# RTL Design and FPGA Implementation of a Real-Time Eye Tracking System using a Novel Projection Algorithm

Pengju Zhang, Kuoyuan Jia, Yubiao Liu, Yuanxiao He, Liang Chen, Cheng Liu*
*School of Microelectronics, Shanghai University, Shanghai , China*

*Corresponding author's e-mail: thomas_liucheng@shu.edu.cn

## ABSTRACT

Software-based eye tracking systems, often leveraging libraries like OpenCV, can encounter significant latency and throughput bottlenecks in demanding real-time scenarios, especially when processing high-resolution video streams. This paper presents an innovative FPGA-based solution to overcome these limitations, detailing the design and implementation of a real-time eye tracking system entirely in Verilog HDL for optimized performance and resource utilization. The primary objective is robust pupil tracking at 720P resolution and 60 frames per second (fps). The system utilizes an infrared camera for video acquisition, processes data on an VF-Ti60F100 development board, and displays the tracked pupil position on an HDMI monitor. Core processing stages include image preprocessing, gradient-based glint detection, and pupil localization within an extracted eye sub-image. Innovations embedded in the hardware design feature: a novel combined vertical projection and pipelined centroid algorithm for efficient glint localization using only vertical image data and on-chip RAM; direct processing of compact eye sub-images centered on the detected glint; and a robust hardware-based blink detection mechanism analyzing glint stability. Experimental results demonstrate that the system successfully achieves real-time pupil tracking at 720p@60fps with high accuracy and extremely low latency, effectively processing input resolutions up to 1280x1024. This underscores the potential of FPGAs for developing high-performance, low-latency HCI applications.

**Keywords:** Eye Tracking, Real-Time Systems, Image Processing, Embedded Systems, FPGA, RTL Design.

## 1. INTRODUCTION

Eye tracking technology offers a powerful avenue for enhancing human-computer interaction (HCI), enabling users to control interfaces and convey information through gaze. While traditional input methods like keyboards and mice can be cumbersome or inaccessible, eye-gaze control presents a more intuitive alternative. However, achieving real-time, low-latency eye tracking, especially at high resolutions, poses significant computational challenges for conventional software-based approaches running on CPUs. Such methods often suffer from processing delays and may require complex algorithms like face detection as a preliminary step, consuming valuable resources.

To address these limitations and achieve higher performance, this research focuses on developing a real-time eye-tracking system implemented entirely in Verilog HDL on an FPGA platform. The primary goals were to accelerate algorithm execution speed, improve resource utilization, and fully leverage the inherent parallelism of FPGAs. Specifically, we aimed to achieve robust pupil tracking at 720P resolution and 60 frames per second (fps). Our methodology avoids preliminary face detection by directly locating the infrared corneal reflection (glint). Initial glint localization occurs within a single frame period, after which a Region of Interest (ROI) around the glint is extracted for precise pupil localization. This significantly enhances real-time performance by reducing processing complexity.

This paper presents the hardware architecture and optimized image processing algorithms, implemented in Verilog HDL,

for the proposed eye-tracking system. Furthermore, experimental results are presented to validate the system's achievement of demanding real-time performance requirements, specifically demonstrating 720p@60fps output capability while processing 1280x1024 input resolution. Potential applications for this technology include assistive technology [1], automotive safety [2], virtual/augmented reality (VR/AR) [3], and fundamental research within psychology and neuroscience [4,5].

## 2. SYSTEM AND METHODS

### 2.1 System Architecture

The eye-tracking system comprises several interconnected hardware modules, designed for efficient real-time processing entirely in Verilog on an FPGA platform. The overall system flowchart, illustrating the data flow from camera to display, is depicted in Figure 1. The implementation relies entirely on Verilog HDL to maximize the benefits of hardware parallelism and pipelining.

The core components include:

- **Data Acquisition:** An SC130GS infrared camera captures high-resolution video frames (up to 1280x1024 pixels). An external 850nm infrared LED provides active illumination, enhancing the visibility of the corneal reflection.
- **FPGA Platform:** An VF-Ti60F100 development board serves as the central processing hub, hosting the Verilog-based processing pipeline.
- **Interface and Decoding:** Camera data is transmitted via the MIPI CSI-2 interface. A dedicated mipi_rx IP core on the FPGA decodes the MIPI stream into image data. Camera configuration is managed via I2C using i2c_timing_ctrl and associated modules.
- **Memory Management:** High-bandwidth external HyperRAM is used for frame buffering, essential for handling high-resolution input. An axi4_ctrl module, implementing the AXI4 bus protocol, interfaces with the HyperRAM control core. This controller manages four FIFO buffers to enable high-speed storage and retrieval of image data streams during processing.
- **Image Processing Pipeline:** Described in detail in Section 2.2, this core logic implemented in Verilog performs the eye tracking algorithm sequentially.
- **Video Output:** The final video stream, with bounding boxes indicating the tracked pupil position, is formatted by the lcd_driver and converted to a HDMI compatible signal by the rgb2dvi module for display at 720p@60Hz.

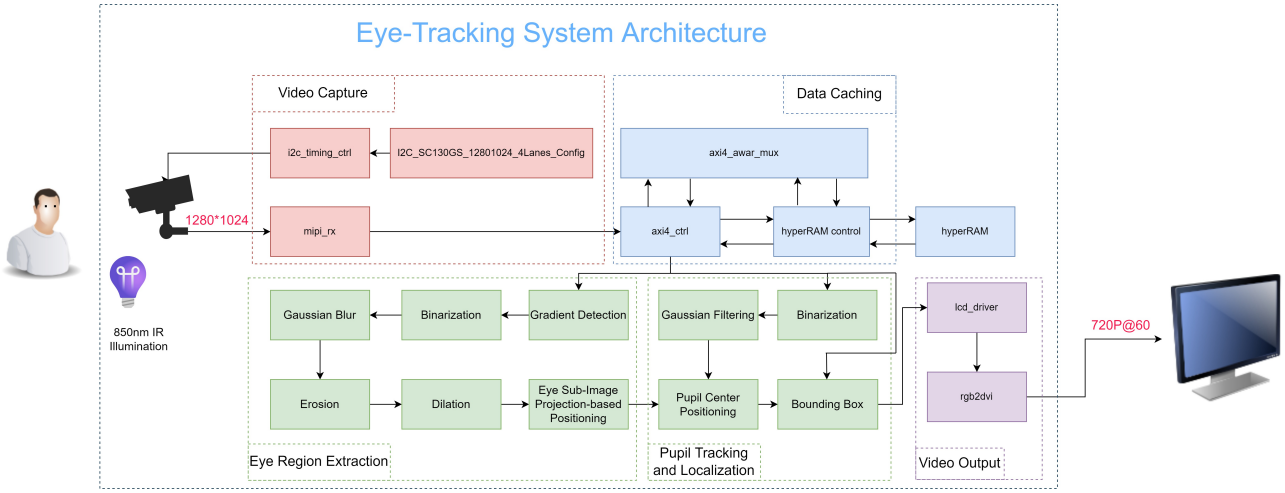



Figure 1. Overall system flowchart, illustrating data flow from camera acquisition through FPGA processing stages to HDMI display output.

### 2.2 Image Processing Pipeline

The Verilog HDL image processing pipeline targets 720p@60fps output, employing pixel-stream processing where feasible to minimize latency. Optimized for FPGA resources and throughput, the pipeline integrates hardware blink detection via glint stability analysis. The main processing stages (illustrated in Fig. 2) are detailed below.

#### 2.2.1 Initial Image Preprocessing

The raw image stream first passes through preprocessing stages to enhance salient features and reduce noise:

- **Gradient Detection:** A gradient detection algorithm enhances image structural data, reducing sensitivity to illumination changes (Fig. 2b).
- **Binarization:** Global thresholding segments the bright corneal reflection (glint) from the background; pixels above the threshold become white (`8'hFF`), others black (`8'h00`) (Fig. 2c).
- **Filtering and Morphology:** A 3x3 Gaussian filter (using line buffers and efficient 4-bit right-shift normalization) first smooths noise while preserving glint structure. This is followed by morphological opening (3x3 kernel, erosion then dilation, also implemented using line buffers) to remove spurious noise pixels and refine glint contours (Fig. 2d).

#### 2.2.2 Initial Glint Localization

Post-preprocessing, gradient analysis identifies candidate glint locations via maximum H/V gradient magnitudes (Fig. 2e). A blink state is flagged upon failure to detect glint candidates over consecutive frames, enabling downstream handling of tracking loss.

#### 2.2.3 Improved Glint Centroid Calculation

Glint coordinates are refined using a custom hardware-optimized centroid algorithm. Relying exclusively on vertical projection data simplifies computation and potentially resource usage.

The algorithm accumulates glint pixel counts ($N_{\text{pixels}}$) and Y-coordinate sums ($\sum y_i$) per column into Block RAMs. The column index ($X$) with the maximum count ($N_{\text{pixels}}$) yields the glint's horizontal position. The vertical centroid ($y_{\text{centroid}}$) for column $X$ is then calculated from the stored sum and count (Algorithm 1).

**Algorithm 1:** Simplified Glint Centroid Calculation via Vertical Projection

**Data:** Input image stream pixels $px(x,y)$, Image dimensions $(W,H)$.
**Result:** Glint X-coordinate ($X_{glint}$), Glint Y-centroid ($Y_{centroid}$).

Initialize $N_{\text{pixels}}[x] \leftarrow 0$ and $\sum Y[x] \leftarrow 0$ for all $x \in [0, W-1]$;
**for** $y \leftarrow 0$ **to** $H-1$ **do**
  **for** $x \leftarrow 0$ **to** $W-1$ **do**
    **if** *pixel* $px(x,y) = 8'hFF$ **then**
      $N_{\text{pixels}}[x] \leftarrow N_{\text{pixels}}[x] + 1$;
      $\sum Y[x] \leftarrow \sum Y[x] + y$;
    **end**
  **end**
**end**
**Find** $X_{glint}$ *such that* $N_{pixels}[X_{glint}] = \max_x(N_{pixels}[x])$ ;
**if** $N_{pixels}[X_{glint}] > 0$ **then**
  $Y_{centroid} \leftarrow \sum Y[X_{glint}] / N_{\text{pixels}}[X_{glint}]$;
**end**
**else**
  $Y_{centroid} \leftarrow 0$;
**end**

This RAM-based, pipelined approach using only vertical data significantly reduces logic complexity and resource utilization, particularly Look-Up Tables (LUTs), enhancing real-time performance (Fig. 2f). The final centroid calculation (division) is implemented using an efficient pipelined hardware divider.

#### 2.2.4 ROI Definition and Pupil Localization

A Region of Interest (ROI) for pupil localization is then defined based on the refined glint location. Typically a 150x100 pixel sub-image centered at the glint coordinates, the ROI is retrieved from the full-frame data cached in external HyperRAM. Processing this smaller ROI significantly reduces computational load for subsequent pupil detection (Fig. 2g).

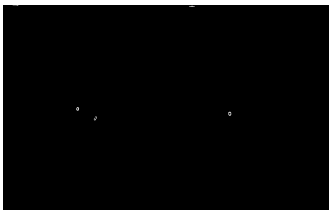
Pupil localization proceeds independently within the ROI. The ROI is binarized and Gaussian filtered using an ROI-specific threshold to segment the pupil. A projection algorithm computes the pupil's centroid coordinates relative to the ROI boundaries. Finally, these relative coordinates are transformed into the full-frame coordinate system to yield the absolute pupil position (Fig. 2h).

#### 2.2.5 Bounding Box Generation

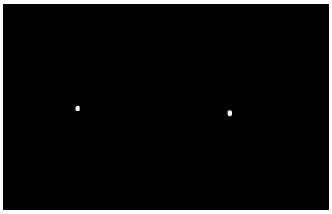
The final pipeline stage visualizes the tracking result by generating bounding boxes centered on the calculated pupil centroid coordinates. Box dimensions are set by hardware constants (`BOX_WIDTH`, `BOX_HEIGHT`). During video output streaming, dedicated logic identifies pixels on the bounding box perimeter and overrides their color values, superimposing the boxes onto the video stream sent to the HDMI display. The final output with this overlay is shown in Fig. 2i.

(a) Original input image.

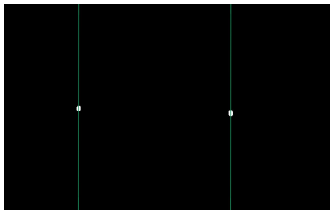
(b) Gradient magnitude.



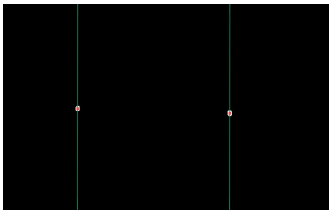
(c) Binarized image.



(d) After morph. opening.



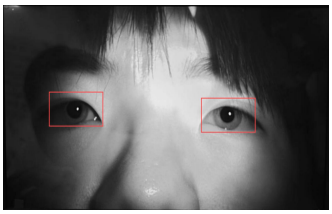
(e) Initial glint candidates.



(f) Refined glint centroid.

(g) Extracted ROI.

(h) Pupil localized in ROI.



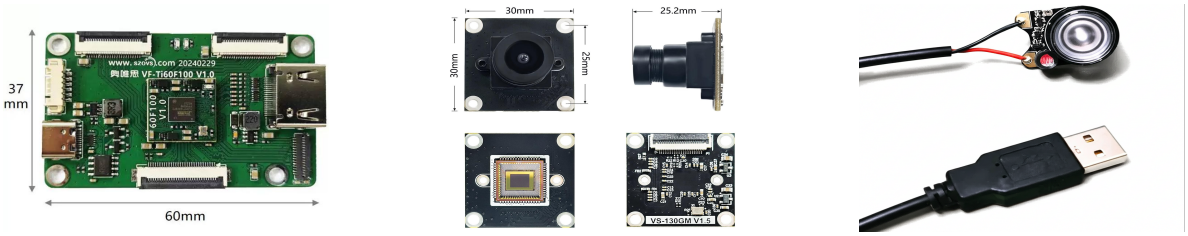
(i) Final output overlay.

Figure 2. Intermediate and final results of the image processing pipeline stages.

## 3. EXPERIMENTAL RESULTS AND ANALYSIS

### 3.1 Experimental Setup

The experimental system was implemented and validated using the hardware configuration detailed below. Central to the system, an VF-Ti60F100 FPGA development board served as the primary processing platform. Video input, at resolutions up to 1280x1024 pixels, was acquired using an SC130GS infrared camera module. Active illumination of the subject's eyes was supplied by an external 850nm Infrared (IR) LED. The processed output, featuring real-time pupil tracking at 720p@60Hz, was displayed on a standard HDMI monitor. The Verilog HDL design was synthesized and implemented using standard Electronic Design Automation (EDA) tools (Efinity). Figure 3 illustrates the key physical components of the experimental hardware used to realize the system architecture detailed in Figure 1.




(a) VF-Ti60F100 FPGA board. (b) SC130GS IR camera module. (c) 850nm IR LED illuminator.

Figure 3. Key hardware components of the experimental setup: (a) VF-Ti60F100 development board, (b) SC130GS infrared camera module, and (c) 850nm infrared LED illuminator.

### 3.2 Evaluation Metrics

System performance was evaluated against primary design goals and comparative benchmarks, as detailed in Table 1. The key metrics for evaluation included: FPGA resource utilization (Logic Cells and Block RAMs), maximum input resolution, real-time capability, tracking latency, smoothness of tracking, the inclusion of blink detection, and maximum output display rate. Qualitative assessments of tracking fidelity and overall functional completeness were also considered.

### 3.3 Performance Analysis and Discussion

Qualitative evaluation corroborated the system's robust real-time performance. The bounding boxes overlaid on the HDMI output accurately tracked the user's gaze across the monitor. As indicated by the "Smooth Tracking (No Tailing)" metric in Table 1, tracking appeared fluid, without noticeable lag or tailing artifacts, even during rapid eye movements. This observation points to the minimal "Tracking Latency" achieved through the pipelined hardware design. The effectiveness of the algorithm in locating the pupil relative to the glint is visually demonstrated through the processing stages depicted in Fig. 2. The system consistently achieved reliable pupil detection and could track both eyes independently when visible, supporting its functional completeness, including the "Blink Detection" capability.

A comparison with representative software-based approaches, such as an OpenCV implementation on a standard CPU (refer to Table 1), underscores the advantages of the proposed hardware implementation. While both methodologies can achieve "Real-time Capability" for basic tracking, the FPGA system provides a notably higher "Max. Output Display" frame rate (720p @ 60 Hz vs. 720p @ 30 Hz) and ensures minimal, deterministic "Tracking Latency." In contrast, software methods are prone to higher and more variable latency contingent on CPU load and operating system scheduling. The design's "Logic Cells (XLRs)" and "Memory Blocks (BRAMs)" usage, documented as 32.6% and 35.2% respectively, indicates a moderate consumption of available FPGA resources. This level of utilization confirms the suitability of the chosen algorithms for efficient implementation within general FPGA fabric while processing a "Max. Input Resolution" of 1280x1024.

These results affirm the viability of the proposed FPGA-based methodology and the implemented algorithms for demand-

ing, high-performance, real-time eye tracking applications. The direct glint processing strategy, combined with the optimized centroid calculation, proves effective in achieving high frame rates within moderate resource constraints.

### 3.4 Limitations and Future Work

Despite the demonstrated real-time performance, several limitations present opportunities for future research. Firstly, the reliance on empirically determined fixed thresholds for binarization may compromise robustness across diverse users, ambient lighting conditions, and camera distances; implementing adaptive thresholding techniques could significantly enhance generalizability. Secondly, qualitative testing revealed reduced tracking sensitivity and accuracy for users wearing eyeglasses, attributed to interfering reflections affecting glint and pupil detection. Future investigations should focus on methods specifically designed to mitigate these effects, potentially by integrating robust iris recognition algorithms capable of addressing such artifacts[5]. Addressing these limitations would improve the system's practical applicability in real-world HCI environments.

Table 1. Performance Comparison and FPGA Resource Utilization.

| Feature / Metric | Verilog Implementation (This Work) | OpenCV Implementation[6] |
|---|---|---|
| Implementation Platform | FPGA (VF-Ti60F100) | CPU (Intel Core i5) |
| Development Language | Verilog HDL | Python |
| Resource Focus | Logic Cells, Block RAMs | CPU Cycles, System RAM |
| Logic Cells (XLRs) | 19827 / 60800 (32.6%) | N/A |
| Memory Blocks (BRAMs) | 90 / 256 (35.2%) | N/A |
| Max. Input Resolution | 1280x1024 | 1280x1024 |
| Real-time Capability | Yes (✓) | Yes (✓) |
| **Tracking Latency** | **Minimal** | **Higher** |
| Smooth Tracking (No Tailing) | Yes (✓) | Yes (✓) |
| Blink Detection | Yes (✓) | Yes (✓) |
| **Max. Output Display** | **720p @ 60 Hz** | **720p @ 30 Hz** |

## 4. CONCLUSION

This paper detailed the successful design, Verilog implementation, and evaluation of a high-performance, real-time eye-tracking system realized on an FPGA. By leveraging FPGA parallelism and an optimized algorithm pipeline—incorporating a novel vertical-projection-only glint centroid method and direct sub-image processing—the system achieved accurate dual-pupil tracking, delivering a 720p@60fps HDMI output while processing input resolutions up to 1280x1024.

Key contributions include the direct glint detection and sub-image processing strategy, which circumvents face detection overhead, and the development of a resource-efficient glint centroid algorithm employing RAM and pipelining techniques. Experimental results validated the system's high speed, low latency, tracking accuracy, and robustness, demonstrating clear advantages relative to comparable software-based approaches, particularly in achievable output frame rate.

While the current implementation shows promise, limitations regarding robustness to eyeglasses and reliance on fixed thresholds were identified. Future work will focus on incorporating adaptive techniques and potentially iris recognition to address these issues, alongside more extensive quantitative validation. This work underscores the efficacy of FPGAs for demanding real-time computer vision tasks like eye tracking and provides a solid foundation for developing advanced HCI applications in diverse fields.

## ACKNOWLEDGMENTS

This research has been supported by Lingang Laboratory Key Project LG-GG-202402-05-02.

## REFERENCES


[1] M. Donegan, L. Oosthuizen, S. P. Arjunan, and M. Rugel, "Eye Gaze Technology and the Application for People with Motor Neuron Disease," *Assist. Technol.*, 32(6), 300–308 (2020).

[2] C. Ahlstrom, K. Kircher, and J. Jansson, "Eye tracking in applied settings: A state-of-the-art review of gaze parameters for assessing cognitive load, drowsiness, and distraction in driving," *Hum. Factors*, 64(7), 1183–1215 (2022).

[3] V. Clay, P. König, and S. U. König, "Eye tracking in virtual reality," *J. Eye Mov. Res.*, 12(1) (2019).

[4] J. Najemnik and W. S. Geisler, "Optimal eye movement strategies in visual search," *Nature*, 434(7031), 387–391 (2005).

[5] Z. Zhang, "Research on Iris Recognition Algorithm and System Implementation," M.S. thesis, Lanzhou Univ., Lanzhou, China (2009).

[6] S. Goñi, J. Echeto, A. Villanueva, and R. Cabeza, "Robust algorithm for pupil-glint vector detection in a video-oculography eyetracking system," *Proc. 17th Int. Conf. Pattern Recognit.*, 4, 941-944 (2004).

[7] Y. Zou, P. Guan, B. Yang, Q. Gong, and H. Xing, "Implementation of Fast Eye Movement Tracking System Based on FPGA and Projection Algorithm," *Eng. Sci. Technol.*, 48(3), 100–106 (2016).

[8] K. Huang, "Research and Implementation of Eye Tracking System Based on FPGA," M.S. thesis, Chongqing Univ., Chongqing, China (2015).

[9] M. Wu, "Research on Real-time Pupil Localization Technology Based on FPGA," M.S. thesis, Xi'an Inst. Opt. Precision Mech., Chin. Acad. Sci., Xi'an, China (2024).