Journal of Computer and Knowledge Engineering, Vol. 8, No. 2, 2025. (10-24)

Ferdowsi University of Mashhad

Journal of Computer and Knowledge Engineering

https://cke.um.ac.ir

Information and Communication Technology Association of Iran

Efficient Implementation of DVI Protocol on FPGA*

Research Article

Sara Ershadi-Nasab[1], Danial Bayati[2], Saeed Yazdani[3]

10.22067/cke.2025.91345.1142

Abstract *This paper presents a general-purpose hardware implementation of the digital visual interface (DVI) protocol on the Xilinx Virtex-6 ML605 FPGA platform for real-time display of digital processing results. The design enables direct output of processed data from the FPGA to an external monitor without relying on external processors or software-based rendering tools. It addresses key challenges in timing synchronization, pixel formatting, and interfacing with the onboard Chrontel CH7301C encoder to support resolutions up to 1920×1080 at 60 Hz. A lightweight processing pipeline is developed in Verilog to convert multidimensional outputs into a sequential stream of pixel data conforming to the DVI protocol. As a case study, a lightweight convolutional neural network trained on the CIFAR-10 dataset is implemented on the FPGA, and its classification probabilities are displayed as a probability map on an LCD. Experimental results confirm low resource utilization and real-time performance, validating the system's applicability in embedded applications such as machine learning inference, image processing, and real-time monitoring. This work demonstrates the feasibility of FPGA- based platforms for efficiently displaying digital video output in intelligent edge systems.*

Key Words Digital visual interface, field programmable gate array, image processing, real-time, video processing, Xilinx Virtex-6.

1. INTRODUCTION

THE integration of neural networks with real-time image processing on FPGA platforms is vital for applications that require low latency and efficient computation. While neural networks are effective at interpreting visual data, displaying their classification results in real time directly from an FPGA presents practical challenges. In software environments such as Python or MATLAB, classification results can be easily shown using high-level functions like plot(). However, FPGA-based systems lack such abstractions; displaying outputs requires manual control over video signal generation and transmission to an external monitor. This makes even basic output display—such as showing the predicted class on an LCD screen—a nontrivial hardware task.

The DVI is a widely accepted standard for transmitting digital video signals, enabling seamless connections between computers and display devices [1], [2]. It requires careful synchronization of pixel data, clock signals, and timing signals to ensure the correct display of visual outputs. Misalignment of these signals can result in distorted or invisible outputs. Therefore, a precise and accurate implementation of the DVI protocol is crucial for verifying the functionality of FPGA- based neural network systems. As the demand for high- quality digital video transmission increases, bridging the gap between legacy interfaces—such as the video graphics array (VGA) and modern digital standards has become essential. Field-programmable gate arrays (FPGAs) provide a flexible and customizable platform for implementing such interface conversions.

In modern image processing and deep learning applications, it is often necessary to display the output of computations on a monitor [3], [4]. This requirement becomes particularly important when processing is carried out on FPGA boards, as the FPGA functions as a specialized integrated circuit (IC) that performs intensive computational tasks. The results must be presented clearly and efficiently to ensure that the system operates as intended. In recent years, implementing deep learning algorithms on FPGA boards has gained significant attention, as these platforms offer high performance, low power consumption, and customizability compared to traditional processors [5], [6], [7]. Many embedded and

* Manuscript received 2024 December 29, Revised 2025 January 24, Accepted 2025 May 31.

[1] Corresponding author, Assistant Professor, Computer engineering department, Ferdowsi University of Mashhad, Mashhad, Iran. **Email:**ershadinasab@um.ac.ir.

[2] M.Sc. Student. Computer engineering department, Ferdowsi University of Mashhad, Mashhad, Iran.

[3] M.Sc. student. Computer engineering department, Ferdowsi University of Mashhad, Mashhad, Iran.

edge applications now utilize specialized ICs, rather than general-purpose processors, to per- form neural network inference. These specialized ICs optimize power, memory usage, and transistor count, making them well- suited for real-time processing. However, to verify the accuracy and effectiveness of neural networks, the processed outputs must be displayed to the user in a comprehensible manner, necessitating the implementation of a video display protocol.

A key limitation in FPGA-based video systems arises from the interface between multidimensional data structures used in AI models and the strictly sequential data format required by video output hardware. While the Chrontel (CH7301C) [8] DVI encoder integrated on the ML605 board handles the encoding of pixel data into the transition minimized differential signaling (TMDS) format for robust transmission over DVI cables, it requires a continuous, precisely timed stream of pixel values along with horizontal and vertical synchronization signals. In contrast, neural networks and image processing algorithms typically operate on two- or three-dimensional data arrays—such as feature maps or RGB images—rather than linear pixel sequences. This structural mismatch demands custom logic to convert high-level, array-based outputs into tightly timed, flattened pixel streams that comply with the DVI protocol. The lack of built-in hardware for this conversion makes it necessary to implement a fully synchronized pipeline capable of feeding the Chrontel encoder accurately and in real time. Addressing this challenge is essential for achieving smooth and coherent display of neural network results in embedded artificial intelligence (AI) applications. This paper presents an efficient hardware-software co-design approach for real-time display of neural network classification results using the DVI protocol on an FPGA. A lightweight neural network, trained on the CIFAR-10 dataset, is implemented within the FPGA, with its weights manually embedded in Verilog. The network processes input images and produces classification probabilities, from which the seven most probable classes are selected for display. These results are formatted as RGB data, which is then converted into a DVI-compatible signal for output to an external monitor. To achieve this, we generate synchronization signals, overlay the classification results on the display, and transmit the output to the onboard Chrontel DVI encoder of the Xilinx ML605 board. This enables real- time rendering of the classification output directly on an LCD screen, alongside the processed input image. By eliminating the need for external CPUs or GPUs, this approach enhances resource efficiency and provides a scalable solution for embedded AI applications. The proposed system demonstrates how FPGA-based deep learning architectures can be directly integrated with standard video output protocols, making them well-suited for edge computing and real-time classification tasks. This paper discusses the process of implementing the DVI protocol on the ML605 FPGA, including the challenges of signal synchronization, the integration of neural network output, and the successful testing of the system for real-time image processing applications. By utilizing the power of the Xilinx Virtex-6 architecture, we aim to showcase the potential of FPGA platforms in the efficient and accurate display of neural network and image processing outputs.

One of the significant challenges in the integration of neural network outputs and real-time image processing in FPGA systems is the precise synchronization of timing signals for video display, especially in legacy-to-modern interface conversions. Existing solutions often focus on individual aspects, such as processing efficiency or display fidelity, without addressing the combined demand for real-time performance and high- quality visualization. Our work introduces a novel approach by integrating DVI protocol on the ML605 FPGA to achieve seamless and accurate displaying of neural network outputs. The system leverages the capabilities of the Xilinx Virtex-6 architecture, ensuring synchronization of pixel, clock, and timing signals with resolutions up to 1920×1080 at 60 Hz. This dual capability of high-resolution rendering and real-time processing represents a significant step forward in the design of embedded systems, particularly for edge AI applications like medical imaging and smart monitoring. In this work, the ML605 FPGA development board, built on the Xilinx Virtex-6 architecture, is employed to prototype and implement complex digital systems. The design is developed using the Xilinx ISE 14.7 toolchain, which fully supports hardware synthesis and configuration for this platform. With its high-capacity FPGA, ample memory, and versatile I/O interfaces, the ML605 is well-suited for system-on-chip (SoC) development and the implementation of advanced video interfaces.

The primary contribution of this work is the real-time implementation of the DVI protocol on the Xilinx Virtex-6 ML605 FPGA platform. This implementation enables direct displaying of neural network outputs and image processing results by addressing key challenges in signal synchronization and high-resolution rendering. Key innovative aspects of the implementation include:

1) **Real-time DVI protocol implementation on FPGA:** This work presents the complete implementation of the DVI protocol on the Xilinx ML605 FPGA board, enabling direct display of neural network outputs on external monitors without requiring a host CPU or GPU.
2) **Direct integration of neural inference with display pipeline:** A novel hardware-software co-design is developed that links a lightweight convolutional neural network (CNN), manually implemented in Verilog, with a real-time pixel stream generator. The system converts multidimensional classification outputs into sequential RGB pixel values for video rendering.
3) **Custom hardware pipeline for TMDS-compatible output:** The system addresses the structural mismatch between array-based AI outputs and the linearly timed pixel streams required by the DVI encoder. A synchronized pipeline is implemented to format and transmit the CNN output via the on-board Chrontel CH7301C encoder using the TMDS standard.
4) **Legacy-to-modern interface conversion:** The design enables legacy VGA-style embedded systems to connect with modern digital displays via the DVI protocol, effectively bridging analog and digital video standards using programmable logic.

5) **Efficient FPGA resource utilization:** The architecture demonstrates minimal consumption of FPGA resources (less than 1% of slice registers and LUTs), allowing significant headroom for additional logic, such as more complex networks or preprocessing units.
6) **Hardware-based rendering of neural outputs:** Classification probabilities are displayed as visual indicators—such as variable-width bars—directly on the LCD without software-based rendering. This hardware-centric approach facilitates real-time feedback in embedded systems.
7) **Scalability for edge AI applications:** The system is well-suited for embedded AI scenarios, such as medical diagnostics, smart surveillance, and industrial monitoring, where low latency, power efficiency, and real-time result visualization are critical.

This novel implementation not only confirms the feasibility of video protocol integration on FPGA platforms but also provides a scalable framework for real-time display in advanced embedded systems.

The structure of this paper is organized as follows: Section II introduces relevant studies addressing the topic. Section III presents a detailed comparison of commonly used display interfaces, including VGA, DVI, and high-definition multimedia interface (HDMI), highlighting their features, limitations, and use cases. The proposed method architecture is outlined in Section IV. The experimental results are provided in Section V. Finally, the conclusion is presented in Section VI.

2. RELATED WORK

Integrating neural networks and real-time image processing outputs on FPGA platforms has garnered significant attention in recent years [9], [5]. Various studies have explored FPGA- based solutions for implementing video display protocols, focusing on DVI and related technologies [10], [9]. These efforts highlight the versatility of FPGAs in bridging legacy and modern systems while ensuring high-performance real- time processing. Recent advancements in AI have driven significant progress in intelligent flexible sensing systems capable of highly efficient data acquisition, analysis, and perception. These innovations enable more sophisticated communication between neural processing units and external sensors, improving real-time monitoring and display capabilities for applications such as flexible sensory systems, humanoid robotics, and human activity monitoring [4]. Similarly, the development of systems-on-chip such as TinyVers, which incorporates state- retentive design for machine learning (ML) inference at the extreme edge, demonstrates the significance of energy-efficient and versatile hardware platforms in supporting real-time AI applications [5]. Recent advancements have showcased the implementation of FPGA-based real-time image processing systems, emphasizing the integration of DVI-compatible video interfaces for effective visualization and synchronization [9]. Optimization techniques for deploying CNNs on FPGA plat- forms have further enhanced the efficiency of hardware- software co-design, enabling seamless display of neural net- work output [10]. High-speed video processing and display integration have also been demonstrated, particularly through FPGA-accelerated object detection systems utilizing edge in-formation, achieving real-time potential in applications [11].

Moreover, digital oscillatory neural network frameworks have been implemented on FPGAs for edge AI applications, highlighting the relevance of video signal generation capabilities in DVI-based visualization systems [12]. The development of real-time systems for processing neuronal network activity on FPGA platforms has further established the critical role of DVI in rendering real-time outputs for high-speed visual feedback [13]. Additionally, FPGA implementations of hyperchaotic neural network systems have illustrated the adaptability of these platforms for complex computations and their corresponding outputs [14].

Efforts have also focused on privacy-preserving authentication protocols for IoT devices, leveraging FPGA capabilities with DVI for secure and efficient visualization [3]. Finally, real-time video enhancement algorithms implemented on FP- GAs have underscored their ability to handle computationally intensive tasks while adhering to DVI standards for high- quality visualization [15].

Another important area of exploration has been the evolution of GPU hardware, which offers significant parallel processing capabilities for neural networks and AI workloads. Peddie [6] provides an in-depth review of the GPU environment and its impact on hardware, highlighting advancements in graphics processing technology and its integration into AI and ML workflows.

These insights are particularly relevant as GPUs and FPGAs continue to coexist as complementary technologies in real-time AI processing. These studies collectively demonstrate the flexibility and efficiency of FPGA platforms in integrating neural networks, real-time image processing, and video display protocols such as DVI. They provide a solid foundation for developing FPGA-based systems that deliver high-speed, accurate visual outputs—crucial for applications in AI, edge computing, and embedded systems.

Wang and Luo [16] emphasize the benefits of FPGA accelerators in optimizing custom hardware architectures for real-time applications. Their review highlights the importance of precision reduction techniques in minimizing latency and enhancing performance—approaches that directly support our objective of achieving high-speed and accurate displaying of neural network outputs.

Recent investigations into FPGA-based visualization systems have also explored optimization strategies aimed at reducing latency and power consumption. In particular, hybrid systems that combine FPGAs with processors or GPUs have received attention for their ability to offload specific tasks, such as preprocessing or feature extraction, to dedicated hardware blocks. This co-design approach is instrumental in meeting the strict timing constraints required for real-time video out- puts [17].

Overall, these studies underscore the potential of FPGA platforms for efficient video protocol integration and real-

time visualization, thereby paving the way for advancements in embedded display systems across applications like medical imaging, edge AI, and smart monitoring. Furthermore, the adaptation of FPGAs for AI-driven video analytics has shown remarkable results. Thyagarajan et al. [18] and Park et al. [15] demonstrated the integration of neural network models with smart cameras, achieving real-time performance in applications like sports analytics and video enhancement. These findings align with the growing need for low-latency, high-throughput FPGA solutions for video-based AI applications. Przesmycki and Nowosielski [19] explored the security implications of compromising emanations in VGA and DVI interfaces, providing insights into the design of secure and reliable FPGA-based visualization systems.

Moreover, Bailey [7] elaborated on the fundamentals of embedded image processing systems on FPGAs, detailing the integration of advanced display protocols like DVI and HDMI for multimedia applications. Hoang et al. [20] presented a pulse-coupled neural network (PCNN) framework implemented on FPGAs for real-time object recognition, showcasing DVI compatibility for visual outputs. Similarly, Fang et al. [21] proposed systematic optimization of spiking neural networks (SNNs) on FPGAs, emphasizing their ability to handle cognitive tasks in real-time scenarios. Farabet et al. [22] developed FPGA-based stream processors for convolutional neural networks, enabling real-time vision tasks with standard DVI connections for video display. These systems demonstrate the capacity of FPGA-based platforms to manage complex visual processing pipelines while ensuring low latency. Additionally, Yildiz et al. [23] and Kayaer et al. [24] explored FPGA implementations of cellular neural networks for preprocessing blocks in high-definition video applications. Their systems utilize DVI interfaces to process and visualize outputs in real time. Abernot [25] investigated the use of oscillatory neural networks on FPGA platforms, highlighting their utility in edge AI systems requiring real-time video processing. This study underscores the adaptability of FPGA designs in integrating learning models with real-time video outputs. Yildiz et al. [26] presented the implementation of preprocessing blocks for cellular neural network-based systems on FPGAs, utilizing DVI for real-time output visualization. This research high- lighted the efficiency of FPGA designs for low-latency video applications. Antonik [27] explored FPGA implementations for hardware reservoir computing and real-time machine learning, emphasizing applications in edge AI. Similarly, Ahilan and James [28] focused on the design and implementation of a real-time car theft detection system, which leveraged FPGA processing and DVI visualization to achieve high-speed image analysis. Davutoğlu et al. [29] designed a real-time frame buffer implementation using external memory on FPGAs. Their study demonstrated how FPGAs can efficiently manage frame data while supporting DVI interfaces for video display. Fasih et al. [30] examined FPGA-based systems for video enhancement in advanced driver assistance systems (ADAS), incorporating convolutional neural networks and DVI outputs to improve video clarity.

TABLE 1 Detailed comparison of Vga, Dvi, and hdmi display interfaces

Feature	**VGA**	**DVI**	**HDMI**
Year of Introduction	1987	1999	2003
Signal Type	Analog	Analog & Digital (DVI-A, DVI-D, DVI-I)	Digital
Maximum Resolution	Up to 1080p	1920x1200 (Single-Link), 2560x1600 (Dual-Link)	8K at 60Hz or 4K at 120Hz (HDMI 2.1)
Color Depth	Limited by analog quality	24-bit (Single-Link) or higher for Dual-Link	Up to 48-bit (HDR supported)
Audio Support	No	No	Yes, with multichannel audio support
Cable Length	Up to 15m with quality loss	Up to 5m for digital, longer for analog	Up to 15m for 4K, shorter for 8K
Compatibility	Legacy monitors and projectors	Transitional systems	Modern displays, TVs, and projectors
Connector Type	15-pin D-Sub	Multi-pin (varied)	Compact (Type-A, Mini, Micro)
Video Signal Quality	Prone to interference	Better than VGA; pure digital avoids noise	Excellent; supports HDR and high refresh rates
Multi-Monitor Support	Not supported	Not supported	Supported via splitters
Data Bandwidth	Not standardized	4.95 Gbps (Single-Link), 9.9 Gbps (Dual-Link)	Up to 48 Gbps (HDMI 2.1)
Use Cases	Legacy monitors and projectors	PC monitors and transitional setups	TVs, gaming systems, multimedia devices
Adapter Availability	VGA to HDMI/DVI with converters	DVI to VGA/HDMI with converters	HDMI to VGA/DVI with converters
Cost	Low	Moderate	Higher (for high-speed cables)

3. COMPARISON OF VGA, DVI, AND HDMI IN FPGA-BASED SYSTEMS

Video interfaces play a crucial role in FPGA-based image processing and displaying the neural network output. Among the widely used standards, VGA, DVI, and HDMI offer different trade-offs in terms of signal quality, bandwidth, and implementation complexity. Table 1. summarizes their key differences, focusing on their impact on FPGA implementation. DVI strikes a balance between complexity and quality, making it a suitable choice for FPGA-based real-time visualization of neural network outputs. Unlike VGA, it provides lossless digital transmission, and compared to HDMI, it avoids the additional complexity of audio and high-bandwidth digital content protection (HDCP) encryption, which are unnecessary for many FPGA applications.

4. PROPOSED METHOD

Fig. 1 illustrates the high-level design of the proposed hardware-based image classification pipeline. This system con- sists of six sequential stages that transform raw image data into classified and visualized output, ultimately displayed in

real time on a digital monitor. The process begins with image processing, where the input image—typically in $W \times H \times 3$ RGB format—is resized, normalized, and optionally filtered to enhance its features and ensure consistency for the classification model. Next, the AI processing stage applies a lightweight classification algorithm, which may be based on neural networks or simpler machine learning methods. This stage extracts relevant features and produces a classification output in the form of a probability map or class index.

The third stage, probability map visualization, converts the AI output into a visual format by mapping probabilities or class indicators into color-coded RGB values. This makes the classification interpretable when displayed on a screen. In stage four, the system generates essential synchronization signals such as horizontal sync (HSYNC), vertical sync (VSYNC), and data enable (DE), along with precise pixel timing to prepare the image stream for display output. These signals ensure that the display device receives video data in a valid scanline order.

Once the pixel data and control signals are properly formatted, the fifth stage interfaces with the Chrontel (CH7301C) DVI encoder chip. This stage handles the conversion of parallel RGB data into TMDS (Transition Minimized Differential Signaling) format, which is the standard for DVI transmission. The sixth and final stage handles the actual DVI output, transmitting the TMDS signals to an external monitor where the classified image is rendered in real time, enabling immediate feedback and visualization.

Fig. 2 provides a detailed RTL schematic that corresponds specifically to the first four stages of the pipeline. These stages are fully implemented in Verilog and deployed on an FPGA platform. The image processing logic is handled by the image_processing module, which receives and formats the incoming image data. This is followed by the ai_processing module, which performs classification using MAC operations and feature extraction based on preloaded filters. The output of this stage is passed to the probability_map_visualizing module, which trans- forms the classification results into pixel-level RGB values based on scan positions.

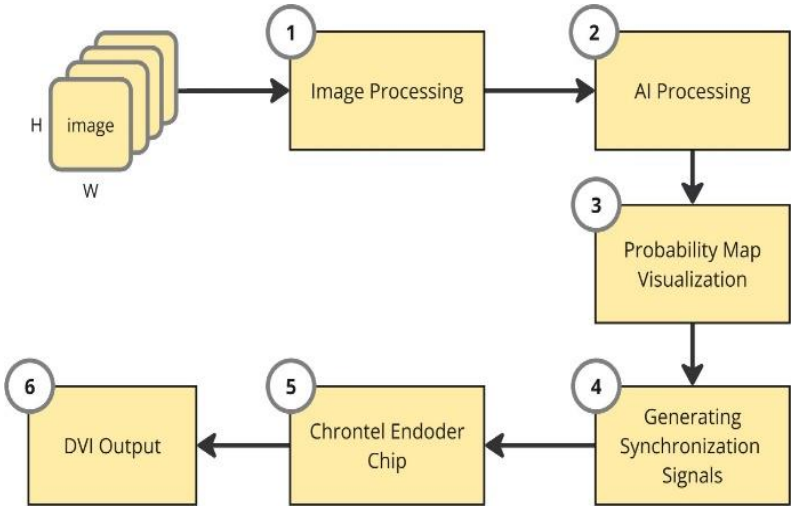

Fig. 1. Overview of the hardware-based image classification pipeline implemented in Verilog. The process is divided into six main stages, from image input to real-time DVI display

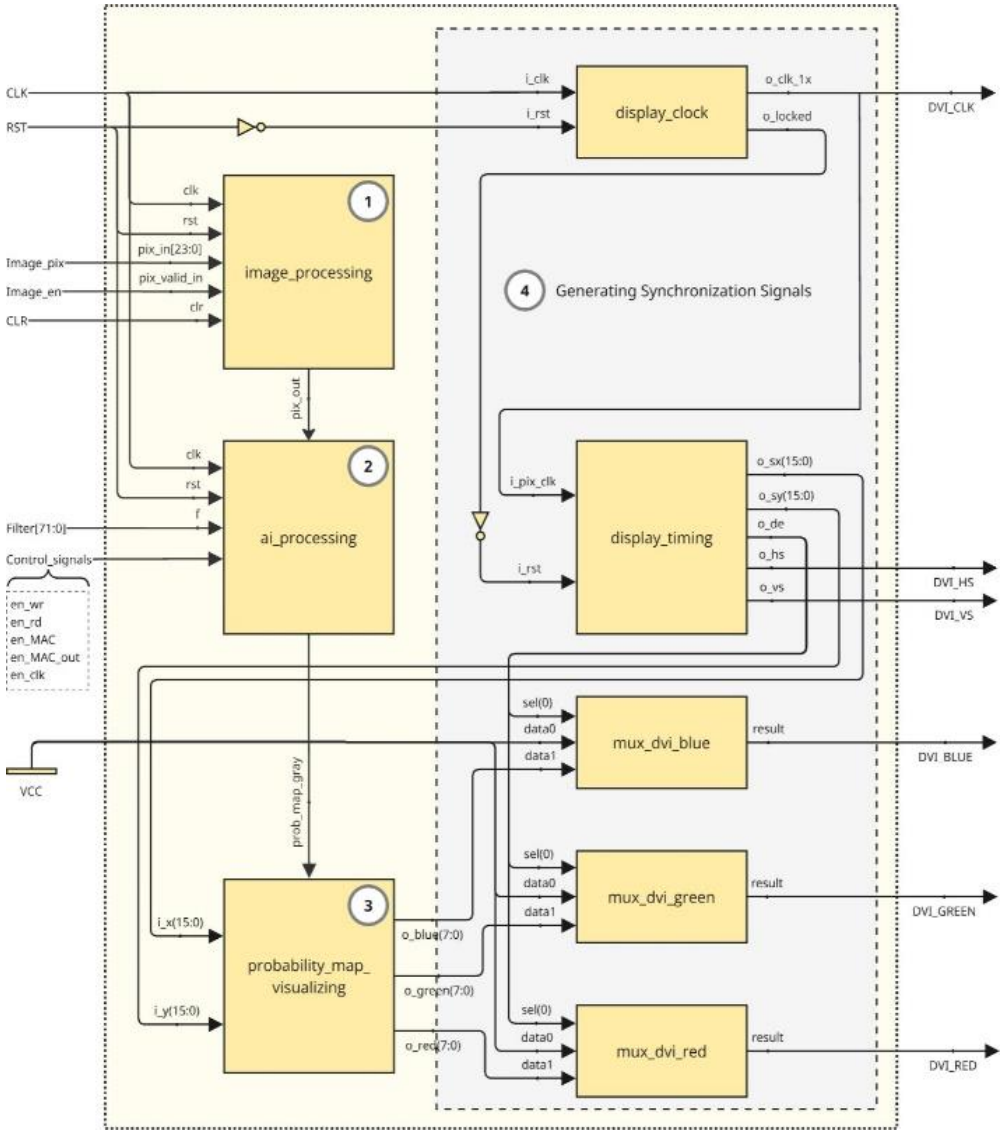

Fig. 2. Register transfer level (RTL) schematic of the Verilog-based image classification system. This diagram focuses on the implementation of Stages 1 through 4 within the FPGA, showing signal flow, control logic, and synchronization modules

To prepare for external video transmission, synchronization signals and timing control are generated by the display_clock and display_timing modules. These provide the pixel clock, scan coordinates, and sync signals required by the downstream encoder. While the RTL diagram concludes at this point, the outputs from Stage 4 are structured specifically for interfacing with the Chrontel encoder in Stage 5, and eventually, real-time rendering on a monitor via Stage 6.

4.1. Image Processing Stage

The Image Processing stage serves as the initial component of our FPGA-based classification pipeline, tasked primarily with preparing raw image data for subsequent analysis by the AI Processing module. This stage plays a fundamental role, ensuring compatibility and quality enhancement of image data, thus directly influencing inference accuracy and computational efficiency.

In the present design, we adopt the widely recognized CIFAR-10 dataset as the primary source of training and evaluation images. CIFAR-10 provides 60,000 images (32×32 pixels, RGB) split into 50,000 training images and 10,000 test images. These images span 10 distinct classes, each containing

6,000 samples. While CIFAR-10 images are natively in color (3 channels).

Initially, input image data arrives in standard RGB format, represented as three separate channels (Red, Green and Blue) with each pixel typically stored at 8-bit color depth. Given the resource constraints and processing requirements of FPGA

hardware, these images undergo several preprocessing steps to enable efficient inference and maintain acceptable accuracy. First, input images are resized to a fixed, uniform resolution compatible with the downstream inference engine

(e.g., 32×32 or 64×64 pixels). This resizing, implemented in Verilog, utilizes hardware-optimized interpolation algorithms

(e.g., bilinear interpolation) to maintain image quality while reducing computational overhead. The choice of a

relatively small, standardized resolution aligns well with the limited memory and computational bandwidth on FPGAs, ensuring predictable timing and efficient parallelization.

Subsequently, normalization of pixel values scales the image data into a suitable numerical range (such as 0–1 in floating- point or Qm.n in fixed-point) to ensure stable arithmetic operations during neural network inference. For this project, an 8-bit RGB input is often transformed into a fixed-point representation (e.g., Q8.8) or scaled floating-point format that fits the FPGA's DSP slices and LUTs. This consistent input magnitude fosters stable training convergence (if on-FPGA training or partial re-training is used) and more accurate inference under resource constraints.

Depending on deployment needs, noise reduction filtering, such as median filtering or Gaussian smoothing, can be added to enhance the signal-to-noise ratio of raw images. In an FPGA context, these filters can be efficiently realized via parallelized convolution modules or simplified averaging techniques. The hardware-level parallelism offered by FPGAs significantly reduces latency for such operations, crucial for real-time applications.

Finally, the processed and normalized image data is buffered in on-chip block RAM or external memory, ready for rapid retrieval during inference. This buffering ensures a smooth pipeline from raw data ingestion to the AI Processing stage, mitigating bandwidth bottlenecks and guaranteeing real-time performance. By streamlining the raw images into a predictable format, the Image Processing stage lays the ground- work for the subsequent hardware-accelerated CNN inference.

4.2. AI Processing Stage

The AI processing stage is the core of our FPGA-based image classification pipeline, where a CNN is implemented directly in Verilog HDL to achieve efficient real-time inference. Leveraging the intrinsic parallel processing capabilities of FPGAs, this design tackles the computationally intensive nature of CNNs while working under the logic, DSP, and memory constraints of devices like the ML605 board.

In our approach, we adopt a six-layer CNN architecture inspired by the work in [31]. The model comprises:

1) Sliding Window Convolution (for feature extraction),
2) ReLU Activation (introducing non-linearity),
3) Max Pooling (down sampling to reduce spatial dimension),
4) Flattening (restructuring 2D features into a 1D vector),
5) Fully Connected (learning global relationships among features),
6) Softmax Activation (producing a probability distribution over the 10 CIFAR-10 classes).

Each layer is coded as a separate Verilog module, allowing straightforward testing and debugging. For instance, the con- volution layer involves efficient hardware-based matrix multi- plication to convolve filters over the input feature maps, while the ReLU module employs a simple conditional operation to clamp negative values to zero. The Max Pooling module further reduces data dimensionality by selecting the maximum value within local neighborhoods of a feature map, improving robustness to minor shifts. Flattening modules then reshape the 2D feature maps into 1D vectors for fully connected processing, and a final Softmax step converts outputs to class probabilities.

Fig. 3 illustrates the internal organization of the inference module. A dedicated memory controller retrieves pretrained weights and biases from off-chip memory (e.g., DDR3 on the FPGA), while the computation engine executes multiply- and-accumulate (MAC) operations in parallel. By instantiating multiple DSP slices for simultaneous MAC operations, the inference pipeline substantially reduces latency compared to software-based implementations.

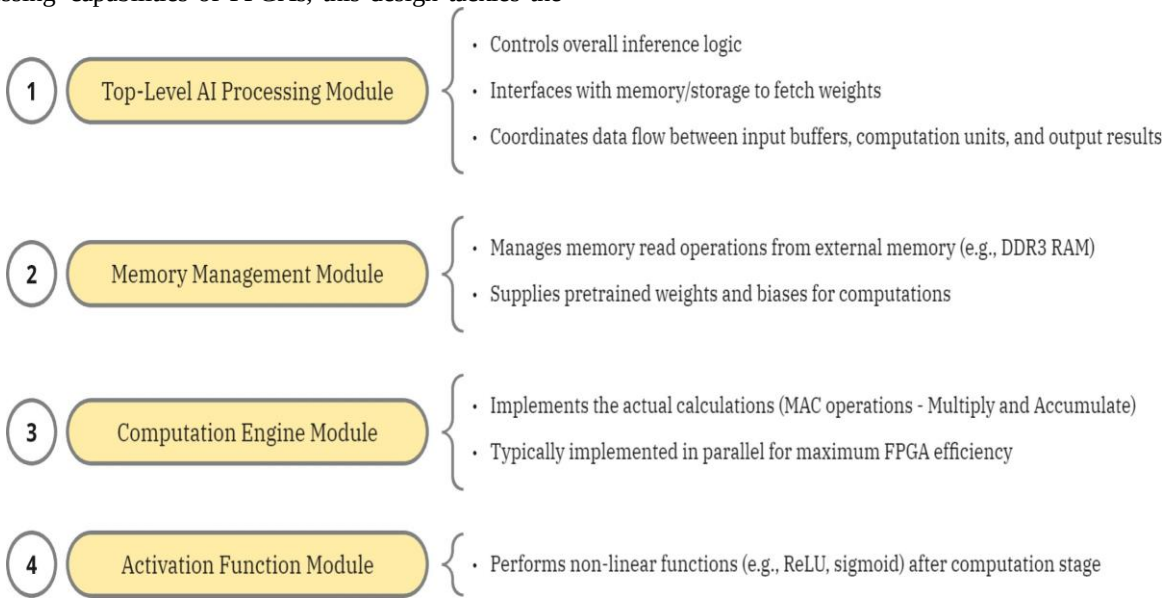

Fig. 3. AI processing stage (inference) module organization.

To achieve consistent performance and accuracy, we pre- trained the CNN offline, using standard frameworks (e.g., Ten- sorFlow or PyTorch) with the CIFAR-10 dataset. During this training phase, high-level floating-point arithmetic was used. Post-training, model parameters were quantized or scaled to fit the fixed-point precision supported by the Verilog modules on the FPGA. This quantization can be as coarse as Q4.12 or Q8.8, depending on resource availability and desired accuracy. An activation function module applies nonlinearities such as ReLU. Compared to sigmoid or tanh, ReLU is both simpler to implement and less prone to saturating at extremes. The AI inference controller, operating in concert with these modules, handles synchronization, data flow, and final classification result generation. Once classification is complete, the output is forwarded to subsequent logic interfaces for digital video output or further processing steps.

By consolidating these hardware modules, we demonstrate a feasible CNN pipeline capable of real-time classification, even on mid-range FPGA platforms. This tightly integrated design exemplifies how FPGAs can address demanding edge inference tasks, combining low-power consumption with competitive throughput for resource-constrained environments. Future enhancements may explore deeper CNN architectures or color- image pipelines once resource usage is further optimized. Nonetheless, the current 6-layer CNN exhibits strong proof- of-concept for FPGA-based deep learning inference on the CIFAR-10 dataset.

4.3. Probability Map Visualization Stage

The probability map visualization stage serves as a critical intermediary within the FPGA-based classification pipeline, connecting AI-generated outputs to the digital video display subsystem. In this stage, the numeric classification results produced by the AI inference engine are systematically converted into clearly distinguishable visual representations suitable for subsequent DVI output. Specifically, classification labels or inference results—initially represented as numeric vectors or encoded class identifiers—are mapped into a predefined color- coding scheme. In our implementation, this involves associating each classified category with a unique color, employing a fixed set of up to ten distinct colors, each corresponding directly to a specific classification result. Such mapping is efficiently realized using lookup tables (LUTs) implemented directly within Verilog code.

The Verilog implementation of the probability map visualization module involves defining a lookup table that associates each of the classification outputs with a preselected RGB color value. This enables immediate, visual differentiation of predicted classes on screen, enhancing interpretability and facilitating rapid decision-making. The hardware module utilizes internal FPGA resources, typically block RAM or distributed LUTs, to perform this quick mapping operation. For instance, an inference result labeled as “Class 1” might correspond to red, while “Class 2” might be displayed as green, and so forth. If the AI inference identifies more than a limited number of classes, a hierarchical encoding strategy can be employed to group classes into broader categories, preserving FPGA resources while maintaining visual clarity.

Additionally, the module handles synchronization tasks necessary for digital video interfaces. It ensures that the converted RGB data stream matches the timing requirements of the DVI protocol, performing frame buffering and pixel synchronization. Precise control of pixel timing, horizontal and vertical synchronization signals, and other required digital video parameters guarantees a stable, high-quality visual output free from artifacts or latency issues.

Moreover, for scenarios involving uncertainty or unidentified classes, an additional category (often represented by a neutral color or grayscale) can be assigned. Implementing

these visual encoding schemes directly through hardware de- scription language allows for seamless, real-time visualization of classification results without latency overhead, which is critical for applications requiring immediate feedback, such as real-time image classification and monitoring systems.

4.4. Generating Synchronization Signals

Stage 4 of the system comprises several essential mod-ules that work in tandem to generate synchronization signals and drive display output. These modules include dis- play_clock, display_timing, and RGB color channel multiplexers (mux_dvi_red, mux_dvi_green, mux_dvi_blue). Together, they handle the timing, pixel positioning, and output format- ting necessary for DVI video transmission, as shown in fig. 2.

The display_clock module is responsible for producing the required video clocks using a mixed-mode clock manager (MMCM). It generates a stable pixel clock (o_clk_1×) for the entire display pipeline and asserts a o_locked signal once clock stabilization is achieved. By manipulating multiplication and division factors, the module ensures that clock frequencies align precisely with resolution-specific requirements, facilitating smooth video playback.

As shown in Table 2, the clock parameters are resolution dependent. The MULT_MASTER parameter sets the base frequency multiplier, while DIV_MASTER, DIV_5×, and DIV_1× divide the result to generate the final pixel clock.

For high-definition formats like 1920×1080, smaller division values (e.g., DIV_1×, × = 5) ensure the required high- frequency clocks are achieved for dense pixel grids.

TABLE 2

Clock setting for different resoulutions

Parameter	**640×480**	**800×600**	**1280×720**	**1920×1080**
MULT_MASTER	31.5	10.0	37.125	37.125
DIV_MASTER	5	1	5	5
DIV_5×	5.0	5.0	2.0	1.0
DIV_1×	25	25	10	5

The display_timing module generates horizontal and vertical sync signals (o_hs, o_vs), display enable (o_de),

and current pixel coordinates (o_sx, o_sy) based on the incoming pixel clock (i_pix_clk). These signals are fundamental for precise raster scanning and timing alignment with modern DVI dis- plays.

Table 3. presents the horizontal and vertical timing parameters. These include the active resolution (H_RES, V_RES) as well as blanking intervals (H_FP, H_SYNC, H_BP, and their vertical counterparts). At higher resolutions like

1920×1080, longer back porch values (e.g., H_BP = 148) allow more time for processing and synchronization. Sync

polarities (H_POL, V_POL) also adapt to modern display requirements—switching to active-high signals for resolutions 800x600 and above.

The output of the display_timing module directly drives the RGB multiplexers: mux_dvi_red, mux_dvi_green, mux_dvi_blue. These modules select between raw image data and AI-generated overlay visuals (received from upstream modules like probability_map_visualizing) using select signals. The chosen color values are then output to the DVI lines: DVI_RED, DVI_GREEN, and DVI_BLUE.

TABLE 3

Display timings for different resoulutions

Parameter	**640×480**	**800×600**	**1280×720**	**1920×1080**
H_RES	640	800	1280	1920
V_RES	480	600	720	1080
H_FP	16	40	110	88
H_SYNC	96	128	40	44
H_BP	48	88	220	148
V_FP	10	1	5	4
V_SYNC	2	4	5	5
V_BP	33	23	20	36
H_POL	0	1	1	1
V_POL	0	1	1	1

4.5. Chrontel Encoder Chip (CH7301C) and DVI Output Stage

The final and crucial step in the FPGA-based image classification pipeline involves the Chrontel (CH7301C) DVI Transmitter chip. This integrated circuit, utilized specifically on the ML605 FPGA development board, is designed for converting digital image data from the FPGA's internal processing units into standardized DVI signals, suitable for high-quality video output.

The Chrontel (CH7301C) is a specialized semiconductor device that accepts parallel digital data (typically in RGB format) from the FPGA and converts it into a serialized digital signal conforming to DVI standards. This chip facilitates the transition from internal FPGA processing outputs into a standard video signal suitable for displays. To achieve this, the (CH7301C) device internally incorporates encoding logic, parallel-to-serial conversion circuits, and synchronization logic. It receives 24-bit parallel RGB data signals along with synchronization signals (horizontal sync, vertical sync, and pixel clock signals) directly from the FPGA output pins. The device then performs parallel-to-serial data conversion, encoding the video data using the TMDS protocol, which is fundamental to the DVI standard.

In practice, once the AI inference results have been converted into color-coded pixel data by the probability map visualization module within the FPGA, these parallel pixel data streams are passed directly into the (CH7301C). This chip organizes the incoming RGB digital signals, applies necessary timing and synchronization adjustments, and serializes the data into TMDS-compliant signals. TMDS encoding ensures minimal electromagnetic interference (EMI), high-speed data transmission, and robust signal integrity, allowing reliable delivery of digital video signals across standard DVI cables to display devices.

The Chrontel chip manages critical video signal timing, including pixel clock generation, horizontal synchronization (HSYNC), vertical synchronization (VSYNC), and data enable (DE) signals. Correct synchronization of these signals ensures stable and flicker-free images. The chip typically supports a wide range of resolutions, accommodating various resolutions defined by FPGA configurations. Internally, the (CH7301C) also integrates modules for color space management and signal integrity control, ensuring consistent output quality and compatibility with digital displays.

The DVI standard itself is a high-speed digital interface widely used for video transmission between source devices (such as FPGA boards or graphics cards) and display monitors. DVI leverages the TMDS standard, effectively minimizing electromagnetic interference and maintaining signal integrity, making it well-suited for high-resolution, high-bandwidth digital video streams. On the ML605 FPGA board, the Chrontel (CH7301C) precisely encodes and transmits the FPGA- generated video signals, ensuring that the classification output images are clearly, accurately, and promptly displayed without data loss or distortion.

Key Components of the DVI Signal:

1) Pixel Data (RGB Values): Represents pixel colors as 8- or 10-bit RGB values. In the provided Verilog code, a 1-bit RGB representation uses the MSB for output, simplifying color processing.

2) Synchronization Signals:

- HSYNC (Horizontal Sync): Marks the start of a new pixel row.
- VSYNC (Vertical Sync): Indicates the start of a new frame, resetting the display for the next frame.

3) Clock Signal (DVI_CLK): Synchronizes pixel data and timing signals with the display's refresh rate for smooth video output.

Timing and Display Processing: The Verilog code manages timing through modules that generate pixel positions (sx, sy), frame signals, and synchronization outputs (h_sync, v_sync). These signals ensure pixel data aligns with the LCD's grid structure.

LCD Display Conversion: The LCD controller

processes incoming RGB data to adjust liquid crystal cells, determining pixel color and intensity. HSYNC, VSYNC, and DVI_CLK ensure data is applied in the correct sequence, maintaining image integrity and frame synchronization.

4.6. Displaying the Output of the Trained Neural Network on Monitor using DVI

Using the DVI to analyze trained neural network results, a system was implemented to display outputs on a monitor. This involves hardware-software integration to enable efficient and accurate visualization.

Process Overview: Neural network outputs, such as class labels or probabilities, are processed into visual formats (e.g., bounding boxes, heatmaps). A lightweight rendering engine maps these data into graphical primitives and generates images or video frames compatible with DVI displays.

DVI Signal Generation and Hardware Integration: A DVI transmitter module encodes visual content into synchronization signals (HSYNC, VSYNC) and RGB pixel data. An FPGA or microcontroller ensures compliance with DVI timing specifications, delivering high-resolution, low-latency output to the monitor

5. EXPERIMENTAL RESULTS

As a case study the proposed CNN for classification of CIFAR-10 images is implemented in Verilog as suggested in [31], to display the output classification of this network we used the proposed method, Result in Table 4. indicate the number of slice registers, slice LUTs, I/O pins and global clock buffers (BUFG) used in CNN architecture. The DVI protocol implementation on the ML605 FPGA was evaluated in terms of device utilization to highlight its resource efficiency. Our analysis focused on key FPGA resources—namely slice registers, LUTs, and logic components. Table 5. presents the synthesis report results, which reflect the resource usage of the DVI protocol module (as illustrated in Fig. 4). The notably low resource consumption (31 slice registers and 90 LUTs) underscores the lightweight nature of our design for generating synchronization signals in stage 4 explained in section IV-D.

TABLE 4
Device utilization summary for slice logic of prorosed CNN for classification of cifar-10 images

Resource	Used	Available	Utilization (%)
Slice Registers (FF)	910	301,440	0.30
Slice LUTs	1,871	150,720	1.24
I/O Pins	357	720	49.58
Global Clock Buffers (BUFG)	1	32	3.13

TABLE 5
Device utilization summary for slice logic for generating synchronization signals

Resource	Used	Available	Utilization (%)
Slice Registers	31	301,440	1
Slice LUTs	90	150,720	1
Logic Components	88	150,720	1

The results demonstrate efficient utilization of FPGA resources, with significant room for additional functionality if needed. The utilization metrics indicate that the design is optimized and suitable for real-time processing while maintaining a low resource footprint. Our experiments confirmed the success of the FPGA-based DVI protocol implementation. The results showed that the system meets DVI standards for signal synchronization and output quality, supports high-resolution video displays, and demonstrates real-time performance for image processing and displaying the predicted output prob- ability map of neural network. The following key points were confirmed through testing:

- Precise signal synchronization with no visible distortion or delay.
- High-quality video output, supporting resolutions up to 1920×1080 at 60 Hz.
- Real-time image processing with a throughput of up to 30 fps at 1280×720 resolution.
- Efficient use of FPGA resources, leaving room for additional tasks or optimizations.

These results validate the feasibility and effectiveness of using the ML605 FPGA for displaying the predicted output probability map of trained model by neural network and image processing applications and pave the way for future improvements in FPGA-based video systems. As shown in Fig. 4, the probability values of the classes in the multi-class trained neural network can be displayed on an LCD. Each class probability is represented by the width of the corresponding column bar, visually indicating the likelihood of each class.

Fig. 4 illustrates the neural network output as it is rendered on an LCD screen via the DVI protocol implemented on the FPGA. The figure shows how classification results—such as class probabilities—are translated into graphical bar elements for real-time display. Each class is represented by a colored bar, where the bar's width corresponds to the predicted probability of that class. This format enables quick interpretation of classification outcomes, supporting applications such as medical imaging or real-time object detection. The FPGA ensures precise synchronization of pixel and timing signals, allowing seamless and low-latency image rendering. This figure demonstrates the system's effectiveness in converting computational results into a clear and immediate display format, highlighting its applicability for embedded and edge AI systems.

Fig. 4. Real-time display of the CNN model's probability map on the CIFAR-10 dataset during the testing phase. The width of each color bar indicates the predicted probability of the corresponding class, with the widest bar representing the most likely classification.

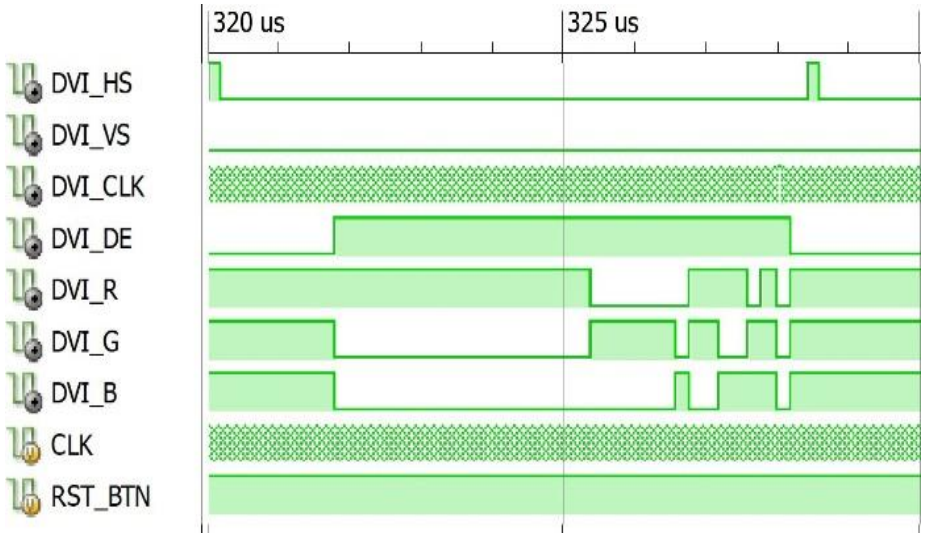

Fig. 5. Signal waveforms for timing, pixel data, and clock synchronization.

Fig. 5. shows the signal waveforms captured during the execution of the design. These waveforms illustrate the synchronization of timing signals, pixel data, and clock signals, which are critical for ensuring the correct operation of the DVI protocol.

The signal waveforms validate the proper implementation of the DVI protocol, showing stable synchronization across all required signals. These results support the design's ability to handle real-time video outputs effectively.

Using on-board LEDs to monitor the status of TMDS signals is an effective and simple method for debugging the DVI connection from an FPGA to a monitor. If the LEDs connected to the TMDS lines, such as data or clock, are not blinking, it indicates that the TMDS signals are not initialized correctly, and the issue is not related to the monitor. To troubleshoot, we ensure proper initialization of TMDS signals, verify clock settings on the FPGA, and confirm that the DVI standard configurations (e.g., resolution, sampling rate) are accurate

6. CONCLUSION

This paper presented a hardware implementation of the DVI protocol on the ML605 FPGA platform for real-time display of neural network and image processing outputs. Leveraging the capabilities of the Xilinx Virtex-6

architecture and the flexibility of FPGA-based design, a system was developed to transmit high-quality digital video signals directly to an external monitor.

The implementation addresses critical challenges such as pixel synchronization, precise timing control, and protocol compliance, thereby enabling accurate rendering of classification results without relying on external processors. The system

supports high-resolution output (up to 1920×1080 at 60 Hz) and demonstrates low resource utilization, making it suitable

for embedded and edge AI applications.

By embedding the DVI output functionality within the FPGA and integrating lightweight neural network inference, this work provides an effective hardware-software co-design framework for real-time feedback in intelligent systems. The approach is particularly valuable for tasks requiring low latency and high reliability, such as smart monitoring and medical imaging.

Future research could explore extending the design to sup- port alternative video standards (e.g., HDMI or DisplayPort), implementing more complex neural architectures, or scaling the design for multi-channel outputs. Additionally, improvements in memory access patterns, dynamic reconfiguration, or adaptive resolution could further enhance system performance and flexibility.

Overall, this work demonstrates the viability of FPGA-based systems for efficient, high-performance neural network inference and real-time display using digital video interfaces, contributing to the advancement of intelligent embedded system design.

7. REFERENCES

[1] J. Park and D. Kim. (2024, Jan.). Statistical Eye Diagrams for High-Speed Interconnects of Packages: A Review. *IEEE Access*. [Online]. *12*, pp. 22880–22891. Available: https://doi.org/10.1109/ACCESS.2024.3359037

[2] S. Singh, A. S. Mandal, C. Shekhar, and A. Vohra. (2017, Jun.). Memory Efficient VLSI Implementation of Real-Time Motion Detection System Using FPGA Platform. *Journal of imaging*. [Online]. *3(2)*, p. 20. Available: https://doi.org/10.3390/jimaging3020020.

[3] J. Plusquellic, E. E. Tsiropoulou, and C. Minwalla. (2023). Privacy-Preserving Authentication Protocols for IoT Devices Using FPGA and DVI Integration. *IEEE Transactions on Emerging Topics in Computing*. [Online]. Available: https://ece-research.unm.edu/jimp/pubs/SiRF_Authentication_FINAL.pdf

[4] H. Park and S. Kim. (2023). Overviewing AI-Dedicated Hardware for On-Device AI in Smartphones. *In Artificial Intelligence and Hardware Accelerators*. Springer International Publishing. [Online]. pp. 127–150. Available: https://doi.org/10.1007/978-3-031-22170-5_4

[5] V. Jain, S. Giraldo, J. De Roose, L. Mei, B. Boons, and M. Verhelst. (2023, Aug.). TinyVers: A Tiny Versatile System-on-Chip with State-Retentive eMRAM for ML Inference at the Extreme Edge. *IEEE Journal of Solid-State Circuits*. [Online]. *58(8)*, pp. 2360–2371. Available: https://doi.org/10.1109/JSSC.2023.3236566

[6] J. Peddie. (2023). The GPU Environment—Hardware. *In The History of the GPU—Eras and Environment*. Springer International Publishing. [Online]. pp. 151–200. Available: https://doi.org/10.1007/978-3-031-13581-1_5

[7] D. G. Bailey, "Design for Embedded Image Processing on FPGAs," John Wiley & Sons, 2023.

[8] Chrontel, Inc. (2014). *CH7301C DVI Transmitter Device Datasheet*, Rev. 2.1. [Online]. Available: CH7301C DVI Transmitter Device.

[9] M. A. Nuño-Maganda, J. H. Jiménez-Arteaga, J. H. Barron-Zambrano, Y. Hernández-Mier, J. C. Elizondo-Leal, A. Díaz-Manríquez, C. Torres-Huitzil, and S. Polanco-Martagón. (2022). Implementation and Integration of Image Processing Blocks in a Real-Time Bottle Classification System. *Scientific Reports*. [Online]. *12(1)*, p. 4868. Available: https://doi.org/10.1038/s41598-022-08777-x

[10] W. Baisi, "A Machine Learning Approach to Optimizing CNN Deployment on Tile-Based Systems-on-Chip," Doctoral dissertation, Politecnico di Torino, 2024.

[11] C. Kyrkou, C. Ttofis, and T. Theocharides. (2011, Sep.). FPGA-Accelerated Object Detection Using Edge Information. In 2011 21st International Conference on Field Programmable Logic and Application, pp. 167–170. [Online]. Available: https://doi.org/10.1109/FPL.2011.38

[12] M. Abernot, "Digital Oscillatory Neural Network Implementation on FPGA for Edge Artificial Intelligence Applications and Learning," Doctoral dissertation, Université de Montpellier, 2023.

[13] D. Guarrera, "Real-time processing of neuronal network activity measured by a high-density microelectrode matrix through an FPGA card," Thesis, Univ. of Padova, 2022.

[14] X. Kong, F. Yu, W. Yao, S. Cai, J. Zhang, and H. Lin. (2024). Memristor-Induced Hyperchaos, Multiscroll and Extreme Multistability in Fractional-Order HNN: Image Encryption and FPGA Implementation. *Neural Networks*. [Online]. 171, pp. 85–103. Available: https://doi.org/10.1016/j.neunet.2023.12.008

[15] J. W. Park, H. Lee, B. Kim, D. G. Kang, S. O. Jin, H. Kim, and H. J. Lee. (2019, Jan.). A Low-Cost and High-Throughput FPGA Implementation of the Retinex Algorithm for Real-Time Video Enhancement. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. [Online]. *28(1)*, pp. 101–114. Available: https://doi.org/10.1109/TVLSI.2019.2936260

[16] C. Wang and Z. Luo. (2022, Oct.). A Review of the Optimal Design of Neural Networks Based on FPGA. *Applied Sciences*. [Online]. *12(21)*, p. 10771. Available: https://doi.org/10.3390/app122110771

[17] X. Liu and Y. Chen. (2023). Hybrid FPGA-GPU Co-Design for Real-Time Neural Network Inference and Visualization. In IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 712–718. [Online]. Available:

https://doi.org/10.1109/ISCAIE57739.2023.10165432

[18] V. Thyagarajan et al. (2023). Video Analytics with FPGA-Based Smart Cameras for Object Recognition in Hockey Games. Presented at International Conference on Networking and Communications (ICNWC). [Online]. Available: https://doi.org/10.1109/ICNWC57852.2023.10127549

[19] L. Nowosielski, R. Przesmycki, and M. Nowosielski. (2016, Aug.). Compromising Emanations from VGA and DVI Interface. Presented at 2016 Progress in Electromagnetic Research Symposium (PIERS), pp. 1024–1028. [Online]. Available: https://doi.org/10.1109/PIERS.2016.7734570

[20] T. T. Hoang, N. H. Nguyen, and X. T. Nguyen. (2012). A Real-Time Object-Recognition System Based on PCNN Algorithm. International Journal of Neural computing [Online]. Available: https://thuchoang90.github.io/assets/pp/2012_ICDV.pdf

[21] H. Fang, Z. Mei, A. Shrestha, Z. Zhao, Y. Li, and Q. Qiu. (2020, Nov.). Encoding, Model, and Architecture: Systematic Optimization for Spiking Neural Network in FPGAs. In Proceedings of the 39th international conference on computer-aided design, pp. 1–9. [Online]. Available: https://doi.org/10.1145/3400302.3415608

[22] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello. (2010, May). Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems. In Proceedings of 2010 IEEE international symposium on circuits and systems, pp. 257–260. [Online]. Available: https://doi.org/10.1109/ISCAS.2010.5537908

[23] N. Yildiz, E. Cesur, K. Kayaer, V. Tavsanoglu, and M. Alpay. (2014, Oct.). Architecture of a Fully Pipelined Real-Time Cellular Neural Network Emulator. *IEEE Transactions on Circuits and Systems I: Regular Papers*. [Online]. *62(1)*, pp. 130–138. Available: https://doi.org/10.1109/TCSI.2014.2345502

[24] K. Kayaer and V. Tavsanoglu. (2009, Apr.). A New Cellular Neural Network Emulator Architecture Processing Video Real-Time. In 2009 IEEE 17th Signal Processing and Communications Applications Conference, pp. 1024–1028. [Online]. Available: https://doi.org/10.1109/SIU.2009.5136446

[25] M. Abernot, "Digital Oscillatory Neural Network Implementation on FPGA for Edge Artificial Intelligence Applications and Learning," Ph.D. dissertation, Univ. of Montpellier, 2023.

[26] O. L. Savkay, N. Yildiz, E. Cesur, M. E. Yalcin, and V. Tavsanoglu. (2013, Sep.). Realization of Preprocessing Blocks of CNN Based CASA System on FPGA. In 2013 European Conference on Circuit Theory and Design (ECCTD), pp. 1–4. [Online]. Available: https://doi.org/10.1109/ECCTD.2013.6662238

[27] P. Antonik, *Application of FPGA to Real-Time Machine Learning: Hardware Reservoir Computers and Software Image Processing*. Springer, 2018.

[28] A. Ahilan and E. A. K. James. (2011, Dec.). Design and Implementation of Real-Time Car Theft Detection in FPGA. In Third International Conference on Advanced Computing, pp. 353–358. [Online]. Available: https://doi.org/10.1109/ICoAC.2011.6165201

[29] D. Davutoglu, N. Yildiz, U. E. Ayten, and V. Tavsanoglu. (2018). Real-Time Frame Buffer Implementation Based on External Memory Using FPGA. *Procedia Computer Science*. [Online]. *131*, pp. 641–646. Available: https://doi.org/10.1016/j.procs.2018.04.307

[30] A. Fasih, C. Schwarzlmueller, K. Kyamakya, and F. A. Machot. (2010). Video Enhancement for ADAS Systems Based on FPGA and CNN Platform. *International Journal of Signal & Image Processing*. [Online]. *1(3)*.

[31] A. Padhi and S. V. (2020). *Image-Classification-Using-CNN-on-FPGA*. [Online].