electronics

MDPI

Article

A Hidden DCT-Based Invisible Watermarking Method for Low-Cost Hardware Implementations

Yuxuan Wang [1], Yuanyong Luo [2], Zhongfeng Wang [1] and Hongbing Pan [1,*]

[1] School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China; b111180146@smail.nju.edu.cn (Y.W.); zfwang@nju.edu.cn (Z.W.)
[2] Huawei Corporation, Bantian, Longgang District, Shenzhen 518116, China; luoyuanyong@yeah.net
* Correspondence: phb@nju.edu.cn

Abstract: This paper presents an invisible and robust watermarking method and its hardware implementation. The proposed architecture is based on the discrete cosine transform (DCT) algorithm. Novel techniques are applied as well to reduce the computational cost of DCT and color space conversion to achieve low-cost and high-speed performance. Besides, a watermark embedder and a blind extractor are implemented in the same circuit using a resource-sharing method. Our approach is compatible with various watermarking embedding ratios, such as 1/16 and 1/64, with a PSNR of over 45 and the NC value of 1. After Joint Photographic Experts Group (JPEG) compression with a quality factor (QF) of 50, our method can achieve an NC value of 0.99. Results from a design compiler (DC) with TSMC-90 nm CMOS technology show that our design can achieve the frequency of 2.32 GHz with the area consumption of 304,980.08 μm^2 and power consumption of 508.1835 mW. For the FPGA implementation, our method achieved a frequency of 421.94 MHz. Compared with the state-of-the-art works, our design improved the frequency by 4.26 times, saved 90.2% on area and increased the power efficiency by more than 1000 fold.

Keywords: watermarking; embedder; blind extractor; invisible; robust; DCT; ASIC; FPGA

Citation: Wang, Y.; Luo, Y.; Wang, Z.; Pan, H. A Hidden DCT-Based Invisible Watermarking Method for Low-Cost Hardware Implementations. *Electronics* **2021**, *10*, 1465. https://doi.org/10.3390/electronics10121465

Received: 6 April 2021
Accepted: 9 June 2021
Published: 18 June 2021

1. Introduction

With the rapid development of the Internet and social media, the spread of digital photos around the world is becoming more and more convenient. Therefore, copyright protection is of concern. Digital watermarking is the process of hiding information in signals such as image, text, video and audio, and is used mainly for the copyright protection of digital content [1].

In recent years, many researchers have been focusing on image watermarking techniques because still images are shared widely throughout the Internet, and many image watermarking methods can also be applied to video watermarking [2]. Generally, image watermarking can be processed in two different domains: the spatial domain [3–6] and the transform domain [1,7–17]. In the spatial domain, watermark data are directly embedded to the pixel values of the host image [18]. Although spatial-domain watermarking methods are easy to implement and usually require less computational resources, they are not robust enough to attacks such as JPEG compression and geometry attacks [19]. As for the transform domain, image watermarking methods are mostly carried out in the discrete cosine transform (DCT) domain [1,7,13], the discrete Fourier transform (DFT) domain [20–22] or the discrete wave transformation (DWT) domain [9,22–26]. Watermarking in the transform domain is often based on the human vision system (HVS) [2,4,10,27,28] model to achieve invisibility and robustness. For example, human vision is more sensitive to low and middle frequency information [18], so most image compressiontechniques remove the high-frequency components which are not perceptible to save space. As a result, watermarks embedded in the low or middle frequencies are more robust against attacks. For instance, in [16], a watermarking technique that adjusts the DCT low-frequency

coefficients through the concept of mathematical remainders was proposed; it enables the watermark to be almost fully extracted even under very high compression ratios.

In this paper, we propose a mixed-strength watermark-embedding approach based-on the HVS model and the DCT approach. Theoretically, after the DCT, three types of coefficient can be chosen:

1. If the DC component is chosen, as in this paper, then there is a risk of changing the contrast of some blocks, as the DC is an average of luminance values.
2. If the low or mid-range frequency coefficients are chosen, the image quality could be affected because the human eye is especially sensitive to these frequencies.
3. If high frequencies are used, they are likely to be removed at compression time.

By specifying the pixel changing values ($\{0, \pm 1\}$ or $\{0, \pm 1, \pm 2\}$) before embedding, we can strictly control the changes in the DC components. Any visible block artifacts can be avoided in advance. Secondly, to eliminate the errors brought about by DCT and IDCT, the optimized workflow conducts DCT and IDCT implicitly. Additionally, computational effort could be saved with the hidden DCT and IDCT. Additionally, 5/6 of the color space conversion can be omitted using our DC component-based approach. As a result, the proposed approach has an obvious advantage over previous works after hardware implementation. Compared with the state-of-the-art works, our design improves the clock frequency 4.26 fold [3], saves 90.2% on area and increases the power efficiency by over 1000 fold [29]. Meanwhile, by changing the threshold and embedding strength in our approach, users can easily modify the original method to make it suitable for applications with different demands. For example, users can choose a low threshold and high embedding strength to increase robustness, and choose low embedding strength to pursue invisibility and high image quality.

The remainder of this paper is as follows. In Section 2, we list the related works and the motivation for our work. Section 3 presents some background by way of basic concepts. In Section 4, the proposed hidden DCT-based invisible watermarking method is introduced. Section 5 presents experimental results of the proposed approach and attack tests. In Section 6, we present the detailed hardware implementation results for both ASIC and FPGA platforms. The elaborated comparison with the state-of-the-art works is shown in Section 7. Section 8 gives a brief conclusion of our work.

2. Related Works and Motivation

Despite the advantages brought by transform-domain watermarking methods, the computational overheads make them hard to implement in many applications, especially on mobile phones where the computational resources are limited. Hence, many researchers have been devoted to inventing low-cost, robust watermark embedding methods. Some researchers focus on improvements in the spatial domain, and some others use novel approaches to reduce the computational burden of transform-domain watermarking.

In [29], the authors proposed an optimized image and video watermarking method using the spatial domain for low-cost applications such as wireless networks. The noise visibility function (NVF)-based mask was adopted in this paper, and the hardware implementation results show that it has significant improvements over previous works in terms of resource utilization and power consumption. However, the highest clock frequency in 90 nm CMOS technology is relatively low due to the long critical path of the divider. Additionally, one divider is shared in the whole process to save computational resources. which limits the throughput of the proposed design.

In [12], the authors used a novel DCT-based approach to achieve fast and robust watermarking. They calculated the DCT coefficient of a specific location, and watermarked bits were embedded by directly modifying the pixel values without the full-frame DCT. Test results proved that the proposed method performs well against noise attacks and compression attacks. However, a detailed hardware implementation was not presented, so its advantages over previous works in terms of area and power consumption remain unknown.

The spatial-domain watermarking and transform-domain watermarking techniques are both widely used in state-of-the-art works. Compared with transform-domain watermarking, the spatial-domain watermarking approaches are very simple and computationally efficient. However, they have relatively low information-hiding capacities and limited robustness to normal media operations, such as filtering or lossy compression. Spatial-domain watermarking approaches also have limited defense against cropping, during which some watermark information is lost [2]. The main focus of a watermarking technique is to achieve robustness and high power efficiency at the same time.

In this paper, a hidden DCT-based invisible watermarking method for low-cost hardware implementation is proposed. Traditionally, DCT-based watermarking methods are supposed to insert the watermark bits in low or middle frequencies. However, the authors of [30] argued that direct current (DC) components are more suitable for watermarking for the following reasons. Firstly, the magnitudes of DC components are much larger than those of any alternating current (AC) components in general, which makes the DC components capable of containing more watermarking information. Secondly, DC components are more robust to attacks such as data compression than the AC components. Meanwhile, the challenges of embedding watermarks in DC components of DCT can be summarized as follows:

1. Compared to doing so with AC components, embedding watermarks into the DC components of the DCT is believed to cause visible block artifacts [30], thereby compromising the image quality. Hence, the first challenge is to embed the watermark in DC components without bringing visible changes to the original picture. Theoretically, any watermarking method would change the pixel values of the target image; otherwise the watermark could not be inserted. From this perspective, if the changing values are constrained to a certain small range, embedding the watermark in the DC components will not necessarily cause more visible block artifacts than any other methods.
2. For the hardware implementation, the fixed-point DCT and IDCT are not completely reversible. DCT matrix elements contain real numbers represented by finite numbers of bits, which inevitably introduce truncation and rounding errors during computation [31]. Conventionally, the longer the bit length, the more accurate the results [32,33]. In general, a longer bit length for the DCT coefficients leads to higher energy consumption during the DCT compression process [34]. Even if we do not apply any changes after the DCT and carry out the IDCT directly, pixel values would also change by between 1 and 2 bits. Watermarks could be polluted by the errors introduced by the DCT and IDCT. Hence, the second challenge is how to optimize the watermark workflow to minimize or eliminate the errors in hardware implementation.

3. Introduction to Basic Concepts

In this section, we explain some fundamental concepts, including the DCT algorithm and the conventional DCT-based approach for digital watermarking. Meanwhile, the drawbacks of the regular approach will also be analyzed in this section.

3.1. The DCT Algorithm

The DCT algorithm is analogous to the discrete Fourier transform (DFT), but it only involves real numbers. For most natural signals (sounds and images), their energies are concentrated in the low-frequency domain after discrete cosine transformation, which is the principle of JPEG (Joint Photographic Experts Group) compression. Hence, the DCT algorithm is widely used in image processing.

The transformation formula of 2D-DCT is as follows:

$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) cos \frac{(2x+1)u\pi}{2M} cos \frac{(2y+1)v\pi}{2N}. \quad (1)$$

In Equation (1), $f(x,y)$ is defined as $\{f(x,y)|x = 0,1,2,\ldots,M-1; y = 0,1,2,\ldots,N-1\}$, representing the matrix of a two-dimensional image block. $C(u)$ and $C(v)$ are the transformation coefficients defined as:

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & u \neq 0 \end{cases}, \text{ and } C(v) = \begin{cases} \frac{1}{\sqrt{2}}, & v = 0 \\ 1, & v \neq 0 \end{cases}. \quad (2)$$

Similarly, the 2D-IDCT (inverse DCT) representation is shown below:

$$f(x,y) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) cos\frac{(2x+1)u\pi}{2M} cos\frac{(2y+1)v\pi}{2N}, \quad (3)$$

where $F(u,v)$ is defined as $\{F(u,v)|u = 0,1,2,\ldots,M-1; v = 0,1,2,\ldots,N-1\}$. $C(u)$ and $C(v)$ are the same as before.

The 2D-DCT converts a picture from the spatial domain into the frequency domain. Additionally, 2D-IDFT reverses the coefficients in frequency domain into their original pixel values.

3.2. A Conventional DCT-Based Watermarking Approach

DCT-based watermarking should have the qualities of robustness and invisibility [16]. The typical workflow [1,7] of DCT-based watermark insertion can be summarized in Figure 1. For example, an original image that is 512 × 512 is firstly divided into a series of 8 × 8 RGB blocks. After color space conversion, the Y channel of the 8 × 8 block is transformed into the frequency domain using the DCT. Meanwhile, the original watermark is prepared and encrypted for embedding. Secondly, the watermark embedder inserts the encrypted watermark data into the 8 × 8 block according to the texture, frequency and direction of image block. If the size of watermark is 64 × 64, then one pixel of the watermark will be inserted in each 8 × 8 block. At last, the embedded Y channel is inversed back into the spatial domain and converted into RGB color space along with the stored U channel and V channel.

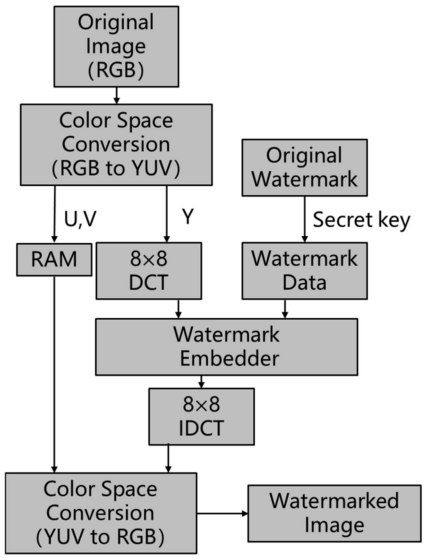

Figure 1. Traditional DCT-based watermarking workflow.

Traditional DCT-based watermarking techniques have the following three drawbacks:

1. The tradeoff between robustness and invisibility: According to the HVS theory, people cannot tell the difference between two pictures as long as the changes of pixel values are under a certain threshold. Much in the same way, there is also a

threshold when it comes to watermarking. When the embedding strength exceeds the threshold, changes of pixel values are visible and the picture begins to appear distorted. However, if the embedding strength is not high enough, the embedded watermark may not be extracted successfully because of the changes introduced to the picture during the process of color space conversion and DCT/IDCT. Considering the formulas of converting RGB signals to YUV signals:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}. \tag{4}$$

Coefficients such as -0.3313 are not represented precisely due to the bit length limits in digital signal processing systems. Pixel values may be different after color space conversion, let alone the DCT and IDCT processes. Hence, it is necessary to strike a balance between robustness and invisibility when using the DCT-based watermarking methods, restricting the applicable scenarios.

2. Computational complexity: Figure 1 reveals that each 8×8 RGB block needs to be converted twice between color spaces and twice between spatial and frequency domain. Nine multiplications and eight additions are required for each pixel to complete the conversion from RGB to YUV judging from Equation (5). For 8×8 2D-DCT, Equation (3) can be represented in matrix form as:

$$[F(u,v)] = [C][f(x,y)][C]^T, \tag{5}$$

where

$$[C] = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix}, \tag{6}$$

and

$$C_i = \frac{1}{2} cos \frac{i\pi}{16},\ i = 1, 2, \ldots, 7. \tag{7}$$

For each coefficient matrix, 64 multiplications and 56 additions are needed. Therefore, 128 multiplications and 112 additions are needed to complete a 8×8 2D-DCT. The total numbers of calculations needed for changing color spaces, spatial and frequency domain can be summarized as follows:

$$\begin{cases} Additons = (8 + 112) \times 2 = 240, \\ Multiplications = (9 + 128) \times 2 = 274. \end{cases} \tag{8}$$

Additionally, the computational complexity will be dramatically increased when the picture's size increases. Take smart phones, for example: the most advanced cell phones have up to 100 megapixel sensors. If we want to add watermarks to the original pictures taken by these cell phones with 8×8 RGB blocks, more than 312 million multiplications and additions are required.

3. Unsuitability for higher embedding ratios: Although the 1/64 embedding ratio is mostly applied in current works, a higher embedding ratio such 1/16 allows us to add more information to the host picture. Current DCT-based watermarking techniques usually add the watermark data to the low or middle frequency bands [18] after the DCT transform. This works well for the 1/64 embedding ratio but fails to achieve

satisfactory results when the embedding ratio is increased to 1/16. It can be deduced that under the 1/16 embedding ratio, the accuracy losses brought by DCT and IDCT are more severe, leading to failure of extracting the valid watermark.

4. The Proposed Method

In this section, the hidden DCT-based invisible watermarking method is proposed. Detailed analysis and proof of our method are presented.

4.1. The Proposed Watermarking Method

As stated before, a low or middle frequency watermarking method is not suitable for higher embedding ratios. Besides a low or middle frequency, watermarking information can also be inserted into the DC component of a picture block. Our proposed method first extracts the DC component. Then, the encrypted watermark is embedded into each picture block according to the texture property of the image region. If the picture block belongs to a flat region, weak embedding is adopted in order to make the watermark invisible. If the picture block belongs to a texture region, strong embedding is adopted in order to make the watermark robust. At last, the change to the DC component brought about by either weak or strong embedding is reflected in RGB channels of the final image.

The workflow of our proposed method can be seen in Figure 2. Compared with the traditional DCT-based process in Figure 1, our method removes the DCT and IDCT steps and the conversion of 5/6 of the color space (our method only needs the RGB to Y transformation).

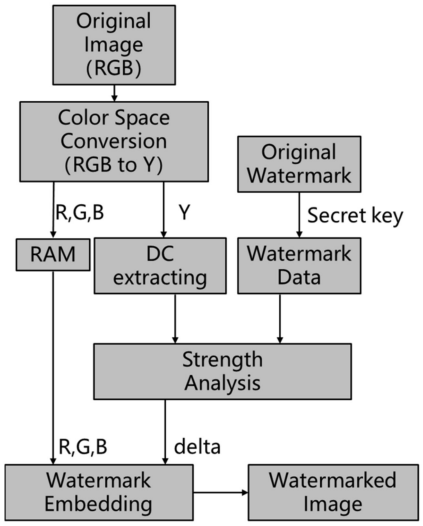

Figure 2. The proposed hidden DCT-based watermarking working flow.

4.1.1. Elimination of DCT and IDCT

The DC component is usually calculated after the picture is transformed into the frequency domain using DCT, which increases the overall computational cost. However, after analyzing the equations to calculate the DC component, we found that the process of DCT can be hidden. According to definitions of the direct current component and Equations (1) and (2), the DC value is calculated through the following equation:

$$\mathrm{DC} = \mathrm{F}(0,0) = \frac{2}{\sqrt{MN}} \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y). \tag{9}$$

Theoretically, the proposed approach can be applied to any kind of rectangular blocks. For convenience, the $N \times N$ 2D-DCT is used to demonstrate our approach, so M in (9) is replaced with N to get the following equation:

$$\mathrm{DC} = \mathrm{F}(0,0) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} Y(x,y), \tag{10}$$

where $Y(x,y)$ stands for each pixel's Y channel value.

In this way, after converting RGB signals to Y channels, it is effortless to get the DC component by summing up the Y channels of the $N \times N$ block without going through the computationally intensive DCT formula.

Watermark information is added to the DC component in our design. Assuming the change to the DC component is ΔM, the DC value after the insertion is

$$\mathrm{DC}' = \mathrm{F}\prime(0,0) = \mathrm{F}(0,0) + \Delta\mathrm{M}. \tag{11}$$

Now we need to use 2D-IDCT to calculate the corresponding Y channel values after adding the watermark. From (3), we can get that

$$Y'(x,y) = \frac{2}{N}C(u)C(v)\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F'(u,v)cos\frac{(2x+1)u\pi}{2N}cos\frac{(2y+1)v\pi}{2N}. \tag{12}$$

Since the watermark information is only added to the DC component, it can be concluded that

$$F'(u,v) = \begin{cases} F(u,v) + \Delta\mathrm{M},\ u = v = 0 \\ F(u,v),\ other \end{cases}, \tag{13}$$

where $F'(u,v)$ stands for the watermarked value and $F(u,v)$ stands for the original value without adding the watermark. After combining (12) and (13), the following equation can be inferred:

$$\begin{aligned} Y'(x,y) - Y(x,y) &= \tfrac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)(F'(u,v) - F(u,v))cos\tfrac{(2x+1)u\pi}{2N}cos\tfrac{(2y+1)v\pi}{2N} \\ &= \tfrac{2}{N}C(0)C(0)(F'(0,0) - F(0,0)) = \tfrac{\Delta\mathrm{M}}{N}. \end{aligned} \tag{14}$$

Hence, IDCT can be saved by adding $\frac{\Delta\mathrm{M}}{N}$ to its original Y channels' values. In conclusion, the proposed hidden DCT-based approach can be used to embed the watermark into the DCT domain without actually carrying out the DCT and IDCT operations.

4.1.2. Color Space Conversion

The watermark was inserted into the Y channels in the last step; now we need to convert YUV to RGB. For example, YUV can be converted to RGB using the following formula:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.344 & -0.714 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U-128 \\ V-128 \end{bmatrix}. \tag{15}$$

From (15), it is obvious that the coefficients of Y and the R channel, G channel and B channel all equal 1. This indicates that the changes to Y values brought about by the watermark will be equally reflected to R, G and B channels. Hence, after calculating $\frac{\Delta\mathrm{M}}{N}$

from the watermark, we can directly add it to R, G and B channels as listed in (16) without first applying it to Y channel and then converting the color space.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \delta, \delta = round\left(\frac{\Delta \mathrm{M}}{N}\right). \tag{16}$$

Since the watermark information can be directly added to R, G and B channels, U and V channels are no longer necessities. In this way, we only need $1/3$ of the calculations listed in (4) to get the Y channel values in order to extract the DC component. The computational cost of (15) can also be omitted. In total, $5/6$ of the computational cost can be saved using our approach when compared with traditional methods in terms of color space conversion. What is more, the calculation errors caused by color space conversion can be avoided too.

4.2. The Watermark Embedding and Blind-Extracting Approach

In order to satisfy the balance between the robustness and the imperceptivity, a mixed-strength watermark embedding approach is proposed. According to (16), the watermark can be directly added to R, G and B channels. Considering that the value changes of RGB color space can only be integers, we introduce a strong-embedding approach which brings changes to $\{0, \pm 1, \pm 2\}$ to RGB channels, and a weak-embedding approach which brings changes to $\{0, \pm 1\}$. Modular arithmetic is also used for blind extraction:

$$\begin{cases} W = 0,\ mod(\mathrm{DC}', F) \in \left[0, \frac{F}{2}\right), \\ W = 1,\ mod(\mathrm{DC}', F) \in \left[\frac{F}{2}, F\right), \end{cases} \tag{17}$$

where F is a positive integer used to distinguish the value of the watermark pixel added to a certain $N \times N$ block. For any block that does not meet the requirement in (17), $\Delta \mathrm{M}$ is combined with watermark information and added to the original DC value.

- For strong embedding, we aim to get a $\delta \in \{0, \pm 1, \pm 2\}$. According to (16), the range of $\Delta \mathrm{M_s}$ can be gotten:

$$\Delta \mathrm{M}_s \in \left[-\frac{5N}{2}, \frac{5N}{2}\right]. \tag{18}$$

Although $\delta \in \{\pm 3\}$ when $\Delta \mathrm{M}_s \in \left\{\pm \frac{5N}{2}\right\}$, $\Delta \mathrm{M}_s$ varies in a continuous range, and the probability of $\Delta \mathrm{M}_s \in \left\{\pm \frac{5N}{2}\right\}$ tends toward zero. It will not affect the distribution of δ due to principle of small probability event.

F is set as $5N$, and the following equation is used to calculate $\Delta \mathrm{M}_s$ to meet (17) and (18):

$$\Delta \mathrm{M}_s = \begin{cases} -r - \frac{5N}{4},\ W = 1, r \in \left[0, \frac{5N}{4}\right], \\ -r + \frac{5N}{4},\ W = 0, r \in \left[0, \frac{5N}{4}\right], \\ -r + \frac{15N}{4},\ W = 1, r \in \left(\frac{5N}{4}, \frac{15N}{4}\right), \\ -r + \frac{5N}{4},\ W = 0, r \in \left(\frac{5N}{4}, \frac{15N}{4}\right), \\ -r + \frac{15N}{4}, W = 1, r \in \left[\frac{15N}{4}, 5N\right), \\ -r + \frac{25N}{4},\ W = 0, r \in \left[\frac{15N}{4}, 5N\right), \end{cases} \tag{19}$$

where r stands for $mod(\mathrm{DC}, 5N)$.

From (19), the values of $mod(\mathrm{DC}', 5N)$ are constrained to $\frac{5N}{4}$ and $\frac{15N}{4}$ after modification, which are the mid-points of two judgment intervals in (17). By doing so, robustness of the watermark is increased. As after attacks, the remainder is more likely to fall into the right judgment interval during extraction if we constrain the remainder to the two mid-points when embedding. In (19), the range of r is divided into three sections labelled

as S1, S2 and S3 in Figure 3. Depending on the value of the watermark bit, different ΔM_s values are added to the original r to move it to the nearest spot according to the principle of proximity. Take (19), for example: When $W = 1$ and $r \in \left[0, \frac{5N}{4}\right]$, it does not meet (17). Although r could be moved to $\frac{15N}{4}$, the changes brought to the pixels will be bigger than when moving it to $-\frac{5N}{4}$. Hence, $\Delta M(-r - \frac{5N}{4})$ is added to the original DC value to move the remainder to its nearest spot (the red trajectory in Figure 3), and the updated r' will be:

$$r' = \mathrm{mod}\left(-\frac{5N}{4}, 5N\right) = \frac{15N}{4} \in \left[\frac{5N}{2}, 5N\right), \tag{20}$$

and

$$\Delta M_s = -r - \frac{5N}{4} \in \left[-\frac{5N}{2}, -\frac{5N}{4}\right], \tag{21}$$

which meets the requirements of (17) and (18).

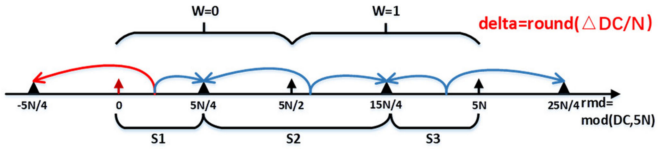

Figure 3. Dividing the range of r into three sections and modifying the DC component according to the principle of proximity.

- For weak embedding, δ should be in the range of $\{0, \pm 1\}$. The range of ΔM_w will be:

$$\Delta M_w \in \left[-\frac{3N}{2}, \frac{3N}{2}\right] \tag{22}$$

F is set as 24, and the following equation is used to calculate ΔM_w to meet (17) and (22):

$$\Delta M_w = \begin{cases} -r - \frac{3N}{4}, \ W = 1, r \in \left[0, \frac{3N}{4}\right], \\ -r + \frac{3N}{4}, \ W = 0, r \in \left[0, \frac{3N}{4}\right], \\ -r + \frac{9N}{4}, \ W = 1, r \in \left(\frac{3N}{4}, \frac{9N}{4}\right), \\ -r + \frac{3N}{4}, \ W = 0, r \in \left(\frac{3N}{4}, \frac{9N}{4}\right), \\ -r + \frac{9N}{4}, W = 1, r \in \left[\frac{9N}{4}, 3N\right), \\ -r + \frac{15N}{4}, \ W = 0, r \in \left[\frac{9N}{4}, 3N\right), \end{cases} \tag{23}$$

where r stands for $mod(\mathrm{DC}, 3N)$.

The strong-embedding approach is more robust than the weak-embedding approach because it brings more changes to the original picture block, which means that the inserted watermark is more likely to survive after attacks. However, the maximum change of value between two blocks equals four in the strong-embedding approach. For some picture blocks with small luminance or color differences, strong embedding will cause visible changes and reduce the image quality, as shown in Figure 4. To solve this problem, a mixed-strength watermark embedding approach is adopted. Parameter L is introduced to represent the image texture of a picture block, and L is defined as follows:

$$L = Y(\max) - Y(\min). \tag{24}$$

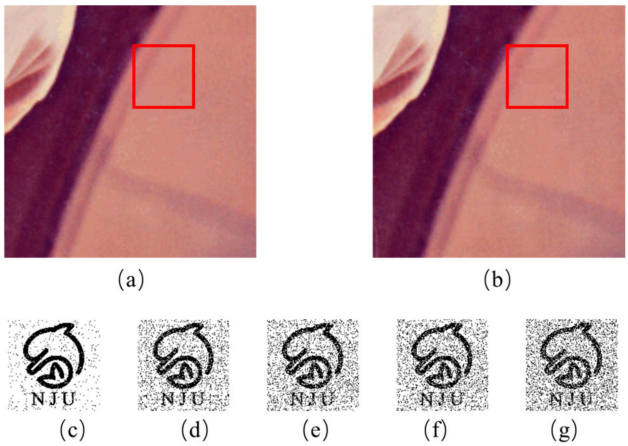

Figure 4. Watermarking experiments. (**a**) Part of the original image. (**b**) Part of the watermarked image when T is set as 0. (**c**–**g**) Extracted watermark patterns after JPEG compression (QF = 80) when T is set to 0, 8, 16, 24 and 255.

We set a threshold T to determine whether to use the strong-embedding approach or weak-embedding approach:

$$\begin{cases} L > T, \text{ strong\_embedding} \\ L \leq T, \text{ weak\_embedding} \end{cases}. \tag{25}$$

5. Experiment Results

In this section, we report experiments of the proposed method conducted on color images. All the test images were 512 × 512 pixels in size. Watermark patterns were set as 32 ×32, 64 × 64 and 128 × 128 pixels (shown in Figure 5) to test our approach in 1/256, 1/64 and 1/16 embedding ratios, respectively. In order to provide objective judgement, metrics such as peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) and normalized correlation (NC) were calculated.

Figure 5. Watermark patterns of different sizes: (**a**) 32 × 32 watermark pattern; (**b**) 64 × 64 watermark pattern; (**c**) 128 × 128 watermark pattern; (**d**) original test image.

PSNR was used to evaluate the image quality of the watermarked pictures, and it is defined as:

$$\text{PSNR} = 10log_{10}\frac{W \times H \times 255^2}{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}(X(i,j) - Y(i,j))^2}, \tag{26}$$

where $W \times H$ represents the image size. $X(i,j)$ and $Y(i,j)$ represent the original image and the watermarked image, respectively.

SSIM is a criterion that reflects the structural similarity between two pictures and is defined as:

$$\mathrm{SSIM} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}. \tag{27}$$

μ_x and μ_y represent the mean values of original and watermarked pictures. σ_x and σ_y represent their variances. Additionally, σ_{xy} represents the covariance between them. C_1 and C_2 are constants added to avoid unstable results; $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$, $k_1 = 0.01$ and $k_2 = 0.03$. L is the dynamic range of a pixel value, which was 255 in our experimental scenario. SSIM is within [0, 1], and it is 1 when two pictures are exactly the same.

NC represents the similarity between the original watermark and the extracted watermark. The definition of NC is as follows:

$$NC = \frac{\sum_{i=0}^{W-1}\sum_{i=0}^{W-1} W_1(i,j)W_2(i,j)}{\sqrt{\sum_{i=0}^{W-1}\sum_{i=0}^{W-1} {W_1}^2(i,j)\sum_{i=0}^{W-1}\sum_{i=0}^{W-1} {W_2}^2(i,j)}}, \tag{28}$$

where W_1 represents the original watermark and W_2 represents the extracted watermark. NC ranges from 0 to 1, and the extraction result is the best when NC equals 1.

5.1. Embedding Strength Tests

We first needed to determine the value of T in (25). For strong embedding $\delta \in \{0, \pm 1, \pm 2\}$, the following equation can be inferred:

$$\begin{cases} P\left((X(i,j) - Y(i,j))^2 = 0\right) = \frac{1}{5}, \\ P\left((X(i,j) - Y(i,j))^2 = 1\right) = \frac{2}{5}, \\ P\left((X(i,j) - Y(i,j))^2 = 4\right) = \frac{2}{5}, \end{cases} \tag{29}$$

where P stands for probability.

Hence, the mean square deviation (MSE) between original and watermarked pictures was 2. According to (25), the theoretical PSNR of strong-embedding approach can be calculated:

$$\mathrm{PSNR}_s = 10log_{10}\frac{512 \times 512 \times 255^2}{512 \times 512 \times 2} = 45.121. \tag{30}$$

Similarly, the theoretical PSNR of weak-embedding approach is:

$$\mathrm{PSNR}_w = 10log_{10}\frac{512 \times 512 \times 255^2}{512 \times 512 \times \frac{2}{3}} = 49.892. \tag{31}$$

Different thresholds for 1/16 embedding ratio were tested, and the results are listed in Table 1. It is obvious that with the increase of the threshold (from 0 to 255, s0 means pure strong embedding and 255 means pure weak embedding), both PSNR and SSIM were improved. PSNR, which is one of the most important criteria in terms of watermarking, saw a significant improvement. Although a satisfying PSNR of 45.1540 was achieved when the threshold was set as 0, we could still observe some visible changes in watermarked images compared with the original images. As shown in Figure 4a,b, the color block marked with a red box has more mosaics in the processed image. In addition, Lena was already a relatively low-resolution and poor-quality image; the color bumps caused by pure strong embedding will be severer in high-quality pictures. However, weak embedding cannot resist attacks effectively. In Table 1, the results of compressed watermarked images (QF = 80) are presented. When the embedding strength was decreased, the NC value gradually dropped from 0.9695 to 0.7994. We can also see from Figure 4c–g that the watermarks extracted after JPEG compression (QF = 80) had better quality when the

embedding strength was guaranteed. The extracted watermark could hardly be recognized if the threshold was set as 255. To strike a balance between robustness and invisibility, the threshold was set as 16 in following tests. Actually, the threshold can be adjusted in different application scenarios with various demands; 16 was just selected for the demonstration.

Table 1. Watermarking results using different thresholds.

Threshold	No Compression			JPEG Compression		
	PSNR	SSIM	NC	PSNR	SSIM	NC
0	45.1540	0.9997	1	31.3602	0.9933	0.9695
8	45.5052	0.9997	1	31.3608	0.9933	0.8722
16	46.9228	0.9998	1	31.3858	0.9933	0.8440
24	47.7266	0.9998	1	31.4103	0.9934	0.8378
255	49.9227	0.9999	1	31.4806	0.9935	0.7994

5.2. Attack Tests

Despite the fact that low-embedding-ratio watermarking (Figure 5a) can already meet the requirements for copyright protection, higher-resolution watermark patterns (Figure 5c) will make the evidence more convincing. Therefore, attack experiments were conducted to test the robustness of our proposed method under different embedding ratios. The testing results are listed in Table 2. JPEG compression is one of the most common attacks seen on the Internet these days. The quality factor indicates the compression ratio: the original picture was more compressed when QF is lower. From Table 2, we can see that the results of NC were the best when the size of the watermark pattern was 32×32, representing a $\frac{1}{256}$ embedding ratio. With increasing watermark size, the NC dropped under all compression ratios, but it still had a satisfying result of 0.7732 when QF = 50 with a $\frac{1}{16}$ embedding ratio. For salt and pepper noise, we can guarantee an NC of over 0.8132 when $\rho = 0.005$ (ρ represent the noise density). Actually, extraction results were still over 0.7314 when increasing ρ to 0.01. However, even when $\rho = 0.005$, the image was significantly polluted, making it completely useless for attackers. We also tested the Gaussian noise with variances (σ) of 0.001 and 0.002. The results were not so great under the $\frac{1}{16}$ embedding ratio. In particular, when $\sigma = 0.002$, the NC value dropped to 0.5365. The reason why it did not work so well at high embedding ratios is that the applied Gaussian noise had a mean value of zero. When the embedding ratio was $\frac{1}{256}$, one bit of the binary watermark was added to a 64×64 block, in which Gaussian noise had a mean value closer to zero than a 4×4 block. Hence, the proposed method did not perform well at high embedding ratios after Gaussian noise attacks. However, the results were still quite good for low embedding ratios, getting an NC value of 0.9833 when $\sigma = 0.002$. Actually, higher variances were also tested under the $\frac{1}{256}$ embedding ratio. We can still guarantee an NC value of 0.7014 when $\sigma = 0.009$, although the attacked image will be complete ruined. Geometric attacks were also tested. As a watermark is randomly distributed in the host image, different kinds of cropping have similar effects on the watermarked image. The NC value will almost equal the percentage of the picture remaining. In our testing cases, the NC values fluctuated around 0.75 when 1/4 of the picture was cropped.

Table 2. Experimental results under different attacks.

			JPEG Compression			Salt & Pepper Noise		Geometric Attacks				Gaussian Noise	
			QF = 90	**QF = 70**	**QF = 50**	**ρ = 0.001**	**ρ = 0.005**	**Center Crop 25%**	**Left Crop 50%**	**Scaling 0.75**	**Scaling 0.5**	**σ = 0.001**	**σ = 0.002**
Original		PSNR	34.0995	32.3854	31.5429	34.9369	28.1906	11.3371	9.1061	35.7365	32.4005	30.0168	27.02
		SSIM	0.9964	0.9947	0.9936	0.9971	0.9863	0.5577	0.4369	0.9975	0.9947	0.991	0.9822
$\frac{1}{256}$	Proposed	PSNR	33.7796	32.1701	31.3638	34.8813	28.1227	11.3358	9.1055	35.2872	32.1953	29.8903	26.9736
		SSIM	0.9962	0.9944	0.9933	0.997	0.986	0.5575	0.4368	0.9973	0.9944	0.9907	0.982
		NC	1	1	0.9972	0.9972	0.975	0.7569	0.5097	1	1	0.9986	0.9833
	[14]	NC	1	1	0.999	-	-	1	0.768	-	0.7056	0.9393	-
	[35]	NC	1	1	0.988	0.9938	0.9781	1	-	-	0.9042	1	-
$\frac{1}{64}$	Proposed	PSNR	33.8144	32.1899	31.3858	34.8602	28.218	11.336	9.1056	35.344	32.2299	29.9061	26.9723
		SSIM	0.9962	0.9945	0.9933	0.997	0.9863	0.5577	0.4369	0.9973	0.9945	0.9907	0.982
		NC	0.9542	0.8854	0.8296	0.9678	0.8132	0.7513	0.5084	0.9282	0.8898	0.8153	0.7359
	[18]	NC	1	0.9999	-	0.9997	-	-	-	-	0.9995	0.9997	0.9992
$\frac{1}{16}$	Proposed	PSNR	33.7798	32.1731	31.3709	34.6731	28.0016	11.3358	9.1055	35.3092	32.2322	29.8815	26.9592
		SSIM	0.9962	0.9945	0.9933	0.9969	0.9857	0.5576	0.4369	0.9973	0.9945	0.9907	0.9819
		NC	0.9973	0.8871	0.7732	0.9804	0.9063	0.7391	0.4799	0.9912	0.8929	0.6464	0.5365
	[18]	NC	1	0.9999	0.9999	0.9996	-	-	-	-	0.9745	0.9987	-

To test the scaling attack, we first scaled the picture according to the scale factor and then rescaled it to its original size. Scaling 0.5 in Table 2 means that the original 512 × 512 watermarked picture was first scaled to the size of 256 × 256. Testing results met the expectations in scaling tests, providing NC values over 0.8898 under different embedding ratios even when the picture was scaled to 1/4 of the original picture.

5.3. Discussion

From the attack tests, we can see that although our approach provided satisfying results overall, it still did not perform so well in some extreme cases. For example, after JPEG compression, the NC value was 0.7732 when QF = 50 under a $\frac{1}{16}$ embedding ratio, whereas the NC in [18] approached 0.9999. Additionally, the results under Gaussian noise attacks were also not good enough compared with other state-of-the-art works such as [18,35]. Although the embedding strength can be adjusted through changing the threshold to increase its robustness, the NC values cannot be improved much. It can be seen that even in pure strong embedding, the PSNR value still has a theoretical limit of 45.121 from (30). Robustness and invisibility are relatively antagonistic. Additionally, if we want to further improve robustness, invisibility will be compromised.

In Table 2, the default setting for $\frac{1}{16}$ embedding ratio is:

$$\begin{cases} T = 0, \\ F = 20. \end{cases} \tag{32}$$

Additionally, δ is within the range of $\{0, \pm 1, \pm 2\}$. Theoretically, if F and the range of δ are increased, robustness will be increased. As more information is added to original image and the judgement ranges for extraction in (17) are widened, the values of $mod(\mathrm{DC}', F)$ are more likely to fall in the right interval after attacks. Results of tests under different F and δ values are listed in Table 3. In Figure 6, different extracting results are presented when QF = 30. The tendency is quite clear: that by increasing the embedding strength, the embedded picture has more noise (lower PSNR values) while having better performance against attacks (higher NC values). Various kinds of combinations of threshold, F and δ values can be chosen in different applications. Moreover, more than one threshold can be set as needed; an extended, multi-staged version of the proposed method can be created.

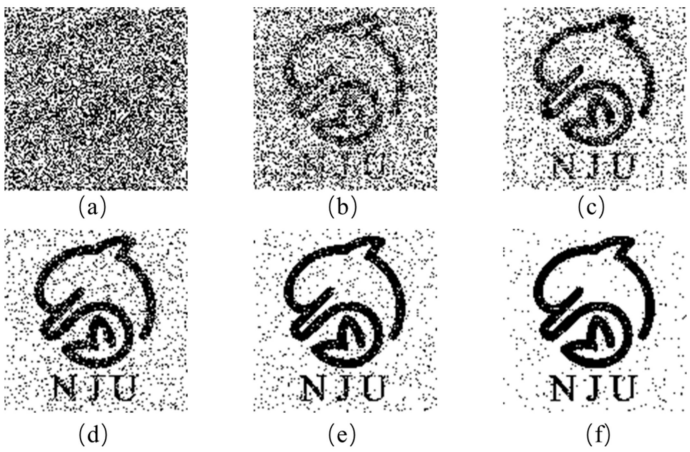

Figure 6. Extracted watermark patterns after JPEG compression (QF = 30) with a QF of 30, when F was set as 20, 28, 36, 44, 52 and 60 (**a**–**f**).

Table 3. Experimental results under different f and δ values.

Attack Type		$F = 28$, $\delta \in \{0, \pm 1, \pm 2, \pm 3\}$		$F = 36$, $\delta \in \{0, \pm 1, \pm 2 \pm 3, \pm 4\}$		$F = 44$, $\delta \in \{0, \pm 1, \pm 2 \pm 3, \pm 4, \pm 5\}$		$F = 52$, $\delta \in \{0, \pm 1, \pm 2 \pm 3, \pm 4, \pm 5, \pm 6\}$		$F = 60$, $\delta \in \{0, \pm 1, \pm 2 \pm 3, \pm 4, \pm 5, \pm 6, \pm 7\}$	
		NC	**PSNR**	**NC**	**PSNR**	**NC**	**PSNR**	**NC**	**PSNR**	**NC**	**PSNR**
None attack		1	42.0993	1	39.8976	1	38.2411	1	36.6941	1	35.3802
JPEG	QF = 50	0.9018	31.2061	0.9671	30.9891	0.9937	30.7164	0.9983	30.4488	0.9995	30.1317
	QF = 40	0.8417	30.8061	0.9180	30.6250	0.9664	30.3895	0.9922	30.0678	0.9983	29.7844
	QF = 30	0.7302	30.2563	0.8586	30.1033	0.9074	29.9036	0.9491	29.6610	0.9758	29.3746
	QF = 20	0.5817	29.3266	0.6929	29.1560	0.7927	29.0179	0.8612	28.8557	0.9020	28.6230
	QF = 10	0.4910	27.2298	0.5344	27.1598	0.5350	27.0678	0.5877	26.9764	0.7043	26.8378
Gaussian Noise	σ = 0.001	0.7967	29.7629	0.8968	29.5979	0.9536	29.4070	0.9833	29.1771	0.9940	28.9117
	σ = 0.002	0.6401	26.8875	0.7577	26.8126	0.8463	26.7230	0.9085	26.5990	0.9510	26.4391
	σ = 0.003	0.5739	25.2106	0.6666	25.1519	0.7666	25.0915	0.8302	25.0015	0.8938	24.9096
	σ = 0.004	0.5341	24.0038	0.6031	23.9540	0.6896	23.8945	0.7686	23.8419	0.8382	23.7694
	σ = 0.005	0.5142	23.0659	0.5642	23.0403	0.6532	22.9942	0.7238	22.9327	0.7927	22.8772
Combination Attack	QF = 50, σ = 0.002, ρ = 0.001, Scaling0.75	0.6111	28.7090	0.7056	28.5960	0.8121	28.4215	0.8668	28.3194	0.9093	28.1249

6. Hardware Implementation

6.1. Details of the Hardware Implementation

The block diagram of the proposed hardware design is shown in Figure 7a. The overall processing flow stayed the same as that illustrated in Section 3. The 4×4 RGB data and the one-bit watermark information arrive synchronously and are stored in the buffer for further use. Meanwhile, 4×4 RGB data are sent to the rgb_to_y module to complete the color space conversion according to (4). Then, the watermark_prepare module calculates the values of $mod(\mathrm{DC}, 12)$ and $mod(\mathrm{DC}, 20)$ by replacing N with 4, as in Section 3, for weak and strong embedding, respectively. Image texture of the 4×4 block is also calculated in this module and is compared with the threshold to determine the embedding strength of this block. Finally, the selected remainder and threshold comparison result are sent to the watermark_process module. For example, if $L > T$ (25), the strong-embedding approach is adopted. The T_result in Figure 7a must be 1 and rmd will be the value of $mod(\mathrm{DC}, 20)$. As the main computing operators used in embedding and extracting of our approach are the same (rgb_to_y module and watermark_prepare module can be used for both embedding and extracting), we made our hardware design capable of being configured in embedding mode and extracting mode with a mode selection signal. In the watermark_process module, either embedding or extracting is carried out according to the mode selection signal (1 for embedding and 0 for extracting). According to the detailed diagram of watermark_process module in Figure 7b, T_result and rmd from the watermark_prepare module are first sent to the mod_sel module to complete the bypass. In embedding mode, T_result and rmd are sent to the embed module. Operations similar to (19) or (23) are executed in the ΔM calculation module to calculate ΔM, which is added to the original RGB channels with 48 adders (the adder for every R/G/B pixel). As the value of ΔM is constrained to $\{0, \pm 1, \pm 2\}$, the length of ΔM is 3 bits. Hence, the two inputs of adders in the embed module are 8 bits and 3 bits respectively. Additionally, some extra logic components are required to make sure the computing results do not exceed $[0, 255]$. In extracting mode, operations in (17) are executed and the extracted one-bit watermark information is obtained (marked as W_extract in Figure 7b). Some other optimizations are also adopted in the hardware implementation.

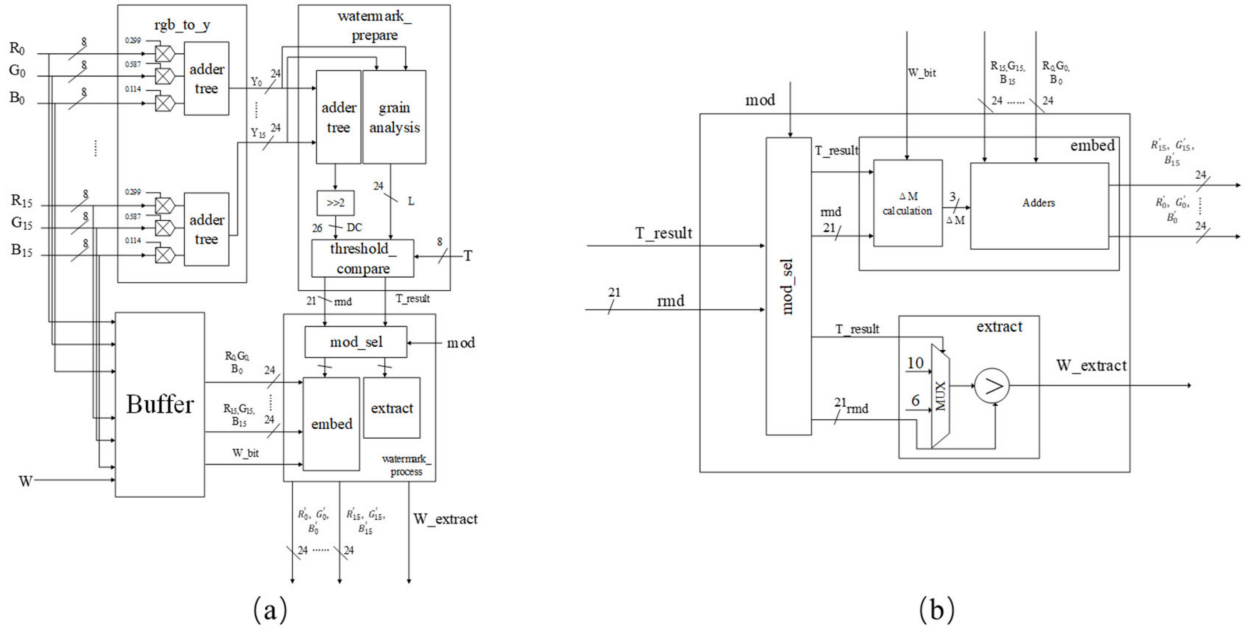

Figure 7. (**a**) Architecture of the proposed approach for a 1/16 embedding ratio. (**b**) A detailed diagram of the watermark_process module.

6.1.1. Parallel Computing and Pipeline

Our first improvement aims at latency reduction. In the rgb_to_y module, input data are the 48 8-bit RGB pieces of data of a 4×4 block. According to (4), all channels need to be multiplied with their constant coefficients. Hence, 48 constant coefficient multipliers are computed in parallel. Then, 16 three-input adder trees are applied to calculate the final Y channel data. In the watermark_prepare module, as shown in Figure 8a, the DC component and image texture of the block need to be calculated. According to (10) and (24), Y channel data are simultaneously sent to a sixteen-input adder tree and the grain analysis module to get the DC component and L, respectively. Since these two operations have no relevance, they are also carried out in parallel to reduce latency.

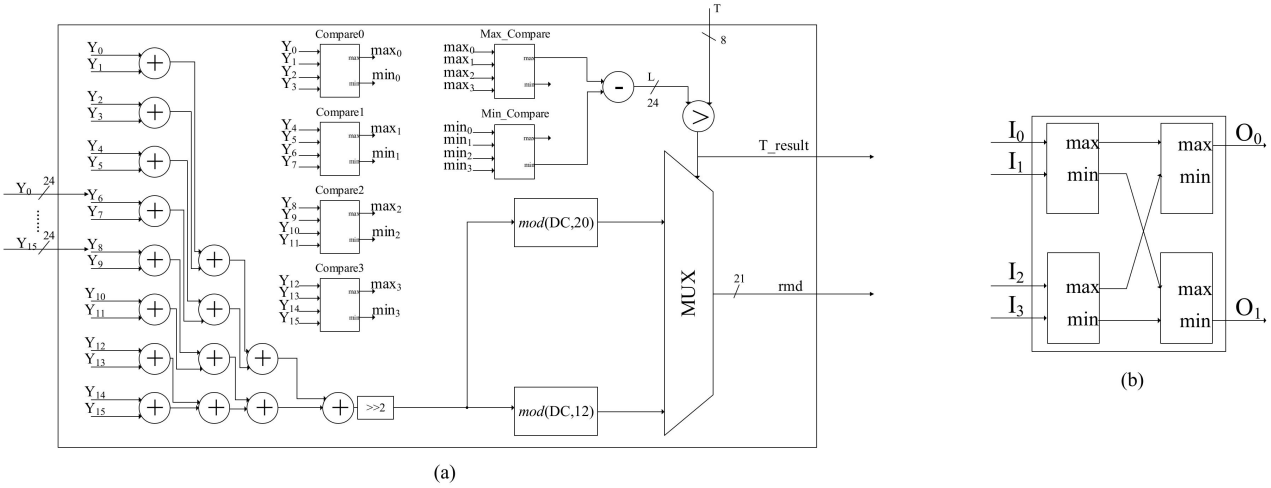

Figure 8. (**a**) Detailed diagram of watermark_prepare module. (**b**) Detailed diagram of compare module in (**a**).

When it comes to pipelining, three 128-bit-width FIFOs and one 1-bit-width FIFO are inserted as buffers to control the data flow. All FIFOs have a depth of 18 to cover the latency of the overall watermarking process. Meanwhile, adder trees and other operators are also fully pipelined to help us get maximum throughput.

6.1.2. Remainder Calculation

As mentioned before, we need to calculate the values of $mod(\mathrm{DC}, 12)$ and $mod(\mathrm{DC}, 20)$ in the watermark_prepare module to get the remainders in weak embedding and strong embedding. However, conventional approaches involve dividers which are not friendly in terms of timing and area when it comes to hardware implementations. Inspired by linear CORDIC, we introduced an iterative algorithm to compute the modulo. Take $mod(\mathrm{DC}, 20)$ as an example; then the following pseudo code can be used (Algorithm 1):

Algorithm 1 Computing the remainder using linear CORDIC.

```
for k = 1:62
    if rmd > 0
        rmd = rmd − 2^(6-k)*20;
    else
        rmd = rmd + 2^(6-k)*20;
    end
end
if rmd < 0
    rmd = 20 + rmd;
end
```

In this way, it takes seven clock cycles to calculate $mod(\mathrm{DC}, 20)$ with the critical path of an adder. Similarly, computing $mod(\mathrm{DC}, 12)$ takes eight clock cycles.

6.1.3. Timing Analysis

In the rgb_to_y module, RGB data are firstly multiplied by constant coefficients and then go through three-input adder trees. Hence, the latency of the rgb_to_y module is:

$$T_{rgb\_to\_y} = T_{mul} + 2 * T_{adder} = 3\ clocks. \tag{33}$$

For watermark_prepare, the sixteen-input adder tree needs four clock cycles to get the DC component. From Figure 8b, it can be deduced that the compare module has a latency of two clock cycles. In order to get the maximum and minimum values of the 16 pixels, two stages of comparison are needed, as shown in Figure 8a. Hence, the grain analysis module needs four clock cycles to get the maximum and minimum values, and one more clock to complete the subtraction, which adds up to five clock cycles. The latency remainder calculation needs eight clock cycles, as stated before. Additionally, another clock cycle is needed to complete the threshold comparison. As the sixteen-input adder tree and grain analysis module work in parallel, the overall latency of the watermark_prepare module is:

$$T_{watermark\_prepare} = 5 + 8 + 1 = 14\ clocks. \tag{34}$$

As for the watermar_process module, when it works in embedding mode, the calculation of ΔM is similar to (19) or (23); it needs one clock cycle, and adding ΔM to RGB channels requires another clock cycle. Hence, the latency is:

$$T_{watermark\_process\_emb} = 1 + T_{adder} = 2\ clocks. \tag{35}$$

When is works in extracting mode, according to (17), only one clock cycle is needed:

$$T_{watermark\_process\_ext} = 1\ clocks. \tag{36}$$

To summarize, the overall latency of our proposed hardware design is:

$$T_{total} = \begin{cases} 3 + 14 + 2 = 19\ clocks,\ embedding \\ 3 + 14 + 1 = 18\ clocks,\ extracting \end{cases}. \tag{37}$$

It is also worth mentioning that the buffered original data are read out of the FIFOs one clock before the final results are calculated, so the depth of the FIFOs is set as 18 to ensure that there is no overflow.

6.2. Implementation Results

Our design was coded in Verilog HDL and synthesized with TSMC 90-nm CMOS technology on the Xilinx Virtex-7 platform. It should be mentioned that the reported ASIC results are pre-layout synthesis results from Design Compiler, and the reported FPGA implementation results were estimated by the Xilinx Power Estimator (XPE) in the Vivado Design Suite. The overall architecture was as in Figure 7a. The synthesized circuit is able to process a single 4×4 block. For example, if we want to embed a 128 × 128 watermark pattern to a 512 × 512 picture with a single core, the overall latency will be:

$$T_{overall} = 19 + (128 \times 128 - 1) = 16,402\ clocks, \tag{38}$$

where 19 is the latency of the first block, and each of following blocks is sent to the process core continuously. In the image process system, DDR is used to transfer data to the process core. Take DDR3, for example: a DDR3 chip usually has a throughput of approximately 20 GB/s and the read channel consumes about half of the throughput which is 10 GB/s. As listed in Table 4, the maximum frequency of our proposed architecture is 2.32 GHz. According to (38), a single core is able to process a 512 × 512 picture within 7053 ns. Hence, a single core has the maximum throughput of 103 GB/s (141,783 fps) with an area of

304,980.08 μm^2 and power consumption of 508.1835 mW. Additionally, the only thing that limits the performance will be interface speed for ASIC applications.

Table 4. ASIC implementation results.

Freq.	Area (μm^2)	Power (mW)	Efficiency (fps/W)
1 GHz	292,349.13	215.3919	2.81×10^5
2 GHz	300,120.61	433.7370	2.81×10^5
2.32 GHz	304,980.08	508.1835	2.79×10^5

a. TSMC-90 nm CMOS technology was used for the ASIC implementation.
b. ASIC implementation results were obtained from DC reports.

For FPGA applications, the clock frequency may not be as high as with ASIC; the bottleneck could be the process speed. Since there are no interactions among all the individual blocks in our approach, we can easily apply the multicore strategy to process a large picture in a shorter time in FPGA applications. In Table 5, the frequencies and numbers of cores were set to match the 10 GB/s interface speed of DDR3. For example, when we set the frequency as 50 MHz, the throughput of a single core was 2.2 GB/s and the number of cores was set as five. Results under different frequencies are listed in Tables 4 and 5. For the ASIC implementation, area and power consumption rose with frequency. However, the power efficiency stayed around 2.8×10^5 fps/W. For FPGA implementation, although power efficiency had a significant decrease when frequency rose from 50 to 250 MHz, the area under 50 MHz was almost five times the area under 250 MHz. To conclude, our design is capable of handling the watermarking tasks in various kinds of scenarios with satisfying performance.

Table 5. FPGA implementation results.

Resources	Available	Used	Utilization	Used	Utilization	Used	Utilization
		50 MHz		100 MHz		250 MHz	
LUT	303,600	27,853	9.17%	16,712	5.50%	5936	1.96%
FF	607,200	24,598	4.05%	14,762	2.43%	4923	0.81%
Power (mW)	-	376		466		430	
Efficiency (fps/W)	-	4.05×10^4		3.92×10^4		3.54×10^4	

a. Xilinx Virtex-7 xc7vx485tffg1157-2 was used for the FPGA implementation.
b. Numbers of cores in the FPGA implementation were set to meet the transfer speed of the DDR3 chip. For 50 MHz, the core number was 5. For 100 MHz, the core number was 3. For 250 MHz, the core number was 1.
c. Power consumption was estimated by the Xilinx Power Estimator.

7. Comparisons

7.1. Hardware Implementation Comparison

Different kinds of approaches have been proposed to realize the hardware acceleration of watermark embedding. Both ASIC and FPGA have been employed for their implementation. Hence, we compare our results with the most recent works in terms of area, power consumption and energy efficiency. For the FPGA implementation, we chose the Xilinx Virtex-7 series vx485t as our target device. The criteria used for evaluating the resources of a design in FPGA applications are the numbers of LUTs, flip flops and DSPs. In order to compare our results with previous works under the same conditions, we adjusted the synthesis strategy in Vivado design tools so that DSPs were not allowed to be used. Hence, resources were constrained to LUTs and flip flops. It is also worth mentioning that in [29],

an Altera Cyclone device was used. It consumes 4582 Les, which are the logic equivalent of 4582 LUTs and 4582 single-bit registers. Compared with other works, our design has a notable advantage in saving logic resources, except for [29,36]. However, [29] showed relatively low throughput (11.8 im./sec), whereas our design is capable of reaching the throughput of 2.57×10^4 fps. FPGA implementation results show that our design achieved the highest frequency of 421.941 MHz with the power consumption of 754 mW. According to (38), the frame rate per Watt of our design equals:

$$\frac{421.941 \times 10^6}{16,402 \times 0.754} = 3.41 \times 10^4 \text{fps/W}. \tag{39}$$

As the input images were 512 × 512 color images, the throughput per Watt of our design was:

$$3.41 \times 10^4 \times 512^2 \times 8 \times 3 = 2.14 \times 10^5 \text{Mbps/W}. \tag{40}$$

Designs in [29] had the throughput of 30.1 im./sec, and the images processed were 640 × 480 grayscale images. Hence, the throughput per Watt can be deduced:

$$\frac{11.8 \times 640 \times 480 \times 8}{0.20516} = 1.41 \times 10^2 \text{Mbps/W}. \tag{41}$$

Although our design consumes more resources than [29], its throughput and throughput per Watt are both three orders of magnitude higher. Compared with [36], our design consumes less LUTs but more flip flops. However, [36] uses four more DSP48E resources than our design. Hence, it is assumed that the design in [36] and our paper are at the same level in terms of hardware utilization and power consumption. Throughput of the design in [36] was 1.676 GBps at 362.58 MHz, which is 1.34×10^4 Mbps. According to (40), the throughput of our design is 1.61×10^5 Mbps, which is one order of magnitude higher than that in [36].

Comparisons of ASIC implementation results are also listed in Table 6. Due to the DCT and IDCT and remainder calculation methods applied in our approach, the critical path of our design is the path of a constant coefficient multiplier. The maximum frequency of the synthesized circuit can reach 2.32 GHz, which is much faster than the other works listed in Table 6. Since we use registers to build the internal FIFOs, the overall power consumption reaches 508.1835 mW at 2.32 GHz. If we do not include the power of the internal buffer, the computational logic consumes 48.6163 mW at 2.32 GHz and 4.0938 mW at 200 MHz. [29] also uses 90 nm CMOS technology, but it can only achieve the maximum frequency of 166.7 MHz for its parallel version due to the introduction of arithmetic operations such as division and square root. Additionally, [29] uses only one divider in the whole design and shares it in different blocks; thus, the throughput is limited to 30.1 im./sec. On the contrary, our design can reach the maximum throughput of 141,783 fps because fully-pipelined architecture is adopted in our design. Compared with [29] in terms of throughput per Watt, our design is three orders of magnitude faster, which is consistent with the FPGA implementation results. In conclusion, the power consumption of the proposed design is lower than in most recent works, such as [29,36,37], due to the eliminations of the DCT and IDCT, and the 5/6 of the computational cost saved in color space conversion. Due to the low computational cost, parallel computing and a pipeline strategy (unlike [29], one divider is shared in all modules) can be used with the proposed method to improve the throughput of our design, which contributes to the high power efficiency.

Table 6. Hardware implementation comparison.

Work	FPGA Type	Process Domain	Watermark Type	LUTs	Flip Flops	Freq. (MHz)	Power (mW)	Efficiency (Mbps/W)
Proposed	Xilinx Virtex-7	DCT	Invisible Robust	6874	4922	421.941	754	2.14 $\times 10^5$
[5]	Xilinx Virtex-II	Spatial	Invisible Robust	1669	896	82.26	1300	-
[38]	Xilinx Spartan-3E	Spatial	Reversible	11,291	9347	98.76	750	1.386
[37]	Xilinx Virtex-7	Spatial	Reversible	50,124	38,774	445.330	1215	-
[29]	Altera Cyclone IV	Spatial	Invisible Robust	4582	4582	79.84	205.16	1.41 $\times 10^2$
[36]	Xilinx Virtex-7	DCT	Invisible Robust	7469	3405	300	-	-
[36]	Xilinx Virtex-7	DCT	Invisible Robust	29,767	7463	54.14	-	-
Work	Process Node	Process Domain	Watermark Type	Area (mm^2)		Freq. (MHz)	Power (mW)	Efficiency (Mbps/W)
Proposed (Hi Freq)	TSMC-0.09 µm	DCT	Invisible Robust	0.30498		2320	508.1835 (48.6163)	1.75 $\times 10^6$ (1.83 $\times 10^7$)
Proposed (Lo Freq)	TSMC-0.09 µm	DCT	Invisible Robust	0.28245		200	43.7063 (4.0938)	1.76 $\times 10^6$ (1.87 $\times 10^7$)
[29]	ASIC-0.09 µm	Spatial	Invisible Robust	2.89 (90.2%)		166.7	4.23	1.75 $\times 10^4$
[39]	ASIC-0.35 µm	DCT	Invisible Robust	3.064		50	62.78	-
[3]	ASIC-0.35 µm	Spatial	Invisible Robust	213.5457		545 (4.26)	2.0547	-
[40]	ASIC-0.25 µm	Spatial	Invisible Robust	16.2		70/280	0.3	-

7.2. Watermarking Performance Comparison

For the watermarking performance comparison, we mainly focus on invisibility and robustness.

The PSNR reflects the image quality of a watermarked picture. Watermarked bits can be considered as noise added to the original image. Hence, for the same picture, if the PSNR value is higher after watermarking, the invisibility of the watermark should be better. First, we compared some of the most recent works with our method using the same image Lena for the invisibility test under a 1/64 embedding ratio, which is widely used. The method in [12] works in the DCT domain, that in [18] works in the DWT domain and that in [34] is based on Tchebichef moments. As shown in Table 7, our PSNR and SSIM values are higher than those in [12,18], which indicates that the proposed method has better invisibility than [12,18].

The NC value between the extracted watermark after attack and the original watermark reflects the robustness of the watermarking approach. According to variable-controlling approach, we set the threshold to 0 and $F = 56$, $\delta \in \{0, \pm 1, \pm 2, \pm 3\}$ to get a PSNR value of 42.1097, which is still better than those in [18,35]. [14,18,35] were among the best in terms of attack tests through using DCT, DWT and Tchebichef moments, respectively. Typical attacks, such as JPEG compression, scaling, Gaussian noise and salt and pepper, were used for comparison. Experimental results are listed in Table 7. For JPEG

compression, the proposed method achieved the best result with an NC value of 1 after 50% compression. For scaling attack, the 512 × 512 image was first scaled to 256 × 256 and then rescaled to 512 × 512. The result shows that our approach was slightly worse than [18] but much better than [14] and [34]. For Gaussian noise and salt and pepper noise attacks, [18] still attained better results than our design, but the differences were very small. Overall, it can be stated that the proposed method shows great advantages in invisibility and is one of the best methods in terms of robustness. Although the DWT-based method in [18] has some slight advantages in robustness tests, the hardware overhead brought on by DWT makes it unsuitable for low-cost applications.

Table 7. Watermarking performance comparison.

Comparison Under No Attacks				
Metric	[12]	[18]	[35]	Proposed
PSNR	42.2	38.2477	40.845	45.0883
SSIM	0.97	0.9991	0.990	0.9997
NC	1	1	1	1
NC Comparison After Attacks				
Attack Type	[14]	[18]	[35]	Proposed
None Attack	1	1	1	1
JPEG (QF = 50)	0.9990	0.9998	0.9880	1
Scaling (scaling factor = 0.5)	0.7056	0.9995	0.9042	0.9973
Gaussian Noise (σ = 0.001)	0.9393	0.9997	-	0.9860
Salt & pepper (ρ = 0.001)	-	0.9985	0.9938	0.9870

8. Conclusions

In this paper, we proposed a hidden DCT-based watermarking method for low-cost hardware implementations. The proposed architecture combines a watermark embedder and a blind extractor in the same circuit using a resource-sharing method. The hardware implementation and simulation results have proven the performance of our design in terms of throughput, efficiency and accuracy. When synthesized with TSMC 90-nm CMOS technology, our proposed circuit was able to reach the frequency of 2.32 GHz, which is 4.26 times higher than the best result reported [3]. Compared to the state-of-the-art design in [29], our design improved throughput per Watt by more than 1000 fold with the energy efficiency of 1.75 $\times 10^6$ Mbps/W. In addition, invisibility and robustness tests showed that the proposed method is among the state-of-the-art methods. The principal contributions of the proposed scheme can be summarized as follows:

1. We improve the invisibility resulting from the conventional DC component-based DCT watermarking method by introducing the HVS model. Changes in DC components are strictly controlled according to the characteristics of the HVS model to avoid visible block artifacts.
2. An optimized workflow is proposed to reduce computational overhead in traditional DCT-based watermarking methods. The hidden DCT-based approach is applied to embed the watermark into the DCT domain without actually carrying out the operations. Additionally, 5/6 of the color space conversion can be omitted using our DC component-based approach. Meanwhile, image quality after watermarking can be improved because the calculation errors can also be avoided.
3. The optimized low-cost watermarking method in this paper is suitable for real-time applications with limited computing resources, such as mobile phone applications and wireless network applications [29]. It is worth noting that the proposed method

is suitable for various kinds of embedding ratios. Additionally, by adjusting the parameters such as threshold in our approach, users can easily adapt it for invisibility-oriented applications or robustness-oriented applications. For example, if we pursue robustness, we can set the threshold to 0 and choose the high F and δ values mentioned to increase the embedding strength.

In future, the proposed method should be optimized for better performance against geometry attacks and JPEG compression attacks. For example, in the cropping attack test, the proposed method did not perform as well as some state-of-the-art methods. Besides, the method in [18] showed advantages over our method in response to JPEG compression attacks. Hence, it is important for us to make efforts to improve our approach in these aspects. Additionally, the high-efficiency characteristic makes our design suitable for video watermarking applications, rather than just still pictures. Some further studies will be undertaken in the video watermarking area.

Author Contributions: Conceptualization, Y.W. and Y.L.; methodology, Y.W.; software, Y.L.; validation, Y.L. and Y.W.; formal analysis, Y.W.; investigation, Y.W.; resources, Y.W.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, Z.W.; visualization, Y.W.; supervision, H.P.; project administration, H.P.; funding acquisition, H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Nature Science Foundation of China via grants 61376075 and 41412020201; in part by the key Research and Development Program of Jiangsu Province under grant number BE2015153; and in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD). The APC was funded by the National Nature Science Foundation of China under grant number 61376075.

Acknowledgments: This work was supported in part by the National Nature Science Foundation of China via grants 61376075 and 41412020201; in part by the key Research and Development Program of Jiangsu Province under grant number BE2015153; and in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Parah, S.A.; Sheikh, J.A.; Loan, N.A.; Bhat, G.M. Robust and blind watermarking technique in DCT domain using inter-block coefficient differencing. *Digit. Signal Process.* **2016**, *53*, 11–24. [CrossRef]
2. Asikuzzaman, M.; Pickering, M.R. An Overview of Digital Video Watermarking. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 2131–2153. [CrossRef]
3. Mohanty, S.P.; Kougianos, E.; Ranganathan, N. VLSI architecture and chip for combined invisible robust and fragile watermarking. *IET Comput. Digit. Tech.* **2007**, *1*, 600–611. [CrossRef]
4. Basu, A.; Das, T.S.; Maiti, S.; Islam, N.; Sarkar, S.K. FPGA based implementation of robust spatial domassin image watermarking algorithm. In Proceedings of the 2009 4th International Conference on Computers and Devices for Communication (CODEC), Kolkata, India, 14–16 December 2009; pp. 1–4.
5. Ghosh, S.; Talapatra, S.; Chatterjee, N.; Maity, S.P.; Rahaman, H. FPGA based implementation of embedding and decoding architecture for binary watermark by spread spectrum scheme in spatial domain. *Bonfring Int. J. Adv. Image Process.* **2012**, *2*, 1–8. [CrossRef]
6. Karybali, I.G.; Berberidis, K. Efficient spatial image watermarking via new perceptual masking and blind detection schemes. *IEEE Trans. Inf. Forensics Secur.* **2006**, *1*, 256–274. [CrossRef]
7. Shoshan, Y.; Fish, A.; Jullien, G.A.; Yadid-Pecht, O. Hardware implementation of a DCT watermark for CMOS image sensors. In Proceedings of the 2008 15th IEEE International Conference on Electronics, Circuits and Systems, Saint Julian's, Malta, 31 August–3 September 2008; pp. 368–371.
8. Erozan, A.T.; Başkır, S.G.; Örs, B. Hardware/Software codesign for watermarking in DCT domain. In Proceedings of the 2013 21st Signal. Processing and Communications Applications Conference (SIU), Haspolat, Turkey, 24–26 April 2013; pp. 1–4.
9. Kiran, S.; Sri, K.V.N.; Jaya, J. Design and implementation of FPGA based invisible image watermarking encoder using wavelet transformation. In Proceedings of the 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), Coimbatore, India, 3 July 2013; pp. 323–325.
10. Mohanty, S.P.; Ramakrishnan, J.R.; Kankanhalli, M.S. A DCT domain visible watermarking technique for images. *Proc. ICME* **2000**, *2*, 1029–1032.

11. Susanto, A.; Setiadi, D.R.I.M.; Rachmawanto, E.H.; Mulyono, I.U.W.; Sari, C.A. An Improve Image Watermarking using Random Spread Technique and Discrete Cosine Transform. In Proceedings of the 2019 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 24–25 July 2019; pp. 168–173.
12. Byun, S.; Son, H.; Lee, S. Fast and Robust Watermarking Method Based on DCT Specific Location. *IEEE Access* **2019**, *7*, 100706–100718. [CrossRef]
13. Hosseini, S.A.; Saboori, A. A new method for color image watermarking based on combination of DCT and PCA. In Proceedings of the 2015 International Conference on Communications, Signal. Processing, and their Applications (ICCSPA'15), Sharjah, United Arab Emirates, 17–19 February 2015; pp. 1–5.
14. Ernawan, F.; Kabir, M.N. A Robust Image Watermarking Technique With an Optimal DCT-Psychovisual Threshold. *IEEE Access* **2018**, *6*, 20464–20480. [CrossRef]
15. Ahmaderaghi, B.; Kurugollu, F.; Rincon, J.M.D.; Bouridane, A. Blind Image Watermark Detection Algorithm Based on Discrete Shearlet Transform Using Statistical Decision Theory. *IEEE Trans. Comput. Imaging* **2018**, *4*, 46–59. [CrossRef]
16. Lin, S.D.; Shie, S.; Guo, H.Y. Improving the robustness of DCT-based image watermarking against JPEG compression. In Proceedings of the 2005 Digest of Technical Papers. International Conference on Consumer Electronics, Las Vegas, NV, USA, 8–12 January 2005; pp. 343–344.
17. Liu, S.; Pan, Z.; Song, H. Digital image watermarking method based on DCT and fractal encoding. *IET Image Process.* **2017**, *11*, 815–821. [CrossRef]
18. Liu, J.; Huang, J.; Luo, Y.; Cao, L.; Yang, S.; Wei, D.; Zhou, R. An optimized image watermarking method based on HD and SVD in DWT domain. *IEEE Access* **2019**, *7*, 80849–80860. [CrossRef]
19. Nikolaidis, N.; Pitas, I. Robust image watermarking in the spatial domain. *Signal. Process. Sp. Issue Copyr. Prot. Access Control.* **1998**, *66*, 385–403. [CrossRef]
20. Tao, P.; Eskicioglu, A.M. An adaptive method for image recovery in the DFT domain. *J. Multimed.* **2006**, *1*, 36–45. [CrossRef]
21. Tsui, T.K.; Zhang, X.-P.; Androutsos, D. Color image watermarking using multidimensional Fourier transforms. *IEEE Trans. Inf. Forensic Secur.* **2008**, *3*, 16–28. [CrossRef]
22. Kang, X.; Huang, J.; Shi, Y.Q.; Lin, Y. A DWT-DFT composite watermarking scheme robust to both affine transform and JPEG compression. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 776–786. [CrossRef]
23. Hu, H.-T.; Hsu, L.-Y. Collective blind image watermarking in DWTDCT domain with adaptive embedding strength governed by quality metrics. *Multimed. Tools Appl.* **2016**, *76*, 6575–6594. [CrossRef]
24. Ghosh, S.; Biswas, A.; Maity, S.P.; Rahaman, H. Design of a Low Complexity and Fast Hardware Architecture for Digital Image Watermarking in FWHT Domain on FPGA. In Proceedings of the 2014 Fifth International Symposium on Electronic System Design, Surathkal, India, 15–17 December 2014; pp. 68–72.
25. Maity, G.K.; Jana, P.; Mandal, H.; Chiu, T.-L. Power-aware VLSI design of reversible watermarking for access control. *Microsyst. Technol.* **2019**. [CrossRef]
26. Bhinder, P; Jindal, N.; Singh, K. An improved robust image-adaptive watermarking with two watermarks using statistical decoder. *Multimed Tools Appl.* **2020**, *79*, 183–217. [CrossRef]
27. Amiri, M.D.; Meghdadi, M.; Amiri, A. HVS-based scalable image watermarking. *Multimed. Tools Appl. Multimed. Tools Appl.* **2019**, *78*, 7097–7124. [CrossRef]
28. Li, J.; Zhang, H.; Wang, J.; Xiao, Y.; Wan, W. Orientation-Aware Saliency Guided JND Model for Robust Image Watermarking. *IEEE Access* **2019**, *7*, 41261–41272. [CrossRef]
29. Pexaras, K.; Karybali, I.G.; Kalligeros, E. Optimization and Hardware Implementation of Image and Video Watermarking for Low-Cost Applications. *IEEE Trans. Circuits Syst. I Regul. Papers* **2019**, *66*, 2088–2101. [CrossRef]
30. Huang, J.; Shi, Y.Q.; Shi, Y. Embedding image watermarks in dc components. *IEEE Trans. Circuits Syst. Video Technol.* **2000**, *10*, 974–979. [CrossRef]
31. Ruiz, G.A.; Michell, J.A.; Buron, A.M. Parallel-pipeline 8/spl times/8 forward 2-D ICT processor chip for image coding. *IEEE Trans. Signal. Process.* **2005**, *53*, 714–723. [CrossRef]
32. Wu, F.S.; Cham, W.K. A comparison of error behaviour in the implementation of the DCT and the ICT. In Proceedings of the IEEE TENCON'90: 1990 IEEE Region 10 Conference on Computer and Communication Systems. Conference Proceedings, Hong Kong, 24–27 September 1990; Volume 2, pp. 450–453. [CrossRef]
33. Shoushun, C.; Bermak, A.; Yan, W.; Martinez, D. Adaptive-Quantization Digital Image Sensor for Low-Power Image Compression. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2007**, *54*, 13–25. [CrossRef]
34. Tang, T.; Goh, W.L.; Liu, X.; Wang, C. A 0.18µm, 0.6V, 83.5µW integer DCT processor for neural signal applications. In Proceedings of the 2016 International Symposium on Integrated Circuits (ISIC), Singapore, 12–14 December 2016; pp. 1–4. [CrossRef]
35. Ernawan, F.; Kabir, M.N. An Improved Watermarking Technique for Copyright Protection Based on Tchebichef Moments. *IEEE Access* **2019**, *7*, 151985–152003. [CrossRef]
36. Cao, Y.; Yu, F.; Tang, Y. A Digital Watermarking Encryption Technique Based on FPGA Cloud Accelerator. *IEEE Access* **2020**, *8*, 11800–11814. [CrossRef]
37. Hazra, S.; Ghosh, S.; De, S.; Rahaman, H. FPGA implementation of semi-fragile reversible watermarking by histogram bin shifting in real time. *J. Real-Time Image Process.* **2018**, *14*, 193–221. [CrossRef]

38. Maity, H.K.; Maity, S.P. FPGA implementation of reversible watermarking in digital images using reversible contrast mapping. *J. Syst. Softw.* **2014**, *96*, 93–104. [CrossRef]
39. Tsai, T.H.; Lu, C.Y. A System Level Design for Embedded Watermark Technique using DSC System. In Proceedings of the IEEE International Workshop on Intelligent Signal Processing and Communication System, Nashville, TN, USA, 9–12 November 2001.
40. Mohanty, S.P.; Ranganathan, N.; Balakrishnan, K. A dual voltage frequency VLSI chip for image watermarking in DCT domain. *IEEE Trans. Circuits Syst. II* **2006**, *53*, 394–398. [CrossRef]