International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 16, No.1, (Jul-24)

http://dx.doi.org/10.12785/ijcds/160131

An FPGA Implementation of Basic Video Processing and Timing Analysis for Real-Time Application

Marwan Abdulkhaleq Al-yoonus[1] and Saad Ahmed Al-kazzaz[2]

[1]*Electrical Engineering Department, College of Engineering, University of Mosul, Iraq*
[2]*Mechatronics Engineering Department, College of Engineering, University of Mosul, Iraq*

Received 14 Oct. 2023, Revised 20 Mar. 2024, Accepted 6 Apr. 2024, Published 1 Jul. 2024

Abstract: For real-time video processing, the analysis time is a big challenge for researchers. Since digital images from cameras or any image sources can be quite large, it is common practice for researchers to divide these large images into smaller sub-images. The present study proposes a subsystem module to read and display the region of interest (ROI) of real-time video signals for static camera applications to prepare for background subtraction (BGS) algorithm operation. The proposed subsystem was developed using Verilog hardware description language (HDL), synthesized, and implemented in the ZYBO Z7-10 platform. An ROI background image of (360×360) resolution was selected to test the operation of the module in real time. The subsystem consists of five basic modules. Timing analysis was used to determine the real-time performance of the proposed subsystem. Multi-clock domain frequencies are used to manage the module operations, 445.5MHz, 222.75MHz, 148.5MHz, and 74.25MHz, which are six, three, two, and one-time pixel clock frequencies, respectively. These frequencies are chosen to perform five basic processing operations in real-time during the pixel period instant. Two strategies are selected to explain the effectiveness of choosing the trigger instant of the used clock signals on the system performance. The operation revealed that the latency of the proposed ROI reading subsystem was 13.468ns (one-pixel period), which matched the requirements for real-time applications.

Keywords: — Background subtraction, Clock domain, Real-time, Region of Interest, Verilog HDL.

1. INTRODUCTION

Field-programmable gate array (FPGA) designs are more easily reconfigured to any functionality than the application of specific integrated circuit (ASIC) designs as they possess hardware resources such as; dedicated block random-access memory (BRAM), multiplexers, and so forth [1]. They also enable flexible trade-offs facilitating the concurrent execution of multiple small heterogeneous tasks or optimizing a single low-latency or high-accuracy task by using more onboard resources for applications that do not fully occupy them [2].

From a technological standpoint, FPGAs are ideal for processing data in parallel at high speeds. Because of this, they allow for a high level of customization [3]. Their fast processing speeds, ability to be implemented in parallel, and ability to produce deterministic performance and latency make them excellent options for real-time detection algorithm implementation [4][5][6].

An area within an image known as a region of interest (ROI) allows the localized application of image-processing operations. This area is typically rectangular in shape. It is defined by Line and Column addresses [7][8]. The ROI or search domain might be diminished to the absolute minimum in order to reduce the processing time and effort of the object detection algorithm [9]. Selecting and reading the ROI to prepare for object detection algorithms like Background subtraction (BGS) is a fundamental step in many computer applications, such as intelligent video surveillance, medical evaluation, human-machine interaction, traffic monitoring, vandalism deterrence, and suspicious object detection systems.

Background subtraction (BGS) is an algorithm commonly used to detect moving objects. The frame differencing method which detects motion by calculating differences in the pixels of two adjacent frames is the simplest method of doing so [10][11][12]. A pixel is classified as a part of the foreground if its difference exceeds a specific threshold (TH), otherwise, it is a part of the background. The aforementioned threshold is determined by observation or experience [13].

Autonomous synthesis tools and hardware description languages (HDL) were first used in the design of large-scale digital systems circa 1990. For the purpose of describing digital electronic circuits, Verilog is standardized

E-mail address: marwanathy1972@uomosul.edu.iq, kazzazs60@uomosul.edu.iq

as IEEE 1364. At the RTL abstraction level, Verilog HDL is primarily utilized for design and verification purposes. An individual system is covered in a single file by a Verilog design. There is a module in the file that contains the system description. This module comprises the description of the behavior as well as the interface to the system (i.e., the inputs and outputs) [14][15][16].

In this work, the implementation of the digital circuit that makes up the ROI reading subsystem and performs basic video/image processing is done using Verilog HDL. The present study is organized as follows. The literature survey and the main points of our contributions are presented in Section 2. Section 3 presents a brief definition of the image ROI. The description of the proposed hardware is presented in Section 4. The hardware implementation of the proposed module and timing analysis are discussed in detail in Section 5. The results and discussion are found in section 6. Finally, the conclusion of this study is presented in Section 7.

2. LITERATURE SURVEY

The BGS algorithm for the ROI obtains a reference image to represent the background scene. This algorithm works correctly when a fixed camera is used [17]. A thorough analysis of a diagram is necessary to understand the operation of this algorithm in real-time surveillance systems, especially when accurate object detection is required. This section provides a summary of some extant studies in this field.

Sousa et al. [8] discussed a dynamic ROI control algorithm that was used to modify the exposure time, gain, and light of an LED source that was connected to a camera head. The suggested system was implemented using an FPGA. It was predicated on an ROI computed using the picture's histogram. The authors used nested multiplexers and simple bit shift processing in place of the divider and multiplier cores that were commonly used in the implementation of PID controllers.

Vidya et al. [12] used a Spartan 6 FPGA kit to develop an area of efficiency detection algorithm by decreasing the number of slice registers in face detection from 861 to 537 and the moving object detection from 365 to 320.

Wang et al. [2] demonstrated a smart camera design based on FPGA that allows streaming processing to accommodate the particular requirements of video surveillance applications. In order to lessen the smart camera's overall system load and enhance resource efficiency, the authors used the background subtraction algorithm to filter out static video frames that did not require processing.

Elisa et al. [18] gave a presentation on the creation and application of specific hardware IP modules for embedded vision systems that will be used to implement the background subtraction algorithm. A variety of quality parameters had been evaluated in order to carry out various algorithms. A low-cost FPGA (SPARTAN-3A) was used to implement five candidate algorithms (the more suitable for implementation in the mentioned device) for image resolutions ranging from QCIF (176 x144) to CIF (352 x 288) formats.

To shorten the design process's time-to-market, Cortes et al. [19] used Vivado HLS to implement widely used image processing algorithms built into Zynq SoC. They suggested that the use of image processing libraries like OpenCV, where software designers were in charge of the design, helped to cut down development time even more because they were already acquainted with them. However, there was a significant increase in the required FPGA resources when using these kinds of libraries.

A low-level processor that was meant to be a component of an algorithm for image analysis and interpretation was presented by Benetti et al. [20]. The low-level adaptive background subtraction that was used as the foundation of the architecture was implemented at the pixel level by utilizing analog fully parallel architecture. By using effective digital methods implemented on an FPGA, the binary image provided by the sensor was directly segmented.

Basic image processing operations had been implemented on an FPGA by Asha et al. [6]. The suggested methodologies involved using Verilog HDL to design and develop the controlling parameters for an image, such as brightness, threshold, and inversion. They used MATLAB to convert the BMP image file to a Hex file.

An alternative version of the BGS method based on independent component analysis (ICA) was proposed by Carrizosa et al. [21]. Four image sequences were examined for motion detection. The results from the implementation that used both the FPGA and the embedded processor of the SoC showed a decrease in runtime from 4326.60 to 125.81 milliseconds compared with others that used only embedded processors. The background estimation and picture capturing was not taken into account during the measurements.

Conti et al. [22] described the testing of two hardware implementations that use RGB and grayscale color spaces, respectively, and an analysis of the system reliability. They showed that processing an RGB sequence was about 6% slower than processing a grayscale sequence, and still adds very little overhead. The developed system had the capability to process data in real-time, ranging from (192×192) base resolution to (640×480), from a commercial Zenith camera at 15 frames per second.

Alejandro et al. [23] implemented real-time event-based filters and featured extractors with minimal resources. Three primary circuit components were used in the VHDL implementation of the filter: (a) Finite State Machines for the control unit, (b) an FPGA-embedded BRAM memory for the 2D array of timestamps, and (c) a counter for

time control. (the FSM made up of flip-flops, adders, comparators, and SRAMs).

The reversible contrast mapping (RCM) algorithm's flaws were examined and fixed by Das et al [24] for invisible watermarking. A gray-scaled video input was used to test the suggested corrected RCM algorithm. the proposed system makes use of parallelism and pipeline structure to attain high performance. The results of a comparative analysis between software and hardware implementations were also examined and briefly discussed.

Liu et al. [15] investigated several image scaling algorithms, weighed the benefits and drawbacks of each, and used a top-down design approach to create FPGA video scaling that was later verified to be accurate. To enlarge the video image, a fuzzy interpolation algorithm was employed. The image was improved through the use of a binary interpolation scaling algorithm, and the computational complexity was decreased through the creation of an interpolation factor table.

A hardware architecture was proposed by Sarkar et al. [25] to link an FPGA to a low-cost digital camera for real-time video processing and recording. Simple logic architectures have been used to implement the Transition Minimized Differential Signaling Encoder (Converter), the optimized Finite State Machine (Controllers), and several interfacing blocks. In order to lower the hardware consumption of various Color Plane Conversion blocks, shifters and adders were employed.

To our knowledge, there is a research area for timing analysis of video processing in real-time systems. In this paper, the main contributions can be listed as:

- An introduction of a novel model aimed at managing subsets of vast data streams generated daily by camera observations, addressing the challenge posed by the sheer volume of data.
- Emphasis on the necessity of efficient data handling, recognizing that a significant portion of acquired data may be irrelevant or redundant.
- A basic video timing analysis that uses a multi-domain clock signal is presented to control the reading procedure and the displaying of the ROI for an input video/image.
- Practical solution offered to enhance efficiency and effectiveness in handling massive volumes of video data in a real-time system.

3. IMAGE REGION-OF-INTEREST (ROI)

The area of an image frame in which the target object is present, as defined by a bounding box, constitutes the ROI. During performing vision tasks, especially in real-time, fewer pixels mean fewer clock cycles, which impacts the post-processing stage. A decrease in the pixel count lessens the computational load and frees up onboard resources and memory for use in other tasks. By using ROI technology, the camera's frame rate might be increased [8][26]. One of the common post-processing stages that depends on the size of the input video/image is the BGS algorithm. The following common equation describes the operation of the BGS algorithm:

$$output(BGS) = \begin{cases} 1 & \text{if } |\text{ I(i,j,t)-Ref(i,j) }| > TH \\ 0 & \text{if } |\text{ I(i,j,t)-Ref(i,j) }| < TH \end{cases} \quad (1)$$

where $I(i, j, t)$ and $Ref(i, j)$ were the pixels of the current input image and the reference image, respectively. The TH was a predetermined threshold. It can be estimated by training the system or via observation and experience [27].

In this study, a fixed ROI was selected with (360×360) size from the full-size (1280×720) resolution for timing analysis as shown in Figure 1. The importance of the suggested subsystem is that it will serve as an initial module to implement the BGS algorithm for the chosen ROI and speed up processing.

Figure 1. ROI size with respect to the input image.

4. SUBSYSTEM HARDWARE DISCRIPTION

Within the selected ROI, basic video/image processing was implemented using the Digilent Zybo Z7-10 device. In order to simulate memory write/read operations, the basic processing consists of converting the RGB input pixels to grayscale, adding and subtracting pixels one at a time using a constant byte stored in a single storage element (D-FF), and then returning the formatted pixels from grayscale to RGB before displaying them at the designated ROI.

To implement a hardware module that has different conditions (states), a finite state machine (FSM) is used for this purpose. From Figure 2, the wait conditions are active in States 1 and 2 until the condition of the positive edge of vertical synchronous (Vsync) and video active line happened, respectively to make a transfer from the current state to the next one (i.e. from state one to state two and so on). In State 3, the pixels/row counter counts to (360 pixels/row) and then jumps to wait for the end of the image's single line in State 4. In the detection of the negative edge of the end of the active line in State 4, the algorithm checks if the selected rows (360 rows in this case)

ended or not. A test condition is always performed at state 5 to determine the end of the ROI which is (360×360) pixels.

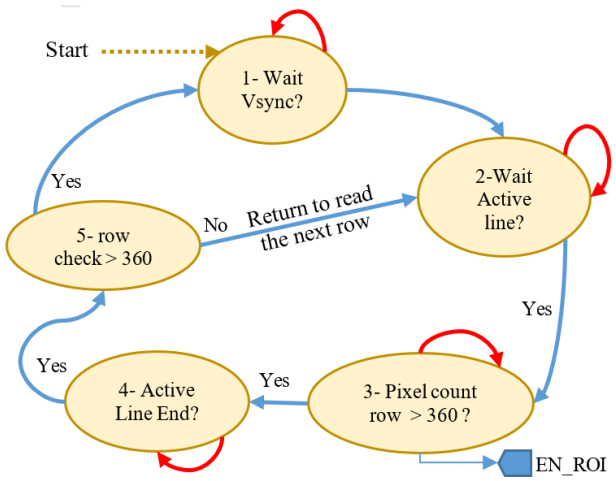

Figure 2. FSM diagram of the ROI image reading (360×360 pixels).

The overall layout indicated by a block diagram of the suggested subsystem is shown in Figure 3. Two sampling units, D-Flip-Flops (D-FF), one placed before the main processing module and the other after it, are used to ensure signal stability at the sampling time which is managed by the input pixel rate. The pixel clock frequency, declared by Clk-D, is used as the sampling frequency of both D-FFs. Five IP modules implemented using Verilog HDL are located in the center block, which is the main processing stage. This stage is operated by four clock frequency domains (Clk-D, 2×Clk-D, 3×Clk-D, and 6×Clk-D). The operation of the main processing stage is managed by multi-domain clock frequencies which are; 148.5MHZ, 222.75 MHZ, and 445.5MHZ, derived from the pixel clock frequency (74.25MHz) as shown in Figure 4.

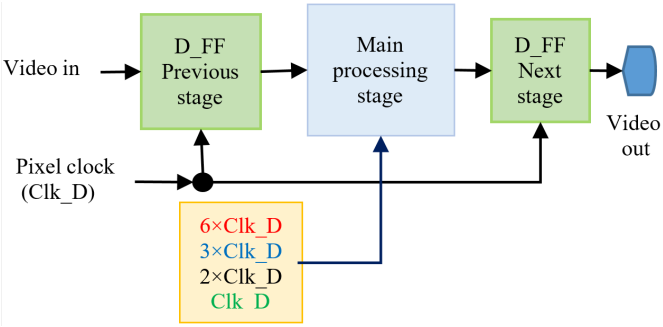

Figure 3. Framework of the proposed ROI reading module and the operating clock frequencies.

To describe the operation order that is used to read the ROI section of an input video/image, the flowchart shown in Figure 5 illustrates the principal actions of the reading control algorithm. The control (command) signals are denoted by the black lines, while the green lines point to the data (pixel) path. To verify the module operation, a

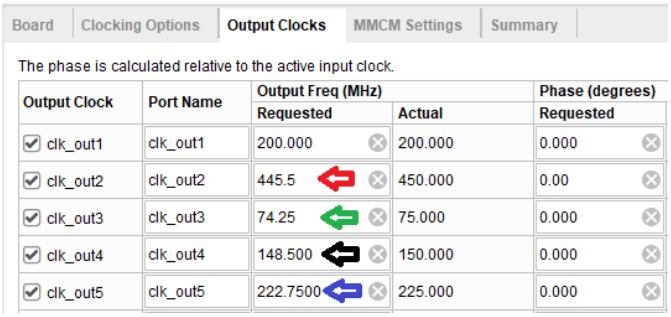

Figure 4. Clocking wizard IP block.

video/image source is taken from a Laptop HDMI port with a frame rate equal to 60 frames/sec.

5. HARDWARE IMPLEMENTATION

This section presents the implementation of the main processing stage using Verilog HDL with IP cores from the Vivado library to complete the basic video/image processing through HDMI ports of the ZYBO board. The complete subsystem processing was implemented only using the programmable logic (PL) part of the board.

The ROI reading Verilog code is described in the shown reading control algorithm. The five states that are used to implement the FSM (see Figure 2) are; Wait-Vsync, Wait-Active-line, Pixel-count-row, Wait-line-end, and row-check. Two counters have been used to read the ROI, one for pixels‘s row reading (resets every 360 pixels/row) and the other was used for the ROI row counting (resets after reading 360 rows/ROI).

The proposed algorithm which is implemented as an IP module block is shown in Figure 6. The Vsync and the Active-line inputs represent the vertical synchronous and the row active line respectively. The preselected ROI from the input video/image signal can be displayed or hidden using the output signal EN-ROI.

Figure 7 depicts the entire real-time video pass system with ROI reading circuit, where the red box contours the main processing stage. Figure 8 shows the basic IP modules that set up the main unit. The main processing stage applies the following five fundamental operations, carried out at the pixel level, to the input video/image stream during a single pixel period:

1- RGB to Grayscale (RGB2Gray) format conversion (24-bit to 8-bit),
2- Read a constant value (reference pixel) from a single memory cell (8-bit D-FF),
3- Add/Sub operation (input pixel ± reference pixel),
4- Grayscale to RGB (Gray2RGB) format conversion (8-bit to 24-bit), and
5- Generate the ROI-Enable signal for a video/image display with a size of (360×360) pixels.

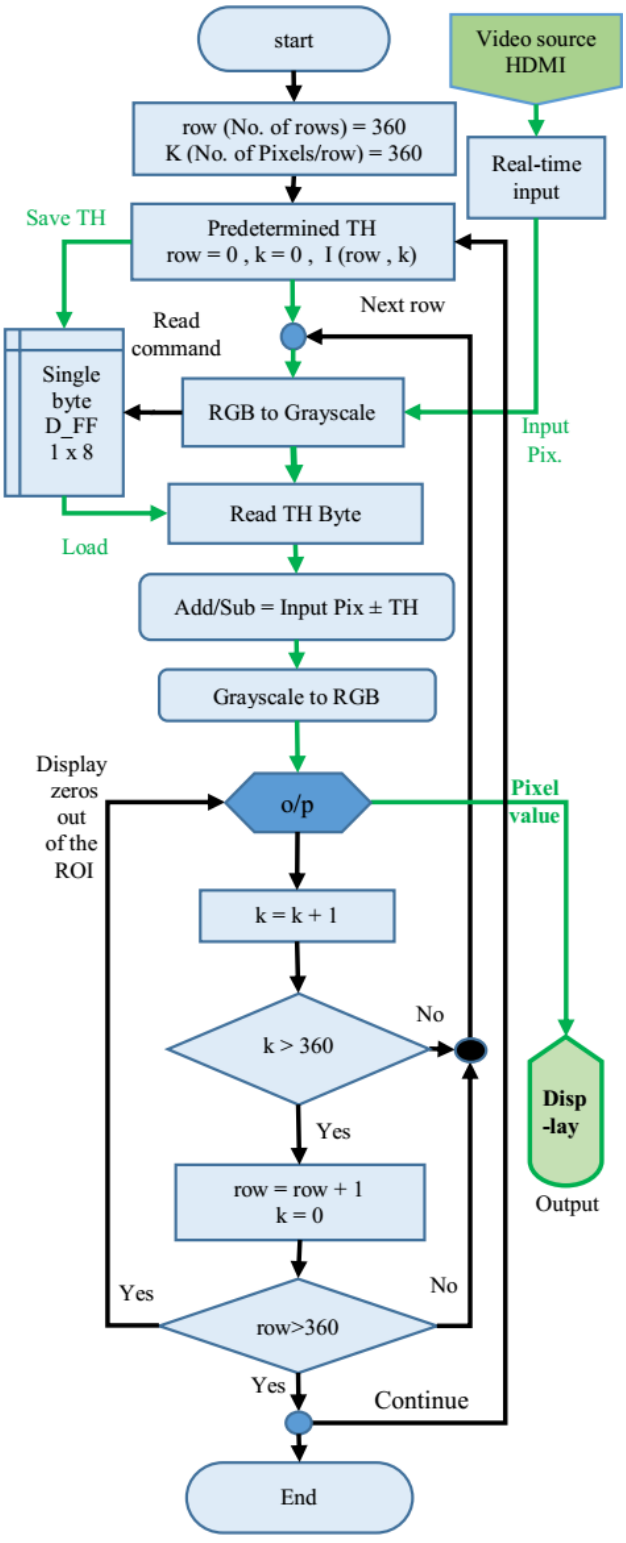

Figure 5. ROI reading algorithm (360×360), the green lines are for data path and the black lines are for algorithm commands. TH; is a predetermined threshold value.

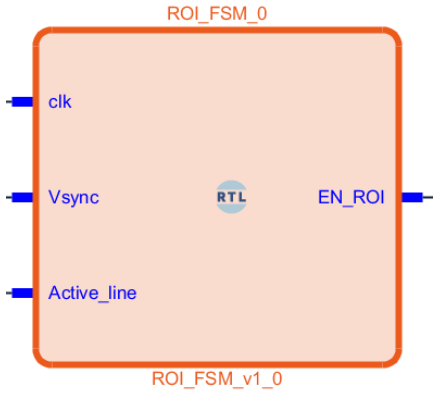

Figure 6. ROI displaying enable IP module (Figure 2) using Verilog HDL.

A linear combination of the RGB signals can be used to calculate the Y signal as explained in eq. (2), which represents the intensity of the grayscale. An approximation is done to implement the RGB to Grayscale IP module which is responsible for converting from RGB (24-bit) to grayscale (8-bit). In order to reduce resource utility and processing time, the shift operation is used instead of multiplication as shown in eq. (3) to obtain the approximated grayscale level of eq. (2) that is defined in eq. (4) which is acceptable in some applications:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \quad (2)$$

$$Y = (R \gg 2) + (G \gg 1) + (B \gg 3) \quad (3)$$

$$Y = 0.25 * R + 0.5 * G + 0.125 * B \quad (4)$$

From eq. (3), shift by two to the right means that R (red) is divided by four, shift by one to the right for G (green) means divided by two, and shift by three B (blue) means divided by eight as explained in eq. (4).

Two strategies are proposed to control the order of the five fundamental operations applied to the input pixels to verify the real-time functioning of the main processing unit. The timing diagram shown in Figure 9 explains the first strategy of the multi-domain clock signals that were used to implement the suggested subsystem to control the operation sequence in real-time. These clock signals can be generated from the clocking wizard IP module (see Figure 4), the orange triangle on the negative edge of the Clk-D clock is the sampling instant of the input pixels. The moment at which the incoming pixel is converted from RGB to grayscale format at 3×Clk-D is represented by the green triangle. The black triangle at the positive edge of 6×Clk-D is for reading the BG pixel or a predefined threshold value. The subtraction operation is done at the positive edge of Clk-D (the red triangle). Finally, the blue triangle indicates the instant of data width back converting from 8-bit to 24-bit. By using a multi-clock domain the latency is equal to a one-pixel clock period (13.468nsec).

Algorithm: Verilog code for ROI module (FSM section)

```
// state machine parameters
parameter Wait_Vsync = 3'b000;
parameter Wait_Active_line = 3'b001;
parameter Pixel_count_row =3'b010;
parameter Active_Line_End = 3'b011;
parameter row_check = 3'b100;
// State Machine Block
always @( posedge clk or posedge Vsync )
if ( Vsync )
EN_ROI <=1'b0;
state_reg <= Wait_Vsync;    // wait for Active_line
else
case ( state_reg )
Wait_Vsync: //////
if ( Active_line && (count_col < 11'd360) )
begin
state_reg <= Wait_Active_line;
EN_ROI <=1'b1;
end
Wait_Active_line: //////
if (count_col >= 11'd360)
begin
EN_ROI <=1'b0;
state_reg <= Pixel_count_row;
end
Pixel_count_row: //////
if (!Active_line)
state_reg <= Active_Line_End;
Active_Line_End:  /////
if (count_row < 9'd360)
state_reg <= Wait_Vsync;
else if (count_row == 9'd360)
begin
state_reg<=row_check;
EN_ROI <=1'b0;
end
row_check:   /////
if ( Vsync ) state_reg <=Wait_Vsync;
default:
state_reg <= Wait_Vsync;
endcase
// Counter Block
always @( negedge clk or posedge Vsync )
if ( Vsync || (state_reg == row_check))
count_row <= 9'd0;
count_col <=11'd0;
else if ((state_reg == Wait_Active_line) || (state_reg
    == Pixel_count_row))
count_col <= count_col + 1;
else if (state_reg == Active_Line_End)
count_col <= 0;
count_row <= count_row + 1;
endmodule
```

The second strategy is shown in Figure 10. This strategy, in contrast to the previous one, involved converting RGB to grayscale at the falling edge of the 2×Clk-D clock signal, which happens when Clk-D='0'. The subtraction process is carried out at the Clk-D signal's positive edge. The Grayscale to RGB conversion is carried out when Clk-D='1' at the rising edge of the 2×Clk-D clock signal.

6. RESULTS AND DISCUSSION

Figure 11 depicts the entire configuration of the prototype system, which includes a display unit (right) that shows the display of the ROI video/image with a size of (360×360) pixels, the ZYBO Z7-10 board (mid), and a laptop that serves as a video/image source via the HDMI port (left). Figure 12 illustrates the pulse signal that permits the chosen 360 pixels/row for the ROI image window's output display.

The signals displayed in Figure 13 have been plotted during the prototype's operation using the integrated logic analyzer (ILA) IP core for the second strategy, which is depicted in Figure 10. Clk-D, 2×Clk-D, and 6×Clk-D are signals that control the order in which the main processing unit operates. The third signal, EN-ROI, is the output of the suggested module that permits the ROI to be displayed. These clock signals are produced using the clocking wizard IP core. The following is a list of the operation sequence declarations:

- White circle: the instant of the enable signal for each image line. This signal holds at level one during (360 pixels/line).
- White arrow: the time instant of the input pixel sampling.
- White solid circle: The time instant of converting the RGB to Grayscale format.
- Red arrow: the instant of subtraction operation.

Figure 14 shows the critical path of data. It is between the first sampling DFF unit (see Figure3) and the RGB to Grayscale IP. The timing report shown in Figure15 indicates the slack value, the source unit, and the destination unit of the data path. The slack value is computed during the implementation process using Equation 5;

$$Slack = DataRequiredTime - DataArrivaltime. \quad (5)$$

By changing the time instant for the operations of the main processing unit as explained in the second strategy (see Figure10), the critical data path is now between the first DFF and the Sub-Byte IP as shown in Figure16. The value of the slack time has been reduced from (-1.63nsec) to (-1.05nsec) using the second timing trigger strategy as shown in Figure17. The type of resources used to implement each module, depending on its purpose, and the clock frequency value of that module are the causes of these timing variations.

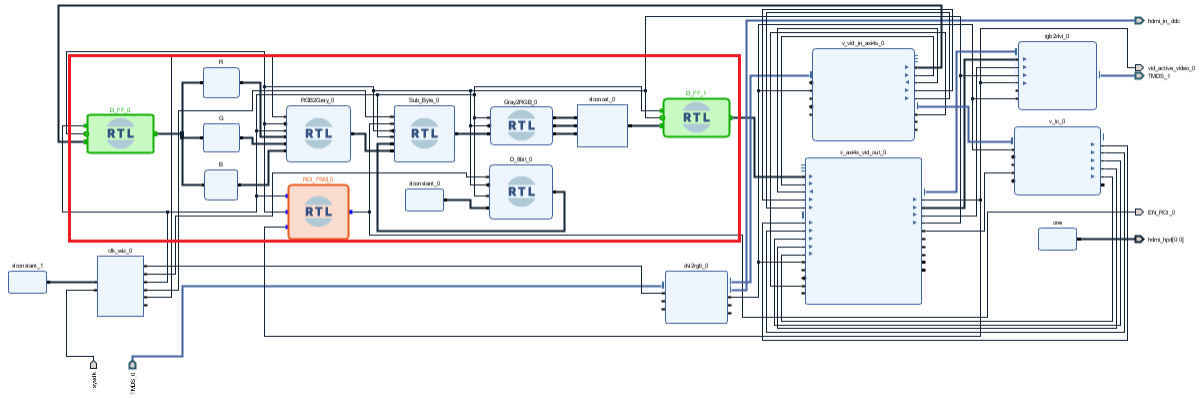

Figure 7. The complete real-time basic video processing system.

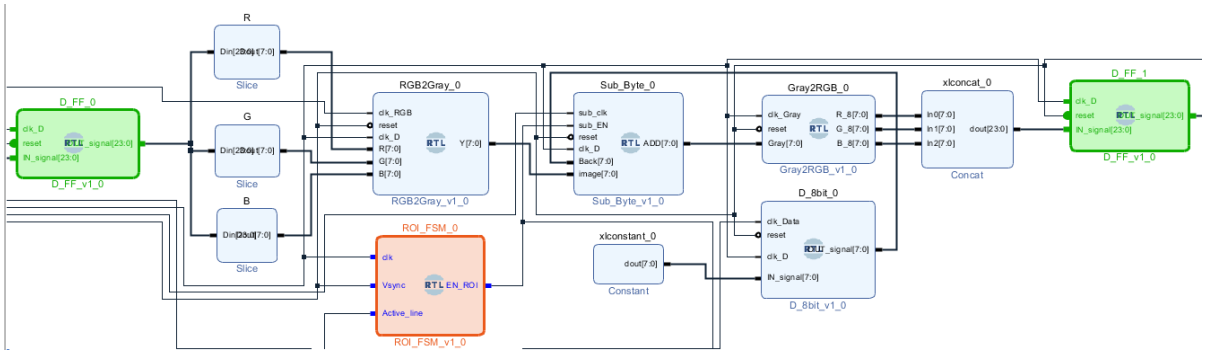

Figure 8. The main processing unit between two samplings units (Green DFF) see Figure3.

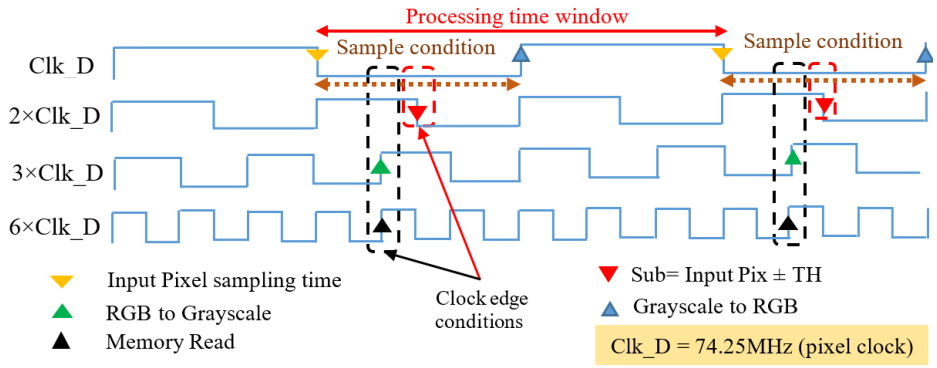

Figure 9. The proposed timing diagram sequence for basic video processing (Strategy 1).

The suggested architecture is successfully implemented in FPGA. Table I provides information about the used logic cells. These numerical results show that the suggested model architecture has low hardware requirements.

Because the related works cover a wide range of hardware types, input source types, and application scenarios, direct comparisons of resource utility and performance are difficult. We used the most widely used parameters in order to enable a fair comparison and to give the reader an overview of the current state of the art in the relevant fields. As a result, Table II gathers the crucial factors from the earlier study publications, including the used device, programming language, synthesis tools, input image/video resolution, input image/video source, data width, application, and the operating frequency.

TABLE I. Resource Utilization (ZYBO-7z010clg400-1).

FPGA Resources	Slice LUTs	Slice Reg.	F7 Mux	F8 Mux	BRAM
Available	17600	35200	8800	4400	60
used	1259	1860	16	8	6
Utility %	7.15	5.26	0.18	0.18	10.0

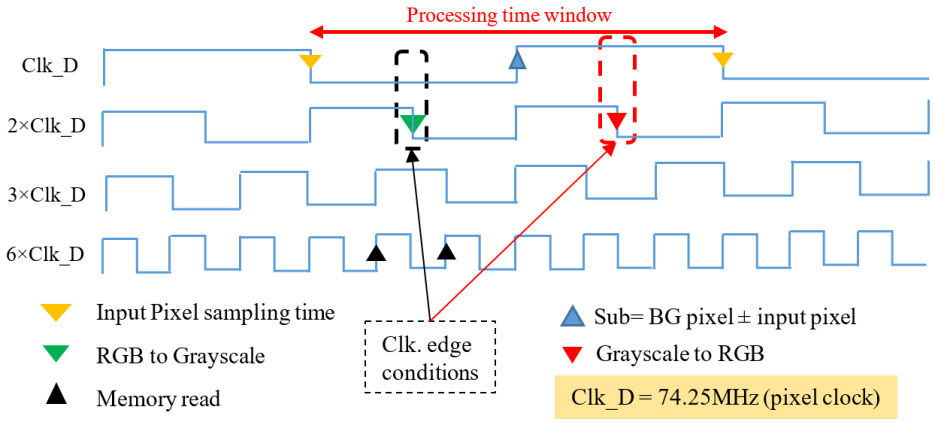

Figure 10. The proposed timing diagram sequence for basic video processing (Strategy 2).

Figure 11. The demonstration prototype for the ROI displaying of video/image signal.

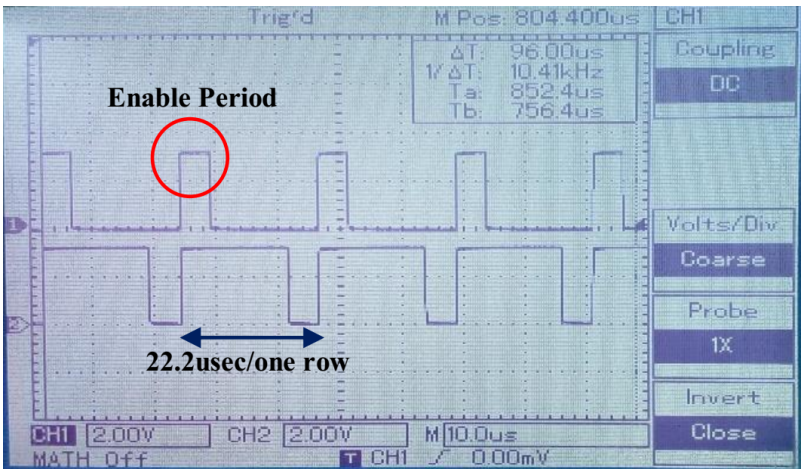

Figure 12. The red circle points at the enable signal for each line within the ROI, and the lower is the video active line signal (the arrow points to one of the video line periods).

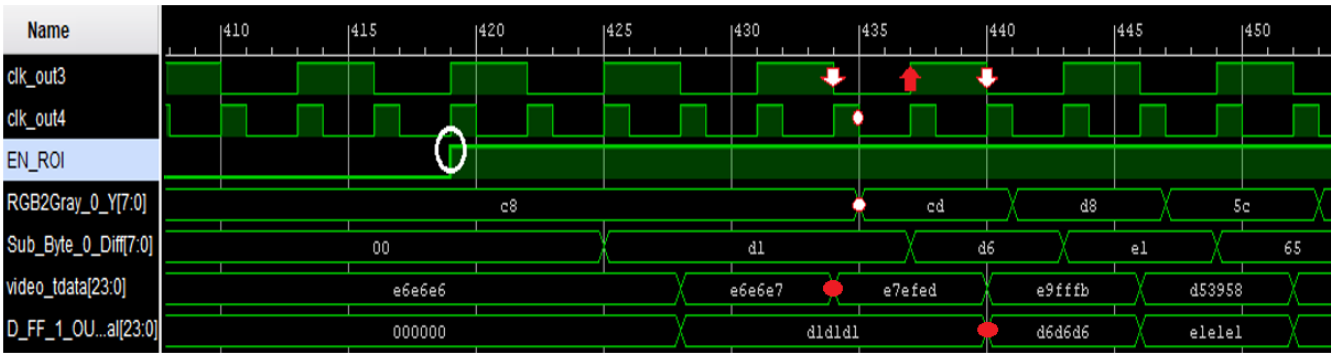

Figure 13. Multi domain clock signals where; clk-out3 is the pixel clock (Clk-D='74.25MHz'), clk-out4 is (2×Clk-D). The 6×Clk-D signal is not displayed here because it is used as an input trigger clock for the ILA core.

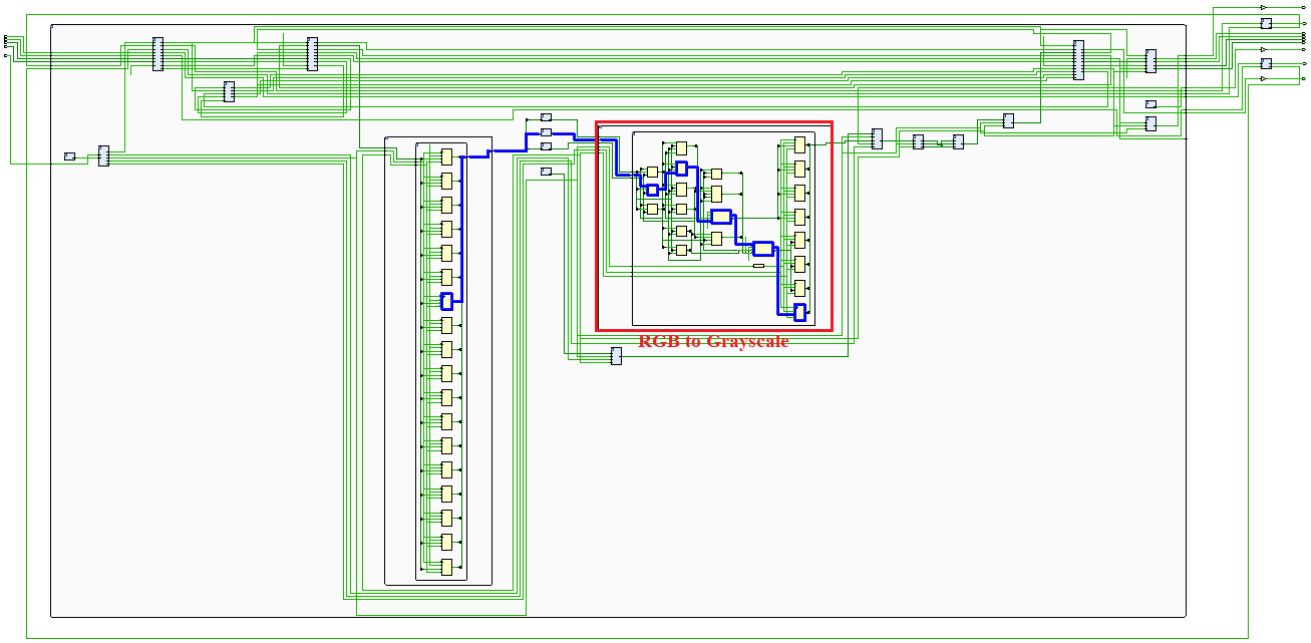

Figure 14. The RTL design (strategy 1) of the whole proposed system and the critical data path (blue line).

Name	Path 253
Slack	-1.633ns
Source	design_1_i/D_FF_0/inst/OUT_signal_reg[10]/C (falling edge-triggered cell FDRE clocked by clk_out3_
Destination	design_1_i/RGB2Gray_0/inst/Y_reg[7]/D (rising edge-triggered cell FDRE clocked by clk_out5_design
Path Group	clk_out5_design_1_clk_wiz_0_0

Slack Equation

Required Time - Arrival Time	
Required Time	7.462ns
Arrival Time	9.095ns

Figure 15. Timing report for the data path between D-FF and RGB to Grayscale IP.

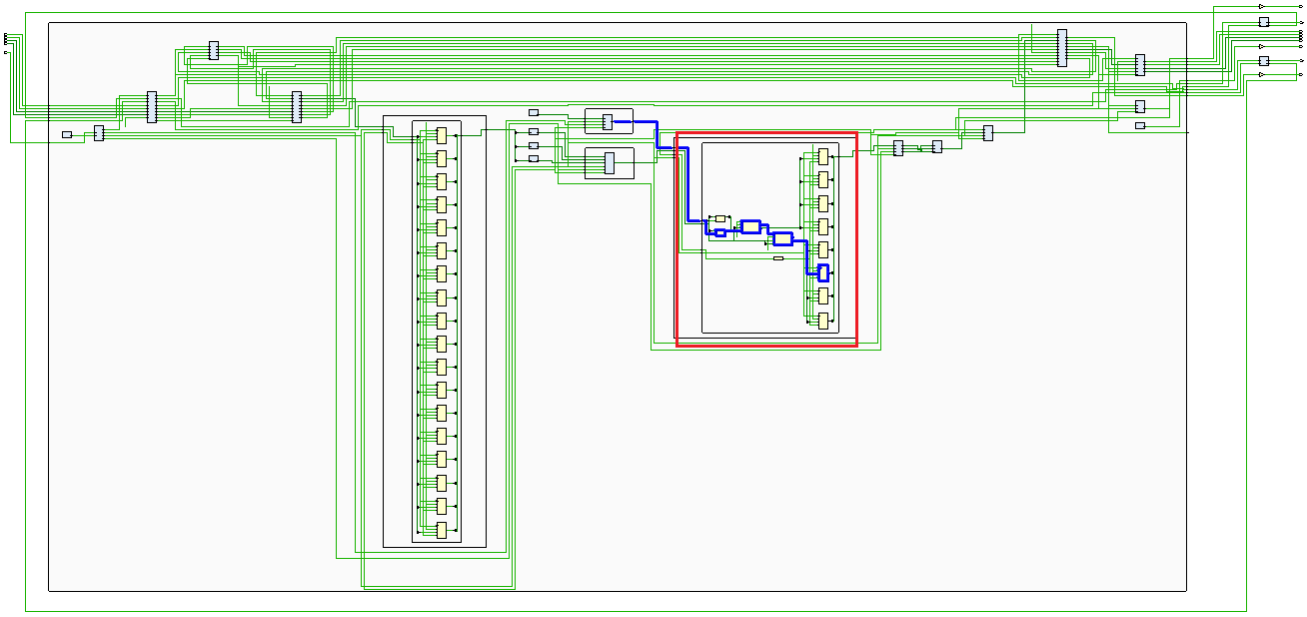

Figure 16. The RTL design (strategy 2) of the whole proposed system and the critical data path (blue line).

Name	Path 121
Slack	-1.050ns
Source	design_1_i/D_8bit_0/inst/OUT_signal_reg[0]/C (rising edge-triggered cell FDRE clocked by clk_out2
Destination	design_1_i/Sub_Byte_0/inst/ADD_reg[5]/D (rising edge-triggered cell FDRE clocked by clk_out3_des
Path Group	clk_out3_design_1_clk_wiz_0_0

Slack Equation

Required Time - Arrival Time	
Required Time	12.165ns
Arrival Time	13.216ns

Figure 17. Timing report shows that the negative slack is between the D-FF (storage device) and Sub-Byte IP.

7. CONCLUSION

Complex analysis of surveillance systems requires an understanding of real-time performance and basic video timing signals. This paper presents an implementation of a basic processing operations-based ROI video/image reading algorithm for real-time applications using FPGA technology. A fixed ROI with a size of (360×360) was selected to demonstrate the idea behind this work. By modifying the setting of the counters, the ROI's display size can be manually adjusted in both vertical and horizontal dimensions. Because Verilog HDL was used in the design of the suggested subsystem, the timing diagram for real-time applications could be understood more clearly. Using multi-clock domain frequencies derived from the operating frequency gives a choice to add a pixel-by-pixel operation during the instant of the sampling time (13.468nsec). For the proposed subsystem, lowering the operating clock frequency for applications with lower video resolutions reduces the negative slack to zero or a positive value. One of the limitations of the proposed design is that the source of the video is the laptop's HDMI port, which outputs 1280x720 pixels at 60 frames per second. As a result, controlling the resolution and frame rate of the video source is not permitted.

Furthermore, because of the laptop's HDMI port as the video source, the suggested system is unsuitable for portable applications. The results in this paper could be used to guide future applications. To regulate the ROI size according to the requirements of a particular application, a software algorithm that integrates adaptive processing could be added. This algorithm could be implemented in the processing system (PS) of the chosen board. Additionally, a design of an IP module by utilizing the PL part to adjust the ROI's size based on the dimensions of the object of interest. The suggested work could be expanded to incorporate additional processing, such as object detection, tracking, and image filtering.

Acknowledgment

This research paper was completed in the University of Mosul laboratory. The authors would like to thank the University of Mosul for their support.

TABLE II. The summary of the most common factors collected from the recent related works with our work. Unavailable data marked as (—).

Comparison factors	Wang [2]	Conti [22]	Das [24]	Liu [15]	Sarkar [25]	Devi [6]	**This Work**
Device (Board)	Zynq UltraScale +MPSoC	Zed board 7020 (CPU+ FPGA)	-Virtex7XC 7V28000T -Zynq XC7Z030	Ultra-large prog. CycloneII	-Nexys video -Zybo Z7-10	Zed board	ZYBO- (7Z010 clg400-1)
Prog. Language	C++	C++ and OpenCV 3.1	C and MATLAB	—	VHDL	C and MATLAB	Verilog+ IP cores
Simulation & synthesis tools	Xilinx SDK & Vivado HLS	Xilinx Vivado HLS	Xilinx Vivado HLS	—	Xilinx Vivado	Xilinx Vivado+ SDK Soft.	Xilinx Vivado
-input image size (Pix.) -fps	360x480 —	640x480 15 fps	640x480 30 fps	1280x720 —	640x480 30 fps	768x512 —	1280x720 60 fps
Image/video source	Video Dataset	Zenith camera	OV7670	Image sensor	-OV7670 -OV9655	Saved image	Laptop HDMI
Data width	8-bit	RGB & 8-bit	8-bit	16-bit,12-bit	16-bit	RGB	8-bit
Operations/ application	GMM algorithm	BGS algorithm	RCM algorithm	Image interpo- lation	Camera interface with FPGA	Image enhanc- ement	Displaying the ROI
Clock freq. MHz	300	—	150	153.12	—	—	-74.25 -148.5 -222.75

References

[1] M. Irfan, Z. Ullah, A. I. Sanka, and R. C. Cheung, "Accelerated updating mechanisms for fpga-based ternary content-addressable memory," *IEEE Embedded Systems Letters*, vol. 13, no. 2, pp. 37–40, 2020.

[2] S. Wang, C. Zhang, Y. Shu, and Y. Liu, "Live video analytics with fpga-based smart cameras," in *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019, pp. 9–14.

[3] R. Kaibou, M. S. Azzaz, M. Benssalah, D. Teguig, H. Hamil, A. Merah, and M. T. Akrour, "Real-time fpga implementation of a secure chaos-based digital crypto-watermarking system in the dwt domain using co-design approach," *Journal of Real-Time Image Processing*, vol. 18, no. 6, pp. 2009–2025, 2021.

[4] P. Sikka, A. R. Asati, and C. Shekhar, "High-speed and area-efficient sobel edge detector on field-programmable gate array for artificial intelligence and machine learning applications," *Computational Intelligence*, vol. 37, no. 3, pp. 1056–1067, 2021.

[5] M. Ravi, A. Sewa, T. Shashidhar, and S. S. S. Sanagapati, "Fpga as a hardware accelerator for computation intensive maximum likelihood expectation maximization medical image reconstruction algorithm," *IEEE Access*, vol. 7, pp. 111 727–111 735, 2019.

[6] D. A. Devi, N. R. Kathula, G. Kalluri, and L. S. Bondalapati, "Design and implementation of image processing application with zynq soc," *International Journal of Computing and Digital Systems*, vol. 14, no. 1, pp. 377–385, 2023.

[7] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[8] R. M. Sousa, M. Wäny, P. Santos, F. Morgado-Dias, and I. Member, "Automatic illumination control for an endoscopy sensor," *Microprocessors and Microsystems*, vol. 72, p. 102920, 2020.

[9] S. Malmir and M. Shalchian, "Design and fpga implementation of dual-stage lane detection, based on hough transform and localized stripe features," *Microprocessors and Microsystems*, vol. 64, pp. 12–22, 2019.

[10] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Rise: An automated framework for real-time intelligent video surveillance on fpga," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–18, 2017.

[11] G. Cocorullo, P. Corsonello, F. Frustaci, L.-d.-l.-A. Guachi-Guachi, and S. Perri, "Multimodal background subtraction for high-performance embedded systems," *Journal of Real-Time Image Processing*, vol. 16, pp. 1407–1423, 2019.

[12] V. P. Korakoppa, H. R. Aradhya *et al.*, "An area efficient fpga implementation of moving object detection and face detection using adaptive threshold method," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2017, pp. 1606–1611.

[13] S. Singh, C. Shekhar, and A. Vohra, "Fpga-based real-time motion detection for automated video surveillance systems," *Electronics*, vol. 5, no. 1, p. 10, 2016.

[14] V. Taraate, *Digital logic design using verilog*. Springer, 2022.

[15] G. Liu, B. Zhou, Y. Huang, L. Wang, W. Wang, and E. Zhao, "Video image scaling technology based on adaptive interpolation algorithm and tts fpga implementation," *Computer Standards & Interfaces*, vol. 76, p. 103516, 2021.

[16] J. L. Brock, *Introduction to Logic Circuits & Logic Design with Verilog*. Spinger, 2019.

[17] M. A. Al-yoonus and S. A. Al-Kazzaz, "Fpga-soc based object tracking algorithms: A literature review," *Al-Rafidain Engineering Journal (AREJ)*, vol. 28, no. 2, pp. 284–295, 2023.

[18] E. Calvo-Gallego, P. Brox, and S. Sánchez-Solano, "Low-cost dedicated hardware ip modules for background subtraction in embedded vision systems," *Journal of Real-Time Image Processing*, vol. 12, pp. 681–695, 2016.

[19] A. Cortes, I. Velez, and A. Irizar, "High level synthesis using vivado hls for zynq soc: Image processing case studies," in *2016 Conference on design of circuits and integrated systems (DCIS)*. IEEE, 2016, pp. 1–6.

[20] M. Benetti, M. Gottardi, T. Mayr, and R. Passerone, "A low-power vision system with adaptive background subtraction and image segmentation for unusual event detection," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 11, pp. 3842–3853, 2018.

[21] F. Carrizosa-Corral, A. Vázquez-Cervantes, J.-R. Montes, T. Hernández-Díaz, J. C. Solano Vargas, L. Barriga-Rodríguez, J. A. Soto-Cajiga, and H. Jiménez-Hernández, "Fpga-soc implementation of an ica-based background subtraction method," *International Journal of Circuit Theory and Applications*, vol. 46, no. 9, pp. 1703–1722, 2018.

[22] G. Conti, M. Quintana, P. Malagón, and D. Jiménez, "An fpga based tracking implementation for parkinson's patients," *Sensors*, vol. 20, no. 11, p. 3189, 2020.

[23] A. Linares-Barranco, F. Perez-Peña, D. P. Moeys, F. Gomez-Rodriguez, G. Jimenez-Moreno, S.-C. Liu, and T. Delbruck, "Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time fpga applications," *IEEE Access*, vol. 7, pp. 134 926–134 942, 2019.

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

[25] S. Sarkar, S. S. Bhairannawar, and R. KB, "Fpgacam: A fpga-based efficient camera interfacing architecture for real-time video processing," *IET Circuits, Devices & Systems*, vol. 15, no. 8, pp. 814–829, 2021.

[26] O. Iqbal, V. I. T. Muro, S. Katoch, A. Spanias, and S. Jayasuriya, "Adaptive subsampling for roi-based visual tracking: Algorithms and fpga implementation," *IEEE Access*, vol. 10, pp. 90 507–90 522, 2022.

[27] X. Ren and Y. Wang, "Design of a fpga hardware architecture to detect real-time moving objects using the background subtraction algorithm," in *2016 5th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 2016, pp. 428–433.

Marwan AL-Yoonus received a B.S. degree in Electrical Engineering in 1994 from the Electrical Engineering Department, University of Mosul, Iraq. Then he was appointed as an assistant engineer in the same department. He received an MSc degree in 2009 from the same department as well. Upon his graduation, he was appointed as teaching staff (lecturer) in the Electrical Engineering Department, University of Mosul. Now, he is pursuing his Ph.D. degree at the same university. His research interests include FPGA platforms for embedded applications in real-time and computer vision.

Sa'ad Ahmed Al kazzaz received the B.S. and M.S. degrees in electrical engineering from University of Mosul, Mosul, Iraq, in 1986 and 1990 respectively, and Ph.D. degree from Indian Institute of Technology Roorkee, Roorkee, India, in 2001. Currently, he is an Assistant Professor with the Department of Electrical Engineering, University of Mosul. His field of interest includes health monitoring of electrical machines and drives, application of FPGA in the field of surveillance system.