Received 24 November 2023, accepted 11 December 2023, date of publication 14 December 2023, date of current version 21 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3342839

RESEARCH ARTICLE

A Resource-Efficient Multi-Function Embedded Eye Tracker System Implemented on FPGA

AHMAD MOURI ZADEH KHAKI[1], **SANGHYEOK YANG**[2], **HYUNSOO KIM**[2], **ANKUR SINGH**[1], **AND BYUNG-GEUN LEE**[1], **(Member, IEEE)**
[1]Department of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea
[2]Electronic Devices Research Team, Energy Materials Research Group, Fundamental Materials Research Center, Uiwang-si 16082, South Korea

Corresponding author: Byung-Geun Lee (bglee@gist.ac.kr)

This work was supported in part by the Nano-Material Technology Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and Information and Communication Technology (ICT) under Grant NRF2022M3H4A1A01009658, and in part by Hyundai Motor Company.

ABSTRACT In this paper, a multi-function embedded eye tracker system (ETS) has been presented and FPGA implementation of that has been described. The image processing algorithm is based on morphological operations to detect centers of pupil (PC) and corneal glint (GC) so that gaze of the user can be estimated by establishing a mapping between obtained PC-GC vector and screen. Moreover, the proposed ETS is capable to detect pupil size which is vitally important in understanding consciousness of the user since it varies in response to light condition. Taking advantage of optimum system design and parallel signal processing on high-speed digital units of the FPGA, the processing time of 1.8 ms was achieved that allows operation of the system up to 500 fps. The results of test with 176×120 images demonstrate maximum vertical or horizontal error of ± 1 pixel. Experimental results imply versatility of the proposed low-power and high-accuracy ETS for real-time and accurate applications such as medical diagnosis and human-computer interface.

INDEX TERMS Embedded system, field programmable gate arrays, gaze tracking, hardware acceleration, morphological operations.

I. INTRODUCTION

Gaze tracking is a well-known approach for a reliable cognitive assessment of human. Gaze tracking is defined herein as the process of detecting, tracking and modeling eyes of users which is the role of an eye tracker system (ETS) [1], [2]. With the increase in demand for ETSs worldwide, many researchers have been trying to introduce and develop eye tracking approaches for several applications.

To date, almost all of the presented ETSs are based on one the following methods: ellipse fitting, pattern recognition and morphological operations. In the method of ellipse fitting after binarizing image the largest contour is considered as pupil. Then, an ellipse is fitted to that contour. By fitting the ellipse, along with center location of the pupil, diameters length of that is provided as well [3]. However, in this method image processing is carried out using a software running on a complex system with high memory usage. Moreover, to remove false pupil candidates that can degrade the performance, extra processing unit is required which increase the system complexity further. In pattern recognition-based works, eye features such as pupil center location and eye corners are extracted using a supervised neural network (basically convolutional neural networks (CNN)) [4], [5] or unsupervised methods such as calculating cross correlation between captured image and a predefined template (a.k.a. filtering) [6]. On the one hand, by exploiting this method a fast and robust ETS can be obtained which is able to extract the required information for eye tracking such as pupil center (PC) location, eye corners and pupil size simultaneously. But on the other hand, the aforementioned advantages can be achieved only at the price of a complex hardware for training network and processing images which result in increase in cost and power consumption of the system. The eye trackers

The associate editor coordinating the review of this manuscript and approving it for publication was Ilaria De Munari.

which employ morphological operations detect PC by processing the binarized image pixel-wise. In fact, each pixel value is determined according to value of the neighboring pixels at each iteration. Thus, after a few iterations undesired pixels including false candidates are removed and only PC remains.

Except for morphological-based eye trackers, other eye tracking methods require a powerful processor and enormous amount of memory. As a result, they consume an extensive power to accomplish image processing. In addition, due to required time for transferring data between memory and processor they are not as fast as morphological-based eye trackers with in-memory processing capability. However, in morphological-based ETSs, false pupil candidates degrade the accuracy drastically. Basically, false pupil candidates can be originated from light source reflection on corneal and undesired pixels like eyelash and eyelids [7]. Consequently, by removing false pupil candidates, an accurate ETS can be achieved that can demonstrate low power consumption and complexity. However, several FPGA-based works fail to achieve high accuracy due to not mitigating this error properly [8]. Also, various works deal with that using software-hardware collaborative design which wastes benefits of FPGA hardware acceleration and is not appropriate for real-time applications [9].

In this work, in order to improve accuracy of the system over previous FPGA-based ETSs we remove false pupil candidates effectively by filling the holes in eye images caused by corneal glint and eliminating unwanted pixels other than pupil prior to detecting PC without the needs of any additional algorithm and the burden of more complexity. The proposed multi-function ETS can obtain crucial data for gaze estimation by detecting the pupil and corneal glint centers besides pupil size of the user based on morphological operations. Taking advantage of FPGA hardware acceleration and utilizing high-speed digital resources optimally, all objectives are detected simultaneously with frame rate up to 500 fps. Also, accuracy and robustness of the proposed design are verified through several tests. Experimental results and comparison with the recent state-of the-art ETSs imply the superiority of our design over previous works and competency of that for real-time applications such as human-computer interface (HCI).

The rest of this paper is organized as follows. In Section II design methodology of the proposed ETS is presented. Section III describes FPGA implementation of the proposed ETS in detail. Experimental results of the prototype ETS are provided and discussed in Section IV. Finally, conclusions are drawn in Section V.

II. PROPOSED ETS DESIGN METHODOLOGY

A. OPERATION PRINCIPLE

Fig.1 illustrates operation principle of the proposed ETS. First, eye image is captured by image sensor. Since the image processing is carried out in digital domain, eye

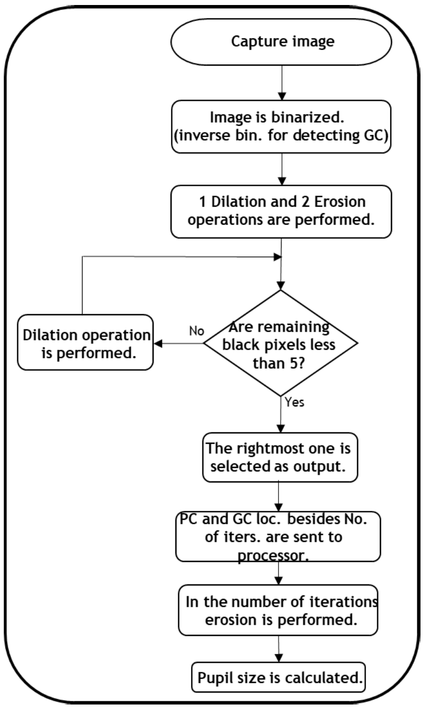

Capture image
Image is binarized. (inverse bin. for detecting GC)
1 Dilation and 2 Erosion operations are performed.
Are remaining black pixels less than 5?
No
Dilation operation is performed.
Yes
The rightmost one is selected as output.
PC and GC loc. besides No. of iters. are sent to processor.
In the number of iterations erosion is performed.
Pupil size is calculated.

FIGURE 1. **Operation principle of the proposed ETS.**

image is binarized using comparator circuit. The comparator threshold value is a critical parameter in morphological operations-based digital ETSs because an inappropriate value for that may result in irrecoverable errors such as wrong PC and GC location. In another case, if chosen value is too large, it causes an increase in the number of iterations and accordingly processing time and power consumption. And in the worst case, if a too small value is chosen for that, failure in obtaining appropriated information due to removing all pixels in the first dilation iteration is highly likely to occur. This parameter is proportional to light condition and object (eyes of the user) distance from light source and image sensor. By using IR LED and photodetector as light source and image sensor, dependence of threshold value on light condition can be mitigated. Also, in head-mounted device (HMD) configuration or an application in which the user is in a static position like medical diagnosis, the object distance to light source and camera is invariant. So, an optimum and permanent threshold value can be set heuristically during system configuration. Subsequent to binarization, several information can be extracted from eye image utilizing morphological operations.

Two well-known morphological operations in image processing are dilation and erosion. Dilation converts a black pixel to white if all neighboring pixels of that are not black. Conversely, erosion flips white pixels to black provided that all neighboring pixels of that are not white. Based on the pattern of sensing neighboring psixels there are three types of kernels for morphological operations: 1) cross, 2) rectangular and 3) ellipse [10]. Appropriately, we extract required information from eye image in this work employing only these two

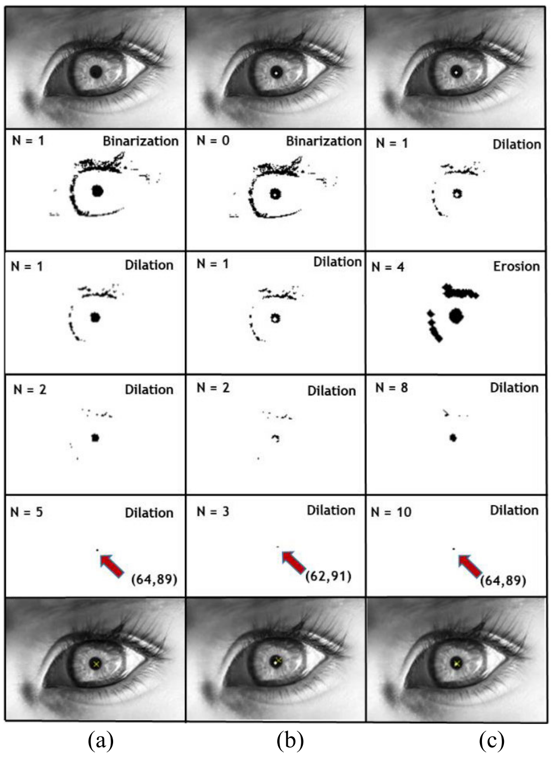


FIGURE 2. PC detection process in the propose ETS. (a) without glint utilizing dilation, (b) with glint utilizing dilation only, (c) with glint utilizing dilation and erosion.

operations as descried in this section. We utilize cross kernel for dilation and rectangular kernel for erosion.

B. PC DETECTION

By performing dilation operation number of black pixels in the image decreases after each iteration which results in removing the small black regions other than pupil. As long as the number of remaining pixels is more than a predefined value, this process is repeated. Then, PC location is represented with the location of the remaining pixels (Fig. 2(a)). Corneal glint reflections cause holes in the image which may lead to wrong PC location (Fig. 2(b)). In order to avoid this error, the holes must be filled prior to series of dilation operations. For this purpose, we perform erosion for two iterations. It should be noted that we perform erosion after one iteration of dilation to ensure removal of unwanted pixels and prevent expanding them. After filling the holes, dilation operation series continues as far as the number of remaining pixels falls below the predefined value which is 10 in this work (Fig. 2(c)).

C. GC DETECTION

In order to track gaze of the user, in addition to location of the PC, GC location is required to establish a mapping between PC-GC vector and screen [11]. In this work we detect GC location with the same algorithm as of PC detection. However, as the corneal glint is the brightest feature in image in contrast with the pupil, we need to perform inverse

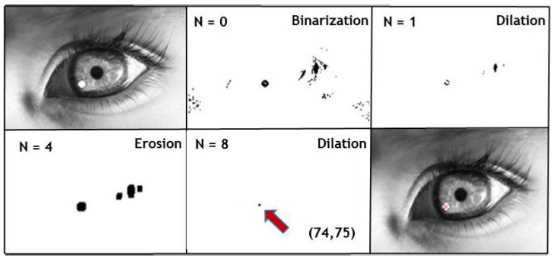


FIGURE 3. GC detection in the proposed ETS.

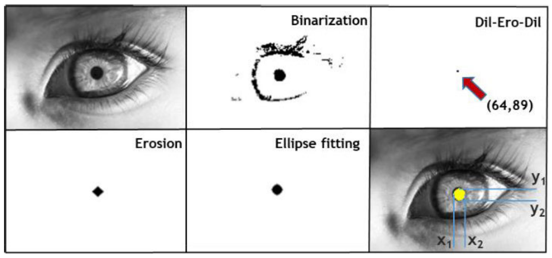


FIGURE 4. Pupil size detection in the proposed ETS.

binarization on image prior to morphological operations. Again, by performing an iteration of dilation at the beginning undesired pixels are diminished. Subsequent to two iterations of erosion followed by the first dilation, GC location is detected through a series of dilation operations with the same stop criterion as of PC detection as shown in Fig. 3. It should be pointed out that for both PC and GC detection if more than one pixel remains, the rightmost one in the lowest row is considered as the output. This rule and number of required dilation and erosion operations were obtained heuristically.

D. PUPIL SIZE DETECTION

Another function that our ETS using FPGA can provide is detecting the size of pupil which is crucial in some applications such as medical diagnosis because diameter of the pupil varies in reaction to illumination. The most commonly used approach of detecting pupil size is finding the largest contour in the image with assumption that it is the pupil and fitting an ellipse or a circle to that [9], [12], [13], [14]. However, an incorrect contour may be mistakenly identified as the pupil. In this work pupil size is detected exploiting the same principle as PC and GC detection (i.e. morphological operations). Since image features are removed by dilation operation, after finding the location of pupil center, pupil shape can be retrieved by processing inversely (expanding the remained black pixel). For this purpose, in the number of difference between number of dilation and erosion cycles we perform expand operation on remaining black pixel. It should be highlighted that we utilize a cross kernel for this function to obtain a diamond that can be fitted to an ellipse as shown in Fig. 4. Then, using 5 obtained points (center of pupil, X1, X2, Y1 and Y2) an ellipse is fitted to pupil the diameters of which provide the size of pupil.

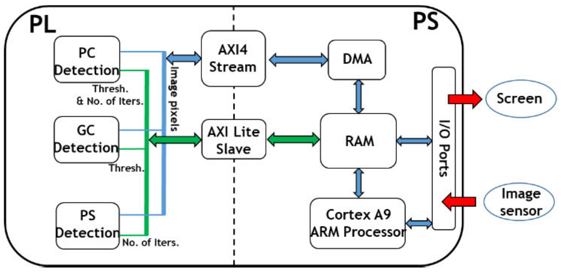


FIGURE 5. Conceptual FPGA implementation of the proposed ETS.

```
 1  typedef ap_axis <8, 2, 5, 6> pixel_type;
 2  typedef hls::stream <pixel_type> AXI_STREAM;
 3  void dil_ero (AXI_STREAM &in_stream, AXI_STREAM
 4  &out_stream){
 5  #pragma HLS INTERFACE axis port=in_stream
 6  #pragma HLS INTERFACE axis port=out_stream
 7  pixel_type input_pixel, output_pixel;
 8  unsigned char INPUT_IMAGE[IMAGE_HEIGHT][IMAGE_WIDTH];
 9  unsigned char OUTPUT_IMAGE[IMAGE_HEIGHT][IMAGE_WIDTH];
10  unsigned char kernel [3][3] = {
11  {0,1,0},
12  {1,1,1},
13  {0,1,0}};
14  int kCenterX = KERNEL_SIZE / 2;
15  int kCenterY = KERNEL_SIZE / 2;
16  for (int i = 0; i < IMAGE_HEIGHT; i++){
17      for (int j = 0; j < IMAGE_WIDTH; j++){
18          input_pixel = in_stream.read();
19          INPUT_IMAGE [i][j] = input_pixel.data;
20          for (int k = 0; k < KERNEL_SIZE; k++){
21              int kk = KERNEL_SIZE - 1 - k;
22              for (int l = 0; l < KERNEL_SIZE; l++){
23                  int ll = KERNEL_SIZE - 1 - l;
24                  int ii = i + (k - kCenterY);
25                  int jj = j + (l - kCenterX);
26                  if (ii >= 0 && ii < IMAGE_HEIGHT
27                  && jj >= 0 && jj < IMAGE_WIDTH){
28                      OUTPUT_IMAGE [i][j] +=
29                      INPUT_IMAGE[ii][jj] * kernel[k][l];
30                  }
31              }
32          }
33          output_pixel.data = OUTPUT_IMAGE;
34          output_pixel.keep = input_pixel.keep;
35          output_pixel.strb = input_pixel.strb;
36          output_pixel.user = input_pixel.user;
37          output_pixel.last = input_pixel.last;
38          output_pixel.id   = input_pixel.id;
39          output_pixel.dest = input_pixel.dest;
40          out_image.write(output_pixel);
41      }
42  }
43  }
```

FIGURE 6. HLS code of a naive implementation of dilation and erosion with 3 × 3 cross-shaped kernel.

III. PROPOSED ETS DESIGN METHODOLOGY

A. SYSTEM CONFIGURATION

Fig. 5 depicts conceptual FPGA implementation of the proposed ETS. In this work FPGA acts as central processing unit of the system. First, it receives data from image sensor through I/O ports connected to programming system (PS) of the chip. The captured images are stored in 512 MB DDR3 SDRAM with 16-bit bus which can be connected to programmable logic (PL) through AXI interface. Then, to accelerate image processing, data is sent to PL to be processed by functions implemented on high-speed digital units with parallel signal processing and pipelining ability.

For each function in the system we have defined a custom IP connected to corresponding AXI DMA. Since each AXI DMA has access to memory separately, and each custom IP utilize PL hardware independently, all functions can be performed simultaneously without any disturbance. To have a perfect performance 8-bit image pixels are transmitted to PL through AXI4-Stream interface that is suitable for transferring image and video because of capability of handling size of the data being transferred and near real-time operation. As can be seen in Fig. 5, in order to fetch data to be sent to PL from memory, and storing received data from PL in the memory, AXI Direct Memory Access (AXI DMA) provides a FIFO-based channel between memory and AXI4-Stream interface. To obtain appropriated information from eye image, morphological operations need to be performed repeatedly on each image. Each function is equipped with block RAM (BRAM) units of FPGA to receive image pixels only once and use them for several times without the need of continuous PL-PS connection. Hence, by excluding required time for continuous transferring data between memory and functions during operation results in lower processing time compared to software-based systems.

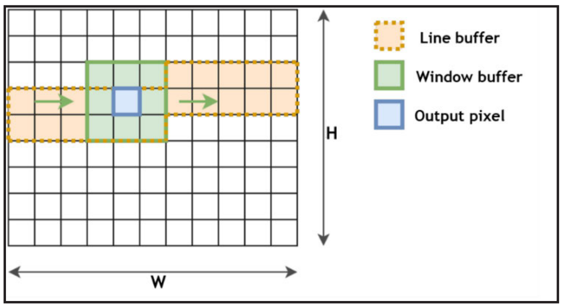
The appropriate threshold values are fed to PC and GC detection functions through AXI Lite Slave interface for image binarization. Also, pupil size detection function receives required number of iterations for erosion from PC detection function through AXI DMA using AXI Lite Slave interface. Finally, PS receives processed data from PL and generates appropriate output that can be sent to screen. In Fig. 5 AXI Lite slave, AXI4-Stream and I/O signals are represented by green, blue and red lines, respectively.

It should be highlighted that even though in this work we exploit PS memory to store captured images, in a more efficient approach for real-time operation in practical condition, functions can receive pixel values directly from image sensor connected to PL of the FPGA. Consequently, by detaching the memory of FPGA chip we can achieve higher rate of image processing with in-memory processing ability of the proposed functions.

B. CUSTOM IP FOR IMAGE PROCESSING

For each function of the system a custom IP has been implemented using Xilinx Vivado HLS 2019.2 in C++. Since fundamentals of each function is based on dilation and erosion, we describe implementing these operations systematically in this section. As mentioned in the previous section, all functions of this work can be obtained by different iterations of these operations. Dilation and erosion are morphological operations based on sliding window convolution between input image and kernel. A naive implementation of this process is shown in Fig. 6. The lines 1-7 and 8-9 define AXI4-Stream interface and 2D memory array on PL, respectively. Moreover, the purpose of the lines 33-40 is to transmit generated output pixels to PS as a stream. The two outer loops iterate over every pixel of input image and the two inner loops iterate over the values within kernel.

It is worth mentioning that owing to pattern of read/write pixels from/to memory in this method, only the inner loop can be pipelined which limits optimizing the hardware design by



(a)

```
1    for (int i = 0; i < IMAGE_HEIGHT; i++){
2       for (int j = 0; j < IMAGE_WIDTH; j++){
3          LineBuffer[0][j] = LineBuffer[1][j];
4          LineBuffer[1][j] = LineBuffer[2][j];
5          LineBuffer[2][j] = INPUT_IMAGE[i][j];
6
7          WindowBuffer[0][0] = LineBuffer[0][j];
8          WindowBuffer[1][0] = LineBuffer[1][j];
9          WindowBuffer[2][0] = LineBuffer[2][j];
10
11         for (int k = 0; k < 3; k++){
12            WindowBuffer[k][2] = WindowBuffer[k][1];
13            WindowBuffer[k][1] = WindowBuffer[k][0];
14         }
15         unsigned char minval = 0;
16         for (int k = 0; k < 3; k++){
17            for (int l = 0; l < 3; l++){
18               if (kernel [k][l] == 1){
19                  minval |= WindowBuffer [k][l];
20               }
21            }
22         }
23         OUTPUT_IMAGE [i][j] = minval;
24      }
25   }
```

(b)

FIGURE 7. **(a) Fundamentals of line buffer and window buffer, (b) HLS code for an efficient implementation of dilation and erosion using line buffer and window buffer.**

pipelining. As a consequence, the bottleneck of this method is limited number of pixels being read/write simultaneously per clock cycle. For example, when we use a 3 × 3 kernel, 9 input pixels have to be read for generating every single output pixel. The Vivado HLS synthesis report shows that in this method calculating only one output pixel takes 5 cycles because each pixel should be accessed several times for calculating different output pixels. It becomes even worse for large kernel sizes. However, since the Vivado tool calculates only the runtime of processing unit excluding inter-unit data transmission time in timing analysis, the estimated time for processing images can be worse in practical condition.

Accordingly, we employ a more efficient method so that each pixel is accessed only once for the whole process. Line buffer and window buffer are two cooperating memory structures that allows to achieve this efficiency. The concept of this method and HLS code implementation of that are illustrated in Fig. 7(a) and (b), respectively. The AXI4-Stream interface and memory definition is the same as of Fig. 6. A line buffer is a 2D memory array with the size of [kernel size] × [image width] ([3] × [176] in this work) and acts as a cache to store input image pixels being processed. The window buffer is an array of registers with the same size as kernel (3 × 3 in this work) that retains the pixel values within kernel in each clock cycle so that they can be accessed at the same time. First, the line buffer loads input image pixels one by one (Lines 3-5). Once it gets adequate elements to fill the rightmost column of the window buffer it shifts vertically at every iteration (lines 7-9). While the line buffer is shifting, elements from the topmost line are discarded one by one. Correspondingly, on each clock cycle only one new pixel needs to be loaded from input stream to the line buffer. After filling the window buffer, at each iteration by performing dilation or erosion operation, the new pixel value is calculated and sent to the output stream. At the next iteration window buffer discards the leftmost column and loads the new column from the line buffer on the right (lines 11-14). While the lines 15-22 of the code illustrates dilation operation, if 0 in line 15 and | (logical or operation) in line 19 are replaced with 255 and & (logical and operation), respectively, the code performs erosion operation.

Utilizing *pipeline* pragma in the code, during each iteration only one read/write from/to the external data stream occurs. Interestingly, benefiting from this technique the optimal throughput of one output pixel per clock cycle can be achieved [15], [16]. It should be pointed out that it is necessary to halt the process until the line buffer gets sufficient elements to produce one output element right from the first iteration. Thus, before running the process on input stream, the line buffer needs to be initialized with a suitable value (0 or 1 depending on image background) otherwise some data in the output image would be missing.

While it takes several iterations to obtain appropriated information from eye images, benefiting from pipelining in the proposed design, different operations are performed simultaneously to achieve lower processing time. For instance, once a pixel is generated at the first iteration of dilation, it is fed to line buffer of the next step immediately for erosion. Moreover, during the operation number of black pixels (pixels with the value of 0) are counted and if it falls below the predefined value the algorithm stops and the result is sent to the processor.

IV. RESULTS AND DISCUSSION

A. TEST SETUP

The proposed ETS was implemented on Xilinx PYNQ Z1 FPGA board. Xilinx Vivado 2019.2 development kit was used to configure hardware of the board. Then, to run the system and obtain the results it was connected to a personal computer and controlled via Jupyter notebook. Eye images were captured using a conventional camera. Nevertheless, the system can be linked with an image sensor as well. Due to lack of IR imaging equipment corneal glints were generated artificially. But, it doesn't discredit the results because the size and brightness of them were set similar to the real case. A conceptual test setup of the proposed ETS is shown in Fig. 8. The utilized FPGA resources has been listed in Table 1. The system operates at default FPGA clock frequency of 100 MHz. Among the used resource, all the

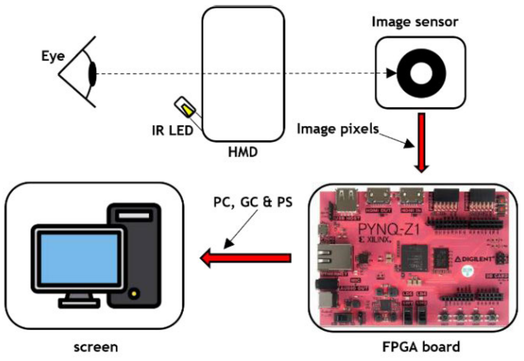


FIGURE 8. The test setup of the proposed ETS.

TABLE 1. The proposed ETS FPGA resource usage.

Resource	Utilization	Available	Utilization %
LUT	9584	53200	18.02
LUTRAM	532	17400	3.06
FF	11750	106400	11.04
BRAM	40	140	28.57
DSP	11	220	5.00

BRAM and DSP, 75.7% of FFs and 71.7% of LUTs were used solely for functions. The AXI interconnections use 24% of FFs, while the processor system reset module utilizes 0.3%. Additionally, 28.13% of LUTs are used for AXI interconnections, and 0.17% are dedicated to the processor system reset module.

B. EXPERIMENTAL RESULTS

We verify the proposed design by inducing several tests on 176 × 120 eye images with different scenarios. First, to demonstrate robustness of the algorithm, we examined the system performance with different types of eyes. As can be seen in Fig. 9, the proposed ETS can detect PC, GC and pupil size properly regardless of the eye shape and color. It is important to note that as the images were captured in divergent conditions, processing of each image might demand different binarization threshold value and number of iterations for detecting appropriated information. These values have been mentioned in Fig. 9 for each image. It can be inferred from the results that as the eye is darker, lower threshold value and higher number of iterations are required for PC detection because more pixels have to be removed. Later, we captured eye images looking to different directions with fixed light condition and camera distance from the eye. In this case, threshold value is identical for processing different images. Fig. 10 shows eye images and detected PC. To evaluate accuracy of the system we obtained these parameters using software simulation along with the proposed FPGA implemented ETS. Detected PC from implemented system and software simulation are represented by yellow × and blue ■, respectively. Herein, we define the error as the discrepancy between measured and simulation results. As can be seen, while the images are occluded by corneal reflection glints, eyelids and eyelashes, the maximum error is ±1 pixel in vertical and horizontal directions. In other words, the maximum diagonal error for all gaze directions is $\pm\sqrt{2}$ pixel which occurs when in both vertical and horizontal directions there is an error of one pixel.

As explained in Section II-A, threshold value is a determining parameter in digital ETSs employing morphological operations. Although we set the threshold value during initial configuration of the system, but due to fluctuation in core voltage and temperature of FPGA this value may vary in practical condition. So, we evaluate our ETS performance over threshold deviation by processing images with random threshold values in the range of ± 100% of optimal value (i.e. 5 for images in Fig. 10) for 1000 epochs. While number of pixels in original image increases with the growth of threshold, number of remaining pixels in processed image is constant for a large portion of the range. Hence, the vertical and horizontal error remain in the range of ±1 pixel for the entire threshold variation range. Therefore, it can be implied that mismatches and environmental conditions do not degrade stability and performance of the system.

We implemented the proposed ETS processing units on PL of the Xilinx FPGA to accelerate functions taking advantage of instructions pipelining and parallel signal processing utilizing fast digital modules. As described in Section III-B, a single output pixel is generated at each clock cycle. In consequence, runtime of the algorithm is calculated by the number of image pixels multiplied by clock period. Particularly, the runtime of processing images with the resolution of 176 × 120 in this work is 211.2 μs using 100 MHz clock frequency. However, taking the required time for receiving data from PS, storing it in PL local memory and sending the processed data to PS into account the total processing time of this work is 1.8 ms. Therefore, it can be said that the system is able to process 500 images per second (i.e. 500 fps). In order to provide a comparative analysis of the obtained processing time with software implementation and underline benefits of hardware acceleration, we implemented the image processing functions on PS of the FPGA using python OpenCV library. While instructions on PL can be performed concurrently and pipelined, they are run sequentially on PS.

Hence, with a considerable increase in the processing time, it takes 25.2 ms to accomplish image processing using PS of the FPGA. As a result, it can be said that hardware acceleration factor of 14 was achieved in this work that is defined as the ratio of processing time using PS to processing time using PL. The total power consumption of FPGA chip is 247 mW with 3.3 V power supply. According to FPGA board control panel, the static and dynamic power consumptions are 141 mW and 106 mW, respectively.

C. DISCUSSION

Table 2 summarizes the proposed ETS specifications and presents a comparison with the state-of-the-art eye trackers. From the Table 2, it can be deduced that there is a noticeable gap between required hardware for implementing an ETS

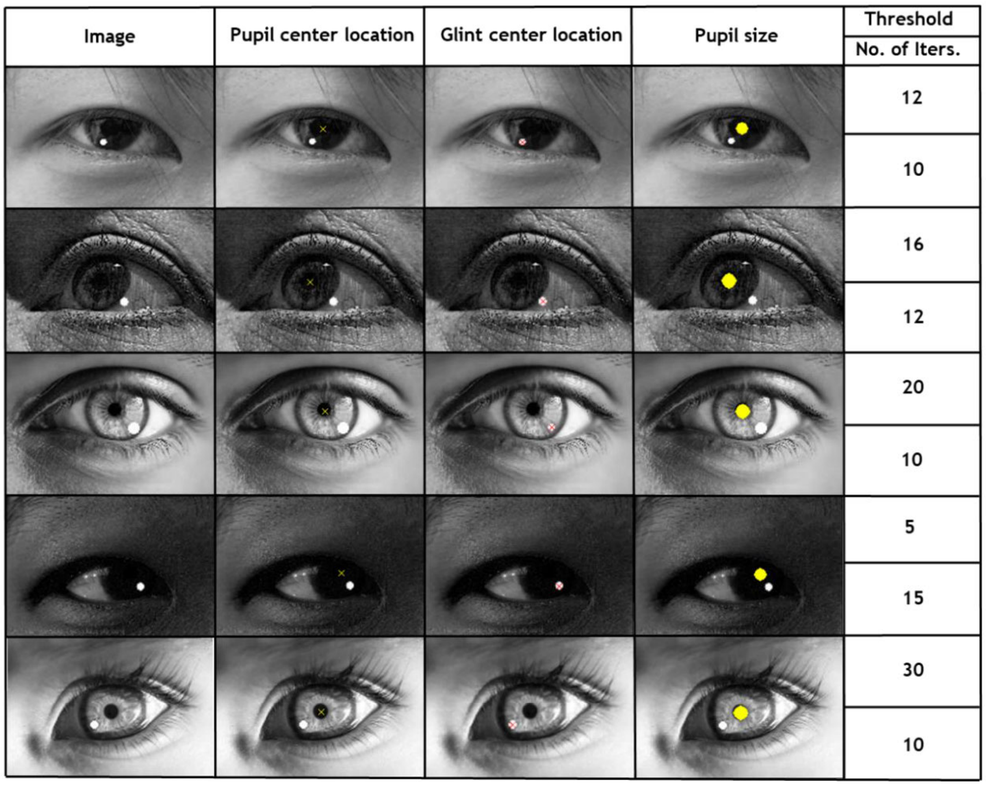


FIGURE 9. The test results of the proposed ETS with different types of eyes.

using FPGA and software-based system disregarding the performance. Additionally, due to high integratability of FPGA components in contrast to required system for software-based eye trackers, FPGA-based works consume drastically less power compared to those systems. Therefore, in applications where power dissipation and system complexity are critical, FPGA-implemented systems are highly preferred.

According to Table 2, our design demonstrates the most resource efficiency among the other works. While [3] captures images at a resolution of 800 × 600, they only process 100 × 100 eye images for pupil detection. Surprisingly, their system utilizes 160 MB of memory, which is over 900 times larger than the memory usage in our work. In addition, ETSs in [6], [8], [12], and [13] process images at double the size, and in [5] at quadruple the size compared to our work. However, considering the relationship between the difference in input image resolution and the disparity in resource usage while they only provide basic functions such as detecting eye location, pupil and gaze direction but our design performs three functions in parallel providing information with high precision at considerably higher frame rate implies superiority of this work over them in terms of resource efficiency.

Owing to high-speed processing algorithm and FPGA hardware acceleration this work demonstrates a remarkable improvement in frame rate among FPGA-based ETSs which allows incorporating our design in near real-time applications. The processing time mentioned in Table 2 encompasses both runtime of the algorithm and consumed

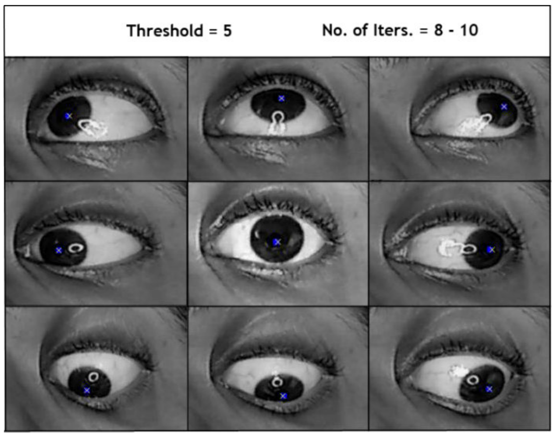


FIGURE 10. The test results of the proposed ETS with constant light condition and user-camera distance.

time for transferring data from memory to processing unit and vice versa. Therefore, to achieve this improvement in this work first the need for continuous inter-unit transferring data in FPGA has been eliminated utilizing PL local memory and data transmission is carried out through ultra-fast AXI4 Stream interface between memory and processing units within the chip only once. Furthermore, employing line buffer facilitates image processing by pipelining and efficient pattern of accessing memory which results in lower runtime. As one output pixel is produced during each clock cycle, by increasing the clock frequency the processing time can

TABLE 2. The proposed ETS performance summary and comparison with state-of-the-art eye trackers.

	[3]	[13]	[12]	[8]	[6]	[5]	**This work**
Obtained info.	PC & GC	Iris	Pupil	Pupil	Eye loc.	Gaze dir.	**PC, GC and pupil size**
Impl. platform	Software	FPGA	FPGA	FPGA	FPGA	FPGA	**FPGA**
Algorithm	Ellipse Fitting	Morph. Oper.	Morph. Oper.	Morph. Oper.	Filtering	CNN	**Morph. Oper.**
Image size	100×100	320×240	320×240	320×240	320×240	640×480	**176×120**
Hardware	Intel i7 7700k	Stratix IV EP4SGX530	Zynq xc7z020	Cyclone IV EP4CE115	Cyclone II 2C70	Cyclone IV EP4CE115	**Pynq Z1 (Zynq xc7z020)**
Frame rate (fps)	723	210	N/A	N/A	8	80	**500**
Processing time	1.38 ms	5.15 ms	6.25 ms[1]	6.5 ms	118 ms	0.52 µs[2]	**1.8 ms**
Memory usage	160 MB	2048 KB	440 KB	1641 KB	52 KB	80 KB	**180 KB**
Processor frequency (MHz)	4200[3]	215	203	111	27	100	**100**
Power (mW)	N/A	N/A	N/A	N/A	375	409	**247**
Max. error (vertical or horizontal)	0.67[4]	N/A	N/A	N/A	±1	N/A	**±1**

[1]Runtime of the algorithm.
[2]Recognition time of CNN.
[3]The system uses 40 % of all CPUs during training and 13%while capturing.
[4]Standard deviation for 4500 samples.

be decreased. Due to PLL limitation the maximum actual clock frequency is 525 MHz on the Xilinx PYNQ Z1 FPGA board. However, we employed the default clock frequency of 100 MHz in this work to highlight the influence of hardware acceleration and perfect design methodology on the achieved efficiency and frame rate enhancement. Though, obviously by employing the maximum clock frequency, a lower runtime and accordingly a higher frame rate can be achieved while using considerably lower hardware resources compared to other works. In practical condition, we can set the clock frequency at maximum for the best performance. Although the ETS in [5] shows the least processing time, but it should be noted that the mentioned time is only recognition time of the CNN. Also, the processing time associated with [12] is solely the runtime of algorithm, calculated by multiplying the clock period by the required number of clock cycles that is still considerably higher than the runtime of our proposed ETS, even when using clocks at double the rate. Thus, we can see that when the total execution time including the time of transferring data between different parts of the system is taken into account, the output frame rate which is obtained considering the whole processing time is remarkably less than our work. In [6], cross correlation with a predefined pattern is calculated through an image. Despite of our design, each pixel needs to be read several times which increase the processing time substantially. Nonetheless, the runtime can be improved by exploiting a higher clock frequency. Yet, the meaningful processing time in [6] underlines the need for optimizing implementation. Taking advantage of modular design and parallel-pipelined implementation, the ETS in [13] shows the best processing time among other FPGA-based works in Table 2. In order to detect iris, the system generates multiple circles as iris candidates. Then, to select the best circle, an iterative algorithm is used which leads to increase in resource usage notably. As can be seen in Table 2, [13] exhibits the highest memory usage among other FPGA-based works. To circumvent iterative optimization algorithm, the ETS proposed in [8] employ a Gaussian filter and adaptive thresholding prior to processing image to reduce imperfections caused by background variations and poor contrast. However, at the price of a slight decrease in memory usage, this work fails to locate the pupil properly in the presence of dense eyelashes and eyebrows. While undesired pixels such as eyelashes and eyebrows at the first dilation operation are removed or shrinked in our proposed ETS. Furthermore, pixel-wise image processing ensures diminishing non-connected components like eyelashes and eyebrows effectively after a few iterations which results in high accuracy as can be seen in Fig. 2. Finally, it should be highlighted that our proposed ETS provides more functions than other works within the mentioned runtime in parallel while consuming less power. To sum up, the proposed design surpasses software implemented counterparts in terms of power consumption, processing time and efficiency in any condition. Also, when an embedded eye tracker with flexibility and multi-functionality is of interest in real-time and accurate applications such as medical diagnosis and HCI, the proposed low-power and high-speed ETS can be preferred to FPGA-based counterparts.

V. CONCLUSION

A high-speed FPGA-based ETS employing morphological operations (dilation and erosion) has been presented in this paper. The proposed multi-function ETS can obtain all of the application required information such as PC and GC locations for gaze tracking along with the pupil size in parallel. Taking advantage of hardware acceleration on PL of the FPGA using line buffers and AXI4 Stream interface, the processing time has been improved by the factor of 14 that allows the proposed ETS to operate up to 500 fps at 100 MHz clock frequency. Experimental results show that the proposed ETS can obtain appropriated information within ± 1 vertical or horizontal pixel error with 247 mW power consumption. Also, the algorithm is robust to FPGA chip voltage and temperature fluctuation, threshold deviation and unwanted corneal reflection glint. The results imply superiority of this work over software implemented eye trackers and competency of that for real-time and precise applications such as HCI and medical diagnosis. Finally, developing this work by designing another custom IP to perform sampling video frames for real-time video processing, and constructing PC-GC vector mapping function for gaze estimation are suggested for future works.

REFERENCES

[1] J. R. Pauszek, "An introduction to eye tracking in human factors healthcare research and medical device testing," *Hum. Factors Healthcare*, vol. 3, Jun. 2023, Art. no. 100031.

[2] T. Duchowski, *Eye Tracking Methodology: Theory and Practice*, 3rd ed. Cham, Switzerland: Springer, 2017.

[3] B. Hosp, S. Eivazi, M. Maurer, W. Fuhl, D. Geisler, and E. Kasneci, "RemoteEye: An open-source high-speed remote eye tracker," *Behav. Res. Methods*, vol. 52, no. 3, pp. 1387–1401, Mar. 2020.

[4] K. Donuk, A. Ari, and D. Hanbay, "A CNN based real-time eye tracker for Web mining applications," *Multimedia Tools Appl.*, vol. 81, no. 27, pp. 39103–39120, Apr. 2022.

[5] Y.-H. Yu, Y.-S. Ting, N. Kwok, and N. M. Mayer, "High-speed gaze detection using a single FPGA for driver assistance systems," *J. Real-Time Image Process.*, vol. 18, no. 3, pp. 681–690, Aug. 2020.

[6] T. Ando, V. G. Moshnyaga, and K. Hashimoto, "A low-power FPGA implementation of eye tracking," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1573–1576.

[7] D. Kim and G. Han, "A 200 μ S processing time smart image sensor for an eye tracker using pixel-level analog image processing," *IEEE J. Solid-State Circuits*, vol. 44, no. 9, pp. 2581–2590, Sep. 2009.

[8] T. M. Khan, D. G. Bailey, M. A. U. Khan, and Y. Kong, "Real-time iris segmentation and its implementation on FPGA," *J. Real-Time Image Process.*, vol. 17, no. 5, pp. 1089–1102, Oct. 2020.

[9] S. Navaneethan and N. Nandhagopal, "RE-PUPIL: Resource efficient pupil detection system using the technique of average black pixel density," *Sādhanā*, vol. 46, no. 3, Jun. 2021, Art. no. 144.

[10] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision With the OpenCV Library*, 1st ed. Springfield, MO, USA: O'Reilly, Sep. 2008.

[11] J. Liu, J. Chi, H. Yang, and X. Yin, "In the eye of the beholder: A survey of gaze tracking techniques," *Pattern Recognit.*, vol. 132, Dec. 2022, Art. no. 108944.

[12] V. Kumar, A. Asati, and A. Gupta, "RE-PUPIL: Resource efficient pupil detection system using the technique of average black pixel density," *IET Image Proc.*, vol. 12, no. 10, pp. 1753–1761, Oct. 2018.

[13] H. T. Ngo, R. N. Rakvic, R. P. Broussard, and R. W. Ives, "Resource-aware architecture design and implementation of Hough transform for a real-time iris boundary detection system," *IEEE Trans. Consum. Electron.*, vol. 60, no. 3, pp. 485–492, Aug. 2014.

[14] J. Avey, P. Jones, and J. Zambreno, "An FPGA-based hardware accelerator for iris segmentation," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Dec. 2018, pp. 1–8.

[15] B. Van Hoorick, "FPGA-based simultaneous localization mapping (SLAM) using high-level synthesis," M.S. dissertation, Dept. Telecommun. Inf. Process., Ghent Univ., Brussels, Belgium, May 2019.

[16] J. Matai, D. Richmond, D. Lee, and R. Kastner, "Enabling FPGAs for the masses," 2014, *arXiv:1408.5870*.

AHMAD MOURI ZADEH KHAKI was born in Ahvaz, Iran, in 1989. He received the M.Sc. and Ph.D. degrees from Islamic Azad University (IAU), Iran, in 2013 and 2020, respectively. From 2014 to 2020, he was teaching with IAU. He is currently a Postdoctoral Researcher with the Gwangju Institute of Science and Technology, Gwangju, South Korea. He is working on human–computer interface systems and hardware implementation of neural networks. His research interests include analog and mixed-signal integrated circuits, optical and RF communication, deep learning, and FPGAs.

SANGHYEOK YANG received the Ph.D. degree in electrical engineering from Korea University, Seoul, South Korea, in 2012. He is currently a Senior Research Engineer with Hyundai Motor Group, South Korea. His research interests include analog integrated circuits, neuromorphic devices, and optical sensors for gas and distance measurement.

HYUNSOO KIM received the Ph.D. degree in electrical engineering from The Pennsylvania State University, in 2010. His Ph.D. research involved the design, modeling, and micro-fabrication of piezoelectric thin film devices. He has joined Hyundai Motor Group, as a Senior Researcher, where his research group has been focused on micro-electromechanical systems (MEMS) based sensors and sensor systems for automotive applications, since 2012. His current research interests include nanofabrication, nanoelectromechanical systems, self-powered devices, optical devices, and neuromorphic devices.

ANKUR SINGH was born in Gorakhpur, India, in 1999. He received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Information Technology Guwahati, Guwahati, India, in 2021. He is currently pursuing the master's degree with the Department of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju, South Korea. His research interests include neuromorphic and in-memory computing.

BYUNG-GEUN LEE (Member, IEEE) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2000, and the M.S. and Ph.D. degrees in electrical and computer engineering from The University of Texas at Austin, in 2004 and 2007, respectively. From 2008 to 2010, he was a Senior Design Engineer with Qualcomm Inc., San Diego, CA, USA, where he had been involved in the development of various mixed-signal ICs. Since 2010, he has been with the Gwangju Institute of Science and Technology (GIST). He is currently a Professor with the Department of Electrical Engineering and Computer Science. His research interests include high-speed data converters, CMOS image sensors, and neuromorphic system design.