International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

High-Speed FPGA-Based Video Watermarking Using LSB Technique in the Spatial Domain

Hajar Maseeh Yasin[1], Amira B. Sallow[2], Riyadh Zaghlool Mahmood[3]

Submitted: 08/10/2023 **Revised**: 26/11/2023 **Accepted**: 06/12/2023

Abstract: In the digital era, watermarking is crucial for copyright protection and content authentication, responding to the challenges posed by digital media's easy duplication and distribution. This paper introduces an innovative approach to digital video watermarking, leveraging hardware implementation for enhanced security and efficiency. By employing the Spartan 3E-XC3S1600 FPGA and VHDL programming, the study focuses on embedding and extracting watermarks in parallel processing on uncompressed digital videos in the spatial domain, using the Least Significant Bit (LSB) technique. MATLAB software supports the pre-processing and post-processing phases. The system's performance is remarkable, achieving a high Peak Signal-to-Noise Ratio (PSNR) of 45.0225 and a Structural Similarity Index (SSIM) score of 0.9867, indicative of quality watermarking with minimal impact on the original video content. Additionally, it demonstrates exceptional processing speed, handling video at 139.47 frames per second (fps) at a frequency of 75 MHz. This study presents a robust solution for digital watermarking. It sets a benchmark for future research in the field, combining hardware efficiency and software flexibility to cater to the evolving needs of digital media security.

Keywords: *MATLAB, Peak Signal-to-Noise Ratio (PSNR), watermarking, benchmark*

1. Introduction

In today's digital age, where multimedia content, particularly videos, is extensively shared over the Internet, the significance of digital watermarking has grown exponentially [1]. This technology protects intellectual property rights, ensures content authenticity, and prevents unauthorized distribution [2]. As videos become a dominant medium for communication and entertainment, safeguarding them from misuse and piracy is paramount [3].

Digital watermarking offers a robust solution to embed invisible marks or identifiers within the media file. These watermarks can be used for various purposes, such as tracking the origin of a media file, asserting ownership, managing digital rights, or even for content authentication and integrity verification [4], [5].

Digital watermarking is broadly classified into two categories: visible and invisible watermarking. Visible watermarking is apparent to the viewer and is often used for branding or as a deterrent against unauthorized use [6]. On the other hand, invisible watermarking is more subtle and is embedded so that it does not perceptibly alter the original media. This form of watermarking is crucial for security and forensic applications where the presence of the watermark should not be evident to the ordinary observer [7].

Moreover, digital watermarking can be classified based on the domain in which the watermark is embedded: spatial and frequency. Each type has advantages and applications for different requirements and scenarios [8].

Focusing specifically on digital video watermarking, the choice between spatial and frequency domains is significant. The spatial domain involves directly modifying pixel values of the video frames, making it more straightforward and less computationally intensive. This simplicity makes it well-suited for real-time applications where speed is essential [9].

In contrast, frequency-domain watermarking, which involves transforming the video data into a frequency space (e.g., using Discrete Cosine Transform or Fourier Transform), offers better robustness against compression and other forms of manipulation [10]. However, this robustness comes at the cost of increased computational complexity, which can be a limiting factor in real-time applications [11].

The preference for the spatial domain in video watermarking, particularly in this thesis, stems from its simplicity and efficiency in real-time processing [12]. The spatial domain allows quicker embedding and extracting of watermarks, which is crucial in scenarios where videos must be processed in real time or on the fly [13]. This immediacy is particularly relevant in live broadcasts,

[1]*Department of Information Technology, Technical College of Informatics, Akre University for Applied Sciences, Duhok, Akre, Kurdistan Region, Iraq. hajar.yaseen@dpu.edu.krd*
[2] *Department of Information Technology, Technical College of Duhok, Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq, amira.bibo@dpu.edu.krd*
[3] *Cyber Security Department, College of Computer Science and Mathematics, University of Mosul, Computer Science, Mosul, Iraq, riyadh.zaghlool@uomosul.edu.iq*

surveillance, and other applications where delays cannot be afforded [14].

Some spatial image watermarking techniques embed a watermark into the least significant bits (LBS) of pixels. The watermark is invisible to human eyes. However, the watermark can be easily destroyed if the watermark is low-pass filtered or JPEG compressed [15]. Embedding the watermark in the most significant bits (MSB) will introduce a more robust watermark, but it will also distort the host image, and this conflicts with the invisibility requirement [16], [17].

Using hardware, especially FPGAs, in digital video watermarking introduces significant advantages, particularly in performance [18]. FPGA-based implementations of watermarking algorithms excel in speed and efficiency, markedly reducing the execution time compared to software-only solutions.

FPGAs provide a high degree of parallelism, allowing for simultaneous processing of multiple video frames or different parts of a frame. This capability is crucial for real-time video processing applications where speed is of the essence [19]. Moreover, FPGAs are reconfigurable and can be programmed to suit the specific requirements of different watermarking algorithms, offering flexibility and adaptability [20].

This paper presents a cost-effective digital video watermarking solution using the LSB technique in the spatial domain, implemented on FPGA hardware for parallel processing. Key contributions include:

1. Implementing FPGA-based parallel processing to accelerate real-time video applications' watermark embedding and extraction.
2. Significantly reducing time for embedding and extracting watermarks, enhancing efficiency in real-time video applications.
3. Introducing a new secret key mechanism to bolster the security of the digital watermarking process.

The remaining structure of this paper is as follows: Section 2 provides a brief overview of related work. Section 3 introduces the proposed system. Section 4 details the hardware implementation and experimental results. In Section 5, a comparison with other studies is made. The paper concludes in Section 6, which also discusses future work.

2. Literature Review

FPGAs are essential in digital video watermarking, offering speed, security, and suitability for real-time applications [21]. The FPGA adaptability and efficiency blend software flexibility with hardware performance [22]. FPGAs excel in parallel watermark embedding and extraction processing, enhancing video watermarking efficiency [23]. The re-programmability of the FPGA allows adaptation to evolving watermarking technologies and security challenges, making them a reliable and versatile tool for academic and professional use [24]. The following paragraph provides an overview of previous research on using FPGAs in watermarking. The following paragraph outlines previous studies that have utilized FPGAs for implementing watermark embedding within raw digital videos in the spatial domain.

K. Pexaras et al. [25] focus on optimizing and implementing low-cost image and video watermarking hardware. It proposes computational optimizations to minimize the size of arithmetic operations, thereby reducing hardware costs. Their study introduces three hardware architecture variants: two for image watermarking (a low-cost serial and a more efficient parallel one) and one for video watermarking (pipelined). These designs leverage small arithmetic units in various computational steps for cost efficiency. Their system's methods balance robustness against attacks and low implementation costs, especially for wireless network applications where resources are limited.

S. Das and et al.[26], concentrate on enhancing a reversible contrast mapping (RCM) algorithm for invisible video watermarking. The methodology involves using an FPGA-based system for efficient hardware implementation. Their study corrects shortcomings of the existing RCM algorithm, ensuring it satisfies invertible properties in all cases. It employs a high-level synthesis (HLS) approach, resulting in a design that uses pipeline structures and parallelism for improved performance. The hardware implementation, verified using the VIVADO HLS tool, shows promise in real-time applications with low cost and high speed. The study also compares software and hardware implementations, emphasizing the effectiveness of the FPGA approach in optimizing video watermarking processes.

K. Sunaniya et al. [27] introduced a novel algorithm for reversible invisible watermarking using FPGA. It proposes an efficient embedding bit rate control based on contrast mapping (EBCRCM). The methodology includes adaptive linear contrast mapping on pixel intensity values controlled by a predefined threshold, ensuring distortion remains within acceptable limits. Their study emphasizes the hardware implementation, leveraging parallel processing to achieve high speed, which is crucial for real-time applications. The algorithm is validated in MATLAB and implemented on FPGA, demonstrating its effectiveness for practical use.

R. Dalbouchi et al. [28] focused on a video watermarking approach using motion vectors. The methodology involves embedding a watermark in the motion vectors of

a video, which are generated during motion estimation. Their study examines both software and hardware implementations, with the hardware aspect leveraging FPGA technology for efficiency. The technique involves embedding a binary sequence into motion vectors' horizontal and vertical components, followed by scrambling to ensure robust video protection. Experimental results indicate that this method maintains video quality while requiring relatively small resources.

3. Proposed Hardware-Based Watermarking System

In this system, we proposed parallel hardware implementation for uncompressed digital video watermarking in the spatial domain leveraging the LSB technique. The proposed system has two algorithms: one for embedding binary watermark images into uncompressed video frames and another for extracting the watermark from watermarked video. These algorithms are implemented on the hardware in a parallel way. Figure (1) presents the overall framework of the proposed system. It initially shows the video loaded by the Personal computer (PC) and the pre-processed data being passed to the FPGA for either the embedding process or the watermark data extraction. Subsequently, this data is passed back to the PC for post-processing. The final output of the entire process could be the watermarked video or the watermarked image.

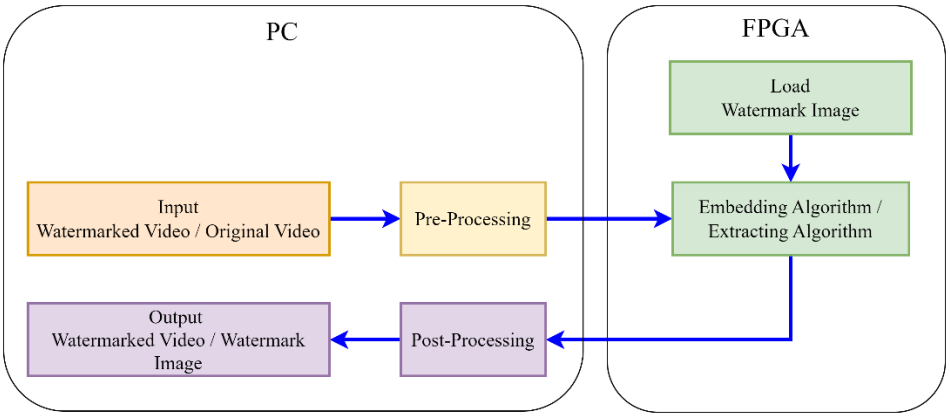


Fig. 1. General Framework of the Proposed System

3.1 Embedding Algorithm

In the proposed system, we have embedded the watermark in all video frames to make it invisible to humans. In this system, watermark embedding involves altering each pixel's second bit (D1) in a video frame, as shown in Figure (2). Instead of directly replacing this bit with the watermark bit, it is modified (from 0 to 1 or vice versa) based on a comparison with the corresponding watermark bit. The reasons for choosing the second bit over the first one are to preserve video quality and to resist better attacks like filtering, cropping, rotation, and compression.

D7	D6	D5	D4	D3	D2	D1	D0

Fig. 2. Bit Position for Watermarking

The system introduces a new secret key for the embedding process, based on a parity check (sum of ones) in both the Most Significant Nibble (MSN) and Least Significant Nibble (LSN) of pixels, compared to the watermark bits. Parity is either even ('0') or odd ('1'). If parity is '0' and the watermark bit is '1', the second bit of the MSN is complemented. No change occurs if both parity and watermark bit are '0'. The exact process applies simultaneously to the LSN, ensuring unchanged MSN and LSN if their parity is '0' and the watermark bit is '0'.

Before watermark embedding, the system undergoes several pre-processing steps to prepare the video data for watermarking. These include extracting frames, isolating the blue channel and dividing it into four parts, and condensing data from two bytes into one, necessitated by the device's limited memory. Additionally, post-processing is conducted on the watermarked data to form the output watermarked video format. The detailed steps for the pre-processing, embedding, and post-processing are listed as follows:

1. From the PC, read the uncompressed RGB video.
2. Extract the first frame N x M (N is the number of columns, and M is the number of rows).
3. Extract Blue Channel.
4. Divide the blue channel into four equal parts. The below Equations calculate the size of each part, where

Equation (1) calculates the number of columns in each part, while Equation (2) calculates the number of rows:

$$\boldsymbol{Pn} = \boldsymbol{N} \quad \text{..... (1)}$$

$$\boldsymbol{Pm} = \frac{\boldsymbol{M}}{\boldsymbol{4}} \quad \text{..... (2)}$$

5. Select the first part.
6. A byte of eight bits is formed by picking the LSN of each pixel in the part, along with the neighboring LSN pixel, by row. The size of a part after the formed byte is calculated by Equation (3) for rows and Equation (4) for columns:

$$\boldsymbol{Pnf} = \frac{\boldsymbol{Pn}}{\boldsymbol{4}} \quad \text{.....(3)}$$

$$\boldsymbol{Pmf} = \boldsymbol{Pm} \quad \text{.....(4)}$$

Pnf is the number of columns in the formed byte matrix, and ***Pmf*** is the number of rows.

7. Send the formed bytes to the FPGA.
8. read the black/white watermark image with size (Wn x Wm) from the FPGA buffer.
9. Divide the watermark into four equal parts, Where the size of each part is calculated by Equation (5) to find the number of bits in each column (***WPn***) and Equation (6) to find the number of bits in each row (***WPm***):

$$\boldsymbol{WPn} = \boldsymbol{Wn} \quad \text{..... (5)}$$

$$\boldsymbol{WPm} = \frac{\boldsymbol{Wm}}{\boldsymbol{4}} \quad \text{..... (6)}$$

10. Pick the first part of the watermark.
11. Divide the frame's part into blocks. The number of blocks in the column is equal to (***WPn***), and the number of blocks in the row is equal to (***WPm***).
12. Calculate block size (Number of Pixels in each Block). Equation (7) calculates the number of pixels in each column (***Bn***), while Equation (8) calculates the number of pixels in each row (***Bm***).

$$\boldsymbol{Bn} = \frac{\boldsymbol{Pnf}}{\boldsymbol{WPn}} \quad \text{..... (7)}$$

$$\boldsymbol{Bm} = \frac{\boldsymbol{Pmf}}{\boldsymbol{WPm}} \quad \text{..... (8)}$$

13. Receive the first watermark bit.
14. Pick the first block.
15. Pick the first pixel of the block.
16. Calculate the parity of the LSN and MSN of pixels in parallel.
17. Check if the parity of the LSN of the pixel is not equal to the watermark bit, complement **D1**; otherwise, leave it unchanged. The exact process is done for the MSN with **D5**. This process works in parallel.
18. If there is a pixel in the current block, pick the next pixel and go to step 16.
19. If there is a block of the current frame's part, pick the next block and next watermark bit, then go to step 15.
20. Send the watermarked part to PC.
21. Divide the watermarked byte back into its constituent LSNs. This will give you two sets of four bits each, corresponding to the LSN of the original and neighboring pixels.
22. Take the MSN of the original and neighboring pixels and merge them with the corresponding LSNs. This will give us the watermarked pixels.
23. Put these newly formed watermarked pixels back into their original positions in the part of the frame from which they were initially extracted.
24. If there is a part of the current frame, receive the next part, load the next watermark part, and then go to step 11.
25. Re-integrate the watermarked part back into its original quadrant within the frame.
26. Combine the Red, Green, and watermarked Blue channels to form the full RGB watermarked frame.
27. Integrate this watermarked frame back into the video sequence. Repeat these steps for all the frames that need to be watermarked to construct the final watermarked video.
28. The steps are repeated until all video frames are completed.
29. End.

The limited memory of the Xilinx Spartan 3E FPGA necessitates dividing a video frame into four parts for processing and forming eight-byte bits. Figure (3) shows all the steps of embedding watermark processing in the proposed system.

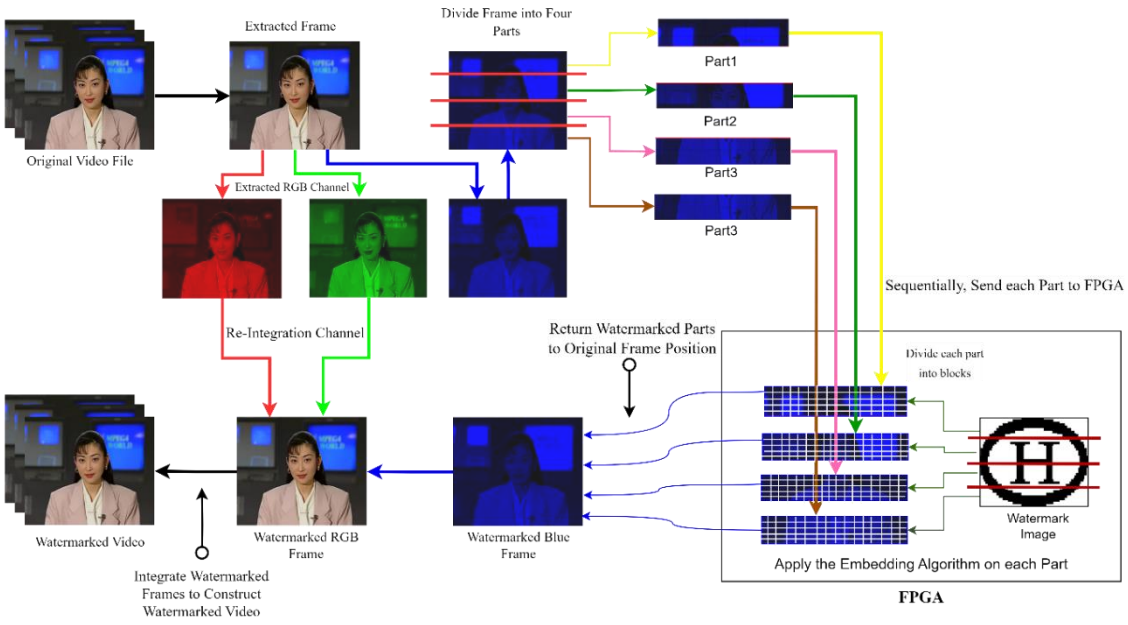


Fig. 3. General Block Diagram for Proposed Watermark Embedding Process

3.2 Extracting Algorithm

The extraction of the hidden watermark is as crucial as the embedding process. The successful recovery of the watermark is essential for establishing ownership of the resources. The extraction procedure involves using the watermarked video as input, and as a result, the system retrieves the hidden watermark image. The following steps illustrate the extraction process:

1. Read the watermarked video.
2. Subject the watermarked video to one of the proposed attacks.
3. Execute the same steps in the embedding algorithm from 2 to 5.
4. Send the formed part to FPGA.
5. As in steps 5 and 6 in the embedding algorithm, divide the part into blocks and determine the number of pixels in each block.
6. Pick the first block.
7. Set the counter to zero.
8. Pick the first pixel of the block.
9. Divide the pixel into MSB and LSB and calculate the parity for each one.
10. If the parity equals zero, increase the counter by one.
11. If there is a pixel in the current block, pick the next pixel of the block and go to step 9.
12. If the counter is greater than the number of pixels in the block, append ‘0’ to the watermark; otherwise, append ‘1’.
13. If there is a block of the current part, pick the next block and go to step 7.
14. Send the watermark part to the PC.
15. If there is a part of the current frame, receive the next part and go to step 5.
16. Construct the watermark image by integrating watermark parts. Output the watermark image.
17. The steps are repeated until all watermarked frames are completed.
18. End.

The watermark extraction algorithm is implemented on each frame of the watermarked video, which has been subjected to one of the proposed attacks. This process involves retrieving the watermark from each frame to achieve the highest quality of the watermark image. Figure (4) illustrates the watermark extraction process using a hardware-based parallel way.

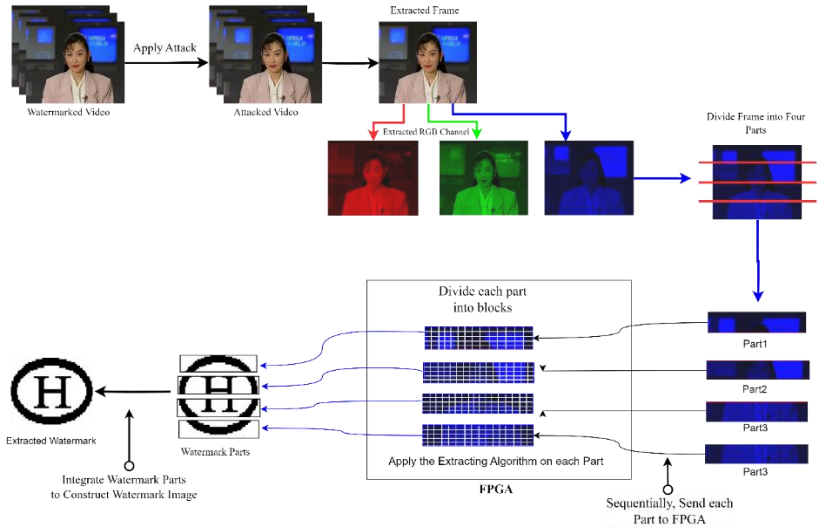


Fig. 4. General Block Diagram for Proposed Watermark Extracting Process

4. Experimental Results

In the following subsections, we present the implementation of the proposed parallel video watermarking system.

4.1 Experimental Setup

The evaluation of the proposed system involved both software and hardware setups. MATLAB 2015b on a Windows 7 computer with an Intel Core i7 CPU and 8 GB RAM was used for software testing. Hardware tests were conducted on a Spartan 3E-XC3S1600 FPGA, utilizing parallel processing. The performance was assessed using three AVI RGB color videos with different motion and texture characteristics: "Akiyo," "Football," and "Foreman," each sized at 768x640. A 64x40 pixel binary watermark was embedded into 100 frames of each video. Figure (5) displays snapshots from all three test videos and watermarks.

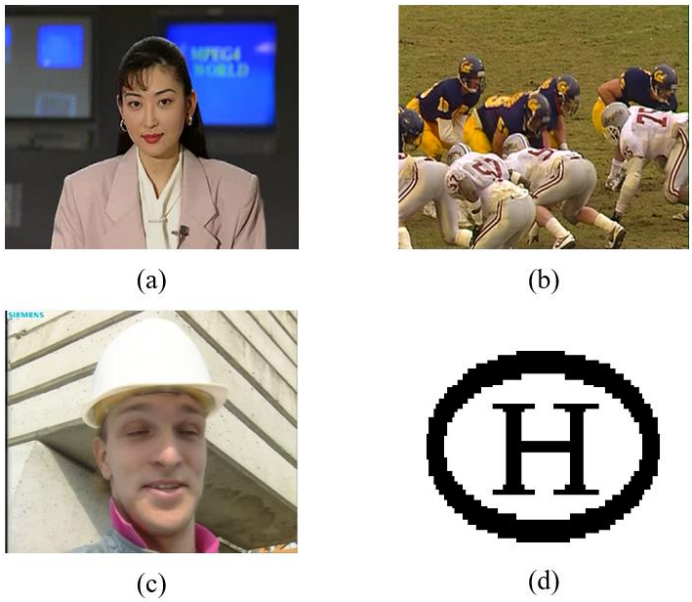


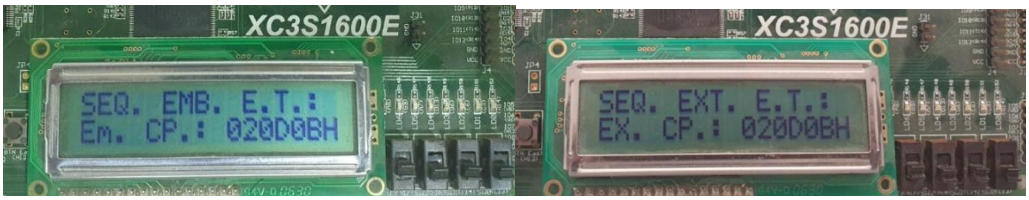
Fig. 5. Snapshots of the test videos with Watermark Image.

(a) Akiyo. (b) Football. (c) Foreman. (d) Watermark.

4.2 Execution Time

The execution time for both embedding and extracting processes was calculated for each tested video and presented in Table (1). It's important to note that these times include the embedding or extraction processes and the data transfer time between the PC and the FPGA, encompassing both pre-processing and post-processing phases. Additionally, the time for executing the

embedding or extracting process inside the FPGA device for one part of a frame was recorded as 20D0B in hexadecimal at a frequency of 75 MHz, as shown in Figure (6).



(a) (b)

Fig. 6. FPGA LCD Display of Execution Times for Parallel Implementation:

(a) Embedding Time. (b) Extraction Time.

The hexadecimal number ***20D0B*** in decimal is ***134,411***. Since one frame contains four parts, the total number of clock cycles for one frame would be (***134,411 * 4 = 537,644***). The time taken for the frame embedding or extracting process is approximately ***13.33 ns***. Therefore, the time FPGA consumes to embed or extract one frame in seconds is equal to (537.644 * 13.33 ns = 0.00717 s) and can process ***548.418 Mbps***. The number of frames processed in one second using this system is ***139.47 fps***.

The video size is 768 x 640, one part is 768 x 160, and the part size becomes 384 x 160 pixels after forming one byte. So, the number of bits to send from PC to FPGA with start and stop bits is (***384 x 160 x 10 = 614,400 bits***). Therefore, the time consumed to transfer these bits to the FPGA using a serial cable with a transfer rate of 700,000 bps is ***0.87714 s*** for one part and ***3.50856 s*** for one frame.

For the extracting process, the transferred data to FPGA is the same as the embedding process, while we receive one part of the watermark. The watermark size is 64 x 40; one part is 64 x 10, so the number of bits received from FPGA with start and stop bits is (64 x 10 x 10 = 6400 bits). Therefore, the time to transfer these bits (one part) to PC is ***0.00914 s*** and **0.03656 s** for the whole watermark image.

Table 1. Embedding vs. Extraction Time per Frame in Serial FPGA-Based System

Video Name	Embedding Algorithm	Extracting Algorithm	Embedding Processing	Extracting Processing
Akiyo	0.00717	0.00717	9.088626	5.62189
Football	0.00717	0.00717	9.094805	5.62167
Foreman	0.00717	0.00717	9.088868	5.63492

4.3 Quality Evaluation

The outcomes of the quality metrics applied to evaluate the performance of the serial FPGA-based embedding algorithm are presented in Table (2).

Table 2. Quality Metrics Values in Average for Parallel Embedding Algorithm

Video	MSE	PSNR	SSIM
Akiyo	3.145953e-05	45.022505	0.986725
Football	3.147473e-05	45.020732	0.973
Foreman	3.06840e-05	45.131215	0.981318

As shown in Table (2), the results indicate high fidelity between the original and watermarked videos. The minimal difference between the values of one standard for each video indicates the system's effectiveness in applying it to various videos regarding the complexity of the scenes and content.

4.4 Robustness Evaluation

The proposed system's robustness was tested against various attacks: noise (salt & pepper noise with 0.4 density), geometric transformations (20° rotation without synchronization), scaling (downscaling to 20% of original size), cropping (20% from each side), and JPEG compression (with quality set to 80). The system's effectiveness against these attacks was evaluated by measuring the Normalized Cross-Correlation (NCC) value between the original and extracted watermarks. We calculate the minimum and maximum values of NCC for each video, as shown in Table (3).

Table 3. Average of Max and Min NCC Values under Various Attacks

	Akiyo		**Football**		**Foreman**	
	Min NCC	**Max NCC**	**Min NCC**	**Max NCC**	**Min NCC**	**Max NCC**
No Attack	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1
	1	1	1	1	1	1
Median Filter	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1
	1	1	1	1	1	1
Cropping 20%	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1
	0.60756	0.60756	0.60756	0.60756	0.60756	0.60756
Rescaling 20%	Frame#49	Frame#10	Frame#86	Frame#23	Frame#41	Frame#11
	0.9664	0.98008	0.95139	0.99502	0.97003	0.98907
Rotation 45^0	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1	Frame#1
	0.70743	0.70743	0.70743	0.70743	0.70743	0.70743
JPEG Compression	Frame#67	Frame#13	Frame#2	Frame#86	Frame#61	Frame#75
	0.66643	0.72165	0.23457	0.75279	0.36358	0.45347

We conclude from the NCC values in Table (3) that watermark extraction is generally close to 1 for most cases, indicating a high level of correlation between the extracted watermark and the original watermark image. Figure (7) shows the watermark image extracted from the tested videos after it was subjected to the previously mentioned attacks with NCC value.

5. Comparison Study

This section addresses the critical comparison points between the proposed system and those presented by previous works. Table (4) compares the proposed watermarking system with other studies, focusing on parameters like FPGA platform, watermarking domain, frequency, embedding throughput, quality assessment, and test video model. The system shows high embedding throughput, suitable for real-time applications, operating at 75 MHz. It demonstrates a balance of speed and quality, with competitive PSNR and SSIM values, highlighting the effectiveness of the spatial domain approach on FPGA, making it efficient for digital video watermarking.

Table (4): Summarization of the Comparison Study with Previous Work

Ref#	FPGA Platform	Processing Domain	Operation Frequency	Embedding throughput	Quality Assessment	Video Model
[25]	Cyclone IV	Spatial	88.03 MHz	93.7 fps	PSNR = 37.5 MSE = 0.1291	Raw Video Grayscale 640x480
[26]	Xilinx Virtex 7-XC7V20 0 0T	Spatial	150 MHz	62.328 Mbps	PSNR = 38.44 SSIM = 0.9647	Raw Video Grayscale (640x480)
[27]	Xilinx Virtex 7-XC7V20 0 0T	Spatial	100 MHz	46.146 Mbps	PSNR = 37.658 SSIM = 0.7445	Raw Video Grayscale
[28]	Virtex6 xc6vlx75tl	Spatial	72 MHz	-	PSNR = 48.46	Raw Video Grayscale (144x176)
Proposed System	Spartan 3E-XC3S1600	Spatial	75 MHz	139.47 fps 1010.84 Mbps	PSNR = 45.023 SSIM = 0.987	Xiph.org Video Test Media (768x640)

6. Conclusion

The study concludes that the proposed FPGA-based parallel video watermarking system, utilizing the LSB technique in the spatial domain, significantly accelerates watermark embedding and extraction for real-time video applications. Experimental results demonstrate the system's efficiency, notably reducing processing time and maintaining high video quality post-watermarking. Future work aims to further explore and enhance the system's capabilities, particularly regarding robustness and adaptability to various video formats and watermarking challenges.

References

[1] M. Kaczyński and Z. Piotrowski, "High-Quality Video Watermarking Based on Deep Neural Networks and Adjustable Subsquares Properties Algorithm," *Sensors*, vol. 22, p. 5376, 2022.

[2] O. F. Mohammad, M. S. M. Rahim, S. R. M. Zeebaree, and F. Ahmed, "A survey and analysis of the image encryption methods," *International Journal of Applied Engineering Research*, vol. 12, pp. 13265-13280, 2017.

[3] K. B. Obaid, S. Zeebaree, and O. M. Ahmed, "Deep learning models based on image classification: a review," *International Journal of Science and Business*, vol. 4, pp. 75-81, 2020.

[4] D. Megías, W. Mazurczyk, and M. Kuribayashi, "Data hiding and its applications: Digital watermarking and steganography," vol. 11, ed: MDPI, 2021, p. 10928.

[5] S. Charles, V. Bindschaedler, and P. Mishra, "Digital watermarking for detecting malicious intellectual property cores in NOC architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, pp. 952-965, 2022.

[6] Z. Gong, N. Qin, and G. Zhang, "Visible watermarking in document images using two-stage fuzzy inference system," *The Visual Computer*, pp. 1-12, 2022.

[7] P. Garg and A. Jain, "A robust technique for biometric image authentication using invisible watermarking," *Multimedia Tools and Applications*, vol. 82, pp. 2237-2253, 2023.

[8] Z. Yuan, Q. Su, D. Liu, and X. Zhang, "A blind image watermarking scheme combining spatial domain and frequency domain," *The visual computer*, vol. 37, pp. 1867-1881, 2021.

[9] Z. Yuan, Q. Su, D. Liu, X. Zhang, and T. Yao, "Fast and robust image watermarking method in the spatial domain," *IET Image Processing*, vol. 14, pp. 3829-3838, 2020.

[10] A. Chopra, S. Gupta, and S. Dhall, "Analysis of frequency domain watermarking techniques in presence of geometric and simple attacks," *Multimedia Tools and Applications*, vol. 79, pp. 501-554, 2020.

[11] P. Garg and R. R. Kishore, "An efficient and secured blind image watermarking using ABC optimization in DWT and DCT domain," *Multimedia Tools and Applications*, vol. 81, pp. 36947-36964, 2022.

[12] H. M. Yasin, A. B. Sallow, and R. Z. Mahmood, "Efficient Watermark Embedding and Extracting in Raw Digital Video: Leveraging the Least Significant Bit Technique in the Spatial Domain," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, pp. 491-504, 2024.

[13] H. M. Yasin, S. R. Zeebaree, M. A. Sadeeq, S. Y. Ameen, I. M. Ibrahim, R. R. Zebari, *et al.*, "IoT and ICT based smart water management, monitoring and controlling system: A review," *Asian Journal of Research in Computer Science*, vol. 8, pp. 42-56, 2021.

[14] V. Jayanthi, S. Pitchai, and M. Smitha, "Design a hybrid FPGA architecture for visible digital image watermarking in spatial and frequency domain," *Journal of Circuits, Systems and Computers*, vol. 31, p. 2250020, 2022.

[15] R. K. Singh, D. K. Shaw, and M. J. Alam, "Experimental studies of LSB watermarking with different noise," *Procedia Computer Science*, vol. 54, pp. 612-620, 2015.

[16] S. Zebari and N. O. Yaseen, "Effects of parallel processing implementation on balanced load-division depending on distributed memory systems," *J. Univ. Anbar Pure Sci*, vol. 5, pp. 50-56, 2011.

[17] S. Zeebaree, S. Ameen, and M. Sadeeq, "Social media networks security threats, risks and recommendation: A case study in the kurdistan region," *International Journal of Innovation, Creativity and Change*, vol. 13, pp. 349-365, 2020.

[18] A. A. Yazdeen, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, O. M. Ahmed, and R. R. Zebari, "FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review," *Qubahan Academic Journal*, vol. 1, pp. 8-16, 2021.

[19] I. M. Zebari, S. R. Zeebaree, and H. M. Yasin, "Real time video streaming from multi-source using client-server for video distribution," in *2019 4th Scientific International Conference Najaf (SICN)*, 2019, pp. 109-114.

[20] X. Wang, Q. Qin, and Y. Cheng, "Design and implementation of digital image watermark based on FPGA," *Recent Advances in Computer Science and Information Engineering: Volume 5*, pp. 223-229, 2012.

[21] S. Zeebaree, "DES encryption and decryption algorithm implementation based on FPGA," *Indones. J. Electr. Eng. Comput. Sci*, vol. 18, pp. 774-781, 2020.

[22] S. R. Zeebaree, A. B. Sallow, B. K. Hussan, and S. M. Ali, "Design and simulation of high-speed parallel/sequential simplified DES code breaking based on FPGA," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 2019, pp. 76-81.

[23] S. P. Maity, "Spread Spectrum Watermarking: Implementation in FPGA," in *Advanced Techniques in Multimedia Watermarking: Image, Video and Audio Applications*, ed: IGI Global, 2010, pp. 455-485.

[24] R. Sinhal, I. A. Ansari, and C. W. Ahn, "Blind image watermarking for localization and restoration of color images," *IEEE Access*, vol. 8, pp. 200157-200169, 2020.

[25] K. Pexaras, I. G. Karybali, and E. Kalligeros, "Optimization and hardware implementation of image and video watermarking for low-cost applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, pp. 2088-2101, 2019.

[26] S. Das, A. K. Sunaniya, R. Maity, and N. P. Maity, "Efficient FPGA implementation of corrected reversible contrast mapping algorithm for video watermarking," *Microprocessors and Microsystems*, vol. 76, p. 103092, 2020.

[27] S. Das, A. K. Sunaniya, R. Maity, and N. P. Maity, "Parallel hardware implementation of efficient embedding bit rate control based contrast mapping algorithm for reversible invisible watermarking," *IEEE Access*, vol. 8, pp. 69072-69095, 2020.

[28] R. Dalbouchi, S. Dhahri, M. Elhaji, and A. Zitouni, "Software/Hardware implementations of a video watermarking scheme based on motion vectors," in *2017 International Conference on Engineering & MIS (ICEMIS)*, 2017, pp. 1-5.