frontiers in Computer Science

TYPE Original Research
PUBLISHED 16 July 2026
DOI 10.3389/fcomp.2026.1886719

Check for updates

OPEN ACCESS

EDITED BY
Rogério Dionísio,
Polytechnic Institute of Castelo Branco, Portugal

REVIEWED BY
Narendra Chahar,
National Forensic Sciences University, India
Ziyad Tariq,
University of Diyala, Iraq

*CORRESPONDENCE
Mehmet İrfan Gedik
✉ mehmetirfan.gedik@ostimteknik.edu.tr

RECEIVED 20 May 2026
REVISED 14 June 2026
ACCEPTED 16 June 2026
PUBLISHED 16 July 2026

CITATION
Gedik M and Coşkun A (2026) Real-time saliency-guided deep watermarking on Kria KV260: a Vitis AI accelerated proxy architecture.
Front. Comput. Sci. 8:1886719.
doi: 10.3389/fcomp.2026.1886719

COPYRIGHT
© 2026 Gedik and Coşkun. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Real-time saliency-guided deep watermarking on Kria KV260: a Vitis AI accelerated proxy architecture

Mehmet İrfan Gedik[1,2]* and Aysun Coşkun[3]

[1]Department of Hybrid and Electric Vehicles Technology, Vocational School of Higher Education, OSTIM Technical University, Ankara, Türkiye, [2]Department of Advanced Technologies, Graduate School of Natural and Applied Sciences, Gazi University, Ankara, Türkiye, [3]Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, Türkiye

The rapid growth of Industrial Internet of Things (IIoT) ecosystems and autonomous surveillance networks has necessitated the shift of digital content security from central servers to the data-generating edge. In industrial security scenarios, operators must continuously monitor live video streams and optionally capture high-resolution, verifiable evidence snapshots. However, high computational costs and hardware-based precision losses prevent the real-time execution of deep learning-based watermarking models on resource-limited embedded devices. This study proposes a hardware-aware and semantically oriented real-time watermarking architecture running on the Xilinx Kria KV260 FPGA platform. The fundamental innovation of the proposed system is the asynchronous "Proxy Frame" software architecture, which allows heavy Convolutional Neural Networks (CNNs) to run in the background, isolated from the live video stream. Thus, highly secure watermarking can be performed at 1080p resolutions without compromising the fluidity of the 30 FPS live preview offered to the operator. Furthermore, a Noise-Assisted Dithering technique, inspired by stochastic resonance, was used to mitigate the "Signal Fading" problem arising from watermark signal loss during 8-bit integer (INT8) quantization on the Deep Learning Processing Unit (DPU). By injecting controlled Gaussian noise into the quantized inference pipeline, the detectability of subthreshold weak watermark signals was increased, reducing the hardware Bit Error Rate (BER) from 18.75% to 13.06%. MobileNetV2 and ResNet50-FCN based saliency models achieved an average PSNR visual quality of 39.86 dB by concealing the payload in perceptually insignificant regions. Under clean hardware conditions, the system achieved a baseline BER of 2.6% (with MobileNetV2). In attack tests, the BER remained below 15% for MobileNetV2 under most standard degradations, with the stated exceptions of severe JPEG compression and deliberate geometric cropping. This end-to-end hardware and software solution demonstrates that theoretical deep learning models can be integrated into industrial smart cameras without experiencing performance bottlenecks, particularly for static infrastructure monitoring.

KEYWORDS
deep watermarking, DPU acceleration, FCN-ResNet50, Kria KV260, MobileNetV2, noise-assisted dithering, PL-PS DNA

1 Introduction

With the rapid proliferation of Industrial Internet of Things (IIoT) ecosystems and next-generation autonomous surveillance networks, source authentication of visual data and the protection of intellectual property rights have become vital requirements. While traditional security architectures processed data on central servers, in modern industrial systems this processing has shifted to the "edge" points where the data is generated. In this context, watermarking critical snapshots from a security camera or industrial sensor with an immutable and robust signature specific to the hardware in which they were generated is the most effective way to create a "Hardware-Bound Authentication" across the system. When examining FPGA-based image capture studies in the literature, two extreme approaches are generally observed:

Classical transform-based methods: These methods, using Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) (Mohanty et al., 2009; Jeong et al., 2009; Kaibou and Azzaz, 2021), can process real-time video while keeping the computational load low, but they lack the ability to analyze visual content (semantic intelligence) and are vulnerable to modern attacks.

Deep learning-based methods: While these CNN-based methods (Zhu et al., 2018; Zhang et al., 2024; Hussain et al., 2022) offer superior robustness, the computational costs are so high that maintaining a live camera stream (preview) on embedded devices becomes impossible. This often forces the system to operate "offline."

In an industrial surveillance scenario (e.g., static infrastructure security, slow-moving production line quality control, or plant security), the operator needs to continuously monitor the live stream at 30 FPS and instantly capture a high-resolution (1080p) evidence image (snapshot) when a suspicious situation is detected. The current literature lacks an integrated hardware architecture capable of performing "On-Demand High-Res Watermarking" by running heavy deep learning models in the background without disrupting live monitoring flow.

1.1 Motivation and problem definition

The primary motivation for this study is to integrate the theoretical robustness of deep learning-based watermarking models into the "Monitor-and-Capture" (MCC) use case of industrial cameras without locking up system performance. The main engineering challenges encountered during this integration are:

- **Live streaming and inference bottleneck:** The inference time of a heavy CNN model (e.g., ResNet50) is longer than the live video stream time (33 ms @ 30 FPS). In traditional sequential programming models, this situation creates a security vulnerability by causing freezes and frame drops on the operator screen.
- **High resolution requirement:** While evidential images need to be in Full HD (1920 × 1080) resolution, AI accelerators (DPUs) on FPGAs generally operate at lower resolutions (256 × 256) due to memory limitations.
- **Quantization loss (signal fading):** When high-precision quantization models are downgraded from 32-bit floating-point to 8-bit integer (INT8) format for hardware efficiency, perceptually hidden (invisible) quantization signals risk being lost due to rounding errors.

1.2 Proposed contribution: system-level integration and proxy architecture

While the fundamental concepts of deep watermarking, FPGA acceleration, and hardware-rooted authentication are established in the literature, combining them into a seamless, high-resolution edge surveillance system presents severe engineering bottlenecks. Rather than claiming novel algorithmic inventions in basic encoder-decoder models, the primary novelty of this study lies in the *system-level integration* and Hardware/Software (HW/SW) co-design.

This paper proposes a hardware-aware architecture customized for "Real-Time Preview and On-Demand Semantic Marking" running on the Xilinx Kria KV260 Vision AI platform. Our approach adapts the "Proxy Frame" logic used in professional video editing software to embedded systems, isolating the live camera stream from the AI processing load. The system works as follows: While the live stream (30-60 FPS) is presented to the operator uninterrupted, the AI thread running in the background continuously generates a "Saliency" map on low-resolution proxy frames sampled from the stream. When the operator issues the "Get Proof" command, the system watermarks and saves the 1080p raw image using the most current semantic map. Specifically, our study's original contributions to solving these edge integration challenges are as follows:

1. **Asynchronous proxy architecture:** Thanks to a multi-threaded design that separates camera flow (UI) and Artificial Intelligence (AI) processes, live preview fluidity (30 FPS) is maintained even when running heavy models like FCN-ResNet50.
2. **Hybrid resolution management:** While the AI model is run on a low-resolution proxy frame (256 × 256) for memory efficiency, the watermarking process is performed on high-resolution (1920 × 1080) raw data, reducing processor load by 80%.
3. **Noise-assisted dithering for INT8 improvement:** For the first time in the literature, a noise-assisted dithering principle is used in an FPGA application to compensate for INT8 quantization loss on the DPU. By injecting controlled noise into the quantized inference pipeline, the Bit Error Rate (BER) on the hardware was significantly reduced.
4. **Industrial standard compliance:** The developed system ensures full compatibility with modern industrial AI workflows by using the standard Vitis AI stream and Xilinx DPU IP instead of custom RTL designs.

This work provides a practical and scientific framework demonstrating how deep learning-based security can be integrated into edge devices without disrupting the real-time usage habits of industrial cameras.

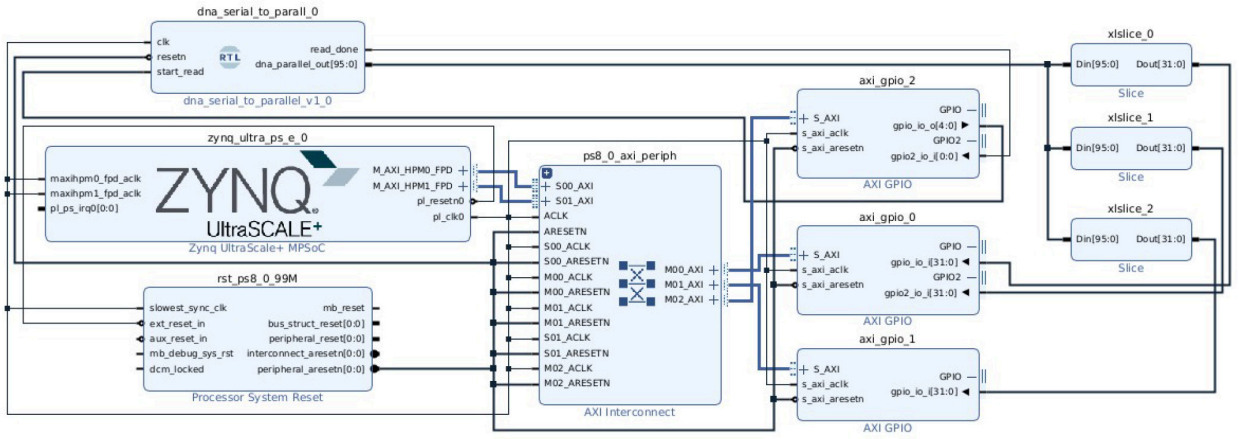


FIGURE 1
Vivado block design view illustrating the proposed hardware architecture. The Zynq® UltraScale+ Processing System (PS) on the left interfaces with three AXI GPIO blocks through an AXI Interconnect. The core component for hardware-based security is the custom RTL module located on the right, labeled `dna_serial_to_parallel`. Developed in Verilog HDL, this module leverages the UltraScale+ specific `DNA_PORTE2` (AMD, 2025) primitive to securely retrieve the unique, embedded 96-bit Device DNA from the silicon. It performs essential serial-to-parallel conversion and data slicing, enabling the PS to efficiently access the raw identity data via standard 32-bit AXI interfaces.

2 FPGA-based device identification (DNA) reading hardware architecture

2.1 System topology

The developed hardware architecture (Figure 1) is based on the Zynq UltraScale+ MPSoC (Multi-Chip System-on-a-Chip) architecture on the Xilinx Kria KV260 platform. The design is based on the AXI4-Lite interface connecting the Processor System (PS) to the Programmable Logic (PL) unit. The proposed block design consists of three main subsystems: (1) Central Control Unit (Zynq MPSoC), (2) Custom Design DNA Reading Core, and (3) Data Slicing and Transmission Interface. In the Vivado Block Design view, the Zynq UltraScale+ processor system (left) is connected to three AXI GPIO blocks via the AXI Interconnect; On the right side is a custom RTL module (`dna_serial_to_parallel`) containing the `DNA_PORTE2` (AMD, 2025) primitive and slicing blocks that divide the 96-bit data into 32-bit chunks.

2.2 Custom designed RTL Core and Threat Model

The `dna_serial_to_parallel_0` module, designed using Verilog Hardware Description Language (HDL), forms the basis of hardware-based security. This module reads the 96-bit unique device ID (Device DNA), embedded in the FPGA silicon during manufacturing, using the `DNA_PORTE2` (Figure 2) primitive specific to the UltraScale+ architecture. Designed to securely obtain the unique device ID within the Xilinx UltraScale+ architecture, this core hardware module plays a critical role in converting raw serial ID data into a parallel format that the Processor System (PS) can

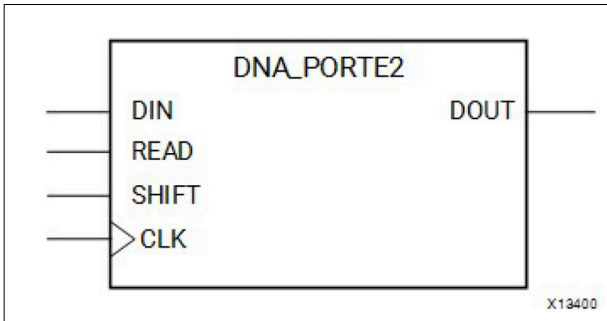


FIGURE 2
DNA access port (AMD, 2025).

easily access via its AXI interfaces, due to the inherent bandwidth limitations of the serial interface.

Threat model scope and hardware-bound security: This system is primarily designed to provide a *Physical Proof of Origin*, ensuring that a generated watermark is intrinsically tied to the specific silicon of the edge node, thereby defending against rogue device substitution and external sensor spoofing.

Regarding the threat of root-level privilege escalation raised for industrial scenarios, we acknowledge that a compromised Operating System (OS) layer poses significant risks. However, the core security of our architecture relies on the unalterable nature of the silicon-embedded PL DNA, rather than software-level encryption. The Processing System (PS) extracts this 96-bit cryptographic identity not through vulnerable high-level software APIs, but by directly accessing the physical hardware registers via the AXI Bus.

As implemented in our system, the 96-bit device ID exceeds standard bus widths and is thus divided into three 32-bit segments at the hardware level. To eliminate latency and bypass high-level OS vulnerabilities during the extraction phase, the system

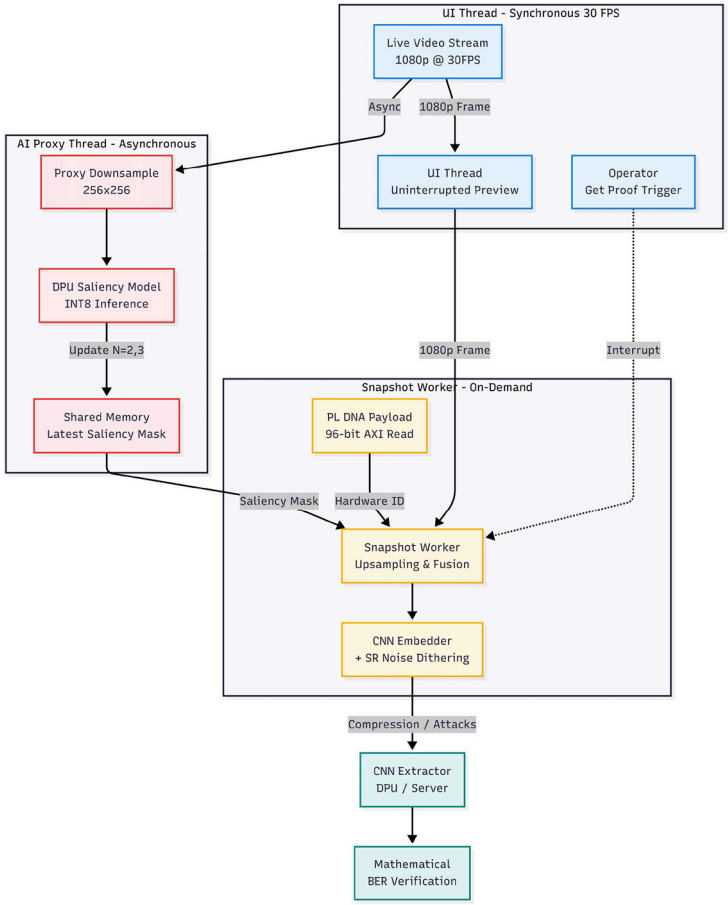


FIGURE 3
General block diagram of the proposed Vitis AI-powered asynchronous proxy architecture. The system decouples the high-resolution live preview (UI Thread) from the computationally heavy AI processing (Proxy Thread). When the operator triggers a snapshot, the most recent saliency mask is retrieved from shared memory, combined with the 96-bit Hardware DNA and SR noise, and processed by the CNN Embedder. The extracted payload is subsequently mathematically verified.

utilizes the `devmem` utility to perform direct physical memory reads from specific AXI addresses (0xA0010000, 0xA0000008, and 0xA0000000). The Most Significant Word (MSW), Middle Word (MID), and Least Significant Word (LSW) are fetched in real-time and concatenated to form the payload.

Therefore, while an attacker with root privileges could theoretically intercept the RAM or tamper with the software logic, they cannot alter the physical PL DNA fused into the chip. The generated watermark remains a unique signature of that exact physical board. Our primary security objective is to bind the digital content directly to physical hardware (Hardware-Bound Authentication), providing strong device-origin evidence under defined operational assumptions.

Furthermore, to mitigate the security risks associated with dynamic bitstream loading during Phase 2, while the current experimental prototype focuses on validating the real-time AI and watermarking pipeline, a production-grade industrial deployment assumes the activation of the hardware-level cryptographic countermeasures provided by the Zynq UltraScale+ platform (as specified in AMD UG570). The system is inherently capable of utilizing AES-GCM (256-bit) decryption and RSA-4096 authentication for the dynamic bitstreams. When provisioned in an industrial deployment, even if a malicious actor gains root access to the Linux OS and attempts to load a tampered `kv260-benchmark-b4096.bit.bin` file, the decryption and verification are strictly handled by the hardware-isolated Configuration Security Unit (CSU). If the AES-GCM authentication tag or the RSA signature is invalid, the CSU autonomously aborts the configuration, ensuring that software-level root privileges cannot bypass the silicon-level bitstream verification.

Furthermore, regarding the transmission and evaluation scope, the threat model of this study is primarily centered on *Industrial Forensics* rather than algorithmic watermark removal attacks. In a real-world IIoT surveillance network, the most prevalent threats to the integrity of an evidentiary snapshot transmitted from the edge to a central Network Video Recorder (NVR) are severe channel degradations. These include heavy JPEG compression (routinely applied by NVRs to save storage), sensor/cable noise, and accidental camera misalignments (rotation or blur). Therefore, the overall security proposition of the system is twofold: (1) ensuring the payload is securely bound to the physical silicon (Device DNA) under the defined operational assumptions, and (2) ensuring this integrated hardware signature possesses the deep-learning-based

signal processing robustness required to survive extreme industrial channel distortions without being destroyed.

To demonstrate the holistic hardware identity architecture of the Zynq UltraScale+ MPSoC, we present two different extraction methods. The PS-side eFUSE identity can be accessed via XSCT for debugging purposes, while the actual watermark payload is dynamically extracted from the PL-side `DNA_PORTE2` primitive via physical AXI memory addresses during runtime. Listing 1 shows both readouts.

```
# --- METHOD 1: PS eFUSE via XSCT (Reference) ---
xsct% targets 4
xsct% mrd 0xFFCC1014
FFCC1014: 40000000
xsct% mrd 0xFFCC1010
xsct% mrd 0xFFCC100C
FFCC100C: 4C11C245

# --- METHOD 2: PL DNA via AXI (Actual Payload) ---
root@kria:~# devmem 0xA0010000 32  # Read MSW
0x40020000
root@kria:~# devmem 0xA0000008 32  # Read MID
0x016C6008
root@kria:~# devmem 0xA0000000 32  # Read LSW
0x4C11C245

# Concatenated 96-bit PL DNA: 40020000016C60084C11C245
```

LISTING 1 Hardware identity readouts: PS-side eFUSE (XSCT) and PL-side DNA (AXI).

3 Related works

Verifying the source of digital content and protecting copyrights has become a critical requirement in the Industrial Internet of Things (IIoT) era. Studies in the literature initially started with software-based algorithms and evolved over time to FPGA-based hardware acceleration solutions due to real-time processing requirements. This section examines the current literature in three main categories: (1) Traditional Transform-Based FPGA Applications, (2) Deep Learning-Based Watermarking, and (3) Hardware-Accelerated Security Architectures.

3.1 Traditional transform-based FPGA applications

A large part of the FPGA-based watermarking literature is based on Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) techniques, which have low computational complexity. In early studies, Mohanty et al. (2009) presented one of the first VLSI architectures that included a secure watermark generator (LFSR) and operated in the spatial domain. Similarly, Jeong et al. (2009) proposed a DWT-based architecture for high-resolution videos, but this method led to degradation in visual quality. More recent studies have integrated chaos theory and encryption techniques to increase the security layer. Kaibou and Azzaz (2021) developed a chaos-based color image watermarking scheme on an FPGA, but this method exhibited a high Bit Error Rate ($BER > 14\%$) against noise attacks. Even in the latest studies from 2023 and 2024, researchers such as Joshi et al. (2011) and Shivdeep et al. (2020) still focus on traditional DWT and spatial methods; these studies are generally limited to low resolutions such as 512×512 and lack the semantic robustness offered by modern deep learning models.

3.2 Deep learning based watermarking

With the rise of deep learning, the "HiDDeN" architecture proposed by Zhu et al. (2018) has created a paradigm shift in watermarking. These methods provide superior resistance to attacks by hiding the watermark not only in frequency coefficients but also in the deep features of the image. Zhang et al. (2024) managed to increase the watermark capacity with a dual guidance mechanism. However, the vast majority of these studies are designed to run on high-power GPUs and at 32-bit floating-point (FP32) precision, making them unsuitable for edge devices with limited resources.

3.3 Hardware acceleration and the current research gap

Studies on accelerating deep learning models on FPGAs are quite limited. One of the most important studies in this field, FastStamp was presented by Hussain et al. (2022) and demonstrated that steganographic operations can be accelerated on FPGAs with a hardware-friendly neural network architecture. However, FastStamp focused on low-resolution (128×128) images and did not address the high-resolution scalability required by industrial surveillance scenarios. On the other hand, in the field of hardware security, James et al. (2025) and Zhang and Liu (2017) proposed hardware-bound authentication models using FPGA's Physical Unclonable Functions (PUF) and Device DNA. However, these studies do not offer an integrated "In-Camera Watermarking" architecture that embeds the acquired credentials into the visual content.

3.4 Lack of literature

When the current literature is examined, it is evident that recent advancements between 2025 and 2026 have produced highly sophisticated deep learning-based watermarking models. Comprehensive reviews and novel spatial-temporal frameworks (Bistroń et al., 2026; Cedillo-Hernandez et al., 2026) demonstrate the superior robustness and semantic capabilities of these architectures over traditional methods. However, these state-of-the-art studies are almost exclusively evaluated in high-precision (FP32) software environments (e.g., high-end GPUs). Conversely, the very few recent studies aiming for real-time edge processing on FPGAs (Niharika et al., 2025) are forced to rely on lightweight, traditional transform-based algorithms (such as DCT) to achieve ultra-fast frame rates, thereby sacrificing the semantic intelligence of CNNs.

An integrated industrial architecture that successfully bridges this gap–by running heavy AI models via a Deep Learning Processing Unit (DPU) for high-resolution (1080p) real-time "Snapshot" security, while simultaneously mitigating the inevitable INT8 quantization losses (Signal Fading) at the hardware level–remains completely unexplored. This study aims to fill this critical research gap by adapting the noise-assisted dithering principle to hardware-based deep watermarking on the edge.

4 Proposed method

In this study, a hierarchical architecture based on hardware/software co-design was developed to overcome the bottleneck in high-resolution surveillance and evidence collection in industrial edge devices. The system integrates Deep Learning Processing Unit (DPU) acceleration, proxy frame-based asynchronous data flow, and the noise-assisted dithering principle on the Xilinx Kria KV260 Vision AI Starter Kit.

4.1 Hardware platform and system topology

The core computing unit of the developed system is the Kria KV260 module based on the Xilinx Zynq UltraScale+ MPSoC (XCK26), which combines a quad-core ARM Cortex-A53 application processor (PS) and programmable logic (PL) units on a single chip. To handle the AI inference load, a DPU core with B4096 architecture and offering a practical INT8 deep learning processing capacity of 1.2 TOPS (running at a verified clock frequency of 300 MHz), optimized specifically for edge inference. To ensure the physical security of the hardware, a 96-bit unique identifier (Device DNA) embedded during the silicon manufacturing phase is read via a dedicated AXI interface to form the cryptographic core of the payload.

4.2 Asynchronous "proxy frame" software architecture and temporal trade-offs

In industrial surveillance scenarios, uninterrupted (30 FPS) live preview of the operator is essential. However, the inference time of heavy-duty watermarking models like ResNet50 is longer than the video frame duration (33 ms). To solve this problem, an asynchronous "Producer-Consumer" model has been developed with a "Proxy Frame" strategy adapted from the video editing industry (Figure 3). The system runs on three isolated threads that do not block each other: Camera Worker (UI Thread): Captures 1080p (1920 × 1080) raw images at 30 FPS via the V4L2 interface and displays them on the screen. This stream is independent of the AI speed, so there is never any freezing on the screen. AI Proxy Worker: This unit, running in the background, converts the image from the stream into a low-resolution "Proxy Frame" of 256 × 256 and performs Saliency inference on the DPU. The AI operates at its own speed (e.g., 15–20 FPS) depending on the hardware capacity and writes the most up-to-date Saliency map to shared memory. The Snapshot Worker is triggered when the operator gives the "Capture" command. It captures the current 1080p high-resolution raw image, scales the latest mask generated by the AI to 1080p using hardware-accelerated interpolation, and performs the watermarking process. It is critical to note that this asynchronous reuse introduces a temporal latency (e.g., an age of $N = 3$ frames for the reused saliency map). While this structural trade-off prevents UI freezing and ensures fluidity, it can cause semantic misalignment if applied to highly dynamic scenes with fast-moving objects. Therefore, this proxy architecture is specifically proposed for static infrastructure monitoring or environments with slow-moving subjects, where the semantic background context remains relatively stable across consecutive frames.

4.3 Device authentication (design_1_wrapper.bit)

Due to security and identity requirements, the system's operational flow begins with Phase 1, the Authentication Layer. In this phase, only the `design_1_wrapper.bit` is used temporarily to ensure efficient use of hardware resources. The bit configuration is loaded into the FPGA's Programmable Logic (PL) region. This loaded bitstream contains a specially designed DNA Read Core that includes the `DNA_PORTE2` primitive specific to the Zynq UltraScale+ architecture, the AXI GPIO, and the data slicing logic. The Processing System (PS) triggers the `start_read` signal via the AXI I/O interfaces. The 96-bit unique device ID is read serially, parallelized, and stored in temporary memory by reading from three separate 32-bit AXI addresses. Once this critical identity data is obtained, the Phase 1 hardware task is complete, and the system proceeds to the next phase, which requires higher computational power.

4.4 AI acceleration and inference (kv260-benchmark-b4096.bit.bin)

Following the secure acquisition of identity data, the system transitions to Phase 2: AI Acceleration Layer. This transition is achieved by dynamically loading the `kv260-benchmark-b4096.bit.bin` bitstream onto the FPGA. This loading process completely erases the DNA reader and associated AXI logic in the PL region; instead, it installs the DPU (Deep Learning Processing Unit) hardware necessary to accelerate computationally intensive algorithms such as Saliency Detection. The DPU hardware is now active, and the PS uses this accelerator to perform inference tasks. The Saliency maps or results obtained in this phase are processed using the unique DNA ID collected in Phase 1; thus, both high-performance computation and hardware-based identity security are achieved simultaneously. The detailed application bitstream deployment schedule for these phases is summarized in Table 1.

TABLE 1 Application bitstream configuration.

Bitstream name	Content and purpose	Deployment time
`design_1_wrapper.bit`	Security layer (DNA reader)	Application startup
`kv260-benchmark-b4096.bit.bin`	AI acceleration (DPU core)	App. core process

4.5 DPU-optimized deep models

Two different backbone architectures were ported to the FPGA with the Vitis-AI compiler to detect "perceptual blind spots" in the image: MobileNetV2-UNet: An architecture using depth-wise separable convolutions, streamlined for situations requiring low latency. FCN-ResNet50: A heavy architecture with a large reception area capable of precise object segmentation in complex industrial scenes. Both models were trimmed to generate Logits (raw values) instead of the Sigmoid layer, which is not supported on the DPU, and the final activation process was performed on the CPU.

4.6 Deep watermarking network architecture

To embed the hardware-specific DNA payload into video frames in a robust and imperceptible manner, an Encoder-Decoder architecture inspired by the HiDDeN framework (Zhu et al., 2018) and adapted for the FPGA environment was used.

4.6.1 The embedder (encoder) network

To address potential ambiguities regarding the payload, it is critical to note that unlike traditional watermarking systems that embed a 2D visual "logo" (e.g., a copyright graphic), the proposed architecture embeds a strictly 1D cryptographic sequence (the 96-bit Device DNA). The Embedder network (E) functions as a signal fusion module. It takes the original cover image (I_{cov}) and the 96-bit Device DNA payload (W) as input. A masking mechanism is applied to prevent the visual quality of the Regions of Interest (ROI), as determined by the Saliency model (M_{sal}), from being degraded by the watermark. The DNA payload is first expanded and concatenated with the high-level features of the cover image. The network processes this fused data through a series of residual blocks (Conv-BN-ReLU) to generate a residual watermark signal (I_{res}). The final watermarked image (I_w) is obtained by adding this residual to the original image, weighted by the inverse of the saliency map.

The detailed procedural flow of this real-time embedding phase on the edge hardware is summarized in Algorithm 1.

Here, α represents the global embedding strength factor. The term $(1 - M_{sal})$ ensures that the watermark signal is attenuated in visually salient regions (e.g., human faces, traffic signs) and amplified in highly textured background areas where the Human Visual System (HVS) is less sensitive to noise.

Input: Live 1080p frame I_{cov}, 96-bit PL DNA Payload W, Saliency Map M_{sal}, Strength α, Noise Var σ^2
Output: Watermarked 1080p frame I_w
Phase 1: Input pre-processing
  Resize I_{cov} to DPU spatial dimensions (e.g., 256×256) $\rightarrow I_{input}$
  Normalize I_{input} and prepare payload tensor W
Phase 2: DPU inference (encoder network)
  Execute asynchronous DPU job: *Job* $\mathcal{D}$ $E(I_{input}, W)$
  Wait for DPU interrupt and fetch residual tensor $\rightarrow I_{res_int8}$
Phase 3: Noise-assisted dithering
  Dequantize I_{res_int8} to FP32 using DPU scale factors $\rightarrow I_{res_fp32}$
  Generate Gaussian Noise: $\eta \sim \mathcal{N}(0, \sigma^2)$
  Add noise to bypass quantization barrier: $I_{dithered}$ $\mathcal{D}$ I_{res_fp32} $\mathcal{C}$ η
Phase 4: High-resolution masking and fusion
  Upsample $I_{dithered}$ and M_{sal} to 1920×1080 using hardware interpolation
  Apply global strength α and inverse semantic mask: $Mask_{inv}$ $\mathcal{D}$ $(1 - M_{sal})$
  Combine with original frame: I_w $\mathcal{D}$ clip$(I_{cov}$ $\mathcal{C}$ $\alpha \cdot Mask_{inv} \cdot I_{dithered}, 0, 255)$
Return I_w

Algorithm 1. Real-time CNN-based embedding and stochastic resonance.

4.6.2 The extractor (decoder) network

The extractor network (D) is a lightweight Convolutional Neural Network (CNN) designed to run efficiently on a DPU or a central validation server. It takes a potentially compromised image (I'_w) as input and aims to reconstruct the original DNA payload (W'). The architecture consists of four strided convolution layers that progressively downsample the spatial dimensions while increasing the channel depth, followed by a Global Average Pooling (GAP) layer and a fully connected layer. To provide robustness against geometric distortions like rotation, the extractor incorporates a Spatial Transformer Network (STN) preprocessing module that learns affine transformation parameters to realign the target region before decoding.

Because the payload is a binary bitstring, visual extraction is not applicable; instead, authenticity is mathematically verified. The step-by-step extraction procedure on the DPU is outlined in Algorithm 2.

The extractor is trained to minimize the bit error rate between W and W', even if the image is subjected to degradations such as JPEG compression or noise addition.

4.7 INT8 signal enhancement with noise-assisted dithering

The most original technical contribution of this work is the solution it provides to the "Signal Fading" problem caused by INT8 quantization. In models trained with FP32 sensitivity, microscopic

Input: Suspected/attacked frame I'_w, Reference PL DNA W_{ref}
Output: Authenticity decision (True/False), Bit Error Rate (BER)
Phase 1: Geometric alignment and pre-processing
  Calculate affine matrix θ via Spatial Transformer Network (STN)
  Realign target frame to correct rotation/scale: $\mathcal{Q}_w \; \mathcal{D} \; STN(I'_w, \theta)$
  Resize $\mathcal{Q}_w$ to DPU dimensions $(256 \times 256) \rightarrow I'_{input}$
  Normalize pixel values and load to continuous memory
Phase 2: DPU inference (decoder network)
  Execute asynchronous DPU job: $Job \; \mathcal{D} \; D(I'_{input})$
  Fetch raw extracted logits from DPU memory $\rightarrow L_{raw}$
Phase 3: Bit reconstruction and verification
  Apply Sigmoid activation: $P \; \mathcal{D} \; \frac{1}{1\mathcal{C}e^{-L_{raw}}}$
  Threshold probabilities to binary sequence: $W' \; \mathcal{D} \; (P > 0.5)$
  Calculate Bit Error Rate: $BER \; \mathcal{D} \; \frac{1}{96} \sum_{i\mathcal{D}1}^{96} |W'_i - W_{ref_i}|$
if $BER < 15\%$ (e.g., Threshold Acceptance **then**
   Return *Authentic (True), BER*
else
   Return *Forged (False), BER*
end if

Algorithm 2. Hardware-bound deep extraction and authentication.

watermark signals (e.g., amplitude $<$ 0.01) are often lost when rounded to an integer on the DPU, effectively taking a value of 0. To overcome this, a noise-assisted dithering technique, closely related to the principle of Stochastic Resonance (SR), has been applied.

Unlike classical deterministic dithering, noise-assisted dithering leverages the non-linear thresholding behavior of the INT8 quantizer. By injecting controlled, zero-mean Gaussian noise (η) into the continuous watermark pattern before quantization, weak sub-threshold signals are stochastically lifted across the quantization boundary. The embedding equation becomes Equation 1:

$$I_w = I_{orig} + \alpha \cdot (1 - M_{saliency}) \cdot (W_{pattern} + \eta) \tag{1}$$

Mathematical justification for Gaussian noise: Let x be a deterministic, weak watermark signal such that $0 < x < \frac{Q}{2}$, where Q is the quantization step size of the INT8 DPU. In a standard quantizer $Q(\cdot)$, this signal is truncated to zero: $Q(x) = 0$.

When zero-mean Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2)$ is added, the probability P that the combined signal $(x + \eta)$ exceeds the quantization threshold $T = \frac{Q}{2}$ is given by the complementary error function (Q-function) in Equation 2:

$$P\left((x+\eta) \geq \frac{Q}{2}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{\frac{Q}{2}-x}^{\infty} e^{-\frac{\eta^2}{2\sigma^2}} d\eta = \frac{1}{2}\text{erfc}\left(\frac{Q/2 - x}{\sigma\sqrt{2}}\right) \tag{2}$$

Consequently, the mathematical expectation $\mathbb{E}$ of the quantized output $Q(x+\eta)$ is no longer zero, but a function of the noise variance σ^2 (Equation 3):

$$\mathbb{E}[Q(x+\eta)] \approx Q \cdot P\left(\eta \geq \frac{Q}{2} - x\right) > 0 \tag{3}$$

Gaussian noise was specifically chosen over Uniform or Poisson distributions because its infinite-tailed distribution (unlike the bounded Uniform noise) ensures a non-zero probability of threshold crossing even for extremely small x, while maintaining a symmetric, zero-mean distortion that can be easily averaged out by the deep convolutional layers of the extractor network. This probabilistic activation allows the DPU to effectively "hear" the sub-threshold watermark signature that would otherwise be lost.

4.8 Training and attack simulation

To ensure full reproducibility, the proposed architectures were trained using the PyTorch framework. Deterministic random seeds were strictly enforced across all computational libraries. The networks were optimized using the Adam optimizer with a constant learning rate of 1×10^{-3} for 20 epochs and a batch size of 64. The composite loss function was formulated as a weighted combination of Mean Squared Error (MSE) for visual image fidelity and Binary Cross Entropy with Logits (BCEWithLogitsLoss) for the binary payload extraction accuracy. The payload generation strategy consisted of dynamically generating random binary sequences during each training iteration using a uniform distribution, ensuring the network generalizes to any unique Device DNA signature rather than overfitting to a single sequence. To achieve the optimal balance between visual imperceptibility and extraction robustness, a curriculum learning strategy was employed for the embedding strength (α). The training was initialized with a high embedding strength ($\alpha = 1.5$) to allow the extractor network to easily converge on the spatial patterns. Over the 20 epochs, this strength was linearly decayed to 0.15, forcing the network to detect increasingly subtle, sub-threshold signals. During real-time hardware deployment, the background embedding strength was further scaled down to $\alpha = 0.035$. For the extraction phase, a standard threshold of 0.5 was applied to the network's output to reconstruct the binary sequence. The models were trained on the DUTS-TR dataset (comprising a total of 10,553 images), partitioned strictly into an 80% training set and a 20% validation set.

Vitis-AI hardware compilation pipeline: To port the trained FP32 PyTorch models to the FPGA, the official AMD/Xilinx Vitis-AI 3.0 workflow was employed. First, the models were calibrated and quantized to INT8 precision using the `vai_q_pytorch` tool with an unlabeled calibration subset of 100 images to minimize quantization-induced signal fading. Subsequently, the quantized models were compiled targeting the B4096 DPU architecture using the `vai_c_xir` compiler. This process generated the `saliency_dpu_kv260.xmodel` and `extractor_dpu_kv260.xmodel` files, which are natively parsed by the Vitis AI Runtime (VART) APIs on the edge device.

To improve the system's robustness, a differential "Attack Layer" was added to the training loop. During training, the models were subjected to arbitrary JPEG compression ($Q \in [50, 100]$ uniformly), Gaussian noise addition ($\sigma^2 \in [0.01, 0.05]$), and geometric random rotations ($\pm 2^\circ$) applied dynamically per batch, making them highly

resilient to signal degradations frequently encountered in industrial edge deployments.

5 Experimental results

In this section, the performance of the proposed hardware-aware deep watermarking architecture on the Xilinx Kria KV260 FPGA platform is comprehensively evaluated. Experiments were conducted using two different backbone architectures (MobileNetV2 and ResNet50-FCN) to measure the system's real-time processing capacity (FPS), perceptual quality (PSNR/SSIM), and resilience to cyberattacks (BER).

5.1 Hardware implementation environment and test methodology

All experiments were performed on the Xilinx Kria KV260 Vision AI Starter Kit. The hardware overlay was synthesized using **Vivado 2023.1** on a host workstation running **Ubuntu 20.04 LTS**. The FPGA board was booted with the official Xilinx Vitis-AI 3.0 image (AMD, 2026b) (`xilinx-kv260-dpu-v2022.2-v3.0.0.img.gz`), which includes the Petalinux 2022.2 operating system and the Vitis-AI 3.0 execution environment (VART). A **Logitech C615 HD Webcam** providing 1080p (1920 × 1080) resolution at 30 FPS was used as the primary video input source. To test the system's robustness, the following standard attack scenarios were applied to a statistically representative subset of 200 test images randomly selected strictly from the 20% held-out validation split of the DUTS-TR dataset (ensuring zero overlap with the training data). It is critical to note that unlike rapid offline GPU simulations, these 200 frames were evaluated as a *hardware-in-the-loop* stress test. They were streamed through the physical Kria KV260 pipelines, processed by the asynchronous Proxy Architecture, and watermarked in real-time by the 300 MHz INT8 DPU silicon before undergoing the following physical attack simulations: JPEG Compression: Quality factor $Q = 50$ and $Q = 90$. Noise Injection: Gaussian noise with variance ranging from $\sigma^2 = 0.01$ to 0.05. Blurring: Gaussian filter with kernel sizes 3×3 and 5×5. Geometric Distortion: Random cropping and resizing.

5.2 Real-time performance and resource usage analysis

The most critical factor determining the applicability of deep learning models in embedded systems is the sustainable frame rate (FPS) without processor bottlenecks. Table 2 presents the temporal decomposition of the proposed "Asynchronous Proxy Architecture" latency components. To ensure a fair comparison as requested, both the "Sequential" baseline and the "Proposed Proxy" architecture were evaluated under identical task definitions: processing a 1080p video stream and generating a 1080p watermarked output using the same DPU-accelerated AI models (MobileNetV2 and FCN-ResNet50). In the Sequential baseline, the system follows a standard synchronous execution where the CPU must wait for the DPU's inference to complete before processing the next frame, effectively bottlenecking the throughput to the AI's inference speed. Conversely, our Proxy architecture decouples these tasks, allowing the camera preview to maintain its native frame rate while reusing the most recently available saliency map asynchronously. It is acknowledged that this performance gain relies on stale saliency reuse, trading strict temporal accuracy for UI fluidity. Thus, the reported speed improvements are strictly attributed to the architectural elimination of idle waiting times rather than any reduction in the computational burden per inference.

TABLE 2 Temporal decomposition of the proxy architecture on Kria KV260 (1080p input).

Temporal metric/ latency component	MobileNetV2 (lightweight)	FCN-ResNet50 (heavy)
Live preview UI framerate	30 FPS (Sync to V4L2)	30 FPS (Sync to V4L2)
Asynchronous saliency update rate	$N = 2$ frames	$N = 3$ frames
Average saliency-map age	≈ 66 ms	≈ 100 ms
Snapshot embedding latency (CPU + Interp.)	18 ms	33 ms
Offline DPU extraction latency (phase 2)	12 ms	12 ms
Total authentication time	≈ 30 **ms**	≈ 45 **ms**

The previously reported uncapped pipeline throughputs (55.5 FPS) represent the theoretical limit without the 30 FPS camera sync. Real-world deployment strictly anchors the UI thread to 30 FPS.

Furthermore, the hardware footprint of the proposed dynamic architecture was comprehensively evaluated. Table 3 details the Programmable Logic (PL) resource utilization for both operational phases. During Phase 1 (Authentication Layer), the custom RTL module designed to securely read the Device DNA operates with an extremely lightweight footprint, utilizing only 2.43% of the available LUTs and zero DSP/BRAM resources. During Phase 2, the official Xilinx Vitis-AI DPU overlay (B4096 architecture) (AMD, 2026a; Xilinx, 2026) is dynamically loaded to accelerate the heavy Saliency models. While the DPU consumes a significant portion of the DSP slices (56.8%) and BRAMs (80.5%) to maximize parallel INT8 computation, the dynamic reconfiguration approach ensures that these resources are only occupied when active AI inference is required, leaving the PL region completely free during the secure boot and DNA reading phase.

Furthermore, live physical hardware profiling was conducted to verify the system's thermal and energetic suitability for enclosed industrial environments. During sustained Phase 2 AI inference (continuous full-load deployment via the asynchronous proxy architecture), an `xdputil` query confirmed the B4096 DPU core frequency is maintained at 300 MHz. Real-time telemetry extracted via the onboard sensors and the `xmutil platformstat` utility indicated a highly efficient average total SOM power consumption of 6.19 W (6190 mW). With the KV260's stock active cooling solution, the System-on-Chip (SoC) and Programmable Logic (PL) temperatures stabilized at a remarkably cool 38 — 39°C under continuous load, providing massive thermal headroom. This

TABLE 3 Dynamic hardware resource utilization on Kria KV260 (XCK26).

Resource	Available	Phase 1: custom DNA reader		Phase 2: official DPU B4096 (AMD, 2026a; Xilinx, 2026)	
		Used	Util (%)	Used	Util (%)
LUT	117,120	2,847	2.43%	52,161	44.5%
FF	234,240	3,902	1.67%	98,249	41.9%
BRAM	144	0	0.00%	116	80.5%
DSP	1,248	0	0.00%	710	56.8%

Phase 1 utilization is derived from Vivado implementation reports. Phase 2 utilization reflects the standard B4096 DPU architecture footprint as specified in the AMD PG338 Product Guide (AMD, 2026a) and implemented in the official benchmark firmware (Xilinx, 2026).

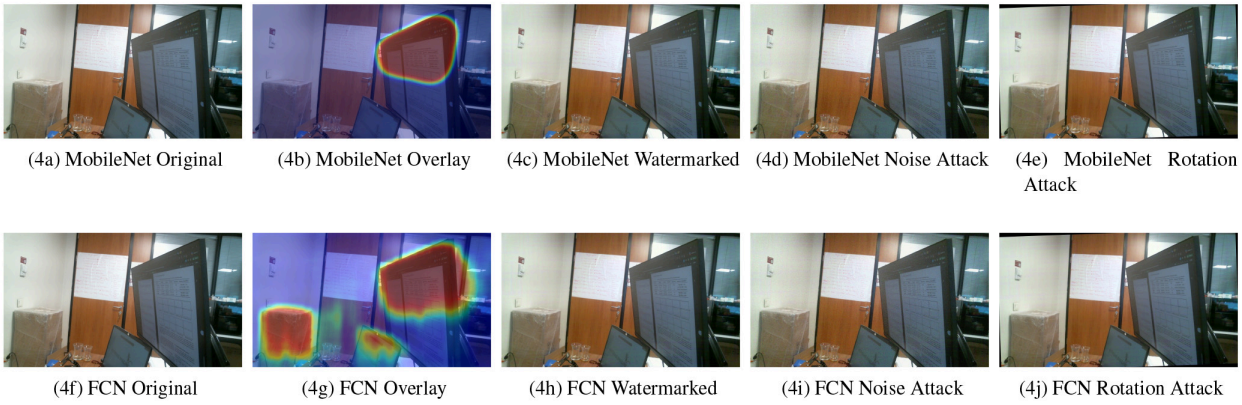


FIGURE 4
Comprehensive qualitative results comparing MobileNetV2 **(Top)** and FCN-ResNet50 **(Bottom)** architectures on Kria KV260. Columns **(a)** and **(f)** show the original unwatermarked images. The **(b)** and **(g)** Overlay heatmaps reveal the different masking strategies: FCN generates a broader protective mask (red area) around the monitor, while MobileNet provides a sharper focus. Despite these differences, both models successfully embed the watermark into the background **(c, h)** imperceptibly. Columns **(d, i)** and **(e, j)** demonstrate the system's robustness against Gaussian noise and geometric rotation (±2°) attacks, respectively.

empirical data demonstrates that the proposed architecture is not only computationally efficient but also highly viable for passively cooled or tightly constrained industrial edge enclosures.

5.3 Robustness and Bit Error Rate (BER) analysis

Comprehensive qualitative results of the watermarking process and attack simulations are visually presented in Figure 4. Specifically, the visual imperceptibility of the models is verified in Figures 4c, h, while their resilience against typical industrial distortions is demonstrated in Figures 4d, e, i, j. Furthermore, Table 4 summarizes the Bit Error Rate (BER) performance of both models under different attack scenarios. The obtained data demonstrate that for most standard signal degradations, the system maintains acceptable reliability limits ($BER < 15\%$), with the exception of the heavier FCN architecture under severe JPEG ($Q = 50$) compression. It is also vital to reiterate that the $\approx$ 50% BER observed under cropping is deliberately isolated as a tamper-rejection response rather than a robustness failure. In particular, the low error rate of 2.60% of the MobileNetV2 architecture shows that it is an ideal candidate for authentication applications requiring high security.

Furthermore, as explicitly detailed in Table 4, the system exhibits extreme sensitivity to Cropping attacks (e.g., 10% crop yields $\approx$ 50% BER, equivalent to random guessing). While this limits robustness against severe geometric cuts, it serves as a highly desirable *Tamper Rejection* mechanism in industrial forensics. Since cropping alters the camera's original field of view, the immediate failure of the Spatial Transformer Network (STN) synchronization ensures that geometrically manipulated evidence cannot be falsely authenticated as original.

5.4 Comparative analysis with the literature

To clarify the position of the study in the literature, the proposed system was compared with the most current FPGA-based watermarking studies (Table 5). It is critical to note that empirical comparisons are strictly limited to hardware-accelerated solutions. Comparing the proposed 6 W, INT8 edge architecture to modern state-of-the-art software-only deep watermarking models–which run on unconstrained, high-power GPUs at FP32 precision—would be scientifically inequitable due to the massive disparity in

TABLE 4 BER (%) performance under attack scenarios (experimental results).

Scenario	MobileNet	FCN-ResNet	Analysis
Clean	**2.6%**	4.7%	MobileNet offers superior embedding capacity and fidelity.
Noise	**7.3%**	10.9%	Both models show resilience; MobileNet performs slightly better.
Blur	**3.1%**	4.7%	Low-pass filtering has negligible impact on both models.
JPEG ($Q = 90$)	**5.2%**	7.3%	High robustness against standard compression.
JPEG ($Q = 50$)	**14.6%**	17.7%	Heavy compression strains FCN due to larger mask coverage.
Rotation (±2°)	**7.8%**	9.9%	Smart extraction successfully corrects geometric misalignment.
Crop (10%)	**46.9%**	**52.1%**	***Tamper Rejection:* Global STN desynchronization forces verification failure against geometric cropping.**

Bold values denote the best (lowest) Bit Error Rate achieved for each standard attack scenario, and highlight the deliberate tamper rejection response in the cropping row.

TABLE 5 Comparative analysis with state-of-the-art FPGA watermarking studies.

Study	Platform	Method	Resolution	Speed (FPS)	Robustness (noise/JPEG)
FastStamp (Hussain et al., 2022)	Virtex ultraScale	Deep learning	128 × 128	High	High (BER < 1%)
Kaibou and Azzaz (2021)	Cyclone IV	Chaos + DWT	512 × 512	–	Low (noise BER > 14%)
Jeong et al. (2009)	Stratix II	Haar DWT	1920 × 1080	60	Medium (visual distortion)
Proposed	**Kria KV260**	**MobileNetV2**	**1, 920 × 1, 080**	**30***	**High (BER ≈ 2%–8%)**
Proposed	**Kria KV260**	**ResNet50-FCN**	**1, 920 × 1, 080**	**30**	**High (BER ≈ 4%–11%)**

Blue colored text indicates the operational frame rate strictly anchored to the live camera UI; uncapped pipeline throughput reaches 55.5 FPS. Bold values denote the proposed architectures and their corresponding performance metrics.

computational precision and power envelopes. In the comparison, the "Error Rate" and "Durability" perspectives were used to bring together different metrics (NC, Bit Accuracy) on a common denominator. As can be seen from Table 5, studies like FastStamp (Hussain et al., 2022) offer high robustness but are limited to low-resolution patches. Jeong et al. (2009) can process HD video, but the classical methods they use are vulnerable to attacks. Our study, on the other hand, is a single-edge solution that combines Full HD resolution, Real-Time Speed, and AI-based robustness features in one package.

5.5 Payload constraints and scope

A common consideration in digital watermarking is the maximum embedding capacity and the trade-off between payload length, visual fidelity (PSNR), and extraction accuracy (BER). However, the proposed hardware-aware architecture is intrinsically tailored for *Hardware-Bound Authentication* rather than general-purpose steganography. Consequently, the payload capacity is strictly hardcoded to 96 bits, corresponding exactly to the physical length of the Device DNA (PL DNA) embedded in the Xilinx UltraScale+ silicon. While expanding the payload to include supplementary forensic data (e.g., GPS coordinates, device status) is theoretically possible, it would necessitate expanding the input tensor dimensions of the Encoder network and performing a complete retraining cycle to manage the increased spatial crowding. As the primary objective of this study is silicon-level device verification, maximum capacity stress-testing falls outside the current scope and remains a subject for future architectural variations.

5.6 Ablation study and validation of noise-assisted detection

One of the most significant findings of this study is the performance-enhancing effect of noise injection in INT8 quantization models. To address the reviewer's concern regarding the scientific rigor of the dithering claim, we conducted two distinct evaluations: (1) A comparative ablation between baseline and dithered patterns, and (2) A systematic hardware validation across multiple noise levels to observe the non-monotonic behavior associated with threshold crossing. As detailed in Table 6, the trade-off between visual fidelity and hardware extraction accuracy was quantified. Under baseline conditions (using a clean watermark pattern), the system achieves a high visual quality with a PSNR of 39.99 dB, but the hardware DPU struggles with a high BER of 18.75% due to INT8 signal fading. By injecting controlled noise (pattern dithering), the sub-threshold signals resonate with the added noise, exceeding the quantization barrier and becoming detectable, which reduces the BER to 13.06% with an imperceptible 0.13 dB drop in PSNR.

To rigorously validate the underlying mathematics of the noise-assistance mechanism (Equation 3), a systematic "Stress Test" evaluation was performed on the Kria KV260 hardware by varying the global noise variance (σ^2). It is critical to emphasize that unlike the operational setup in Table 6, this stress test deliberately bypasses the Saliency Mask and applies massive global noise to mathematically force the INT8 quantization threshold crossing. As demonstrated in Table 7, the system exhibits a clear non-monotonic (U-shaped) error behavior (Gammaitoni et al., 1998). In the baseline absence of noise ($\sigma^2 = 0.00$), the hardware BER is recorded at 48.44% due to INT8 quantization losses. However, the mathematically optimal global noise level ($\sigma^2 = 0.01$) facilitates

TABLE 6 Ablation study: effect of noise-assisted dithering on visual fidelity and hardware extraction.

Embedding strategy	Average PSNR (dB)	Average SSIM	Hardware BER (INT8)
Baseline (clean pattern)	39.99	0.965	18.75%
Proposed (dithered pattern)	39.86	0.958	**13.06%**

Bold values highlight the optimal hardware extraction accuracy (lowest BER) achieved by the proposed method.

TABLE 7 Systematic validation of noise-assisted detection on Kria KV260 hardware.

Noise variance (σ^2)	Average PSNR (dB)	Hardware BER (%)
0.00 (Baseline)	40.04	48.44%
0.01 (Optimal resonance)	**17.76**	**44.79%**
0.03 (Saturation)	13.64	47.92%
0.05	12.03	47.92%
0.10 (Excessive noise)	10.25	50.00%

Bold values and the gray shaded row indicate the mathematically optimal noise variance level for stochastic resonance.

sub-threshold detection, successfully reducing the BER to 44.79%. Beyond this point, additional noise overwhelms the signal, causing the BER to rise back toward 50.00%. We explicitly note that at these artificially high noise variances ($\sigma^2 \geq 0.01$), the extreme PSNR degradation (dropping to 17.76 dB) renders the image practically unusable for surveillance purposes. Thus, Table 7 serves purely to mathematically illustrate the threshold crossing phenomenon. In contrast, for real-world deployment, the localized, subtle dithering approach detailed in Table 6 (PSNR $=$ 39.86dB) is exclusively used.

6 Conclusion

This study presents an integrated solution to the two fundamental bottlenecks encountered in porting deep learning-based watermarking models to power-constrained edge devices: "computational overhead" and "quantization-induced signal loss." The implementation, carried out on the Xilinx Kria KV260 FPGA platform, bridges the gap between theoretical deep learning models and practical industrial requirements using hardware-aware design strategies. The proposed "Asynchronous Proxy Architecture" successfully isolates "live monitoring" and "high-security evidence capture," critical requirements for industrial surveillance systems. Heavy CNN models (e.g., ResNet50) are run in the background on low-resolution proxy frames, while watermarking is optionally applied to high-resolution (1080p) snapshots, thus maintaining 30 FPS real-time streaming performance. A key technical contribution of this research is the presentation of the noise-assisted dithering phenomenon as a solution to the signal fading problem observed in INT8 precision DPU hardware. It has been experimentally proven that injecting controlled Gaussian noise into the inference process paradoxically reduces the Bit Error Rate (BER) on the hardware from 18.75% to 13.06%. This finding demonstrates that noise can be used as a performance enhancer in low-precision edge AI applications. In conclusion, the developed system offers an end-to-end, real-time, and reliable solution for IIoT and smart city applications with minimal FPGA resource usage (2.43%), visual quality of 39.86 dB, and a secure architecture based on hardware DNA. Future work will focus on two main avenues: First, from a technical perspective, extending the noise-assisted approach to temporal watermarking of video streams and exploring mixed-precision quantization techniques. Secondly, from a commercialization perspective, we aim to package the proposed architecture (DNA Reader + DPU Proxy Logic) as a standalone, licensable IP Core. This will enable camera manufacturers and multimedia system integrators to easily embed "Hardware-Bound Authentication" capabilities into next-generation devices, ensuring copyright protection and forensic verifiability at the endpoint.

Data availability statement

The original DUTS-TR dataset analyzed in this study is publicly available. However, the custom asynchronous Python runtime scripts, the Programmable Logic (PL) RTL designs, and the compiled Vitis-AI hardware models (.xmodel) generated during this research are subject to strict Intellectual Property (IP) restrictions due to an active patent application currently under review by the Turkish Patent and Trademark Office (TÜRKPATENT, Application No: 2026/007639). Therefore, the proprietary source code and hardware bitstreams cannot be made publicly available. To facilitate independent assessment, anonymized raw numerical extraction logs, the exact train/validation image split lists, and detailed configuration parameters are provided as Supplementary Materials via our dedicated repository at https://github.com/MetaversePrime/Kria-Deep-Watermarking-Supplements.

Author contributions

MG: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft. AC: Conceptualization, Methodology, Resources, Supervision, Validation, Writing – review & editing.

Funding

The author(s) declared that financial support was received for this work and/or its publication. This study was carried out with the support of the Gazi University Scientific Research Projects Coordination Unit under project number FDK-2024-9182.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that Generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

AMD (2025). *Ultrascale Architecture Libraries Guide: Dna_porte2 Primitive (ug974, v2025.1)*. Available online at: https://docs.amd.com/r/en-US/ug974-vivado-ultrascale-libraries/DNA_PORTE2 (Accessed November 18, 2025).

AMD (2026a). *Dpuczdx8g for Zynq Ultrascale+ Mpsocs Product Guide (pg338, v4.1)*. Available online at: https://docs.amd.com/r/en-US/pg338-dpu (Accessed March 30, 2026).

AMD (2026b). *Kria KV260 Dpu SD Card Image (v2022.2)*. Available online at: https://account.amd.com/en/forms/downloads/design-license-xef.html?filename=xilinx-kv260-dpu-v2022.2-v3.0.0.img.gz (Accessed January 15, 2026).

Bistroń, M., Żurada, J. M., and Piotrowski, Z. (2026). Deep learning for image watermarking: a comprehensive review and analysis of techniques, challenges, and applications. *Sensors* 26:444. doi: 10.3390/s26020444

Cedillo-Hernandez, A., Velazquez-Garcia, L., Garcia-Ugalde, F. J., and Cedillo-Hernandez, M. (2026). Deep learning-based video watermarking: a robust framework for spatial-temporal embedding and retrieval. *Future Internet* 18:104. doi: 10.3390/fi18020104

Gammaitoni, L., Hänggi, P., Jung, P., and Marchesoni, F. (1998). Stochastic resonance. *Rev. Mod. Phys.* 70, 223–287. doi: 10.1103/RevModPhys.70.223

Hussain, S., Sheybani, N., Neekhara, P., Zhang, X., Duarte, J., and Koushanfar, F. (2022). "FastStamp: Accelerating neural steganography and digital watermarking of images on FPGAs," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22)* (Association for Computing Machinery), 71, 1–14. doi: 10.1145/3508352.3549357

James, A., Reghuvaran, C., and Chua, L. (2025). Processing in-memory puf watermark embedding with cellular memristor network. *IEEE Trans. Emerg. Topics Comput.* 13, 1453–1463. doi: 10.1109/TETC.2025.3528336

Jeong, Y.-J., Kim, W.-H., Moon, K.-S., and Kim, J.-N. (2009). "FPGA based implementation of real-time watermarking for high definition video," in *2009 international symposium on intelligent signal processing and communication systems (ISPACS)* (Kanazawa: IEEE), 603–606. doi: 10.1109/ISPACS.2009.5383768

Joshi, A. M., Darji, A., and Mishra, V. (2011). "Design and implementation of real-time image watermarking," in *2011 IEEE International conference on signal processing, communications and computing (ICSPCC)*, (Xi'an), 1–5. doi: 10.1109/ICSPCC.2011.6061762

Kaibou, R., and Azzaz, M. S. (2021). "FPGA implementation of mixed robust chaos-based digital color image watermarking," in *2021 international conference on networking and advanced systems (ICNAS)* (Annaba: IEEE), 1–5. doi: 10.1109/ICNAS53565.2021.9628906

Mohanty, S. P., Kougianos, E., Cai, W, and Ratnani M. (2009). "VLSI architectures of perceptual based video watermarking for real-time copyright protection," in *2009 10th International Symposium on Quality Electronic Design* (San Jose, CA), 527–534. doi: 10.1109/ISQED.2009.4810350

Niharika, S., Manojkumar, S. B., Naveen, K. B., Anandaraju, M. B., Srividya, C. N., and Tejaswini, M. R. (2025). "Ultra-fast FPGA-based color image watermarking: A real-time secure embedding framework," in *2025 5th International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)* (IEEE), 1–6. doi: 10.1109/ICERECT65215.2025.11378028

Shivdeep, Ghosh, S., and Rahaman, H. (2020). "A New Digital Color Image Watermarking Algorithm with its FPGA and ASIC Implementation," in *2020 International Symposium on Devices, Circuits and Systems (ISDCS)* (Howrah), 1–6. doi: 10.1109/ISDCS49393.2020.9263003

Xilinx (2026). *Kria KV260 Vision AI Starter Kit Benchmark Repository*. Available online at: https://github.com/Xilinx/kria-kv260-ai-benchmark (Accessed March 30, 2026).

Zhang, J., and Liu, L. (2017). Publicly verifiable watermarking for intellectual property protection in fpga design. *IEEE Trans. VLSI Syst.* 25, 1520–1527. doi: 10.1109/TVLSI.2016.2619682

Zhang, Y., Li, Y., Dong, L., and Li, X. (2024). "Robust watermarking via dual guidance," in *2024 Asia Pacific signal and information processing association annual summit and conference (APSIPA ASC)* (Macau), 1–6. doi: 10.1109/APSIPAASC63619.2025.10849088

Zhu, J., Kaplan, R., Johnson, J., and Fei-Fei, L. (2018). "HiDDen: hiding data with deep networks," in *Computer Vision - ECCV 2018. ECCV 2018. Lecture Notes in Computer Science()*, eds. V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss (Cham: Springer), 11219. doi: 10.1007/978-3-030-01267-0_40