1+a$;如果加密水印的对应像素是 0,将对应的左上角的奇异值 $S1-a$;这其中 a 是嵌入强度,一个可设定的值,但在一次嵌入和提取的算法流程中是一个固定的常数。

整个算法的原理如图 3 所示。

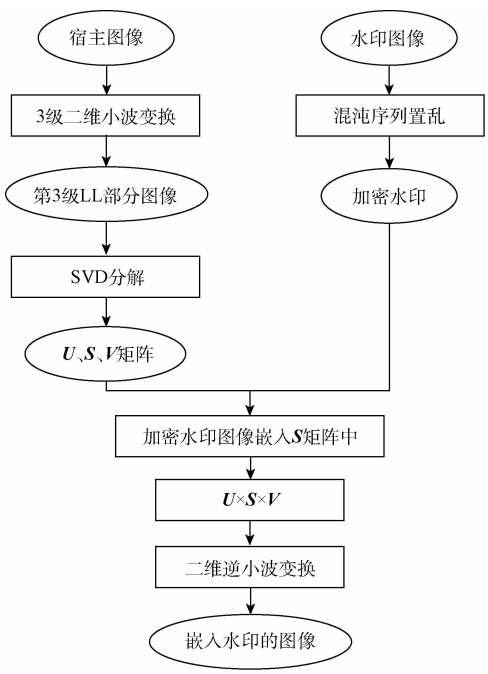

图 3　整体算法原理

2　FPGA 仿真与实现

本设计中算法实现的关键点分别是二维小波变换、混沌置乱加密、SVD 分解，因此，使用合理的硬件结构实现算法是本文的重点。

2.1　二维提升小波变换算法的 FPGA 实现

本设计中使用 9/7 DWT，将变换分为两个模块并行工作，模块 1 完成第 1 级 DWT 行列变换，模块 2 以分时复用的方式进行第 2 级和第 3 级 DWT 行列变换。在 DWT 硬件实现结构中，小波系数将以流水线的方式按照特定的顺序连续输出到后级进行处理。算法计算方式为：

$$\begin{cases} y(2n+1)=x(2n+1)+\alpha[x(2n)+x(2n+2)], & -2\leqslant n<N/2+1, \quad (\text{第 1 步提升}),\\ y(2n)=x(2n)+\beta[y(2n-1)+y(2n+1)], & -1\leqslant n<N/2+1, \quad (\text{第 2 步提升}),\\ y(2n+1)=x(2n+1)+\gamma[y(2n)+y(2n+2)], & -1\leqslant n<N/2, \quad (\text{第 3 步提升}),\\ y(2n)=x(2n)+\delta[y(2n-1)+y(2n+1)], & 0\leqslant n<N/2, \quad (\text{第 4 步提升}),\\ y(2n+1)=Ky(2n+1), & -1\leqslant n<N/2, \quad (\text{量化}),\\ y(2n)=y(2n)/K, & 0\leqslant n<N/2, \quad (\text{量化}), \end{cases} \tag{1}$$

其中，$x(n)$ 为输入图像的像素数据，$y(2n)$ 为小波低通系数，$y(2n+1)$ 为小波高通系数，式(1)中的常系数 $\alpha\approx-1.586$，$\beta\approx-0.053$，$\gamma\approx0.883$，$\sigma\approx0.444$，$K\approx1.149$。从式(1)中可看出，各步骤的提升计算过程很近似，各步中均含有 2 次加法运算和 1 次乘法运算，区别仅在于各步的输入乘法系数不同。以第一步为例：

$$y(2n+1)=x(2n+1)+\alpha[x(2n)+x(2n+2)]=\alpha x(2n)+x(2n+1)+\alpha x(2n+2) \tag{2}$$

式(2)可再分为两步运算，即简单变形为：

$$g_1=\alpha x(2n)+x(2n+1)$$

$$g_1=g_2+\alpha x(2n+2) \tag{3}$$

故可将乘法器和加法器时分复用，并运用 FSM 选择控制的方法由同一个电路结构分时完成两次运算。

根据以上分析，本设计使用 1 个乘法器、1 个加法器、5 个选择器、1 个寄存器和 1 个移位器来完成单步提升结构。其硬件结构图如图 4 所示。

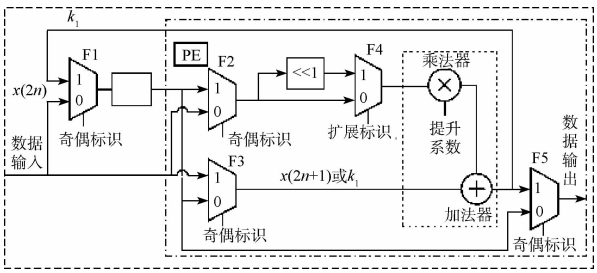

图 4　单步提升结构

那么，9/7 DWT 的所有 4 个提升运算都可以使用上述相同的单步提升结构来完成，需要改变的是输入数据和乘法运算系数。为了加快计算速度，本设计采用了流水线(Pipeline)式的设计方式，即例化出 4 个独立的单步提升结构级联并行运算，如此可实时实现 1 次行变换。在各独立提升结构的输入输出路径中，通过插入 D 除法器来缩短路径之间由于较长的组合逻辑路径引起的时延。

行变换的输出结果存入两个寄存器后进入列变换模块。为了使行列变换可以并行运算，在列变换时需要对整行数据进行操作。因此在列变换中不使用寄存器而是用 DPRAM 来缓存输入数据和中间结果。完成两次行变换之后存入 RAM，当第 2 行第 1 个数据输入时就开始列变换。如此便可以完成 1 次提升小波变换。如果需要大于 1 次的 DWT，可使用时分复用的方式来完成后面的变换，此时需要增加一些额外的控制逻辑，但所增加的逻辑资源很少。复用控制逻辑主要包括奇偶标识信号、变换层次标识信号和 RAM 控制信号。本设计中需要完成 3 次提升 DWT。行变换和列变换逻辑结构如图 5 所示。

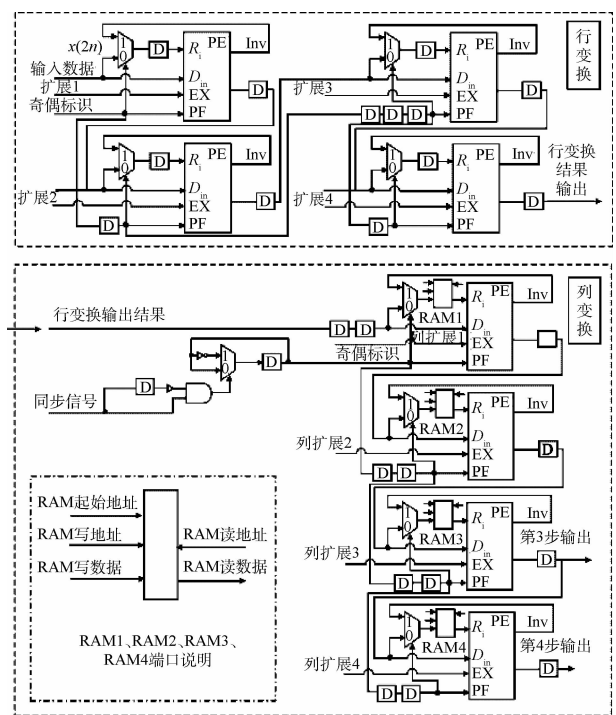

图 5　行变换和列变换逻辑结构

2.2　混沌置乱加密算法的 FPGA 实现

本设计中，使用改进型 Logistic 映射方式形成混沌系

列,这种映射算法简单且性能较好,适于硬件实现。映射关系为:

$$x_{i+1} = c(x_i - x_i^2) = cx_i(1 - x_i) \tag{4}$$

$$x_i = \text{round}[\text{mod}(1\,000x_i, 256)] \tag{5}$$

i 取 $[1, N\times N]$,c 是常数,用于控制分布的幅值,初值 x_1 可取 $[-1,1]$ 之间的数值,为了方便后面的计算,可以设置为 $[0,1]$ 的任意数值。经过计算将产生 $N\times N$(即二值水印矩阵的像素点个数)个数值,产生的数值将由式(5)调整为 0～255 之间的数,最后每个 x_i 与 127.5 比较产生二值化序列,该序列即为混沌序列 m。在该序列的一个周期内,0.1 分布具有随机性、良好的自相关特性和互相关特性及理想的线性复杂度。

在硬件实现中,式(5)将采取第二个等号右边的方式进行运算,这样一个乘加器就可以完成式(4)的运算,免除了第一个等式右边中的平方运算所需的多余的硬件消耗;取余运算将使用 FPGA 的除法器 IP 核,由于每个 FPGA 厂商都提供了这样的除法器,故 FPGA 型号不限定。同时,由于有乘 1 000 运算,为了节省硬件资源,可考虑通过将数据向右移 10 位来实现,同时将取余底数 256 改为 262。后面的取整运算和比较运算在 HDL 语言中比较容易实现。

2.3　矩阵 SVD 分解算法的 FPGA 实现

本设计中需要设计一个使用 FPGA 实现的 2×2 矩阵 SVD 分解算法硬件结构,然后在整个算法过程中复用此模块即可。SVD 分解可使用基于 systolic 结构完成。本文使用双边 Jacobi SVD 算法来实现 SVD 矩阵分解。

设需要分解的矩阵 $\boldsymbol{G} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$,其中,左右算转角与 $\boldsymbol{G}$ 中矩阵元素的关系为:

$$\begin{cases} \theta_R + \theta_L = \arctan\left(\dfrac{c+b}{d-a}\right) \\ \theta_R - \theta_L = \arctan\left(\dfrac{c-b}{d+a}\right) \end{cases} \tag{6}$$

$$\begin{pmatrix} \cos\theta_L & -\sin\theta_L \\ \sin\theta_L & \cos\theta_L \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \cos\theta_R & \sin\theta_R \\ -\sin\theta_R & \cos\theta_R \end{pmatrix} = \frac{1}{2}\begin{pmatrix} A+C & B+D \\ B-D & C-A \end{pmatrix} \tag{7}$$

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} a-d \\ b+c \end{pmatrix} \tag{8}$$

$$\begin{pmatrix} C \\ D \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} a+d \\ b-c \end{pmatrix} \tag{9}$$

其中 $\alpha = \theta_R + \theta_L$,$\beta = \theta_R - \theta_L$。在 FPGA 中,反正切算法可以使用 CORDIC 核方便地实现,故 2×2 矩阵 SVD 分解运算可结合 CORDIC 核实现。

本设计中,SVD 分解采用流水线结构,其硬件实现图如图 6 所示。

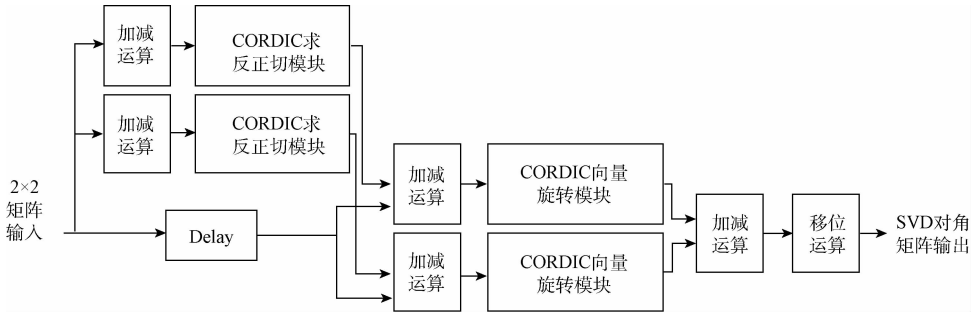

图 6　2×2 SVD 模块结构

实现了单个 2×2 矩阵 SVD 分解结构后,在实际的 HDL 中,可按照水印大小、像素点个数因素适度的重复例化多个此结构,以提高运算的速率。

2.4　整体仿真

仿真的过程中,在 MATLAB 中将灰度图像素值存为 coe 文件,在 Modelsim 中读入"MYMfopen("sz_im.txt", sz_im)",并在 testbench 文件中按照摄像头采样图像数据的方式将图像数据接入上面算法 IP Core 电路的输入端。在 HDL 仿真的过程中,图像的矩阵像素被转换成行和列数据处理,得到的结果也是序列型数据,最后根据对应的地址值将序列重新排列才能得到图像矩阵数据。Modelsim 中仿真时序图的部分截图如图 7 所示。

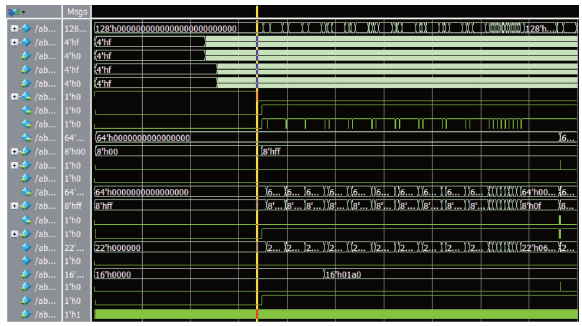

图 7　HDL 仿真时序图

经过 HDL 仿真后,在 Modelsim 中使用命令"MYMfwrite("wm_output.txt", wm_output)"HDL 仿真得到的图像数据结果按顺序存入 TXT 文本中,再在

Matlab 中读出此数据值，即可检查仿真结果。最终，仿真的结果如图 8 所示。其中图 8(a)为二进制水印图，图 8(b)为混沌置乱加密后的水印图，图 8(c)为原始宿主图像，图 8(d)为经过 HDL 仿真的嵌入水印后的图像。

(a) 二值水印图　(b) 置乱加密图　(c) 原始宿主图　(d) 嵌入水印图

图 8　HDL 仿真的嵌入水印图

3　实验结果与分析

将 HDL 程序仿真得到的图像矩阵存为 TXT 文本后，导入 MATLAB 中进行攻击实验，并计算原始水印与提取水印的归一化相关系数(normalized correlation，NC)。NC 值越大，则表示提取出来的水印越完整水印的鲁棒性越好。由仿真实验数据可以看出在受到不同方式的攻击时，在 MATLAB 中使用不同的方式进行攻击，包括高斯噪声、椒盐噪声、高斯滤波、中值滤波、左上剪切。对仿真结果进行剪切攻击的过程如图 9 所示。限于篇幅要求，此处不冗列出其他的攻击方式了。最后，得到的结果如表 1 所示。

图 9　剪切攻击

表 1　仿真结果经过攻击后的 *NC* 值

攻击方式	椒盐噪声	高斯滤波	左上剪切	高斯噪声
NC 值	0.968 1	0.996 9	0.988 1	0.992 6

从攻击后的 NC 值结果看出，本文经过 HDL 仿真得到的嵌入水印图像的鲁棒性非常强，在各种不同方式的攻击下，均能保持较高的 NC 值。可见本文的 FPGA 实现方法是可行的。

本设计中仍然存在不足，亦即只实现的水印嵌入算法的 FPGA 实现，由于本设计是对 FPGA 实现嵌入算法的尝试，而结果的验证是在 MATLAB 中进行的，故水印提取算法的 FPGA 实现并未涉及。但由于提取算法是嵌入算法的逆运算，HDL 代码中可以通过 FSM 等方式，改变相关的系数符号和运算流程的顺序，而不消耗多余的硬件资源，即可实现嵌入算法的逆运算。

4　结　　论

本论文设计了一种高鲁棒性的数字水印算法的 FPGA 实现，本算法的抗攻击性非常强，通过仿真验证可以看出硬件实现该水印算法是切实可行的。在很多嵌入式系统中，重负荷的算法可以放到 FPGA 中完成算法加速。本文尝试了将复杂的提升小波变换、混沌置乱加密、矩阵 SVD 分解通过 FPGA 硬件的形式实现，根据最后的实验结果分析，得到了非常好的效果。针对本设计中仍然存在的不足，将会在后面的设计中继续完善。

参考文献

[1] 刘海，陈军. 基于 QR 分解的 Contourlet 域鲁棒盲水印算法[J]. 计算机应用与软件，2016，33(6)：306-310.

[2] 包观笑，孙刘杰，于海娇. 基于拉普拉斯金字塔的数字水印防伪技术[J]. 包装工程，2016(1)：130-133.

[3] 黄晓红，张凯月，王冉，等. 基于 QR 分解和小波变换的彩色图像水印算法[J]. 河北联合大学学报：自然科学版，2016，38(2)：70-80.

[4] 毛运柳，黄东军. 基于 SURF 视频分割的视频水印算法[J]. 计算机工程，2010，36(19)：241-243.

[5] 邹其军. 基于 FPGA 的 DCT 域图像水印算法设计及其优化技术研究[D]. 南京：南京理工大学，2008.

[6] 刘国贺，李玉惠，李勃，等. 基于 FPGA 的数字图像水印实时嵌入系统的设计与实现[J]. 电子技术应用，2010，36(3)：27-30.

[7] 刘金安，金聪. 基于离散小波变换的双重水印方案[J]. 电子测量技术，2012，35(11)：45-48，66.

[8] 付炜，廖晓玉. 基于多通道小波变换的彩色图像数字水印嵌入算法研究[J]. 仪器仪表学报，2010，3(4)：824-831.

[9] 何选森，陈利，吴良敏. 基于奇偶量化的图像水印嵌入与检测方法[J]. 电子测量与仪器学报，2011，22(12)：34-39.

[10] 周美丽，白宗文. 基于 HVS 的压缩域数字水印嵌入系统的设计[J]. 国外电子测量技术，2015，34(4)：78-80.

作者简介

孙峰，工学硕士，主要研究方向为基于 FPGA 的数字图像处理、模式识别、机器学习等。

E-mail：walkstep@163.com