Hardware Architecture for Video Authentication using Sensor Pattern Noise

Amit Pande *Member, IEEE*, Shaxun Chen, Prasant Mohapatra *Fellow, IEEE*, and Joseph Zambreno, *Senior Member, IEEE*

***Abstract*—Digital camera identification can be accomplished based on sensor pattern noise which is unique to a device and serves as a distinct identification fingerprint. Camera identification and authentication has formed the basis of image / video forensics in legal proceedings. Unfortunately, real-time video source identification is a computationally heavy task, and does not scale well to conventional software implementations on typical embedded devices. In this paper, we propose a hardware architecture for source identification in networked cameras. The underlying algorithms, an orthogonal forward and inverse Discrete Wavelet Transform (DWT) and Minimum Mean Square Error (MMSE) based Estimation have been optimized for 2D frame sequences in terms of area and throughput performance. We exploit parallelism, pipelining and hardware reuse techniques to minimize hardware resource utilization and increase the achievable throughput of the design. A prototype implementation on a Xilinx Virtex-6 FPGA device was optimized with a resulting throughput of 167 MBps, processing 30 640×480 video frames in 0.17 second.**

***Index Terms*—digital camera identification, hardware architecture, video security**

I. Introduction

DIGITAL camera identification has multiple applications in real-world scenarios. For example, when presenting a video clip as evidence in a court of law, identifying the source (acquisition device) of the video is as important as the video itself [1]. Not doing so can lead to legal challenges which may render the evidence invalid. Another example is the movie industry, where significant revenue losses are caused every year by secretive recording in movie theaters and the subsequent illegal distribution. Video source identification can be employed to track down such piracy crimes [2], [3]. Similarly, images or videos shared using Flickr, Facebook or other social networking sites or through personal email can be authenticated and tied to the user device (in this case, the smartphone or personal camera).

Easier access to high-quality digital camcorders and sophisticated video editing tools further motivates the improvement of video source identification techniques. The issue of digital image or video authentication can be approached in several different manners.

A. Pande, S. Chen and P. Mohapatra are with the Department of Computer Science, University of California Davis, 2063 Kemper Hall, One Shelds Ave, Davis, CA-95616 USA. Phone: (530) 752-0870, Fax: (530) 754-4767, e-mail: {pande, sxch, pmohapatra}@ucdavis.edu,

J. Zambreno is with the Department of Electrical and Computer Engineering, 2215 Coover Hall, Iowa State University, Ames, IA-50011, USA. e-mail: zambreno@iastate.edu

Manuscript received 7 March, 2013; revised 28 April and 11 June 2013; accepted 07 July 2013

The simplest strategy would be to inspect the digital file itself and look for header clues or any other attached information. The EXIF header format [4], supported by many camera manufacturers contains information about the digital camera type and geo-location. However, this header data is unavailable if the video is transcoded or re-compressed. Moreover, such tags can trivially be modified by software.

Another strategy is to equip digital cameras with an invisible, yet fragile watermark carrying information about camera, location, time and personal biometric data. Such approaches are used in some high-end cameras by Epson, Kodak and Canon [5], [6]. However, not every camera is equipped with such sensors. The existing deployments such as surveillance camera networks or commercial image sharing in smartphones are not equipped with such 'secure-cameras'.

The most reliable method reported so far for video source identification is based on the sensor pattern noise which is unique to each camera. This noise results from the non-uniformity of each sensor pixel's sensitivity to light, and can be treated as the inherent fingerprint of a video capture device [7].

The scheme presented in [7] involves image denoising using the Discrete Wavelet Transform (DWT) followed by subband level denoising using MMSE estimation procedure. In our experiments, we found computational requirements leading to large processing time (in the order of seconds per frame on multicore desktops for small resolution videos). The 'db8' DWT filter used in [7] has high computational requirements owing to the presence of irrational coefficients and a large number of taps. The MMSE estimation task uses 2D processing and is the most computationally expensive task (taking 99% of the entire processing time). Processing a single video frame (640×480 resolution) on an Intel core i7 laptop takes about 5 seconds, giving an effective throughput of only 184 KBps.

The expensive computation overhead will become a bottleneck when fast identification is needed. An example is detecting video camera spoofing attacks using source identification techniques. An adversary can compromise a legitimate camera, and then send fake video to the sink using the victim's identity. Such an attack is called camera spoofing attack, which introduces severe security threats if the camera is used for surveillance or other security purpose. Moreover, given the increasing popularity of wireless video cameras, such attacks are becoming easier to launch. The sensor pattern noise based source identification method is naturally a good candidate to detect this attack; however, it requires performing source identification in a real-time fashion.

In this paper, we propose a hardware architecture for video authentication using a pixel-non-linearity noise scheme, and illustrate an acceleration of this task using a conventional hardware implementation. Our prototype implementation was mapped to a Xilinx Virtex-6 FPGA XC6VLX75. The main features of our hardware implementation are as follows:

1) We propose a modified filter bank based implementation of the orthogonal 'db2' DWT filter and compare its performance with the existing 'db8' filter for image denoising. With an insignificant decrease in authentication or source identification accuracy, we find enormous savings in hardware (7× savings in number of multipliers in design) and improvements in clock frequency (2.65×).
2) We propose a 2D systolic array based implementation of an 'image denoising' algorithm. The proposed architecture uses parallelism, pipelining and hardware reuse to obtain a clock frequency of 397 MHz. This implies a processing time of 0.07 second per 1 second of video at 640×480 resolution and 30 fps, making it possible to identify source cameras in real-time scenarios.

The rest of the paper is organized as follows: Section II gives an overview of existing work in this direction. Section III presents an overview of existing algorithm and our approach. Section IV describes architectural details of DWT implementation while Section V gives details for MMSE implementation. Section VI evaluates the performance of proposed implementation in terms of accuracy in detection of source cameras and shows the results of prototype implementation on a Xilinx Virtex 6 XC6VLX75 FPGA. Section VII concludes the paper.

II. Literature Review

A. Source Identification in Images and Videos

The research on image source identification emerged a few years prior to video source identification, and the techniques are often similar. Kharrazi et al. [8] proposed a novel idea for camera identification based on supervised learning. They compute image features in spatial and wavelet domain and then train a Support-Vector-Classifier to find camera model. A multi-class SVM classifier is used to identify and classify images from 5 different cameras with an accuracy of $78 - 85\%$. Similarly, Celiktutan et al. [9] defined a set of similarity measures using KNN and SVM for classification operation. Choi et al. [10] include intrinsic lens radial distortions as part of the features and improve classification. Popescu [11] uses the Expectation Maximization algorithm to identify the demosaicing algorithm that a camera uses, based on which different image sources are classified. However, all these methods are only capable of detecting the model or the manufacturer of the device, instead of identifying the individual camera that produced the image.

The following techniques focus on the specific device identification, which is desirable for the forensic applications. The Canon Data Verification Kit [6] calculates the hash of images and uses a special secure memory card to enable tracing the image to a camera, but only high-end Canon DSLR cameras support this solution. The same applies to embedding watermarks into images, which is only applicable for specially designed devices rather than commodity devices.

Geradts et al. [12] proposed to utilize sensor hot pixels or dead pixels to identify the image source. It performs nicely even for JPEG compressed images. However, all cameras do not have such defective pixels, and many cameras post-process to remove such defects from output images.

Kurosawa et al. [13] measured the dark current noise of the sensor and used it as the device fingerprint. Since the dark current noise can only be extracted from dark frames, this method is restricted to the videos that contain dark frames.

Lukas et al. [7] employed sensor pattern noise as an inherent fingerprint of the camera for source identification. More specifically, they use Photo-Response Non-Linearity (PRNU) noise to identify the individual video camera. So far, the sensor pattern noise based schemes report the most reliable results. Kang et al. [14] model this noise as a white noise signal to improve the detection statistics in cases of images suffering from interference and losses by JPEG compression and the camera signal processing. Li [15] proposed to use adaptive weighting to improve the performance of this approach. Recent work by Li et al. [16] consider the interference caused by interpolation process in color filter arrays in PRNU extraction and propose a color-decoupled PRNU extraction process.

Chen et al. [3] extend this prior work to networked videos. However, they require as long as 10 minutes of processing time for low resolution (264×352) and 40 seconds for higher resolution (536×720) videos. The work of [17] improves this value to 10 seconds ($\sim 300 - 400$ frames) using network characteristics.

Houten et al. [18] uses this algorithm to classify multiply compressed networked videos of length 30 s with satisfactory performance. Hyun et al. [19] propose a new MACE (Minimum Average Correlation Energy) filter to improve correlation efficiency in Chen's algorithm.

B. Hardware Architectures

To the best of our knowledge, there is no related work in this direction. There has been work in direction of optimizing the DWT implementation in hardware [20]–[22], but it mainly focused on image and video compression applications, not on denoising. There are a few implementations of image denoising using FPGA [23], [24]. The underlying algorithm used for denoising [25] is based on DWT and MMSE estimation and has been shown to be efficient to extract PRNU noise. Burg et al. [26] present a VLSI architecture for MMSE estimation with a focus on MIMO-OFDM systems.

III. Overview

An overview of the basic algorithm is explained in Fig. 1. A surveillance camera or smartphone takes still images or video frames (N) and transmits them to a central station using the network infrastructure. The camera has the following elements: lens, Color Filter Array (CFA), sensor, demosaicing algorithm and compression Codec. The lens captures light rays from the view and passes it on to CFA. The light is next passed through sensor which adds various noise to the image or video frames, namely shot noise, fixed-pattern noise and Photo-Response Non-Uniformity (PRNU) noise. Shot noise is a random component which can be averaged out and removed. It varies amongst subsequent frames shot from a camera and is therefore not reliable. Fixed-pattern noise is caused by pixel-to-pixel difference when the sensor is not exposed to any light. However, real-life images and videos do not correspond to such scenes as they are typically full of real-world information. Also, this technique is affected by humidity and temperature, and is therefore non-reliable for video authentication.

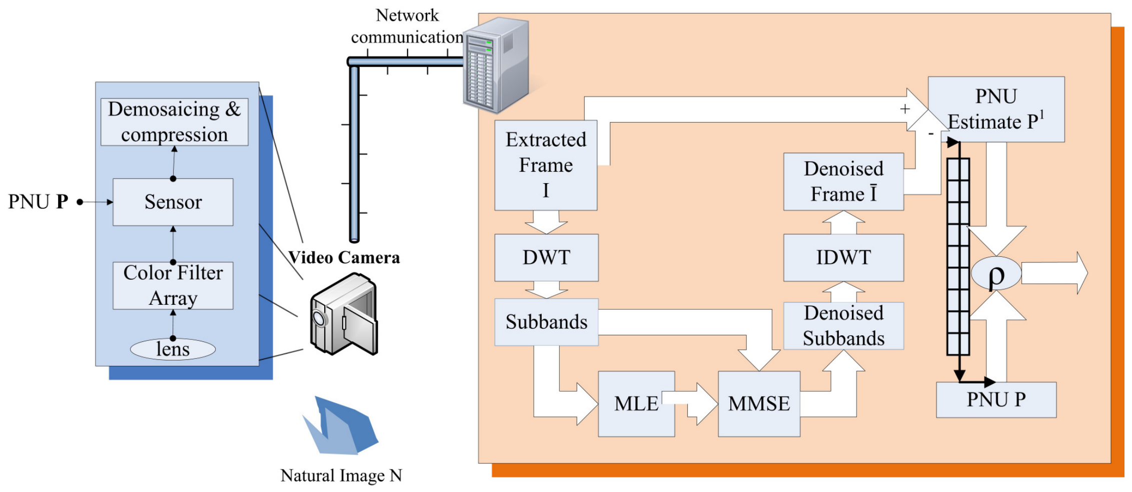

Fig. 1. Block Diagram of video communication and authentication setup. A surveillance camera / smartphone takes shots (N) and transmits it to a central station using the network infrastructure. The camera inherently inserts a PNU (P) into the captured image/ video which can be extracted to authenticate that the video is shot using desired camera. The extraction step involves Forward and Inverse Discrete Wavelet Transform (DWT and IDWT), image denoising using Maximum Likelihood Estimation (MLE) and Minimum Mean Square Estimation (MMSE) and PNU estimation (P^1). It is then compared (correlated ρ) with reference PNU P. Multiple frames are averaged before correlation in the case of video, due to the small resolution and high compression ratios.

The PRNU noise has a low frequency component caused by light refraction in air, dust and optical surfaces and zoom settings. Apart from this, it has a Pixel Non-Uniformity (PNU) Noise (P) component due to the varying sensitivity of pixels to light caused by the inhomogenity of silicon wafers and imperfections during the sensor manufacturing process. Thus, PNU noise is unique to the device and independent of temperature and humidity, making it an excellent choice for video authentication purposes. Detailed description of these noises and modeling is presented in [7].

The camera inherently inserts a PNU footprint (P) into the captured image or video which we plan to use for authentication purposes. The extraction step involves Forward and Inverse Discrete Wavelet Transform (DWT and IDWT) to convert the image to frequency subbands which are then processed independently. The denoising model assumes mixture process of independent component fields having zero mean, unknown variance Gaussian distribution and is shown to be highly robust and effective. A Maximum Likelihood Estimate (MLE) is used for variance estimation of each subband followed by Minimum Mean Square Error (MMSE) estimation procedure. The denoised frame is subtracted from the extracted frame to get an estimate of the PNU watermark. P_i^1 stands for the estimated PNU of the i-th image of a camera. The reference watermark P is not directly accessible from most cameras, therefore we estimate it by averaging the past values of P_i^1 over a large number of past samples ($N \geq 50$).

$$P = \frac{1}{N}\sum_{i=1}^{N} P_i^1$$

For high resolution images, P^1 itself can be computed from single image. For low resolution videos, P_i^1 stands for the estimated PNU of the i-th frame of a video camera averaged over a window of M consecutive frames .

$$P_i^1 = \frac{1}{M}\sum_{j=1}^{M} P_{i+j}^1$$

Additional details of algorithm and hardware implementation are presented in subsequent sections.

The process of denoising typically removes all noise and extracts the original image or frame. Wavelet-domain denoising using MMSE estimation has been found to be very efficient in this task and takes care of natural image boundaries which appear in other denoising algorithms as part of noise pattern. The denoised frame $\overline{I} \approx N$ is subtracted from the extracted frame (EF) to obtain an estimate of the PNU (P^1)

$$I = N + P + noise$$

$$P^1 = I - \overline{I} = P + noise$$

IV. Discrete Wavelet Transform

The obvious first and last step in denoising is the forward and inverse Discrete Wavelet Transform operation.

A. Background

Applying a 2-D DWT to an image of resolution $M \times N$ results in four images of dimensions $\frac{M}{2} \times \frac{N}{2}$: three are detailed images along the horizontal (LH), vertical (HL) and diagonal (HH), and one is a coarse approximation (LL) of the original image. LL represents the low frequency component of the image, while LH, HL, and HH represent the high frequency components. This LL image can be further decomposed by DWT operation. Three levels of such transforms are applied and shown in Fig. 2. The coarse information is preserved in the LL3 image and this operation forms the basis of Multi-Resolution Analysis for DWT [27].

Bi-orthogonal Wavelet Filter Banks (BWFBs) are commonly used for DWT for image compression but as they have irrational coefficients, the associated DWT requires a high precision implementation, leading to an increased computational complexity. In a hardware implementation, rational binary coefficients can help in achieving a multiplier-free implementation of filter coefficients [21], [28], [29]. These multiplier-free implementations

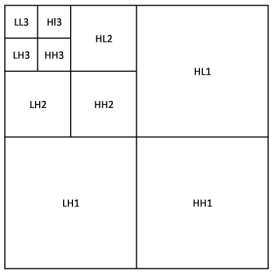

Fig. 2. Result of three level 2-D wavelet transform operation on an image

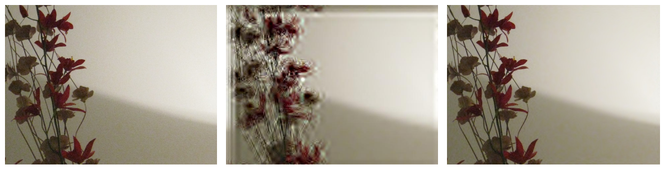

Fig. 3. Extra distortions are obtained when using optimized biorthogonal wavelets for image denoising. (a) Original image, (b) Denoised using biorthogonal filter [21], (c) orthogonal filter [7]

and other optimizations [30]–[32] involve image reconstruction quality trade-offs and have not been tested for denoising applications.

B. Hardware Implementation of DWT

Much research has been done in the development of DWT architectures for image processing [21], [29], [30], [33]–[35]. A good survey on architectures for DWT coding is given by [36], however the focus has been primarily on image compression applications.

The DWT architectures can be broadly classified into lifting based, convolution-based and B-spline based architectures. The lifting based architectures are popular and became the mainstream because they need fewer multipliers and adders and have a regular structure. Similarly B-spline-based architectures have been proposed to minimize the number of multipliers by using B-spline factorization [37]. However, the lifting based architecture has a larger critical path. Convolution-based approaches have a lower critical path but require a larger number of multipliers.

These filters designed for image compression and efficient implementation degrades quickly for image denoising applications. The 9/7 poly-DWT filter in [21] has best known image compression and hardware-efficient implementation. Figure 3 shows this effect where denoising causes distortions when using 9/7 filter. This is because denoising applications typically use orthogonal wavelets while compression codecs use CDF 9/7 and similar filters which are based on bi-orthogonal wavelet construction.

C. Filter Design

In [7], the authors propose using ‘db8’ orthogonal wavelet for denoising operation. Named after Ingrid **Dau**b**e**chies who did monumental research on wavelets and their applications, ‘db8’ is an orthogonal and asymmetric wavelet filter. The filter coefficients are irrational and asymmetric and 16 taps are present in both decomposition low pass Lo_D and high pass Hi_D filters. The coefficients are given in Table I. They are all distinct

TABLE I
COEFFICIENTS OF ‘DB8’ WAVELET FILTER

i	Lo_D	Hi_D
1	-0.000117477	-0.054415842
2	0.000675449	0.312871591
3	-0.00039174	-0.675630736
4	-0.004870353	0.585354684
5	0.008746094	0.015829105
6	0.013981028	-0.284015543
7	-0.044088254	-0.000472485
8	-0.017369301	0.128747427
9	0.128747427	0.017369301
10	0.000472485	-0.044088254
11	-0.284015543	-0.013981028
12	-0.015829105	0.008746094
13	0.585354684	0.004870353
14	0.675630736	-0.00039174
15	0.312871591	-0.000675449
16	0.054415842	-0.000117477

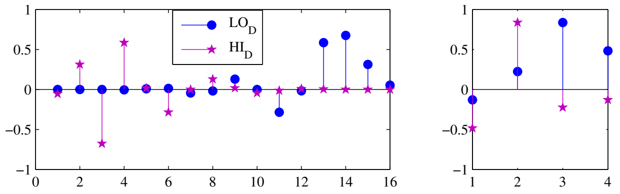

Fig. 4. ‘db8’ and ‘db2’ filter coefficients respectively: Low pass Lo_D and high pass Hi_D

and irrational (truncated values are shown). Consequently, a direct implementation in hardware will require 16 multipliers and subsequent 15 adders to get a high or low pass output. The filter is asymmetric and no coefficients are same across high and low pass filter (see Figure 4). Use of 32 multipliers and 30 adders to obtain a single level of wavelet decomposition will lead to significant area and computational requirements. It is also possible to represent ‘db8’ filter coefficients using lattice implementation as follows:

$$\begin{bmatrix} Lo_D(z) \\ Hi_D(z) \end{bmatrix} = \sum_{i=2}^{8} \left(\begin{bmatrix} 1 & -\alpha_i \\ \alpha_i & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-2} \end{bmatrix} \right) \begin{bmatrix} 1 & -\alpha_1 \\ \alpha_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ z_{-1} \end{bmatrix} \tag{1}$$

where $\alpha_i, i \in \{1, ...8\}$ are the lattice coefficients. This implementation on hardware will require 16 multipliers but will greatly reduce the throughput and latency owing to large critical path (for low pass filter).

We want to simplify this design, leading to area, computational and power savings in the design. For denoising applications, it is not possible to simplify the coefficients, an approach presented in [21], [38], because that will lead to visible distortions.

Rather, we propose to use ‘db2’ filters. The filter coefficients for the ‘db2’ filter are represented as:

$$Lo_D(z) = a_1 + a_2 z^{-1} + a_3 z^{-2} + a_4 z^{-3}$$
$$Hi_D(z) = b_1 + b_2 z^{-1} + b_3 z^{-2} + b_4 z^{-3}$$

TABLE II
HARDWARE REQUIREMENTS OF DWT FILTERS

	'db8' FB	'db2' lattice	'db2' FB	'db2' MFB
Add.	32	5	6	6
Mult.	30	5	8	4

where a_1, a_2, a_3 and a_4 are low pass filter coefficients and $b_4 = a_1$, $b_3 = -a_2$, $b_2 = a_3$ and $b_1 = -a_4$ respectively. The lattice representation of the same filter is given by following equations:

$$\begin{bmatrix} Lo_D(z) \\ Hi_D(z) \end{bmatrix} = K \begin{bmatrix} 1 & -\alpha_2 \\ \alpha_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-2} \end{bmatrix} \begin{bmatrix} 1 & -\alpha_1 \\ \alpha_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ z_{-1} \end{bmatrix} \tag{2}$$

where K is a constant and α_1, α_2 are coefficients for lattice representation. It can be seen that 'db2' filter requires fewer adders and multipliers than 'db8' filter. For the 'db2' filter, the lattice approach requires only five multipliers while Filter Bank based approach requires 8 multipliers.

Figure 5(a) shows the basic architecture for lattice implementation of 'db2' filter. Figure 5(b) shows architecture for Filter Bank implementation of 'db2' filter. We observe the redundancy in multiplications (the eight multipliers perform only four distinct unsigned multiplications). Thus, we introduce additional buffers to present a Modified Filter Bank (MFB) implementation which reuses the multiplier computations and re-uses them using time-buffers. This design is shown in Figure 5(c) and it leads to a saving of four multipliers in the design. It introduces three cycles of delay in high-pass filter calculations. Mathematically, we can write it as follows:

$$Lo_D(z) = \overbrace{a_1}^{A_1} + \overbrace{a_2 z^{-1}}^{A_2} + \overbrace{a_3 z^{-2}}^{A_3} + \overbrace{a_4 z^{-3}}^{A_4}$$

$$\begin{aligned} Hi_D(z) &= b_1 + b_2 z^{-1} + b_3 z^{-2} + b_4 z^{-3} \\ &= -A_4 z^3 + A_3 z^1 - A_2 z^{-1} + A_1 z^{-3} \\ &= z^3 \left(-A_4 + A_3 z^{-1} - A_2 z^{-3} + A_1 z^{-6}\right) \end{aligned}$$

This implementation is shown in Figure 5(c). The hardware resource requirements of direct implementation of the above discussed filters are provided in Table II.

D. Image Parsing

Because the image is effectively parsed through a one-dimension linear systolic array, it is important to efficiently parse the two-dimensional frame. We use the periodic extension mode of DWT which effectively generates least wavelet coefficients and thus maximizes throughput. The raster scan order of coefficients is followed i.e. parsing the image from left to right, then top to bottom with accompanied padding for periodization of pixels. This ensures efficient usage of pipelining registers and flushing them only at the end of a given row.

V. PNU EXTRACTION

After the DWT operation, we next perform denoising of subbands using MMSE estimation. Four levels of DWT were considered in this scenario and all subbands except LL4 (see Figure 2) are passed independently through MMSE estimaton and denoising process. Let $S(i,j)$ denote the pixel in a subband.

In each subband, we estimate the local variance of the original noise-free image for each wavelet coefficient. This is done using the Maximum A Posteriori (MAP) estimation performed for 4 sizes of a square $N \times N$ neighboring mask. The MAP can be used to obtain a point estimate of an unobserved quantity on the basis of empirical data. It employs an augmented optimization objective which incorporates a prior distribution over the quantity one wants to estimate. It is a regularization of ML Estimation. The value of this mask is selected to be $N = \{3, 5, 7, 9\}$ [7]. Although rectangular masks can also be chosen, square masks are chosen for generality. Typically a large mask is adaptively chosen for general low detail regions of image while a small mask is chosen for edge regions. For each mask, the variance is calculated as $\widehat{\sigma_W}^2$.

$$\sigma_W \widehat{(i,j)}^2 = max\left(0, \frac{\sum_{\text{all i,j}} S^2(i,j) - \sigma_0^2}{N^2}\right) \tag{3}$$

for the W^{th} mask where N is the number of pixels in the mask. The minimum value is chosen over these masks, indicating that the chosen mask is good for noise estimation. The MMSE Estimation obtains the estimated variance σ using the following relation:

$$\min_{W \in \{1,2,3,4\}} \left\{\widehat{\sigma_W}^2\right\} \tag{4}$$

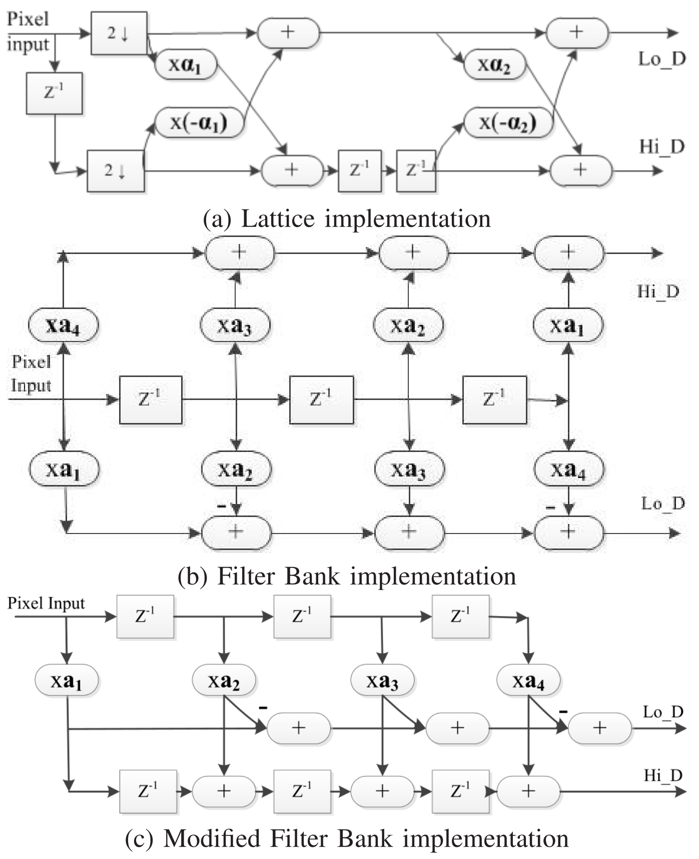
Pixel input
2↓
Z⁻¹
xα1
x(-α1)
xα2
x(-α2)
Lo_D
Hi_D
(a) Lattice implementation
xa4
xa3
xa2
xa1
Pixel Input
(b) Filter Bank implementation
Pixel Input
(c) Modified Filter Bank implementation

Fig. 5. Proposed DWT architectures for image denoising using 'db2' filter. The MFB implementation is neatly pipelined, requires the fewest number of multipliers and achieves the highest clock frequency.

Under the assumptions of independence and Gaussianity, the optimal predictor for a denoised subband is given by the following relation:

$$\overline{S(i,j)} = \frac{S(i,j) \times \widehat{\sigma^2(i,j)}}{\widehat{\sigma^2(i,j)} + \sigma_0^2} \tag{5}$$

The value of σ_0 is chosen as 5 for 8 bit pixels. This process is done to recover all denoised subbands which are then applied with inverse DWT to obtain denoised image ($\overline{I}$). Subtracting denoised image from extracted image gives us an estimate of PNU P^1.

Maximum Likelihood Estimation

This is the most computationally-expensive portion of the algorithm. This requires estimation of the underlying variance field, which is estimated based on local neighbourhood of the data point in the wavelet domain [25]. Lukas et al. [7] use a square mask for image denoising, following the trend of [25] and employ masks of size 3×3, 5×5, 7×7 and 9×9 for uniformity. However, we observe two irregularities with this choice:

1) A rectangular mask, biased towards horizontal pixels should give a more accurate description of image since regularity in natural images is maintained more along the horizontal direction. Consider a scenery or a human snapshot, the horizontal information changes less frequently as the vertical information. For a sunset scene, we find similar pixels along the horizontal line than the vertical line and same for objects like tables and walls.
2) A square-sized mask leads to larger I/O overhead than a rectangular mask of same size. More pixels need to be read simultaneously for a single pixel output.

For these reasons, we propose use of rectangular masks, of size 3×3, 3×5, 3×7, 3×9 and 3×11. Since these masks are only used for variance estimation, their impact on denoising performance should be minimal.

A. Architecture

There are many choices for implementing a PNU extraction module in hardware. Direct implementation of the scheme is fairly tedious. We need different masks to implement computation for each window size, then compare the minimum of them and select it for MMSE estimation.

The architecture for PNU extraction is shown in Figure 6. It consists of three blocks which are explained below:

1) **DWT:** As previously discussed in last section.
2) **Pre-processing**: In this stage each pixel is squared and subtracted with variance to obtain $X(i,j)$ value. Three instances are used in our case to remove the need of any extra buffer and facilitate coordination with next stage where a 2D systolic array of width 3 is used.
3) **PE array**: A 2D systolic array of Processing Elements (PEs) is used to make the MLE computations identified previously. It is described in next subsection. The output of this module is a numerical value corresponding to the minimum of the values outputted by rectangular masks.
4) **Post-processing:** The final stage computes the PNU estimate from the original pixel value and the denoised estimate which is correlated with a stored reference value and compared to a threshold (frame-wise). A correlation higher than a pre-defined threshold implies successful authentication.

Pre-processing

We introduce a pre-processing block to reduce redundant computations in our PE array and introduce hardware reuse. The squaring operation (of subband coefficient done for each computation) is redundant. Hence, instead of squaring for each operation, we input squared values of the pixels themselves and normalize them with the input variance value. The pre-processing stage is described below:

$$X(i,j) = S(i,j)^2 - \sigma_0^2$$

This leads to significant computational and power savings. We need to buffer these values and then input them to the 2D array. This implementation is referred to as a Full buffered (*F-B*) implementation. However, such an implementation will significantly increase the usage of input buffers which need to buffer an entire subband worth of values. Since we are using a rectangular mask, we can efficiently remove the need of such a buffer by implementing three pre-processing blocks, one for each row.

PE array

The PEs are arranged in a 2D systolic array. We note some interesting properties which help us to optimize the implementation of the MLE block:

1) **Pipelining:** Since the computations between subsequent pixels reuse most of the pixels (except one row / column which needs to be input), we use a pipeline which inputs along the short edge (row/ column). Thus, effectively only three pixels are input every clock cycle. We refer to this as our *Naive* implementation.
2) **Parallelism:** The larger windows overlap over smaller windows, making it possible to do the computations concurrently. This step leads to 5X speedup because the number of computations required are greatly reduced and can be reused amongst the masks.
3) **Scan pattern:** Typical image-parsing algorithms use a raster scan to parse the elements. With this approach, the buffer needs to be refreshed with every new row. We refer to this as our *Raster* implementation. We propose an alternate implementation, where we parse odd rows from left to right. At the end of odd row, we parse top-bottom one pixel and then we parse right to left for even row. When we reach the beginning of an even row, we parse top-bottom one pixel again and continue the process. This leads to a faster implementation.

A PE consists of no functional units, but instead is used simply to route the input values efficiently to other pixels (see Figure 7). There is an input mux which collects input $X(i,j)$ values from one of the four directions and passes it to the adder below. These values are passed on the next PE in next cycle, with the direction of traversal depending upon the image parsing scheme. The edge PEs may have unconnected edges (or inputs). The input mux and output demux are controlled by 2 bit signals each which select the direction of PE to receive input and to

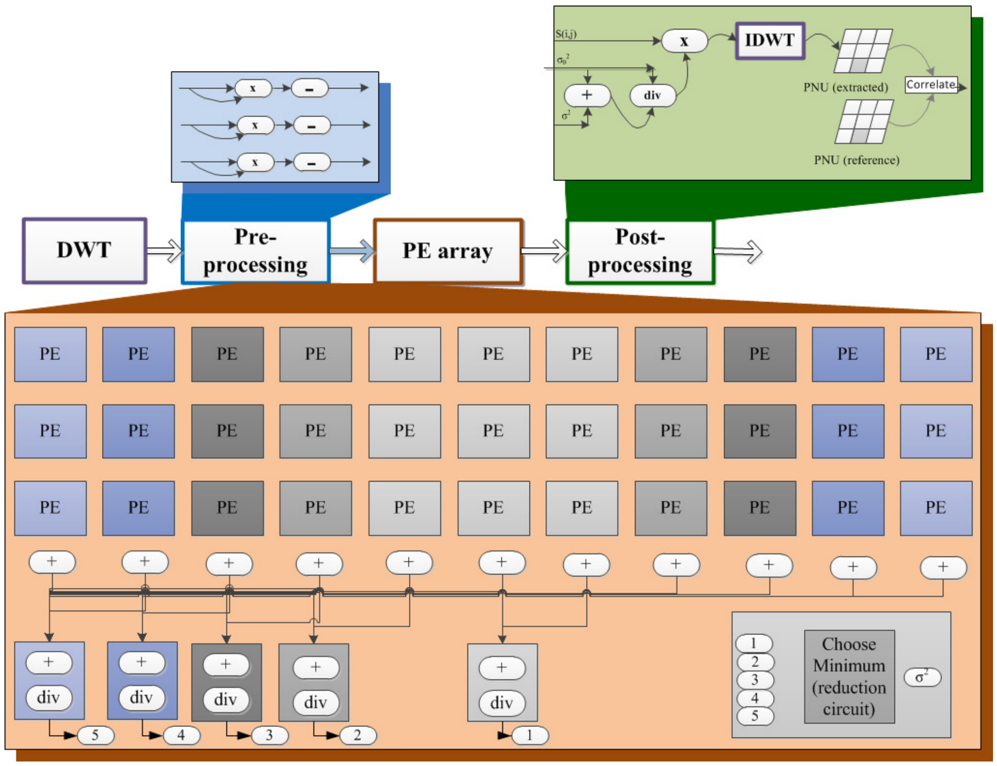

Fig. 6. Architecture for PNU extraction. The pre-processing stage squares and normalizes the input pixels (obtains X values). The PE array has a number of processing elements which are interconnected to yield the MLE outputs for different masks concurrently. The third stage performs MMSE estimation and PNU prediction for each pixel which is correlated and compared to a threshold to complete the source identification problem.

forward input to. In case of left to right traversal, the mux input will be on the left while demux output will be on the right.

The five sums are computed corresponding to the five square masks and then averaged and then the minimum is computed. To compute the minimum, we just parse the unsigned values obtained earlier bit-wise and maintain a flag (counter modulo 5). As we reach the first '1', we eliminate that variable. In case of a 'crucial-tie' (for example, two or three '1's for same bit positions for all two or three variables remaining in the list), we ignore it and continue parsing. This is implemented using a simple reduction circuit.

Post-processing

The denoised subband pixel is obtained as described in equation 5 above using the MMSE value from the PE array and the subband value. Then, the PNU estimate is obtained as P^1

$$P^1(i,j) = I(i,j) - \overline{I(i,j)}$$

where image $\overline{I}$ is obtained after inverse DWT operation on the denoised subbands. We skip discussion on IDWT in this section, as it is implemented similar to DWT in last section, with orthogonal filter coefficients.

$$\overline{I} = \text{IDWT}\left(\text{all subbands } \overline{S}\right)$$

Next, we compute the correlation between pixels in PNU P and P^1. If this correlation is above a specific threshold, the source is considered matched.

$$\rho = corr(P, P^1) = \left(\frac{(P^1 - \overline{P^1})(P - \overline{P})}{\|P^1 - \overline{P^1}\|\|P - \overline{P}\|}\right)$$

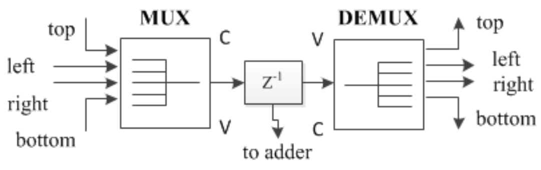

Fig. 7. Inner schematic of a Processing Element (PE).

where $\overline{P}, \overline{P^1}$ denotes mean value of pixel in P and P^1 respectively. The value of ρ above a pre-defined threshold (0.1, obtained from exhaustive experiments) indicates successful authentication of the video by identification of correct camera.

VI. Implementation

In this section, we would like to discuss the performance of our algorithm and proposed architecture.

We used 6 available surveillance cameras along with 1 laptop camera for the experiment. The details of camera models is given in Table III. They are the most popular brands in the market and have very similar specifications, which places a higher requirements on the source identification algorithm (to distinguish amongst these cameras). We applied the sensor pattern noise extraction algorithm to every channel (every component) of the video frames, and then joined them together as a whole, which is later used to calculate the correlation coefficient. In these experiments, we assume no packet loss conditions and connect these cameras to a Cisco Linksys WRT160N V2 wireless-N router which is connected to our processing modules. To make the experimental settings close to physical settings, we set the resolution to 640×480 at a frame rate of 30 frames per second. This is the maximumm resolution

TABLE III
DETAILS OF CAMERA MODELS USED FOR EXPERIMENTS

Model	Number
LinkSys WVC80N	4
D-link 942L	1
Axis M1011-W	1
Lenovo X301 webcam	1

of Linksys, D-link and Axis commercial grade surveillance cameras used in our experiments. Thus, the X301 webcam was also set to the same settings. MPEG-4 codec was used, with the GOP size set between 15 to 20 depending on the camera model. The distance between two anchor frames (I or P) is 2 or 3. At 30 fps, it requires about 26.7 MBps throughput to process them in real-time. The software implementation, on the other hand, takes 5 seconds per frame on a core I7 computer. Hence, we look towards hardware acceleration to make real-time video authentication available in commercial scenarios. We conducted tests over 140 such samples collected in a number of trials and then check the accuracy and throughput of our approach.

We evaluate our approach on the Xilinx Virtex 6 XC6VLX75 FPGA by generating the different architectures proposed earlier. We prefer FPGAs because they provide a means for rapid implementation of proposed architectures and support functional parallelism. However, the designs presented don't use any reconfiguration-specific properties and can be further accelerated for performance in VLSI technology. The architectures presented in this paper have been analyzed in terms of kernel area clock frequency and throughput considerations.

Our design is written in VHDL and synthesized using Xilinx ISE Design Suite 12.4. iSim simulations were performed to test the waveforms. The extracted frame can be loaded to the FPGA using Xilinx framebuffer module onto SRAM allowing us to make row and column accesses to extracted frame in single cycle.

A. DWT

The DWT and IDWT operations are identical and can be performed using a similar architecture (by minor modification in signs of coefficients). Before we move ahead, we would like to ensure that simplifying the filter design from 'db8' to 'db2' will not have any significant impact on our PNU extraction process.

Foremost, we compare the authentication performance of 'db2' filter over the 'db8' filter. Figure 8 shows the results of this test with varying length of each clip. It can be observed that the two filters perform similarly in terms of identification accuracy. With 100 frames of a video, we obtain an average accuracy of 60% with 'db2' while this is 63% with the 'db8'. To obtain a reasonable accuracy, we need a larger number of video frames. We need roughly 500 frames to achieve 100% accuracy with the 'db8' filter while about 525 frames give this accuracy with the 'db2' filter.

DWT Implementation

A direct implementation of the 'db8' filter using Filter Bank scheme requires 32 DSP slices, where each slice consists of multiplier-accumulate unit. The design achieves a clock

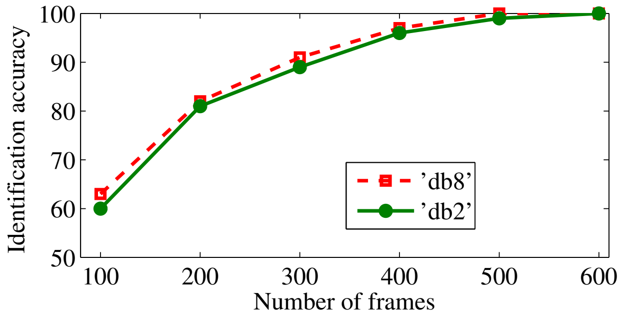

Fig. 8. Comparison of identification accuracy of the two DWT filters. The results are averaged over 140 video samples of resolution 640 × 480 from 4 cameras. The 'db2' filter performs almost as good as 'db8' filter in source identification while enabling significant hardware savings.

TABLE IV
IMPLEMENTATION OF DWT FILTERS IN XILINX XC6VLX75 FPGA

	'db8'	'db2' lattice	'db2' MFB
slices	112	88	96
LUTs	48	99	138
Registers	112	88	96
DSP48E1	32	5	**4**
Frequency (MHz)	45.56	105.94	**166.5**

frequency of 45.56 MHz. The detailed hardware resources are shown and compared in Table IV. We implemented the 'db2' filter using both the lattice approach and the Filter Bank approach. The lattice DWT kernel achieves a clock frequency of 106 MHz while requiring 5 DSP slices on-board. The architecture for lattice, Filter Bank (FB) and Modified Filter Bank (MFB) implementations is shown in Figure 5(a-c).

The MFB implementation is neatly pipelined and it achieves a higher clock frequency of 167 MHz (corresponding to 167 MBps) on target device while requiring 4 DSP slices. Details of other resources (LUTs, slices and registers) is given in Table IV. Multiple independent kernels can be launched for DWT kernel to accelerate processing. Since each kernel requires little hardware resources and an 8-bit (pixel) input every cycle, loop unrolling gives linear improvements in performance.

B. MMSE Estimation

After DWT, the image is buffered and the subbands are parsed to denoise them using MMSE estimator.

We first test our earlier assumption that rectangular masks should give better performance than square masks. [7] use four square masks of size 3×3, 5×5, 7×7 and 9×9 each to make ML Estimation and then compute MMSE values. This requires processing 81 values to output a single coefficient. A 2D systolic array implementation would require 9 inputs per cycle while other values can be reused.

Instead, we propose using rectangular masks of size 3×3, 3×5, 3×7, 3×9 and 3×11 respectively. The width of 3 is chosen to consider one pixel border around top and bottom of the current value. The length of mask is varied from 3 to 11. This implementation would require only 3 inputs per cycle (along three rows) while other 30 values can be reused. Moreover, only 33 computations need to be made to compute the MMSE value.

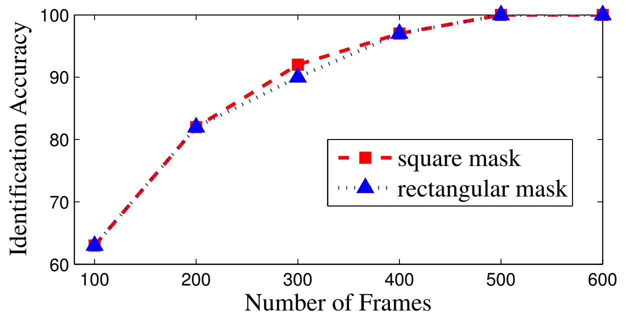

Fig. 9. Comparison of identification accuracy for square vs. rectangular masks. With almost similar identification accuracy, rectangular masks simplify and accelerate the architectural implementation. Input per cycle is reduced by 3X while hardware requirements (number of multiplications) is also reduced by 2.5X.

TABLE V
IMPLEMENTATION OF MMSE FILTERS IN XILINX XC6VLX75 FPGA

Design	Naive	Proposed
Adders	32	35
Multipliers	33	**3**
Registers	69	72
Mux	165	165
Slice Registers	1778	2306
LUT flip flops	3504	3503
Frequency (MHz)	240	**397**

Next, we compare the identification accuracy of source videos using these masks. Figure 9 shows the identification accuracy of rectangular mask, as compared with using large square masks. The curves overlap for except one data-point. This is noteworthy because we are able to obtain almost same classification accuracy with an input reduction by 3 and computational savings of 2.46.

As discussed earlier, the pre-processing module squares the magnitude, normalizes it by subtracting σ_0^2 and inputs them to the PE Array. This reduces the multiplication operations per input from 33 to 3. We implemented the individual PEs onto VHDL and them constructed a PE Array of size 3x11, which can traverse in any of the four directions. The values for each mask is computed in parallel and the minimum is found using a reduction circuit. The divide operation was implemented using Xilinx Coregen program which uses the Radix2 algorithm to allow it to be efficiently implemented using flip-flops instead of multipliers. This is used for computing the denoised subband value.

On a Xilinx XC6VLX75 FPGA device, the module requires 3 DSB48E1 slice, 3503 LUT flipflop pairs and 2306 slice registers. It achieves a clock frequency of 397 MHz. Details are given in Table V. Compared to the *naive* implementation, we achieve a savings of over 30 hardware multipliers and an 65% improvement in clock frequency. The *F-B* scheme achieves the same throughput but requires an extra buffer to store subbands value (1.4 Mb on chip memory). The proposed scan architecture is 5% faster than the *Raster* scan implementation (due to reductions in pipelining flush at end of rows).

Table VI gives the details of identification accuracy using our modified scheme versus the original scheme. There is a slight decrease in identification accuracy as reported. However, the scheme uses a significantly lesser number of pixels and hardware resources than the original. We obtained a throughput of 167 MBps for DWT kernel (which is the limiting factor for entire implementation) which makes it feasible to use commodity Virtex 6 FPGA for real-time video authentication.

TABLE VI
IDENTIFICATION ACCURACY FOR ORIGINAL ALGORITHM AND PROPOSED ALGORITHM (USING 'DB2' FILTER AND RECTANGULAR MASKS). THERE IS A SLIGHT DECREASE IN IDENTIFICATION ACCURACY BUT MANIFOLD IMPROVEMENT IN IMPLEMENTATION SPEED

# Frames	100	200	300	400	500	600
'db8' & 'square'	63	82	92	9	100	100
'db2' & 'square'	60	81	89	96	99	100
'db8' & 'rect.'	63	82	90	97	100	100
'db2' & 'rect.'	58	76	89	92	97	98

VII. CONCLUSIONS

In this paper, we proposed architectures for hardware acceleration of video authentication algorithm using pixel-nonlinearity noise to identify the original camera. Our algorithm is able to accurately authenticate source camera using 650 frames from source video.

We proposed a modified filter bank approach for DWT and IDWT implementation which reduces the hardware requirements and achieves a clock frequency of 167 MHz. We also presented a 2D systolic array architecture for wavelet subband denoising which was optimized for hardware requirements and performance using rectangular masks and suitable design choice. It achieved a clock frequency of 397 MHz with 3 multipliers and 5 dividers in the design. The overall system will run at the lower of the two (167 MHz) clock frequency processing one pixel every second. Hardware prototyping was done on Xilinx Virtex-6 FPGA XC6VLX75.

REFERENCES

[1] O. Kerr, "Searches and seizures in a digital world," *Harvard Law Review*, vol. 119, p. 531, 2005.

[2] F. Lefèbvre, B. Chupeau, A. Massoudi, and E. Diehl, "Image and video fingerprinting: forensic applications," *Media Forensics and Security*, pp. 725 405–725 405–9, 2009.

[3] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, "Source digital camcorder identification using sensor photo response non-uniformity," in *Proceedings of the SPIE*, vol. 6505, 2007.

[4] K. Cohen, "Digital still camera forensics," *Small Scale Digital Device Forensics Journal*, vol. 1, no. 1, pp. 1–8, 2007.

[5] P. Blythe and J. Fridrich, "Secure digital camera," in *Digital Forensic Research Workshop*, 2004, pp. 11–13.

[6] "Canon data verification system," online, http://cpn.canon-europe.com/content/education/infobank/image_verification/canon_data_verification_system.do, 2013.

[7] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[8] K. Mehdi, H. Sencar, and N. Memon, "Blind source camera identification," in *International Conference on Image Processing*, vol. 1. IEEE, 2004, pp. 709–712.

[9] O. Çeliktutan, B. Sankur, and I. Avcibas, "Blind identification of source cell-phone model," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 553–566, 2008.

[10] K. Choi, E. Lam, and K. Wong, "Automatic source camera identification using the intrinsic lens radial distortion," *Optics Express*, vol. 14, no. 24, pp. 11 551–11 565, 2006.

[11] A. Popescu and H. Farid, "Statistical tools for digital forensics," in *Information Hiding*. Springer, 2005, pp. 395–407.

[12] Z. Geradts, J. Bijhold, M. Kieft, K. Kurosawa, K. Kuroki, and N. Saitoh, "Methods for identification of images acquired with digital cameras," *Enabling technologies for law enforcement and security*, vol. 4232, no. 1, pp. 505–512, 2001.

[13] K. Kurosawa, K. Kuroki, and N. Saitoh, "CCD fingerprint method-identification of a video camera from videotaped images," in *Proceedings International Conference on Image Processing*, vol. 3. IEEE, 1999, pp. 537–540.

[14] X. Kang, Y. Li, Z. Qu, and J. Huang, "Enhancing source camera identification performance with a camera reference phase sensor pattern noise," *IEEE Transactions on Information Forensics and Security,*, vol. 7, no. 2, pp. 393–402, 2012.

[15] C. Li, "Source camera identification using enhanced sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 280–287, 2010.

[16] C.-T. Li and Y. Li, "Color-decoupled photo response non-uniformity for digital image forensics," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 2, pp. 260–271, 2012.

[17] S. Chen, A. Pande, K. Zeng, and P. Mohapatra, "Video source identification in lossy wireless networks," in *IEEE International Conference on Computer Communications, (Infocom) mini-conference*. IEEE, 2013.

[18] W. Van Houten and Z. Geradts, "Source video camera identification for multiply compressed videos originating from youtube," *Digital Investigation*, vol. 6, no. 1, pp. 48–60, 2009.

[19] D. Hyun, C. Choi, and H. Lee, "Camcorder identification for heavily compressed low resolution videos," *Computer Science and Convergence*, pp. 695–701, 2012.

[20] T. Acharya and C. Chakrabarti, "A survey on lifting-based discrete wavelet transform architectures," *The Journal of VLSI Signal Processing*, vol. 42, no. 3, pp. 321–339, 2006.

[21] A. Pande and J. Zambreno, "Poly-DWT: Polymorphic wavelet hardware support for dynamic image compression," *ACM Transactions on Embedded Computing Systems*, vol. 11, no. 1, pp. 6:1–6:26, Apr. 2012. [Online]. Available: http://doi.acm.org/10.1145/2146417.2146423

[22] P. Tseng, Y. Chang, Y. Huang, H. Fang, C. Huang, and L. Chen, "Advances in hardware architectures for image and video coding-a survey," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 184–197, 2005.

[23] M. Katona, A. Pižurica, N. Teslić, V. Kovačević, and W. Philips, "Fpga design and implementation of a wavelet-domain video denoising system," in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2005, pp. 650–657.

[24] M. Katona, A. Pizurica, N. Teslic, V. Kovacevic, and W. Philips, "A real-time wavelet-domain video denoising implementation in FPGA," *EURASIP Journal on Embedded Systems*, vol. 2006, no. 1, pp. 6–6, 2006.

[25] M. Kivanc Mihcak, I. Kozintsev, and K. Ramchandran, "Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6. IEEE, 1999, pp. 3253–3256.

[26] A. Burg, S. Haene, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," in *Proceedings IEEE International Symposium on Circuits and Systems*. IEEE, 2006, pp. 4–pp.

[27] M. Vetterli and J. Kovačevic, *Wavelets and subband coding*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[28] D. Redmill, D. Bull, and R. Martin, "Design of multiplier free linear phase perfect reconstruction filter banks using transformations and genetic algorithms," in *Proc. Intl. Conf. Image Processing and Its Applications*, Jul. 1997.

[29] M. Martina and G. Masera, "Multiplierless, folded 9/7 - 5/3 wavelet VLSI architecture," *IEEE Trans. Circuits and Systems II*, vol. 54, no. 9, pp. 770–774, Sep. 2007.

[30] J. Ritter and P. Molitor, "A pipelined architecture for partitioned dwt based lossy image compression using FPGAs," in *Proc. Intl. symposium on Field Programmable Gate Arrays (FPGA)*, 2001, pp. 201–206.

[31] M. Alam, C. Rahman, W. Badawy, and G. Jullien, "Efficient distributed arithmetic based dwt architecture for multimedia applications," in *Proc. Intl. Work. SoC for Real Time Applications*, 2003, pp. 333–336.

[32] M. Martina and G. Masera, "Low-complexity, efficient 9/7 wavelet filters implementation," in *Proc. IEEE Intl. Conf. Image Processing (ICIP)*, Sep. 2005.

[33] A. Benkrid, K. Benkrid, and D. Crookes, "Design and implementation of a generic 2D orthogonal discrete wavelet transform on FPGA," in *Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM)*, Apr. 2003, pp. 162–172.

[34] A. Benkrid, D. Crookes, and K. Benkrid, "Design and implementation of a generic 2D biorthogonal discrete wavelet transform on an FPGA," in *Proc. IEEE Symp.Field-Programmable Custom Computing Machines (FCCM)*, 2001, pp. 190–198.

[35] K. Kotteri, S. Barua, A. Bell, and J. Carletta, "A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting," *IEEE Trans. Circuits and Systems II*, vol. 52, no. 5, pp. 256–260, May 2005.

[36] P. Tseng, Y. Chang, Y. Huang, H. Fang, C. Huang, and L. Chen, "Advances in Hardware Architectures for Image and Video Coding - A Survey," *Proc. IEEE*, vol. 93, no. 1, pp. 184–197, Jan. 2005.

[37] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "VLSI architecture for discrete wavelet transform based on B-spline factorization," *Proc. IEEE Work. Signal Processing Systems, 2003. SIPS 2003*, pp. 346–350, Aug. 2003.

[38] S. Murugesan and D. Tay, "New techniques for rationalizing orthogonal and biorthogonal wavelet filter coefficients," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 3, pp. 628–637, 2012.