TELKOMNIKA, Vol.17, No.6, December 2019, pp.3003~3009
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v17i6.12710

Transformation and dynamic visualization of images from computer through an FPGA in a matrix of LED

Edwar Jacinto Gómez*[1], Fernando Martínez Santa[2], Holman Montiel Ariza[3]
Universidad Distrital Francisco José de Caldas, Facultad Tecnológica,
Cll 68 D Bis A Sur No. 49F–70, Bogotá D.C., Colombia, South America
*Corresponding author, e-mail: ejacintog@udistrital.edu.co[1],
fmartinezs@udistrital.edu.co[2], hmontiela@udistrital.edu.co[3]

Abstract

This article shows the implementation of a system that uses a graphic interface to load a digital image into a programmable logic device, which is stored in its internal RAM memory and is responsible for visualizing it in a matrix of RGB LEDs, so that This way, the LEDs show an equivalent to the image that was sent from the PC, conserving an aspect ratio and respecting as much as possible the color of the original image. To carry out this task, a Matlab script was designed to load the image, convert and format the data, which are transmitted to the FPGA using the RS232 protocol. The FPGA is in charge of receiving them, storing them and generating all the signals of control and synchronization of the system including the control of the PWM signals necessary to conserve the brightness of each one of the LEDs. This system allows the visualization of static images in standard formats and, in addition, thanks to the flexibility of the hardware used, it allows the visualization of moving images type GIF.

***Keywords**: digital image processing, dynamic visualization, FPGA, LED array*

1. Introduction

The processing and visualization of digital images using complex computer algorithms makes it imperative to use FPGA's [1, 2], the performance of this type of devices is crucial for the correct functioning of the different human machine interfaces [3], where information processing is required with high speed of refreshment, quality and use of complex algorithms with low delay [4], which causes them to be used in high speed image management applications. In visualization systems that require complex calculations [5], with changes of view of the image to be displayed [6], writes must be made in the internal memory of the device without losing processing speed [7, 8], together with the possibility of having a block of information of variable size so that the implementation is viable [9], characteristics that only has a reconfigurable system type FPGA [10].

Around the transformations [11], extraction of characteristics [12] or any other type of massive matrix operation, object detection and recognition and route planning, Programmable Logic Devices (PLD) have been used, since they require low power and a small amount of space to work [13, 14] in addition to having the ability to perform real-time implementations, working with static and moving images [15], which requires thousands of iterations per second, with maximum use of FPGA resources, both in the combinatorial part (LUT) as in its storage part (memory bank), making efficient the implementations made with this type of devices [16, 17]. The acquisition of real-time images has become an everyday task, using conventional cameras and stereo cameras for capturing in 3D [18, 19], in which applications with digital programmable devices have been made to perform high-speed processing, achieving solutions where a single FPGA is used that replaces systems of multiple architectures that required connections and conversions of information, slowing down actions on processes in jobs where short response times are required [20, 21].

The flexibility and performance of hardware development using standard hardware description languages, regardless of the size and architectures of different FPGA's, offers the possibility to test and compare different algorithm resolution methods, which could give different solutions and indicate what family and what architecture would be the most appropriate to use in digital image processing applications [22, 23], using the combination of Soft Cores together with hardware designed to accelerate information processing. An application was

Received March 19, 2019; Revised July 2, 2019; Accepted July 19, 2019

designed that allows visualization of a resident image in a personal computer, which, through a simple application and a custom designed hardware with a low energy consumption [24, 25], allows its visualization in a matrix system of LEDs of a considerable size.

2. Methodology

In this application, a methodology that complies with the Top-Down philosophy has been considered as one of the most used in reconfigurable hardware systems, therefore, a general block diagram has been made of the implemented solution and from this, each one of the functional blocks has been designed, see Figure 1. The work done begins with the design of a software script on the personal computer, which sends the data of the digital image in binary-ascii format through a serial port emulated towards the FPGA, after that, the programmable logic device processes the information, sends the data to a power system that supports the electrical requirements at the speed required by the array of LEDs and in this way the correct visualization of the information is made.

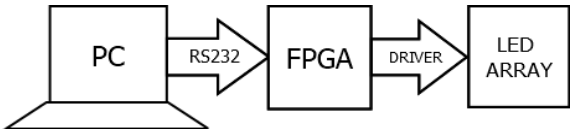

Figure 1. Generalized diagram of the application

3. Implementation and Results

The first part of the work is shown in the following code, the script reads different formats of images (jpg, png and gif), it will make the conversion to a matrix of valor to create the ROM file, it will be read for the FPGA and put in to LED matrix.

```
imread () /imshow () to verify that it is in full form for processing.
im = imread('Mario_8bits.PNG');
figure ()
imshow(im)
```

In the following code segment, to scale the image the imresize () function is used, which gives us the facility of scaling the value of a square or rectangular matrix to the size to be visualized in the array of LEDs, a bicubic interpolation algorithm was used, a process that guarantees that do not drastically change the information that will be sent to the FPGA.

```
im_small = imresize(im, [32 32], 'cubic');
figure ()
imshow(im_small)
```

The next code shown that the matrices for the images that are entered were defined for a size of 32x32 being this a square matrix, the selection of the size is based on the space to occupy within the ROM memories that will be of 1024 bits maximum size of each block of memory of a standard FPGA, which has a capacity of 8 times the value of each block generated.

```
im_small_r = im_small(:,:,1);
im_small_g = im_small(:,:,2);
im_small_b = im_small(:,:,3);
```

The image already scaled is broken down into the 3 matrices of the channels that make up a digital image R (network), G (green) and B (blue), maintaining the intensity characteristics to conform the different color possibilities, in each matrix the values of each pixel have a magnitude between 0 and 255. The combination of these channels conforms the scaled image, which will be used to occupy value by value the array of LEDs for their respective

visualization. Figure 2 shows the matrices or channels R, G and B in which the most characteristic values of the image to be displayed can be observed. If they are compared, the first array has a prominent white value, which means a 255 magnitude or a strong red color, in the image where it is observed completely and with darker tones blacks are 0 or a red value in a blue representation image.

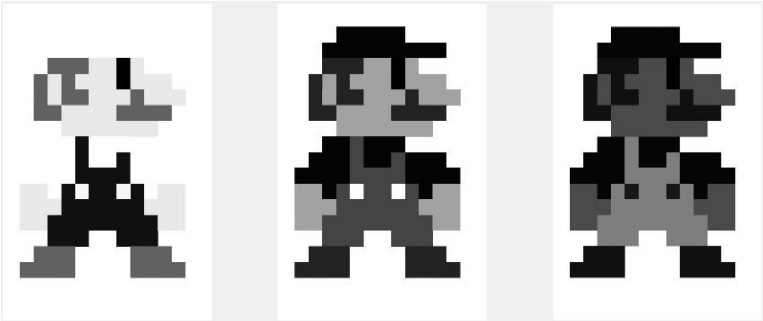

Figure 1. RGB channels of the image that will be scaled and sent to the FPGA

3.1. Transmission of Information Blocks from Matlab to FPGA-send in RGB

For the reception of the data is done with a CORE performed directly by Xilinx [17], as one of the modules implemented for the 8 Bit PICOBlaze processor, since this CORE is performed at very low level only uses 22 Slices of the one FPGA Spartan type to perform the data reception. Figure 3 shows the block diagram of the reception design for RS232 on a Xilinxs FPGA made by Mr. Chapman.

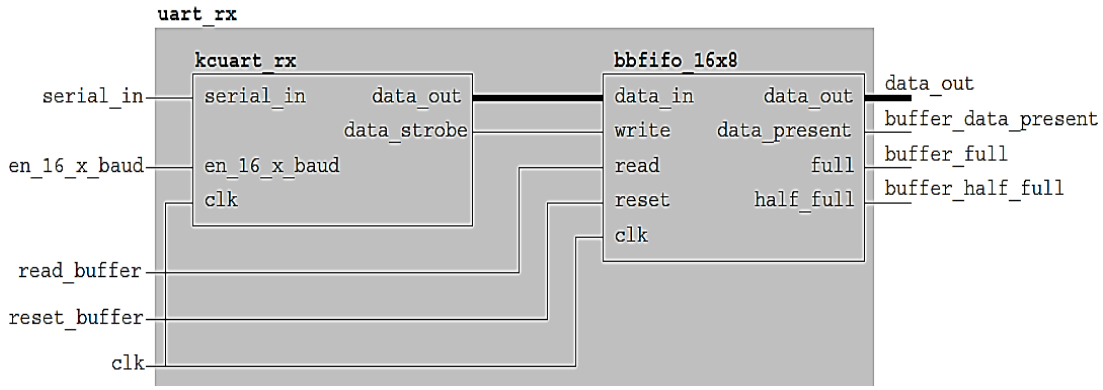

Figure 2. RS232 reception block; core provided by Xilinx for PicoBlaze [26]

The input channel of the module (`serial_in`) receives the data to work with the waiting of a rising edge for the transmission of the 8 bits. (`data_out`) is the parallel output of the received data. (`read_buffer`) in high state of responsible for the transmission of the data to a FIFO memory. (`reset_buffer`) in active state delete the data saved under the count of 16 data blocks. (`en_16_x_baud`) in active state makes the counting of 16 cycles of 8 bits which culminates the storage so that the entire block created in the 16x8 memory will be transmitted. (`buffer_data_present`) gives an input prompt or that there is information block in the memory. (`buffer_full` or `buffer_half_full`) indicates in high mode the full state of the memory, indicates how their names say full or half full 16x8/8x8 when it is full memory communicates this state and activates the complete transfer of the memory block wherever your processing is necessary. (`clk`) system clock for a synchronous multiple system.

3.2. Reception and Storage of Blocks of Information in RAM

In the design and implementation of the system it is observed how the CORE of the Uart provided by Xilinx is controlled, by means of a finite state machine, this is in charge of reading the data from the FIFO memory and passing it to the internal RAM memory of the FPGA, in this way it is possible to read blocks of information from the USB-Serial port of the computer to the internal memory, which will then be responsible for displaying the information, as shown in Figure 4.

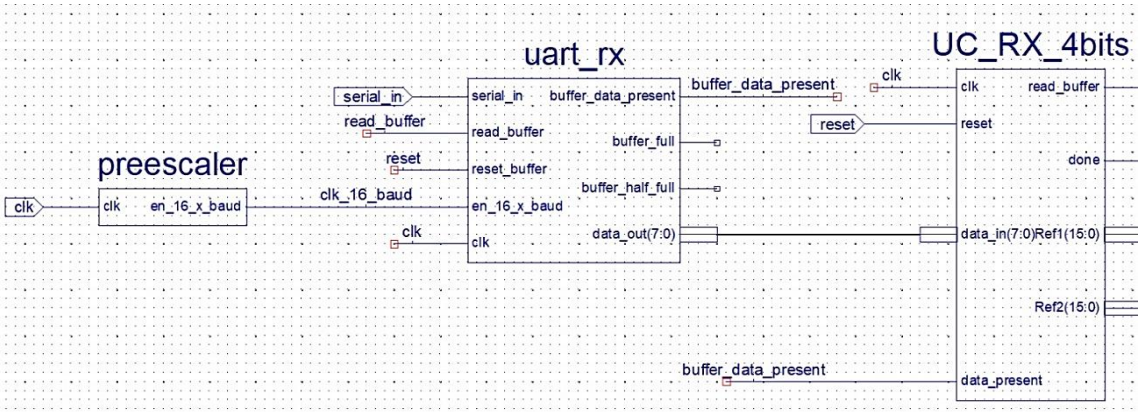

Figure 3. Block diagram of the serial reception system

3.3. Generation and Control of PWM

Next, one of the most important hardware blocks is described, which allows to read the value of each of the pixels in all its channels and convert them into a brightness intensity value of the points (LEDS) in the matrix, varying from dynamically the pulse width, all this must be done at the frequency at which the dynamic display works, in other words, at the same time that the data are being enabled to be shown on the display, the pulse width must be changed. It gives the intensity to each of the points of the RGB matrix. The following block of code describes the 8-bit PWM module in the VHDL language, a highly used language, since it is a standard language that allows the description of hardware at a high level of abstraction.

```
entity PWM_01 is
generic (generic : width := 8);
    Port ( clk,reset   : in  STD_LOGIC;
           Duty            : in  STD_LOGIC_VECTOR (width-1 downto 0);
           PWM_out         : out  STD_LOGIC);
end PWM_01;
```

In this section of code, the declaration of the inputs and outputs of the system is made; a clock and reset as base signals in any description of a sequential circuit are obtained, in addition to the Duty input that comes from the RAM memory, where they are stored every one of the values of the intensity of the pixels to be displayed in the LED's matrix and finally the output of the pulse width modulation, which will be the last signal to be displayed in each of the points of the matrix.

```
architecture Behavioral of PWM_01 is
signal counter : STD_LOGIC_VsECTOR (width-1 downto 0);
begin
process(clk,reset)
begin
if reset = '1' then
    counter <= (others => '0');
    PWM_out <= '0';
elsif rising_edge(clk) then
    counter <= counter + 1;
    if counter <  Duty then
```

```
        PWM_out <= '1';
    else
        PWM_out <= '0';
    end if;
end if;

end process;
end Behavioral;
```

In the architecture the description of a counter is shown, but in this particular case it is compared with the input of Duty, which gives the value that the useful cycle of the system will be given and will give the intensity of brightness of each one of the LED's.

3.4. Dynamic Visualization on LED Array

When performing a dynamic visualization in a LED array, the value of the current necessary to see each of the points correctly and have the necessary brightness to handle an RGB matrix must be taken into account. A circuit must be made to deliver the necessary current with the required speed, in this case, some tests were performed with the integrated circuit MAX7219, driver for 8x8 LED array management, which delivers an output of 500 milli Amperes, but the refresh rate of the system is not achieved, since, although the working speed is 10 Mhz for its SPI interface, it only achieves a refresh rate of 800 Hz. For these reasons, it was decided to make a circuit with a Mosfet N channel and its respective driver, since this power element gives a maximum current of 12 Amps continuously and up to 30 Amps peaks and a working speed over 300 Khz, remembering to respect the minimum on and off times of the device, avoiding that the switch is in short circuit or in an indeterminate state.

Having a system that requires handling 32 rows and 32 columns to use the lowest number of outputs of the FPGA, the columns were handled with two shift registers, which only require the input of `Din` and `Clk` to perform said task, the clock must have a working frequency of 17.2 kHz to achieve a refresh rate of each of the columns at 45 Hz, this circuit with shifting registers is handled with a Darlington-type current switch of 1.5 amps output for each of the 32 columns and a maximum working speed of 40 Khz, avoiding having pulse widths less than 10% and greater than 90%.

On the other hand, there were 8 outputs of PWM, which were "demultiplexed" to handle the 32 rows, each of them has a power circuit with a Mosfet Channel N in sink configuration. In other words, the current is injected through the columns and the PWM module controls the passage of the current through each LED and towards the ground, it is required that the frequency of this PWM is at least 10 times of the speed of visualization of the columns, that is why the PWM module of 176 Khz was configured with an 8 bit resolution, the pulse width is read from the RAM memory where the channels of the image to be displayed are stored. Figure 5 shows a block diagram detailing the system. It is important to emphasize that both the frequency of the PWM of each channel, the multiplexing speed of the signals and the speed of reading of the memory, would be difficult to achieve with other types of devices than an FPGA.

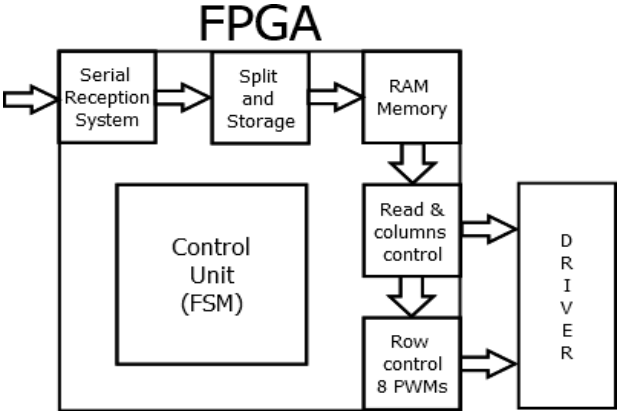

Figure 5. Detailed block diagram

4. Conclusions

The treatment of images on the architecture of FPGA's is facilitated by having the ability to perform processes in parallel, both for preprocessing and for the visualization of an image, together with some algorithms that require high demand for embedded processing, but with low amount of hardware, this advantage is evidenced in the refresh rate of the frames shown in the array of LEDs per second, at this point it is important to emphasize that it would be very difficult to increase the speed of work of the system if you were not working with a FPGA.

Thanks to the multiple hardware tools of the FPGA, it is possible to work in conjunction with the computer to perform a fundamental part of the implementation in the reading and storage of FIFO memories, through the USB port to the FPGA and then have data for the visualization in the array of LEDs. The great demand of power in this type of applications requires a control for the injection of current to each one of the points of the matrix of LED, for that reason the capacity of handling of the signals of control and feeding is required, the FPGA fulfills with the technical specifications of this type of implementations. A circuit for dynamic visualization is shown, where it was necessary to design a high-speed current amplifier circuit to perform the correct visualization of the images already processed according to the theoretically planned form.

Acknowledgment

This work was supported by the Universidad Distrital Francisco José de Caldas Technological Faculty. The views expressed in this paper are not necessarily endorsed by the University. The authors thank the research group ARMOS for the evaluation carried out on prototypes of ideas and strategies.

References

[1] Zhang H. *Realization on image 2D-DCT sparse transform based on FPGA*. 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). Chengdu. 2017: 175–178.

[2] Zhang S, Zhu J, Wang C. *Application of TEXTIO in the Simulation of FPGA Image Processing Algorithm*. *3rd International Conference on Information Science and Control Engineering (ICISCE)*. Beijing. 2016: 235-238.

[3] Rupani A, Whig P, Sujediya G, Vyas P. *A robust technique for image processing based on interfacing of Raspberry-Pi and FPGA using IoT*. 2017 International Conference on Computer, Communications and Electronics (Comptelix). Jaipur. 2017: 350-353.

[4] Hu T, Ikenaga T. *FPGA implementation of high frame rate and ultra-low delay vision system with local and global parallel based matching*. 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA). Nagoya. 2017: 286-289.

[5] Morales-Romero JJ, Gomez-Castaneda F, Moreno-Cadenas JA, Reyes-Barranca MA, Flores-Nava LM. *Time-multiplexing cellular neural network in FPGA for image processing*. 2017 14th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). Mexico. 2017: 1-5.

[6] Zhang C, Liang T, Mok PKT, Yu W. FPGA Implementation of the Coupled Filtering Method and the Affine Warping Method. *IEEE Trans Nanobioscience*. 2017; 16(5): 314–325.

[7] Funasaka T, Iwase M, Fujisawa K, Hatakeyama S. Visualization of Stability of Dynamical Systems by 3D Graphics Supported by Cluster Computing. *Work Intell Data Acquis Adv Comput Syst Technol Appl*. 2015: 588–592.

[8] Liu H, Yu F. *Research and Implementation of Color Image Processing Pipeline Based on FPGA*. 2016 9th International Symposium on Computational Intelligence and Design (ISCID). Hangzhou. 2016: 372-375.

[9] Son TN, Hoang TM, Dzung NT, Giang NH. *Fast FPGA implementation of YUV-based fractal image compression*. 2014 IEEE Fifth International Conference on Communications and Electronics (ICCE). Danang. 2014: 440-445.

[10] Rahangdale S, Keijzer P, Kruit P. *MBSEM image acquisition and image processing in LabView FPGA*. 2016 International Conference on Systems, Signals and Image Processing (IWSSIP). Bratislava. 2016: 1-4.

[11] Akkad G, Elhassan M, Ayoubi R. *FPGA hardware architecture for stereoscopic image compression based on block matching, watermarking and hamming code*. 2016 International Image Processing, Applications and Systems (IPAS). Hammamet. 2016: 1-5.

[12] Dhanabal R, Sahoo SK, Bharathi V, Dowluri K, Varma BSRP, Sasiraju V. *FPGA based image processing unit usage in coin detection and counting*. 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]. Nagercoil. 2015: 1-5.
[13] Atay M, Yalcin ME. *A parallelized distance transformation architecture for FPGAs*. 2013 European Conference on Circuit Theory and Design (ECCTD). Dresden. 2013: 1-4.
[14] Son TN. *Efficient implementation of a fractal color image compression on FPGA*. 2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR). Hanoi. 2013: 184-189.
[15] Manu KS, Rekha KR, Nataraj KR. *FPGA Implementation of Image Block Generation and Color Space Conversion for the Gaussian Mixture Model*. 2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT). Bangalore. 2017: 24-28.
[16] Ryoo JR, Lee ES, Park HK. *Real-time implementation of an LUT-based image warping system*. IEEE ISR 2013. Seoul. 2013: 1-4.
[17] Kiran S, Nadhini S, Jaya J. *Design and implementation of FPGA based invisible image watermarking encoder using wavelet transformation*. 2013 International Conference on Current Trends in Engineering and Technology (ICCTET). Coimbatore. 2013: 323-325.
[18] Michalik S, Michalik S, Naghmouchi J, Berekovic M. *Real-Time smart stereo camera based on FPGA-SoC*. 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids). Birmingham. 2017: 311-317.
[19] Jin KC, Lee KS, Kim GH. *High-speed FPGA-GPU processing for 3D-OCT imaging*. 2017 3rd IEEE International Conference on Computer and Communications (ICCC). Chengdu. 2017: 2085-2088.
[20] Xu Y, Zhou Q, Gong L, Zhu M, Ding X, Teng RKF. High-Speed Simultaneous Image Distortion Correction Transformations for a Multicamera Cylindrical Panorama Real-time Video System Using FPGA. *IEEE Transactions on Circuits and Systems for Video Technology.* 2014; 24(6): 1061–1069.
[21] Mazinan AH, Esmaeili A. *An algorithm for extracting the phase of the fringe patterns with its applications to three-dimensional imaging through FPGA based implementation*. 2016 International Conference on Industrial Informatics and Computer Systems (CIICS). Sharjah. 2016: 1-5.
[22] Frid N, Mlinaric H, Knezovic J. *Acceleration of DCT transformation in JPEG image conversion*. 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija. 2013: 1292-1295.
[23] Padmavati S, Meshram V, Jayadevappa. *A hardware implementation of discrete wavelet transform for compression of a natural image*. 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET). Chennai. 2017: 1-5.
[24] Taher F, Zaki A, Elsimary H. *Design of low power FPGA architecture of image unit for space applications*. 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS). Abu Dhabi. 2016: 1-4.
[25] Babu L, Nm L. *FPGA Implementation of Energy Efficient Approximate Multiplier with Image Processing*. 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI). Tirunelveli. 2018: 637-641.
[26] Chapman K. Pico Blaze DS2432 Communicator. Xilinx Ltd. 6th April 2006.