Architecture for Blocking Detection in Wireless Video Source Authentication

Amit Pande[1], Shaxun Chen[1], Prasant Mohapatra[1] and Gaurav Pande[2]
[1]Department of Computer Science, Univerisity of California, Davis, CA, USA
[2]Department of MCA, R.K. Goel Institute of Technology, Ghaziabad, UP, India
Email: {pande, sxch pmohapatra}@ucdavis.edu, reachgauravpande@gmail.com

***Abstract*—Blocking is a common artifact in wireless video streaming services, mainly attributed to packet loss degradation in real-time transmission scenarios. In this paper, we present a simple algorithm and architecture for robust detection of blocking artifact. We first take Discrete Wavelet Transform of the original video frame followed by utilizing a unique property of staircase sized repeated pattern in the videos. The variance of this pattern is measured as the extent of blocking in a video frame. We propose two architectures for blocking detection: one using orthogonal wavelets which can be seamlessly integrated to video source authentication, and the other based on bi-orthogonal wavelets, and can be used in robust stand-alone blocking detection. A prototype implementation on a Xilinx Virtex-6 XC6VLX75 FPGA device was optimized to obtain a clock frequency of 167 (396) MHz for orthogonal (and bi-orthogonal wavelets) using 4 (0) multipliers in the design respectively.**

***Index Terms*—digital camera identification, hardware architecture, blocking**

I. Introduction

Video surveillance is an important application in both consumer (home), enterprise and defense scenarios. With the prevalence of small portable wireless video cameras, the amount of video crossing the network, originating from embedded devices is increasing at enormous rates. Traditional authentication mechanism are not suitable for these surveillance cameras as they put an enormous computational load leading to power drainage. Therefore, there have been efforts to develop schemes for real-time video authentication in wireless networks.

The idea behind recently proposed video forensics based schemes [1]–[3] is inherently based on pixel-non-uniformity (PNU) noise present in camera sensors by virtue of their construction. Every camera inserts a unique PNU noise to the images and videos acquired using the device, which can be used to authenticate that the recorded video was indeed streamed from the specific device [1]. The authenticity of the camera is as important as the camera footage itself [4]. This evidence has been used in the court-of-law for some time now, to establish the authenticity of videos or images shown in court and this field is known as image or video forensics. It can also be used against movie piracy [5], [6]. However, with the emergence of surveillance industry, it is desired to extend the analysis to real-time videos streamed in a networked domain to authenticate that the feed is coming from desired camera. This is applicable to versatile cameras installed in drones used by military for aerial surveillance and commodity surveillance cameras used by shopkeepers. The commodity cameras are not equipped with special watermarks, now embedded by camera makers into the high-range models, however, PNU noise is present. A hardware implementation of this forensics scheme, making it real-time on Virtex-6 FPGA device was performed by Pande et al. [2].

Fig. 1. Illustration of blocking artifact in network transmitted video

However, in the case of wireless cameras streaming real-time video (over UDP) to the client, the proposed scheme fails to work because of the blocking and blurring artifacts introduced by wireless channel losses. Figure 1 gives an illustration of blocking artifacts in a wireless streamed video at different channel conditions. It can be seen that blocking artifacts can be quite severe even at normal packet losses. These artifacts limit the performance of video source authentication algorithm. Chen et al. [3] propose removal of wireless artifacts to speed up the performance of source authentication in wireless networks and make it real-time. A hardware implementation for blocking detection is proposed in this paper.

Video and image compression codecs such as MPEG or JPEG use block coding techniques where a video frame or an image is typically divided into small blocks of 8x8 or 16x16 pixels and subsequently processed by frequency transform, motion compensation, quantization and entropy encoding procedure. Reduction of data rate for compression and loss of packets during transmission leads to several artifacts in the received video (image) such as blocking, blurring and ringing. Of these, blocking is the most prominent artifact. Prior research [3], [7] has shown that blocking and blurring artifacts are highly correlated in wireless transmission scenarios where packet loss is the main culprit. Blocking artifacts appear as a regular pattern of visible block boundaries. This degradation is a direct result of the coarse quantization of the coefficients or loss of packets and the independent processing of the blocks which does not take into account the existing correlations among adjacent block pixels.

As we mentioned earlier, these artifacts are mainly present in the form of blocking artifacts i.e. small rectangular block regions are whitened or blackened or hanged out in the video

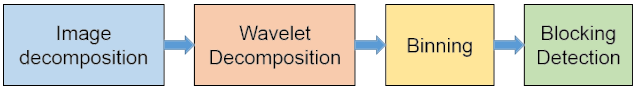

Fig. 2. Block Diagram of Blocking extraction process

playback. In this paper, we present a simple algorithm for accurate detection of such artifacts in videos and then present its prototype implementation on Xilinx Virtex-6 FPGA. The algorithm is based on observation of staircase like pattern observed in diagonal wavelet decomposition coefficient of blocked images.

The paper is organized as follows: Section II discusses related works in video source identification and blocking detection in videos. Section III gives a brief explanation of blocking detection algorithm followed by a prototype implementation in Section IV. Section V gives the performance improvements in source identification algorithm achieved using blocking detection and implementation details on Virtex-6 FPGA followed by Conclusions in Section VI.

II. Related Works

Canon Data Verification Kit [8] 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.

Kharrazi et al. [9] 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. Geradts et al. [10] proposed to utilize sensor hot pixels or dead pixels to identify the image source (unique device identification). However, all cameras do not have such defective pixels, and many cameras post-process to remove such defects from output images. Lukas et al. [1] employed sensor pattern noise as an inherent fingerprint of the camera for source identification. More specifically, they use Pixel Non-Linearity (PNU) noise to identify the individual video camera. So far, the sensor pattern noise based schemes report the most reliable results. Pande et al. [2] give a hardware architecture for proposed scheme. Chen et al. [6] 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 [3] improves this value to 10 seconds ($\sim 300 - 400$ frames) using network characteristics, however, blocking detection is done as a post-processing task.

Existing blocking detection schemes [11], [12] are based on weighted mean-square difference along block boundaries or second and third order statistical features and cannot distinguish how much of the gray level difference between block boundaries is due to real blocking discontinuity or the oscillation of original signal itself. The blocking detection algorithm proposed in this work is based on Wavelet domain alignment of coefficients, and elaborated next. There is no known hardware implementation of these schemes.

2-Dimensional Discrete Wavelet Transform (DWT) is used to transform image into wavelet domain. 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 [13].

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 [14]–[16]. These multiplier-free implementations and other optimizations [17]–[19] involve image reconstruction quality trade-offs and have not been tested for denoising 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 [20]. 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. The 9/7 poly-DWT filter in [16] has best known image compression and hardware-efficient implementation.

However, our application uses orthogonal db8 filter instead of biorthogonal filters used for DWT-based image compression applications. This is because forensics require denoising that typically uses orthogonal wavelets while as against CDF 9/7 and similar filters which are based on bi-orthogonal wavelet construction.

III. Algorithm

Now, we focus exclusively on the scope of this paper: to present a simple algorithm and architecture for blocking detection in wirelessly transmitted videos. The main steps are given in Figure 2.

In our observation, this method gives good results on the images/frames in which the blocking is obviously composed of small "unit blocks" (like 4×4, 8×8 or 16×16 squares), and the "unit blocks" are well aligned. for example, two samples attached below can give good results. This method was tested for motion JPEG, Mpeg-1 and Mpeg-4 videos, that we obtained with our surveillance cameras (detailed in Section V).

First of all, we transform the image into smaller tiles or blocks of small size. This decomposition helps us to locally detect blocking within an image. The block size was taken to be 32×32 in our settings. A smaller block size also reduces the computational requirements of block processing. However, choosing a small size, such as 8×8 may be unsuitable when the codec has block size of 16×16 or 32×32. When we transform an block over to wavelet domain, we obtain a peculiar property in diagonal details coefficients. We observe that the coefficients have a staircase or saw tooth pattern which repeats every 4 values if blocking is present in the videos. Thus, if we sum the coefficients across rows or columns and plot the histogram of the coefficients, the histogram is 'staircase-like' in the case of frames with blocking. A sample histogram is shown in Figure 3, using a tile size of 512×512 to illustrate the staircase pattern. We use this property for detection of blocking in videos.

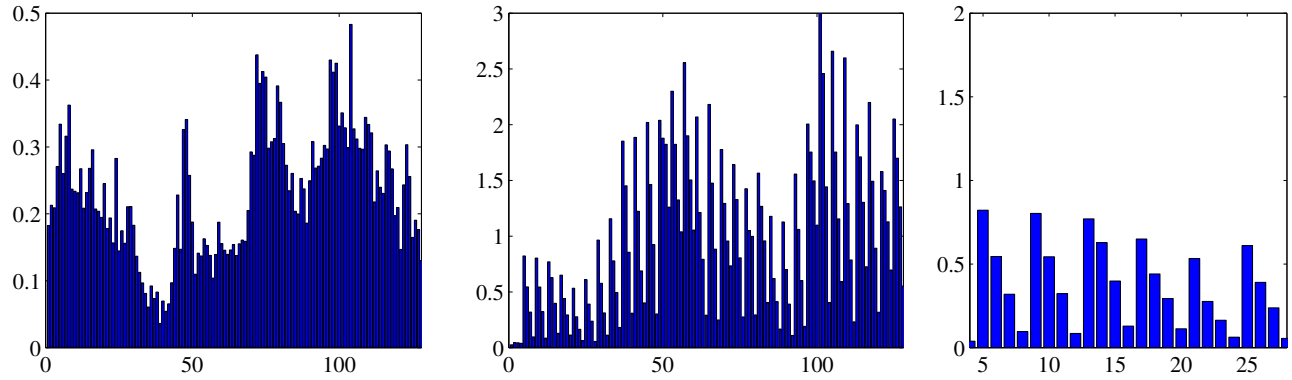

Fig. 3. Sample histogram plot for transform coefficients across rows and columns for diagonal coefficient of wavelet decomposition using orthogonal wavelet for (a-left) original frame, and (b-middle) frame with blocking. (c-right) shows zoom-in to (b) showing clear staircase like pattern with repetition every 4 pixels. The entire sample frame (512×512 pixels) is considered as a single tile for illustration purposes.

The algorithm can thus be summarized as follows:

1) Divide the image or frame into tiles (32×32).
2) Obtain the Discrete Wavelet Transform of each tile.
3) Sum the coefficients across row and bin them into 4 (A,B,C and D bins) by adding the values.
4) Bin the coefficients into 4 bins and find the accumulated value for each bins. Sort the values. If the coefficients are in the ratio 4:3:2:1, blocking is present in the frame. Else, if the coefficients are almost equal, blocking is absent.

IV. Architecture

A. Image Decomposition

We use the periodic extension mode of DWT which effectively generates least wavelet coefficients and thus maximizes throughput. One level of DWT implementation involves applying the low and high pass filter across rows and columns successively. As mentioned before, tile size of 32×32 is used and DWT is applied independently to each one. Since each pixel is 8 bits input, we need only 4Kb memory which is available on-board in modern FPGAs.

B. DWT Filter Design

1) Orthogonal Filter: In [1], the authors propose using 'db8' orthogonal wavelet for denoising operation for PNU extraction. The same filter is to be used for extracting the blocking pattern, to maintain the possibility of hardware reuse. Named after Ingrid **D**au**b**echies 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. They are all distinct 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. 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

TABLE I
Hardware requirements of DWT Filters

Filter Details	'db8' FB	'db2' lattice	'db2' FB	'db2' **MFB**	**CDF binary**
Adders	32	5	6	6	9
Multipliers	30	5	8	4	0

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 [16], [21], because that will lead to visible distortions. Rather, we propose to use 'db2' filter. The filter coefficients for the 'db2' filter are represented as:

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

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 4(a) shows the basic architecture for lattice implementation of 'db2' filter. Figure 4(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 4(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:

$$Lo_D(z) = \overbrace{a_1}^{A_1} \quad \overbrace{+a_2z^{-1}}^{A_2} \quad \overbrace{+a_3z^{-2}}^{A_3} \quad \overbrace{+a_4z^{-3}}^{A_4}$$

$$\begin{aligned} Hi_D(z) &= b_1 + b_2z^{-1} + b_3z^{-2} + b_4z^{-3} \\ &= -A_4z^3 + A_3z^1 - A_2z^{-1} + A_1z^{-3} \\ &= z^3\left(-A_4 + A_3z^{-1} - A_2z^{-3} + A_1z^{-6}\right) \end{aligned}$$

This implementation is shown in Figure 4(c). The hardware resource requirements of direct implementation of the above discussed filters are provided in Table I.

2) Biorthogonal Filter: The bi-orthogonal filters are more popular in image compression algorithms and many architectures are proposed in literature to efficiently map them in hardware. The CDF 9/7 filter is commonly used and it has irrational coefficients. They can be rationalized by adding more degrees of freedom on the Lagrange Half-Band Filter equation and imposing the condition of rational coefficients [22] along with the condition for perfect reconstruction. We refer to the implementation in [16] which provides a hardware-efficient multiplier-less approach to implement CDF 9/7 filter with rationalized coefficients. The filter coefficients are symmetric, hence they are clubed together to simplify the computationsL

$$w_p = \text{pixel}(i-p) + \text{pixel}(i-p), \quad p \in \{0,1,2,3,4\}$$

The low and high pass coefficients are obtained using shifts and add logic operations on these values, instead of using dedicated hardware multipliers.

$$\begin{aligned} Lo_D(i) &= 1/2 \times w_1 + w_0 - 1/32 \times w_0 + 1/64 \times w_4 \\ Hi_D(i) &= -1/16 \times w_2 - 1/8 \times w_1 + 3/8 \times w_0 - 1/32 \times w_1 \end{aligned}$$

The architecture is presented in Figure 4(d). Division by 4,8 or 16 is implemented using arithmetic shift operations. This eliminates the need of multipliers in the circuits.

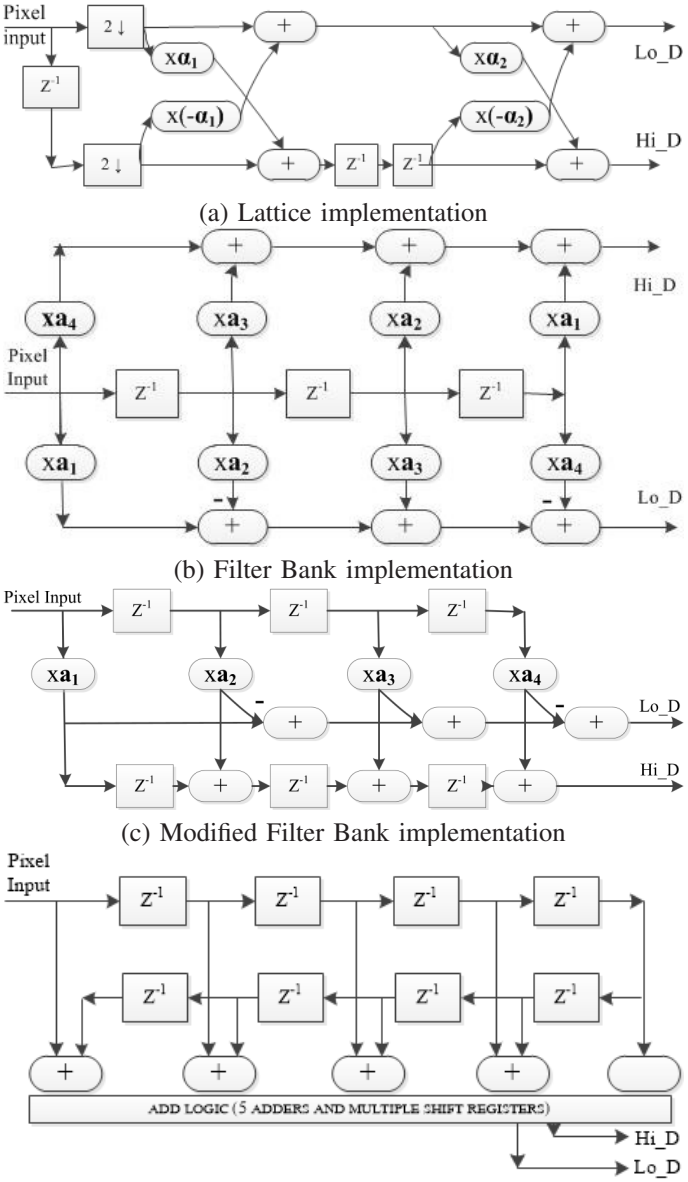

Fig. 4. Proposed DWT architectures for blocking detection. (a-c) show 'db2' based implementations with optimizations. The MFB implementation is neatly pipelined, requires the fewest number of multipliers and achieves the highest clock frequency. (d) shows multiplierless implementation (suitable for blocking detection for image compression applications but unsuitable for denoising or PNU extraction)

C. Blocking detection

The accuracy of blocking detection and measure of blockiness in the given frame is determined by the extent of staircase like structure in the current frame. In this setting, we took tiles of size 32×32, hence the size of diagonal coefficients is 16×16 values . We accumulate (add) the values of pixels over the rows and then aggregate them along 4 bins. The four bins are placed across the columns while all the pixels in rows are summed. Mathematically, this can be described as follows:

$$bin(j\%3) \quad += HH4(i,j)$$

Now, the values in these four bins are sorted into A,B,C and D such that $(A > B > C > D)$. As we mentioned earlier, for tiles with blocking, A:B:C:D is approximately 4:3:2:1, while the ratio is 1:1:1:1 for non-blocking cases. Sorting the bins is easy because we can implement the design in signed mathematics and bit-comparisons can lead to solution. A direct implementation of comparison logic in hardware would require several multiplier, divider and comparator circuits.

To efficiently implement this in hardware, we use the following approximate logic and implement this with help of shift registers and comparators:

$$\begin{aligned} E &= A + B + C + D \\ \text{if} \quad (A > (E \gg 2 + E \gg 4) \quad &\& \quad D < (E \ll 3)) \\ \text{blocking} &= \quad \textbf{true} \end{aligned}$$

Here, $\gg$ and $\ll$ refer to linear right and left shift operations. The idea is that A must be greater than 31% of sum while D must be less than 12.5% of the sum E to declare blocking

in the tile. This threshold is chosen between the two extreme boundaries (4:3:2:1 and 1:1:1:1) and is also simple to implement in hardware because it uses binary arithmetic.

This process is repeated for every tile. For each tile, we obtain a binary value (O for no blocking and 1 for blocking). We can average this value over a frame to obtain a normalized index of blocking in the given image (ranging from 0 to 1 to show percentage of blocking in the image). Existing blocking detection schemes don't give any normalized score, hence this is an added advantage of our scheme.

For source camera identification, we need to instead quarantine the blocks with blocking from the algorithm used subsequently for camera identification.

V. Implementation

A. DWT implementation

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 waveform. The extracted frame can be loaded to the FPGA using Xilinx frame buffer module onto SRAM allowing us to make row and column accesses to extracted frame in single cycle. We focus on DWT because it forms the cruz of computational time of this algorithm (subsequent blocking detection requires only two comparators).

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 verified that simplifying the filter design from 'db8' to 'db2' will not have any significant impact on our PNU extraction process. The CDF filter, although excellent in blocking detection performance is unsuitable for denoising step involved in source identification.

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 frequency of 45.56 MHz. The detailed hardware resources are shown and compared in Table II. 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 4(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 II. 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.

TABLE II
Implementation of DWT Filters in Xilinx XC6VLX75 FPGA

Filter Details	'db8' FB	**'db2'** lattice	**'db2'** **MFB**	**CDF binary**
slices	112	88	96	245
LUTs	48	99	138	175
Registers	112	88	96	210
DSP48E1	32	5	**4**	**0**
Frequency (MHz)	45.56	105.94	**166.5**	**396**

B. Performance

We used 6 available surveillance cameras along with 1 laptop camera for the source authentication experiment.This comprises 4 Linksys WVC80N, 1 Dlink 942L, 1 Axis M1011-W and 1 Lenovo X301 web cam. To make the experimental settings close to physical settings, we set the resolution to 640×480 (maximum possible) at a frame rate of 30 frames per second. MPEG-4 codec was used, with the GOP size set between 15 to 20 depending on the camera model. 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. The FPGA modules proposed above can process at 167 or 396 MHz per coefficient, which can be augmented to [2] to make a practical real-time authentication system. Against the requirement of 26.7 MBps, this system can achieve a throughput of 167 Mbps.

The speedup in video source authentication achieved by augmenting the blocking module is shown in Figures 5 and 6 respectively. In Figure 5, the dotted lines indicate the performance of existing detection scheme without removing the false positives caused by blocking. The dotted lines show the performance of existing approach while the solid lines show the improvement caused by incorporating blocking detection. At 500 kbps, camera identification accuracy is at most 60% after processing 30 seconds (90 seconds) of transmitted video but it increases to 100% in 25 seconds only with our approach. For 1.5 Mbps video, 100 accuracy is reached 2.3X faster (10 seconds against 23 seconds). Order of magnitude improvements can also be observed in cases of heavy blocking where identification accuracy increases from 20% to 100% for low bitrate video (500 kbps)

VI. Conclusion

In this paper, we presented a simple approach to detect blocking artifacts in images and video frames. This approaches uses 'stair-case' property observed in diagonal wavelet decomposition coefficients. We presented two architectures, based on orthogonal and bi-orthogonal wavelets, to realize this scheme. The bi-orthogonal wavelet based architecture is simple and efficient for hardware implementation, requires no multipliers and gives higher clock frequency. The orthogonal implementation is helpful for integration and re-use in video forensics application where it is used for real-time camera source detection in surveillance feeds. We showed that the system can provide upto

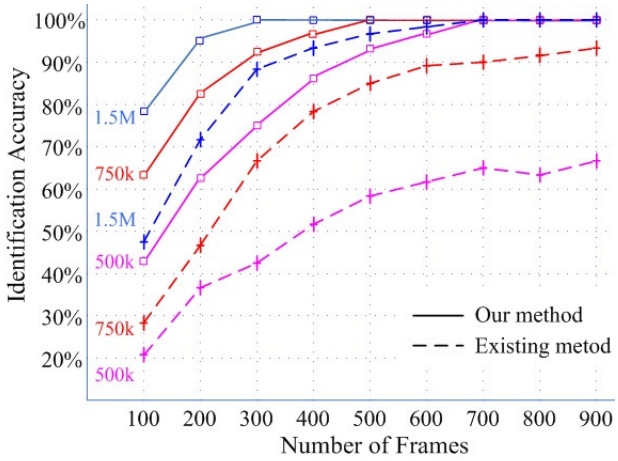

Fig. 5. Improvement in identification accuracy of video surveillance feeds using blocking detection approach in case of light blocking (light network losses)

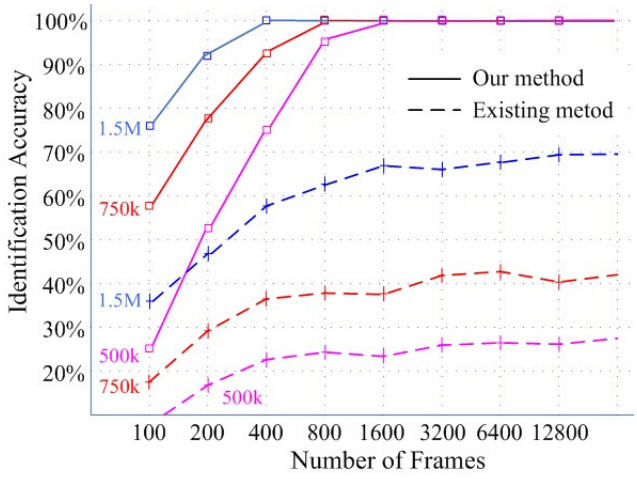

Fig. 6. Improvement in identification accuracy of video surveillance feeds using blocking detection approach in case of heavy blocking (heavy network losses)

167 Mbps speed and significantly improve the performance of source camera authentication schemes.

The future work will involve implementation and deployment of prototype implementation on hardware boards with surveillance cameras to learn difficulties in a practical settings.

REFERENCES

[1] 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.

[2] A. Pande, S. Chen, P. Mohapatra, and J. Zambreno, "Hardware architecture for video authentication using sensor pattern noise," *IEEE Transactions on Circuits and Systems for Video Technology*, to appear.

[3] S. Chen, A. Pande, K. Zeng, and P. Mohapatra, "Video source identification in lossy wireless networks," in *The 32nd IEEE International Conference on Computer Communications (IEEE Infocom mini-conference)*, 2013, pp. 215–219.

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

[5] 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.

[6] 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.

[7] P. Mcdonagh, A. Pande, C. Vallati, P. Mohapatra, P. A. Perry, and E. Mingozzi, "Investigation of scalable video delivery using h.264 svc on an lte network," in *14th Symposium on Wireless Personal Multimedia Communications (WPMC 2011)*, 2011, pp. 1–5.

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

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

[10] 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.

[11] H. Wu and M. Yuen, "A generalized block-edge impairment metric for video coding," *IEEE Signal Processing Letters*, vol. 4, no. 11, pp. 317–320, 1997.

[12] Z. Wang, A. C. Bovik, and B. Evan, "Blind measurement of blocking artifacts in images," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3. Ieee, 2000, pp. 981–984.

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

[14] 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.

[15] 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.

[16] 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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] D. Tay, "Rationalizing the coefficients of popular biorthogonal wavelet filters," *IEEE Trans. Circuits & Systems for Video Technology*, vol. 10, no. 6, pp. 998–1005, Sep. 2000.