Designing and Real-Time FPGA-Based Implementation of a Chroma-Key Effect System Using Xilinx System Generator

H. Merah, L. Merah, N. Chaib, and A. Ali-Pacha

***Abstract*—Chroma key is a popular visual effect and post-production technique today. The aim of this work is to design and implement the Chroma-key effect on an FPGA in real-time. The XSG (Xilinx System Generator) tool is used to design the effect system, and the Xilinx ISE (Integrated Synthesis Environment) tool is used to simulate, synthesize, and implement it. Despite the fact that other techniques for achieving the Chroma-key effect exist, we have based our work on the color space conversion technique (RGB to HSV) to extract the image's background. The mathematical model of the color space converter (RGB to HSV) of the designed system is based on the MATLAB *rgb2hsv* function. The designed XSG-based system is stimulated under Simulink/Matlab. The obtained results are compared with the results obtained from a Chroma-key effect implemented using Matlab scripting. The obtained results have confirmed the good function of the designed XSG-based system. The designed system is then implemented on the FPGA, and a real-time video application is executed to extract the green background and replace it with a new background. The obtained results demonstrated that the designed Chroma-key effect system worked properly. The paper is presented as a guide with technical details about designing and real-time implementation of the Chroma-key effect, and also any other video processing application on FPGA.**

***Index Terms*—Image processing, chroma-key, green screen, real-time, Xilinx, XSG, FPGA.**

I. INTRODUCTION

Today, the digital image is part of our daily life. Television, mobile phones, medical imaging, traffic control, robotics, etc., are all areas where the digital image plays a key role. Thanks to the availability of sophisticated computers, digital image processing has become an area of active research. Its use has grown exponentially over the past few decades. Its applications range from medicine and entertainment to geological processing and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing [1]. Digital image processing refers to all the techniques used to modify a digital image with the aim of improving it or extracting information from it.

One of the most commonly used image processing techniques in video broadcasting and cinema (science fiction) is the Chroma-key (or keying) effect. It is the most used and oldest visual effects technique. Simply put, Chroma is shooting a subject against a solid colored background, then removing that background in post-production and replacing it with transparency. Then the subject can be placed in front of any new background. The most commonly used key colors are green and blue. Why these two colors? They are in opposite contrast to the color of human skin.

Generating a Chroma-key effect using a blue or green background is not the only way to achieve a result. An alternative is to focus on differences. The idea behind differences is based on storing the background image and calculating the difference between the pixels' values of the background image and those of any moving object through this background. When operating in Chroma-key mode, the program replaces anything it thinks belongs to part of the original background with the material we wish to key in as the new background [2]. A common Chroma-key method is to find a mask (including the Fixed Key Method and the K-Means Clustering Method [3], [4]) representing the foreground region extracted from the foreground frame [5].

Another possible alternative (which is our case) is to key on a color specified by its saturation and value (HSV). The HSV model provides a more natural way to select color, and it may be easier to choose a key color using this model than with the RGB components [2].

Real-time image processing requires powerful computing platforms. One of the most popular computing platforms today is the FPGA (Field-Programmable Gate Array). Today, FPGA circuits have become truly revolutionary devices that combine the advantages of hardware and software. An FPGA is an integrated circuit designed to be configured by a designer after manufacture—thus the term "Field-programmable" [6]. The FPGA configuration is usually specified using a Hardware Description Language (HDL), similar to that used for an Application-Specific Integrated Circuit (ASIC). FPGAs offer considerable performance due to the parallel structure of their programmable logic resources. This advantage allows FPGAs to become the right choice to implement algorithms that require high performance, for example image processing algorithms.

The purpose of this work is to implement an algorithm to extract and replace the background of an image or video on an FPGA for real-time applications. The principle is to choose a range of background colors to extract and replace

Manuscript received October 9, 2022; revised November 30, 2022.

Hocine Merah is with the Physics department, École Normale supéRieure of Laghouat–Oasis Nord, Laghouat, 03000, Algeria (e-mail: merah.hossein2@gmail.com).

Lahcene Merah is with the Electronics Department, University Amar Telidji, BP 37G, Laghouat, Algeria (e-mail: l.merah@lagh-univ.dz).

Noureddine Chaib is with the Computer Science and Mathematics Laboratory, University Amar Telidji, BP 37G, Laghouat, Algeria (e-mail: n.chaib@lagh-univ.dz).

Adda Ali-Pacha is with the the Department of Electronics, University of Science and Technology of Oran (USTO), Oran 31036, Algeria (e-mail: a.alipacha@gmail.com).

with a different image. The RGB/HSV converter is the key component of our design. Using the HSV color space will simplify the selection of the color range.

We have used the XSG tool to design the Chroma-key effect system, including the RGB/HSV converter. This tool makes it possible to create fairly complicated digital systems without prior knowledge of hardware description languages such as VHDL. XSG adds blocks to the Simulink/Matlab library. Some of these blocks are used to design and simulate the Chroma-key effect system, and then to automatically generate the VHDL code. The generated VHDL code will be exported to the ISE (Integrated Synthesis Environment) tool in order to complete the remaining real-time implementation steps.

This paper is organized as follows: In Section II, the conversion from the RGB space to the HSV space and the implementation of the Chroma-key effect using Matlab scripting are achieved. The simulation results will be used later as references to the Chroma-key effect results achieved using XSG. Section III presents the design, simulation, and real-time implementation steps of the Chroma-key effect system using XSG and ISE tools. Section VI is devoted to presenting the simulation results of the XSG-based designed Chroma-key effect system. The FPGA-based implementation and the real-time evaluation of the designed Chroma-key effect system are the subjects of section V. The paper is ended by a conclusion about the achieved results.

II. Chroma Key Effect Using Matlab

Color is a visual sensation which is produced by the interaction of the light spectrum on the photoreceptors represented by the cone cells in the retina of the eye [7]. Cone cells allow us to see color. There are three kinds, each responsible for seeing Red, Green, or Blue, respectively (RGB); the three primary colors that combine to create the multitude of color beauty we see in our world.

In a digital color image, each pixel is composed of three color elements, RGB, to form a variety of colors. There are other color space systems, for instance CIE XYZ, HSV, YUV, CIELAB, etc. [8]. The General HSV color space is another expression of the RGB color space. As it expresses the perception of color contacts clearer than RGB, and the calculation is very simple, it is widely adopted in image processing [9].

As previously stated, the design and implementation of the Chroma-key effect on FPGA is based on the MATLAB algorithm (RGB to HSV conversion) dedicated to this purpose. Hence, this section is devoted to presenting this algorithm and performing some practical experiments using it. Converting an image from RGB to HSV color space is the most important way of producing the Chroma-key.

The RGB/HSV conversion is performed using the MATLAB rgb2hsv function dedicated to this purpose. $H = rgb2hsv\ (M)$ converts an image M of RGB colors to an image of HSV colors. This function is based on the following mathematical model:

$$mx = \max(\max(R,G),B)\,; \qquad (1)$$

$$\Delta = (V - min(min(R,G),B))\,; \qquad (2)$$

$$V = \frac{mx}{255}; \qquad (3)$$

$$S = \begin{cases} 0\,, & mx = 0 \\ \frac{\Delta}{mx}\,, & mx \neq 0 \end{cases} \qquad (4)$$

For H, we have three cases:

- If R = mx $\quad H = \frac{G-B}{\Delta} \times \frac{1}{6}$
- If G = mx $\quad H = (2 + \frac{B-R}{\Delta}) \times \frac{1}{6}$
- If B = mx $\quad H = (4 + \frac{R-G}{\Delta}) \times \frac{1}{6}$

If a negative value for H is obtained, then $H = H + 1$.

A. Application Example

For the application example, the RGB image shown in Fig.1 is used. The following MATLAB code is used to convert the image from RGB color space to HSV:

```
im = imread('image.bmp'); % Import the RGB image
[H,S,V] = rgb2hsv(im); % Convert the RGB image to HSV space
```

Table I shows some randomly chosen RGB values and the corresponding obtained HSV values after conversion. It is obvious from Table I that, despite the RGB component values changing, the value of H remains nearly constant (we limit ourselves to only two numbers after the decimal point). This shows that the HSV color space is better for background extraction because the user can easily define the range of colors to be extracted.

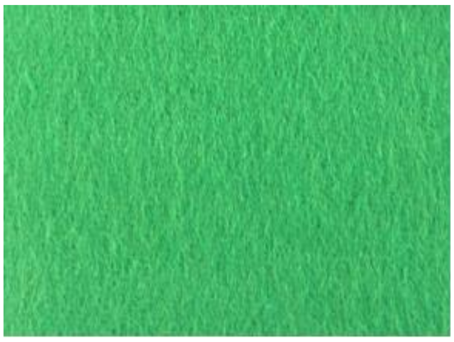


Fig. 1. The RGB image used in the application example.

TABLE I: Randomly Chosen RGB Values from the Fig.1 and the Corresponding HSV Obtained Values after Conversion

R	G	B	H	S	V
78	199	113	0.38	0.60	0.78
72	193	107	0.38	0.62	0.75
74	195	109	0.38	0.62	0.76
71	192	106	0.38	0.63	0.75
66	189	103	0.38	0.65	0.74
67	192	105	0.38	0.65	0.75
64	189	102	0.38	0.66	0.74
72	196	109	0.38	0.63	0.76
82	207	119	0.38	0.60	0.81
77	204	115	0.38	0.62	0.80
73	200	111	0.38	0.63	0.78
75	202	113	0.38	0.62	0.79
81	208	119	0.38	0.61	0.81
72	199	110	0.38	0.63	0.78
65	190	102	0.38	0.65	0.74
72	196	108	0.38	0.63	0.76

For instance, the following MATLAB code takes an image's background and replaces it with a different one:

```
I1=imread('testkey2.bmp'); % import the initial RGB image
I2=imread('fond.bmp'); % import background RGB image
R2=I2(:,:,1); R2=R2(:); % convert the Red component  to a
Vector
G2=I2(:,:,2); R2=R2(:); % convert  the Green component to a
vector
B2=I2(:,:,3); R2=R2(:); % convert the Blue component to a
vector
[H,S,V]=rgb2hsv(I1); % convert the initial RGB image to HSV.
msk = (H>0.25 & H<0.55); % selection of the H interval to be
extracted.
% Replacing the background with second image.
I1(find(msk(:))+numel(msk)*0)= U(find(msk(:))+numel(msk)*0);
I1(find(msk(:))+numel(msk)*1)=U(find(msk(:))+numel(msk)*1);
I1(find(msk(:))+numel(msk)*2)=U(find(msk(:))+numel(msk)*2);
imshow(cat(2,I1)); % Displaying the result.
```

According to the code above, we can see that the interval chosen for H is from 0.22 to 0.55, which represents the vast majority of the green color possibilities. The obtained result is depicted in Fig. 2, whereas Fig. 3 shows the results of the effect of changing the interval of the H.

III. Design of Chroma-Key Effect System Using XSG

In this section we will present the design steps of an RGB/HSV converter based on XSG. The authors in [8] have designed an optimized RGB to HSV converter using XSG. For our design, we follow the optimization done by the authors in [8] but with some modifications; the RGB to HSV converter in our case is based on the mathematical equations (1), (2), (3), and (4) as used exactly by the *rgb2hsv* function in Matlab.

A. Design of Maximum and Minimum Calculator

We begin by constructing the first block, which is used to compute the maximum and minimum of R, G, and B using XSG. The achieved block is depicted in Fig. 4.

According to Fig.4, the block has 3 inputs which represent the RGB color components coded in 8 bits for each one, and 3 outputs: the maximum, the minimum, and the index. The index represents the output of a multiplexer. We have 3 states for the output of this multiplexer; 0, 1, and 2. The first if Red is the maximum, the second if Green is the maximum, and the last if Blue is the maximum. This output will indicate later what value H should take.

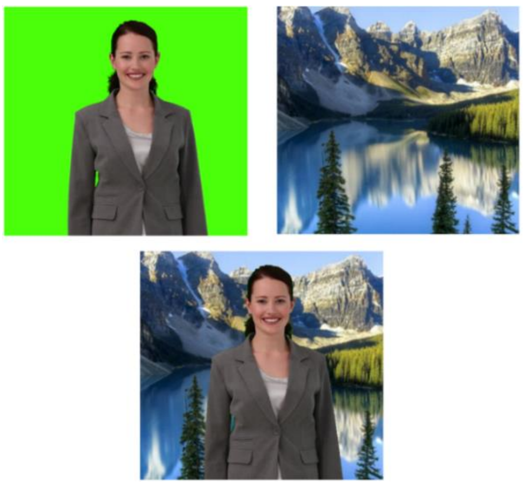


Fig. 2. Example of background extraction using Matlab.

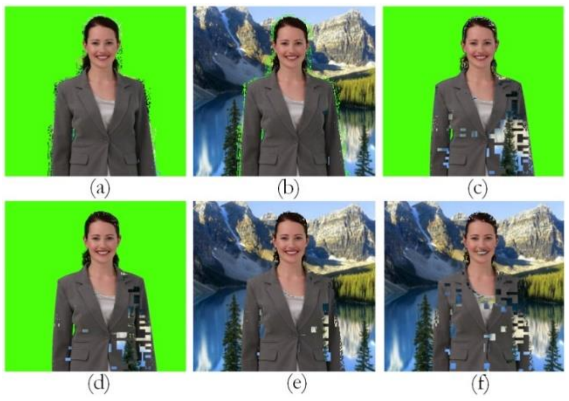


Fig. 3. Results obtained for different values of H, (a) H = [0.29-0.31], (b) H = [0.29-0.31], (c) H = [0.29-0.31], (d) H = [0.29-0.31], (e) H = [0.29-0.31], (f) H = [0.29-0.31].

B. The Arithmetic Subsystem

The following figure represents the second part of our system; it is the arithmetic part where all the necessary calculations are carried out:

According to Fig.5, the subsystem has 5 inputs (the RGB color components, the control index signal, the max and the min) and 3 outputs (H, S, and V). For the calculation of H, the main block for this operation is the *Devider Generator 3.0*. This block allows us to perform complicated division operations. For more information on this block, see [10].

For good optimization, only one divider block is used in our circuit. This is because, according to the H calculation formula, only one division operation is performed per time cycle. The index signal controls this operation. Fig. 5 shows that the output of *Subtractor S1* yields the following result:

- G-B if *index* =0 (condition where R is max),
- B-R if *index* =1 (condition where G is max),
- R-G if *index* =2 (condition where R is max).

The divider output (*div_res*) is the result of dividing the *S1* output by Δ. According to the formula for calculating H, the divider block result is added to:

- 0 if *index* =0,
- 2 if *index* =1,
- 4 if *index* =2.

Finally, the obtained result will be multiplied by 1/6 as shown in Fig. 5. For the calculation of S and V, the operation is quite easy. We can obtain the value of S by dividing the max by 255. Similarly, V can be obtained as follows: V = (max-min)/255.

IV. Simulation Results

The obtained results using the XSG-based design in terms of the HSV results and graphically by extracting the background of an RGB image are compared with results obtained using the *hsv2rgb* Matlab function.

A. Numerical Comparison

It should be noted that the designed XSG-based system uses 18Q12 fixed-point arithmetic precision (18 bits of length with 12 bits for the fractional part). We should compare the results from XSG and Matlab to make sure that our system is working correctly.

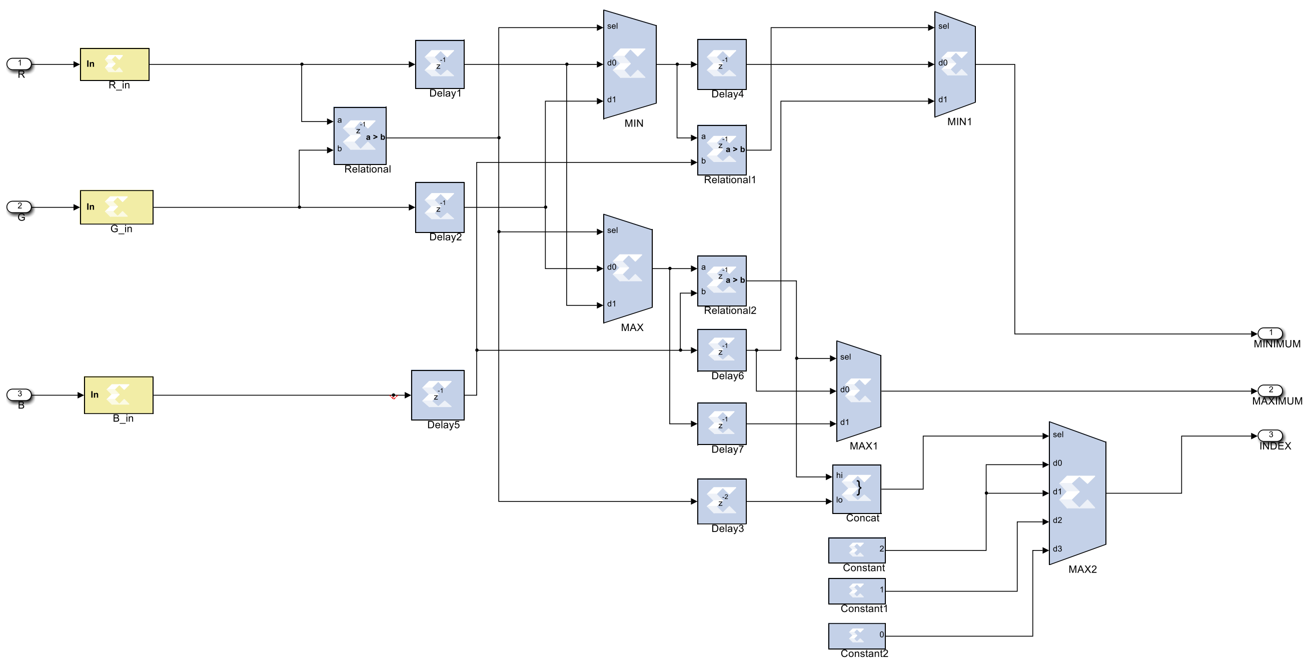


Fig. 4. Subsystem based on XSG for calculating the maximum and minimum of R, G and B.

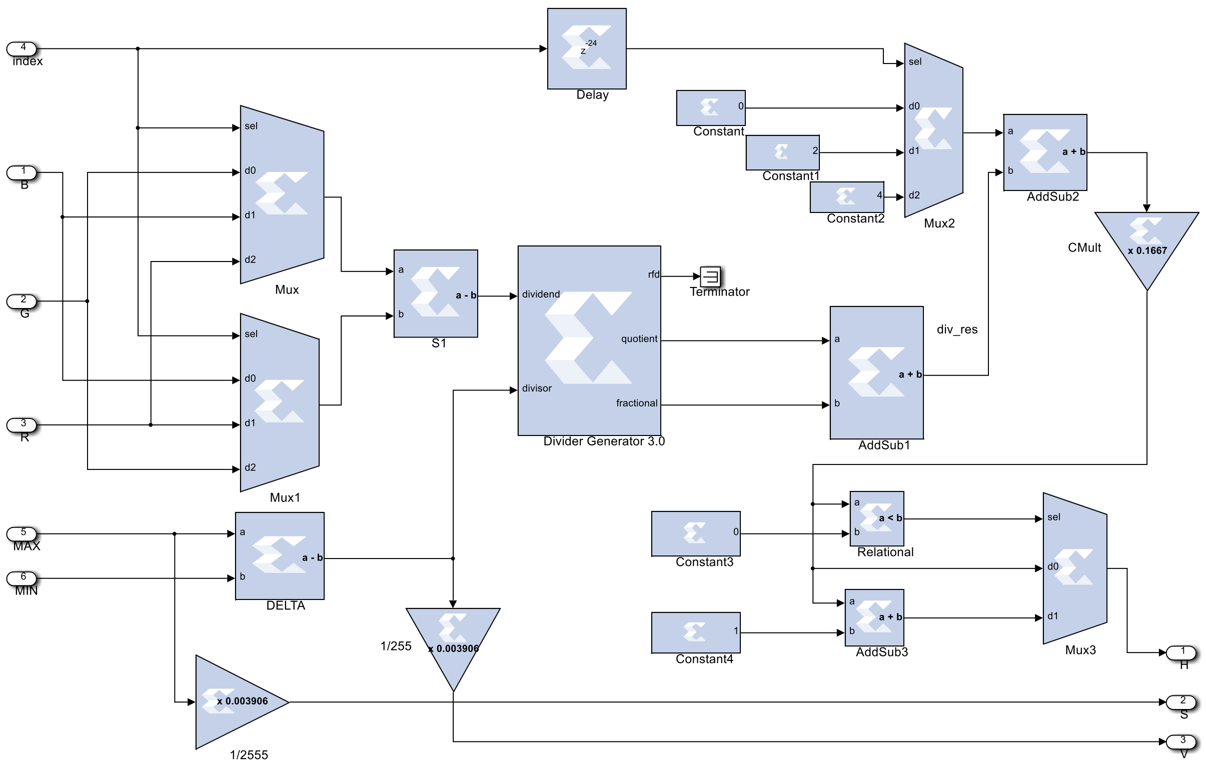


Fig. 5. XSG-based arithmetic subsystem.

Table II presents the obtained results (results of RGB to HSV conversion) for a given image with Matlab and those obtained with XSG. We notice that even though Matlab uses a large arithmetic precision (double precision with 64 bits), the results from XSG (fixed-point precision with 18 bits) are the same if only two digits after the decimal point are used, which is good enough for our application.

B. Graphical Comparison

In this part, we will compare the obtained results with the two implementation methods (Matlab and XSG) by extracting the background of a color image. Other blocks are added to the converter based on XSG. The role of these blocks is to select the interval of H that we must detect and replace with the chosen background pixels (Fig.6). Fig. 7 presents the obtained results. The results shown in Fig.8 make it very clear that the results from the code written in Matlab and the results from XSG are the same.

TABLE II: COMPARISON BETWEEN THE RESULTS OBTAINED WITH MATLAB AND THOSE WITH XSG

Results obtained with Matlab			Results obtained with XSG		
H	S	V	H	S	V
0.1372	0.1515	0.4063	0.1363	0.1573	0.4078
0.8386	0.0972	0.4141	0.8346	0.1065	0.4157
0.9414	0.1098	0.4453	0.9496	0.1118	0.4471
0.9390	0.1345	0.4883	0.9396	0.1349	0.4902
0.8899	0.1032	0.5469	0.8796	0.1027	0.5490
0.8931	0.1098	0.6563	0.8900	0.1029	0.6588
0.9092	0.1146	0.5898	0.9119	0.1157	0.5922
0.8833	0.1190	0.5742	0.8879	0.1106	0.5765
0.8770	0.1206	0.5430	0.8733	0.1229	0.5451
0.8862	0.1220	0.5703	0.8781	0.1247	0.5725
0.8887	0.1140	0.4531	0.8808	0.1127	0.4549
0.9092	0.1310	0.4141	0.9069	0.1384	0.4157
0.8896	0.1272	0.4609	0.8876	0.1193	0.4627
0.8833	0.1420	0.4102	0.8862	0.1476	0.4118
0.8770	0.1378	0.3633	0.8743	0.1357	0.3647
0.8862	0.1282	0.4023	0.8791	0.1292	0.4039
0.8801	0.1417	0.4453	0.8820	0.1465	0.4471

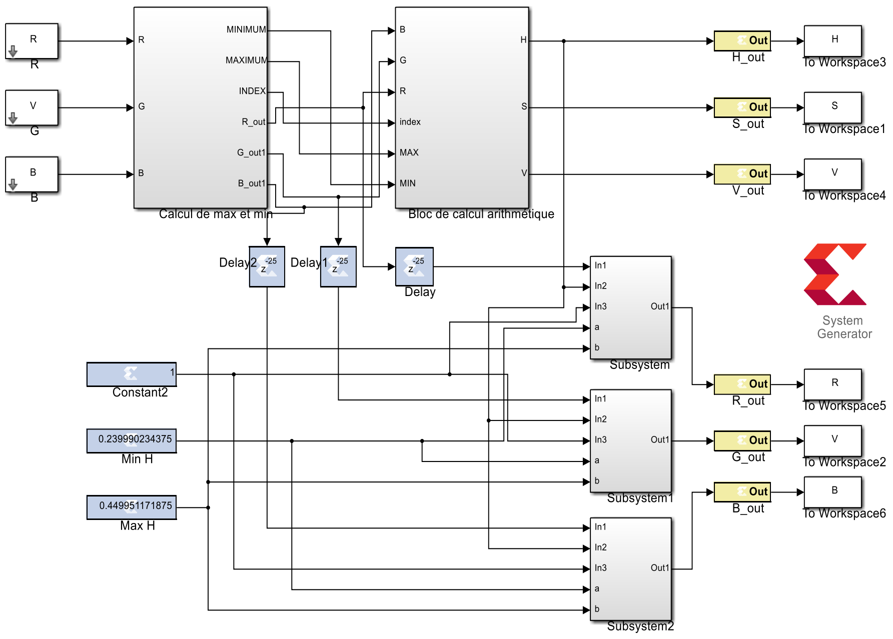


Fig. 6. The whole Chroma-key system based on XSG to extract the background of an image.

V. IMPLEMENTATION ON FPGA

After confirming the proper functioning of the XSG-based Chroma-Key system, we move on to its FPGA-based hardware implementation. This section presents the different steps for completing this point.

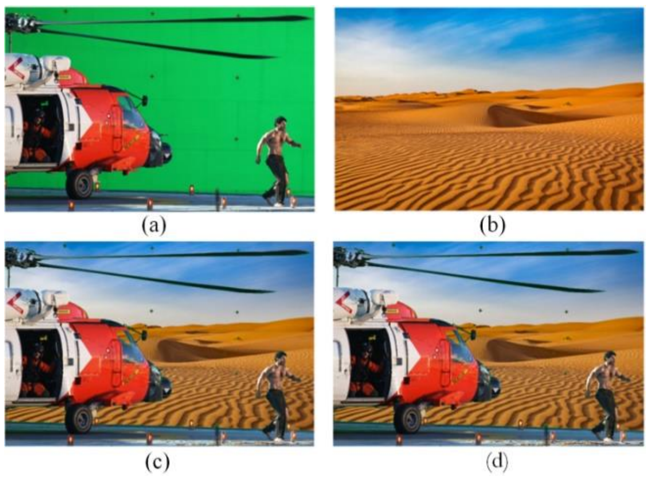


Fig. 7. Graphical comparison between the results obtained with Matlab and those with XSG, (a): original image, (b): background image, (c): result obtained by Matlab, (d): result obtained by XSG.

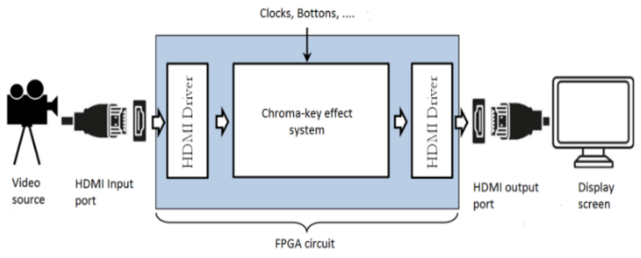
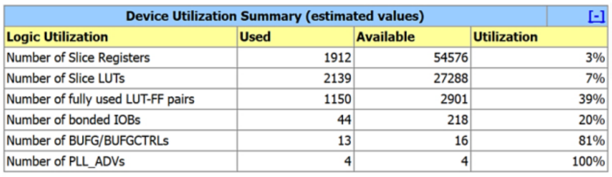


Fig. 8. Block diagram of the complete real-time Chroma-key effect system.

We have used the Atlys board for the implementation. The Atlys board is a complete, ready-to-use digital circuit development platform based on a Xilinx Spartan-6 LX45 FPGA, speed grade -3. The large FPGA and on-board collection of high-end peripherals including Gbit Ethernet, HDMI Video, 128MByte 16-bit DDR2 memory, and USB and audio ports make the Atlys board an ideal host for a wide range of digital systems, including embedded processor designs based on Xilinx's MicroBlaze [11].

It must be taken into consideration that the source of the image as well as the display of the image obtained in real time from the Atlys board is ensured by the HDMI interfaces. Controlling these interfaces with VHDL is a difficult task that is beyond the scope of this work. We send the VHDL files generated by XSG to a project already created under ISE and which contains the necessary files to ensure communication with the HDMI ports. The whole project scheme is presented in Fig. 8.



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1912	54576	3%
Number of Slice LUTs	2139	27288	7%
Number of fully used LUT-FF pairs	1150	2901	39%
Number of bonded IOBs	44	218	20%
Number of BUFG/BUFGCTRLs	13	16	81%
Number of PLL_ADVs	4	4	100%

Fig. 9. Resources used by the implemented system.

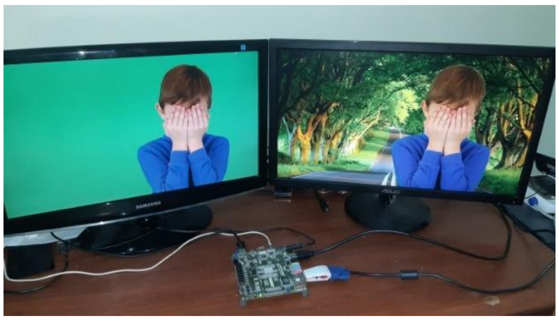


Fig. 10. FPGA-based real-time evaluation of the designed Chroma-key effect system.

The last step of our project is to achieve the hardware implementation of the Chroma-key effect system on the FPGA platform. The first thing is to generate the equivalent VHDL code for our system from the XSG. Then send the generated code to the ISE-created project, which contains additional files to control the HDMI port.

After compiling the ISE project, a programming file will be generated for the configuration of the Atlys board. The FPGA-based system will drive the background image signal and the image with a green background to the Chroma-key system. After the input images are processed, the result will be shown on a screen through the HDMI port. The Fig. 9 presents the report of the resources consumed by the implemented system, including the other components, such as the drivers of the HDMI ports.

After configuring the FPGA board, we can check the system operation in real-time, as shown in Fig.10. It is clear from the obtained result that the implemented system detects perfectly the green background of the original image and replaces it with the new background.

VI. Conclusion

In this work, we presented the steps of designing a Chroma-key system and its real-time FPGA-based implementation, from the first step (validation by simulation) to the last step (implementation on FPGA (real-time validation)). We used the XSG tool, which allowed us to create this system quickly and easily; the design of this system may be difficult and time-consuming if using a hardware description language (VHDL, Verilog). XSG allowed us to do the simulation in Matlab as well as the generation of the equivalent VHDL code of the designed system automatically. The designed system consists of an RGB/HSV converter. This converter allows us to easily detect the colors of the pixels to be extracted. In addition, another sub-system extracts the background pixel and replaces it with the pixel of the new background image. The obtained simulation as well as the FPGA-based real-time results showed the good functioning of our system with good performance. Indeed, there is a lot of software to achieve this graphic effect, but little hardware to do that. This project was done on a low-cost FPGA board. It will be very useful for people who want to achieve this effect in real-time. Finally, there are still several things we need to do to improve this project in the future.

References

[1]. V. M. Gelson, "Eduardo AB da Silva," *The Electrical Engineering Handbook*, 2004, vol. 891.

[2]. M. Karen and R. S. Ferguson, *A Hitchhiker's Guide to Virtual Reality*, CRC Press, 2007.

[3]. N. T. Sang and T. Q. Vinh, "FPGA implementation for real-time chroma-key effect using coarse and fine filter," in *Proc. Computing, Management and Telecommunications (ComManTel)*, 2013.

[4]. T. Sameera. (2022). Chroma Keying MATLAB Implementation 1.0. [Online]. Available: https://thilinasameera.wordpress.com/2010/12/03/chroma-keying-matlab-implementation-1-0/

[5]. S. N. Thanh and T. Q. Vinh, "FPGA implementation for real-time chroma-key effect using coarse and fine filter," in *Proc 2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, 2013.

[6]. X. Fan, and M. M. Tanik. "An educational environment for evolutionary and adaptive circuit DESIGN," 2011.

[7]. M. Alina, S. Aupetit, and M. Slimane, "Improving web accessibility for dichromat users through contrast preservation," in *Proc. International Conference on Computers for Handicapped Persons*, Springer, Berlin, Heidelberg, 2012.

[8]. W. Long *et al*., "Implementation of RGB to HSV color space conversion with Xilinx system generator," *Advanced Materials Research*, vol. 816, Trans Tech Publications Ltd, 2013.

[9]. S. Li and G. Gaizhi, "The application of improved HSV color space model in image processing," in *Proc. 2010 2nd International Conference on Future Computer and Communication*, vol. 2, 2010.

[10]. LogiCORE IP DividerGenerator v3.0, 2011.

[11]. Atlys Board Reference Manual, 2011.

Merah Hocine obtained his engineering degree in electronics (communication) from Amar Telidji University of Laghouat –Algeria in 2009 and the magister degree also from Ferhet Abbess University in setif, Algeria in 2012. He got his PhD from Amar Telidji University of Laghouat –Algeria in 2019.Due to his efforts, he benefited from an exceptional national program scholarship from the Algerian government (2018-2020) to Canada, immediately after this training he worked as a head of the physics department at the level of ENS Laghouat , Algeria . Currently , he is an associate professor in the école normale supérieure of laghouat –Algeria. He is mainly interested in research fields like: performance analysis of multicarrier modulation, communication systems, network, multi-carrier waveforms. He is also concerned with certain aspects such as : channel coding, signal processing, PAPR reduction, information security and channel estimation.

Merah Lahcene received the engineering degree in electronics (instrumentation) from University Amar Telidji of Laghouat - Algeria in 2004. He worked as an instrumentation engineer within the national company SONATRACH (2006-2008), where he acquired good experience in the maintenance of gas turbines. He received the M.Sc and Ph.D degrees in telecommunication systems from the University of science and technology of Oran - Algeria in 2010 and 2016 respectively. He is currently a lecturer at the department of electronics, University Amar Telidji of Laghouat. His research interests include are information security, chaos-based secure information, Random number generators, and signal processing on reconfigurable hardware. He has a number of published works in these contexts. He joined a number of research laboratories such as Advanced Microsystems Engineering Laboratory - the University of Quebec (Ottawa - Canada), Coding and Information Security Laboratory, Coding and Information Security Laboratory, the university of science and technology of Oran (Algeria), and Signals and Systems Laboratory, the University Amar Telidji of Laghouat (Algeria).

Noureddine Chaib received the B.E. degree in computer science engineering from the University of Laghouat, in 2007, the M.E. degree in computer science from the University of Batna, in 2011, and the Ph.D. degree in computer science from the University of Laghouat. He serves as a Treasurer of the IEEE Algeria Section. He is currently an Associate Professor with the Computer Science Department and the Chief Information Officer of the University of Laghouat. His research interests include security, privacy, mobile, and vehicular networks and other networking topics. He is a member of the Computer Science and Mathematics Laboratory (LIM) and the IEEE Vehicular Technology and the Intelligent Transportation Systems Societies.

Ali-Pacha Adda was born in Algeria. He received the engineering degree in telecommunications from the Institute of Telecommunication of Oran-Algeria in 1986; also, he got university degrees in mathematics in 1986 from university of Oran I- Algeria and a magister in signal processing in November 1993, and later he obtained his Ph.D. in safety data in 2004 from the University of Sciences and Technology of Oran. He worked in the telecommunications administration (PTT Oran) in the position of head of telephone traffic for two years (1986 -1988), He is currently a professor (teacher/researcher) in the Electronics department of the University of Sciences and Technology of Oran (U.S.T.O). His research interests are coding, cryptography and security, and digital signal processing using reconfigurable hardware. He is currently the head of LAboratory of COding and Security of Information (LACOSI laboratory).