ISSN 2395-8618

Raster data implemented in a FPGA device

J. Sandoval-Gutierrez, J.A. Alvarez-Cedillo, J.C. Herrera-Lozada, T. Alvarez-Sanchez and M. Olguin-Carbajal

Abstract—**The instrumentation for image processing sends the information through a communication interface, and the applications are developed by two general methods: a Software Development Kit as a PC-based programming language and a hardware implementation. In the first method, the users are limited by other processess that are being executed at the same time, energy consumption, size of the system, mobility, and the visual interface. Conversely, an embedded system provides more efficient features than general purpose software. To confirm this idea, in this paper VGA display is used as the output reference to compare the results applying a set of raster data as portable pixmap (*.ppm), graymap (*.pgm) and bitmap (*.pbm). Two tests in a different field of study are comparing: pre-processing of an image format with four operations: original, grayscale, binary and inverted; the other test is a laser triangulation measurement system. In the tests: RPLIDAR A1M1-R1 Development Kit, ImageJ, GIMP and a Spartan 3E FPGA hardware 12 bits RGB output image was used as a reference at 640x480 pixels in a conventional computer monitor. The method proposed as an image processing was compared with a conventional computer, and the results in the visualization were similar in both cases, but with less energy consumption, less size and capacity for mobile systems.**

Index Terms—**FPGA, Imaging processing, Lidar, Netpbm format, VGA.**

I. Introduction

IMAGE processing is a part of signal processing that uses some segmentation that researchers are using in many fields, such as measuring [1] [2] laser scaning [42] [43] [44] [45], food [3], surgery[4], corrosion [5], industrial [6] [7], particles [8] and others. A general framework image processing according to [9] [10] [11] is:

- Image acquisition
- Pre-processing
- Segmentation
- Representation
- Classification

A. Image acquisition and pre-processing

There are two ways to obtain image data either cases (Electronic device or software) the result is a digital image storage in an array of bits, within a memory using a particular format file. In a review of various applications aimed at image processing, the characteristics of an image were found in different disciplines. Specifically an overview of the major file formats currently used in medical imaging, define universal concepts to all file formats such as pixel depth, photometric interpretation, metadata and pixel data [12]. A particular software package for image processing of electron, micrographs, interpretation of reconstructions, molecular modeling and general image processing generate a text file [13].

Some image file format provided by GIMP software are: Animation .flic, Animation .mng, PostScript .ps, Icon .ico, Digital Imaging and Comunications in Medicine .dcm .dicom, BMP Image .bmp, Photoshop .psd, Encapsulated PostScript .eps, GIF .gif, IRIS de Silicon Ghraphics .sgi, JPEG .jpg, PBM .pbm, PGM .pgm, PIX .pix, PNG .png, PNM .pnm, PPM .ppm, SUN .im1, im8 .im24 im32, TarGa .tga, TIFF .tif, X BitMap .xbm,X pixMao .xpm, Zsoft PCX .pcx, KISS CEL .cel, OpenRaster .ora, GIMP. pat, PDF .pdf and Flexible image .fit. For example JPEG 2000 standard (Joint Photographic Experts Group) file format is used widely on the internet, color facsimile, printing, scanning, digital photography, remote sensing, mobile, and others. It is processed with the block tiles to produce a JPEG file [14] such as occurs with BMP, PNG, TIFF, among others. All formats have implicit features as image nature, resolution, number of colors [15] [16] even a Holographic Data System applies a similar storage [17].

This paper focused on three file formats as mentioned above: PPM, PGM and PBM [18] [19] [20] in order to share the data with other devices. The Netpbm is a toolkit for the manipulation of graphic images including conversion of images from a variety of different formats. Also it is portable to Unix-based systems, Windows, Mac OS X, VMS and Amiga OS. Netpbm was developed to be a single source for all the primitive graphics utilities [21] and in this paper on hardware applications.

B. LiDAR

LiDAR is a distance sensor [42] that allows showing environmental visual information through a grid map or point-cloud. Normally it is mounted on mobile systems such as vehicles [42] [44], UGV [41], coordinate motion [45] and static environment [43]. An RPLIDAR A1M1-R1 module tested with the SDK was connected to USB from a conventional computer and a set of points over a radar background is shown on a screen as in Figure 6. The raw data sent by the LiDAR is an array of values representing a distance and orientation in digital bits. The hardware implementation proposed avoids connecting the device to a computer, but allows drawing the color of each pixel as an image pre-processing.

Manuscript received Month DD, YYYY; revised Month DD, YYYY.

We would like to thank the Cátedras CONACYT for the financial support of this work

J. Sandoval-Gutierrez is with the Universidad Autónoma Metropolitana at Lerma, J.A. Alvarez-Cedillo, J.C. Herrera-Lozada and M. Olguin-Carbajal are with the Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC), Instituto Politécnico Nacional (IPN), Juan de Dios Bátiz s/n, C.P. 07700 D.F., México (e-mail: jacobosandoval@hotmail.com; jaalvarez,jlozada@ipn.mx) T. Alvarez-Sanchez is with the Centro de Investigación y Desarrollo de Tecnología Digital(CITEDI), Instituto Politécnico Nacional (IPN), Av. del Parque No. 1310, Mesa de Otay, Tijuana, Baja California, México

ISSN 2395-8618

(a) PGM Image

```
P2
# test.pgm
19 7
15
0 0  0  0  0  0  0 0  0  0  0  0  0 0 0 0 0 0 0  0
0 15 15 15 15 0 0  11 11 11 11 0 0 5 5 0 5 5  0
0 15 0  0  15 0 0  11 0  0  0  0 0 5 0 5 0 5  0
0 15 15 15 15 0 0  11 0  11 11 0 0 5 0 5 0 5  0
0 15 0  0  0  0 0  11 0  0  11 0 0 5 0 5 0 5  0
0 15 0  0  0  0 0  11 11 11 11 0 0 5 0 5 0 5  0
0 0  0  0  0  0 0  0  0  0  0  0 0 0 0 0 0 0  0
```

(b) PGM code

Fig. 1: PGM image representation

C. Visualization

After the digital image has been stored in any electronic device by any format, the image is displayed using a screen electronic device without specific software [6] [7] [28], MATLAB® [1] [2] [3] [5] [8] [43] [25] [26] , JAVA ® [1], C language [26] [42] [44], ROS [43], Qt SDK [41] and also alternative methods for MPI CUDA in HPC [27].

This process is a common task and known as visualization. However, when a device with embedded screen (display size, resolution, and color) shows the raw data in real time, there is no possibility of knowing if the file format shared is the same as the original. Since the process is a Black-box for the users. An alternative solution is a visual direct manipulation as a software [22]. In this paper, an FPGA-based implementation is shown using a VGA display.

D. Netpbm kernel

PPM is a raw ASCII image format and is a suitable string representation of an image in a file. Each pixel contains ASCII information in an arbitrary size. In the first line, a P3 tag(color file format) is used, in the second line the columns and rows number must be added, in the third line an RGB maximum number value, and in the other lines the rest of the data.

Another format is PBM, where each pixel is represented with 0 or 1 (black and white); white space in the raster section is ignored and the heading in the first line is P1 instead of P3 used by PPM [29].

PGM is a format consisting of four lines, providing a maximum of 256 gray scale levels or 8 bit data per pixel [30] [32].

A sample code of PGM file is shown with a P2 indicating a gray level from 0 up to 15 values, 19 columns, seven rows and ASCII information of one character is equal to one pixel. In Figure 1 the result of this code is shown (test.pgm), and the file was generated by ImageJ and GIMP Software.

E. VGA Display

The general considerations for VGA display controller have been referenced by development in Verilog Hardware Description Language [33] and VHDL [34] [36] [37].

In Table I VGA signal 640 x 480 @ 60 Hz Industry standard timing is shown.

TABLE I: Timing

Horizontal timing (line)

Scanline part	Pixels	Time (μs)
Visible area	640	25.422045680238
Front porch	16	0.63555114200596
Sync pulse	96	3.8133068520357
Back porch	48	1.9066534260179
Whole line	800	31.777557100298

Vertical timing (frame)

Frame part	Lines	Time (μs)
Visible area	480	15.253227408143
Front porch	10	0.31777557100298
Sync pulse	2	0.063555114200596
Back porch	33	1.0486593843098
Whole frame	525	16.683217477656

The Spartan-3A FPGA Starter Kit board includes a VGA display port via a DB15 connector with a red, green, and blue signal. VGA display port provides 4-bit RED, 4-bit GREEN, 4-bit BLUE, (444 color), or 4,096 possible colors. In (1) the color output is described.

$$\text{color}_{\text{out}} = \frac{\text{vga}[3:0]}{15} \times \text{color} \tag{1}$$

II. Test design

This section specifies the characteristics utilized to produce a VGA output on FPGA. The first step is to read a file with the raw data saved in the RAM memory block. The RAM memory has three parameters to set: the address vector (depth), width vector (value) and writing in an enable signal.

Read after write is used to compute three functions through a processing module: inverted function (2) gray (3) and binary (4).

$$\text{Inverted}_{\text{out}} = 2^{\text{ vga}[3:0]} - \text{ color}_{\text{out}} \tag{2}$$

$$\text{Gray}_{\text{out}} = \frac{\sum\limits_{RGB} (\text{color}_{\text{out}})}{3} \tag{3}$$

$$\text{Binary}_{\text{out}} \begin{cases} 1 & \text{if } (\frac{2^{\text{ vga}[3:0]}}{2} > \text{color}_{\text{out}}) \\ 0 & \text{else} \end{cases} \tag{4}$$

This module reads all the addresses of the raw data, computing (4), (3) and (2), and writing in a new RAM memory section.

The VGA controller with 25 MHZ clock (clk2) reads all the memory block and creates a synchronization with the data and the addresses. The data must be written in the vector before the horizontal and vertical requires it.

Finally, VGA output port receives four images of VGA controller and shows the results on the computer screen. A diagram of the design is shown in Figure 2.

ISSN 2395-8618

A. Reading a COE file

A memory coefficient (COE file) loaded in the initialization with a single port A, 12-bit width, (25 600 deep RAM). The syntax is:
memory_initialization_radix = 16;
memory_initialization_vector = 100, 200, . . . 100, 200;

B. VGA controller

A set of six signals selected is: address, data, clock 2, synchrony, horizontal and vertical value for VGA controller. The process begins when the horizontal value is greater than 144 and less than 784, and the vertical value is greater than 31 and less than 511 while another parallel process reads the data with its corresponding address. The relation (5) describes its values by section with two clocks (clkdiv = 25MHz and clk = 50 Mhz).

$$[\mathrm{h}][\mathrm{v}]_{\mathrm{value}} \times \mathrm{clkdiv} = (\mathrm{data}_{\mathrm{address}} \times \mathrm{clk}) \times \mathrm{sync} \qquad (5)$$

C. Processing module

The data of RAM memory is divided into four sections. The processing module reads and computes (4), (3) and (2) address by address in a parallel process to the VGA controller. The algorithm uses a single instruction multiple data streams (SIMD) [38] [39].

D. VGA output port

The Spartan® 3A FPGA Starter Kit board, includes an HD-DB15 female connector with the horizontal sync signal (row), the vertical sync signal (column); these two continuously running counters from the address into a video display buffer (RGB Values) [40].

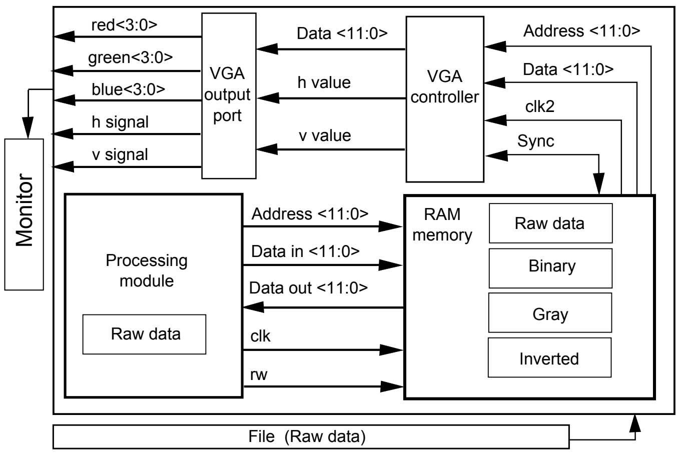
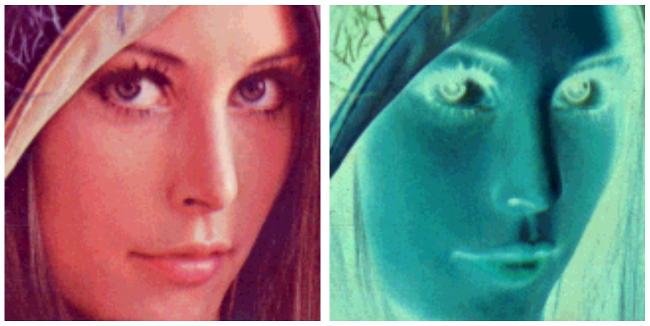


Fig. 2: Test design of FPGA implementation

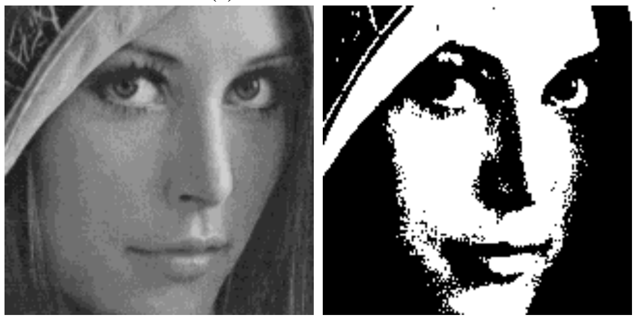
III. Testing

A. Image Proccessing Software

A Lena image has been tested and the result of ImageJ software is shown in Figure 3a 12-bit RGB (1), 3a inverted (2), 3b gray (3) and 3b binary (4).



(a) 12-bit RGB and inverted



(b) Gray and bin

Fig. 3: Lena image processing using ImageJ software

B. Image LiDAR

The original image has a set of 360 distances and angles that are display in the demo application developed by the SDK of the manufacturer. The on-line data is shown in Figure 4, but it is not clearly visible to the human eye.

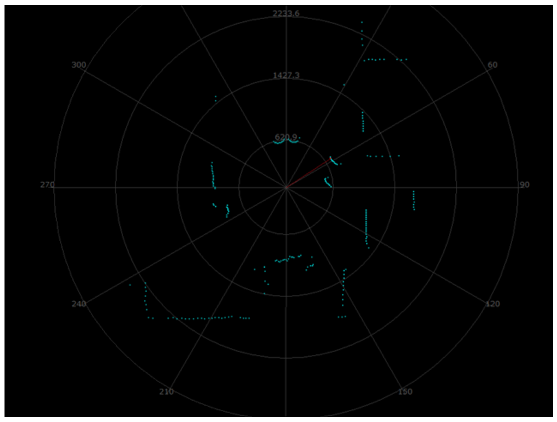


Fig. 4: LiDAR image by SDK's manufacturer

The final tests are shown in Figures 5 and 6, the outputs by software is shown on the left and the hardware output is shown on the right.

IV. Analysis and Discussion

In the first test a comparative table in II was filled out with five features: hardware type, software or file format, energy consumption, mobility and finally the system size. The four cases were included in the proposed design having an FPGA implementation with Mif file [33] and Hexadecimal file [35]. It

ISSN 2395-8618

means that an embedded application requires a more suitable format in order to be manipulated, but there is no problem with the format file in PC-based processing. The computer uses 1000 % more energy than FPGAs implementation, and this wasted energy avoids reducing the size of the system and consistently only static applications could be developed.

TABLE II: First test comparative features with the proposed design

Application	Hardware	Software / File Format	Watts	Mobility	Size
Proposed design	FPGA		+5 W	Yes	Small
Image [33]	FPGA	mif file	+5 W	Yes	Small
Image [34]	FPGA	—	+5 W	Yes	Small
Image [35]	FPGA	Hex File	+5 W	Yes	Small
Image [3]	PC	Matlab	+65 W	No	Normal
Image [4]	PC	Matlab	+65 W	No	Normal
Image [5]	PC	Matlab	+65 W	No	Normal
Image [6]	PC	SDK	+65 W	No	Normal
Image [8]	PC	MatLab	+65 W	No	Normal
Image [13]	PC	Bsoft	+65 W	No	Normal
Image [22]	PC	Palimpses	+65 W	No	Normal
Image [43]	PC	Matlab	+65 W	No	Normal
Image [24]	PC	Java	+65 W	No	Normal
Image [25]	PC	Matlab	+65 W	No	Normal
Image [26]	PC	Matlab	+65 W	No	Normal
Image [30]	PC	Java	+65 W	No	Normal
Image [31]	PC	Matlab	+65 W	No	Normal

In the second test a comparative Table in III as in TableII was compared. The most similar applications are developing with a low consumption energy technology [41] where the software provides sufficent resources. The other applications are a typical PC-based SDK with high wasted energy and normal size that is not efficient for a mobile robot applications.

TABLE III: Second test comparative features with the proposed design

Application	Hardware	Software	Watts	Mobility	Size
Proposed design	FPGA		+5 W	Yes	Small
Robot [41]	Intel Atom	Qt SDK	+9 W	Yes	Small
Measure [1]	Pc	Kinect	+65 W	No	Normal
Measure [2]	PC	MatLab	+65 W	No	Normal
Measure [42]	PC i7	Open CV	+95 W	No	Normal
Measure [43]	PC	ROS	+60 W	No	Normal
Measure [44]	PC i5	C++	+73 W	No	Normal
Measure [45]	Pc core2	unknown	+65 W	No	Normal

V. Conclusion

Raster data as Netpbm is a compatible file format that could be implemented in embedded systems such as the FPGA proposed design and other similar cited papers. The compressed algorithm used by JPGE, PGN and others is not a suitable format for the hardware applications. While a 444 RGB and 160 x 160 pixels *.jpg and *.ppm file use a variable size from 10 KB up to 346 KB in the hard disk. The memory in the FPGA uses a fixed size of 38.4 KB. The most common applications in image processing are developed using an SDK tool on the computer, but the problem is that the energy consumption is more than a 100 times the embedded application. The design proposed has capacity to be implemented in a mobile robot platform, because it satisfies three necessary conditions. A small size, less than $2\times10^{-3}m^3$, low consumption around $5W$ (consequently avoids a cooler system) and the electronic supports vibration. Only a smart computer has similar characteristics, but this requires an OS sharing the resources and a heat sink that avoids damaging the components. Figure 5 and 6 compare the final VGA output with their counterpart (personal computer). Another characteristic is that it only takes a few seconds to boot the embedded application; conversely the PC lost time booting the OS.

Appendix A
PPM, PGM and PBM File Format Tested

PPM file.

```
P3
160 160
5 1 4 . . .
```

PGM file.

```
P2
160 160
16
3 3 3 . . .
```

PBM file.

```
P1
160 160
1 1 1 . . .
```

Appendix B
Test Code for 8-bit VGA Image and RAM Black Out

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity vga is
 port(
 sw        : IN STD_LOGIC_VECTOR(1 downto 0);
 Led       : INOUT STD_LOGIC_VECTOR(0 DOWNTO 0);
 wea1      : INOUT STD_LOGIC_VECTOR(0 DOWNTO 0);
 addra1    : INOUT STD_LOGIC_VECTOR (14 downto 0);
 dina1     : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
 douta1    : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
 clk       : IN STD_LOGIC;
 red_out   : OUT STD_LOGIC_VECTOR(2 downto 0) ;
 green_out: OUT STD_LOGIC_VECTOR(2 downto 0) ;
 blue_out : OUT STD_LOGIC_VECTOR(1 downto 0) ;
 hs_out    : OUT STD_LOGIC;
 vs_out    : OUT STD_LOGIC_VECTOR
 );
end vga;

architecture Behavioral of vga is

COMPONENT ram
  PORT (
    clka  : IN STD_LOGIC;
    wea   : INOUT STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(14 DOWNTO 0);
    dina  : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
```

ISSN 2395-8618

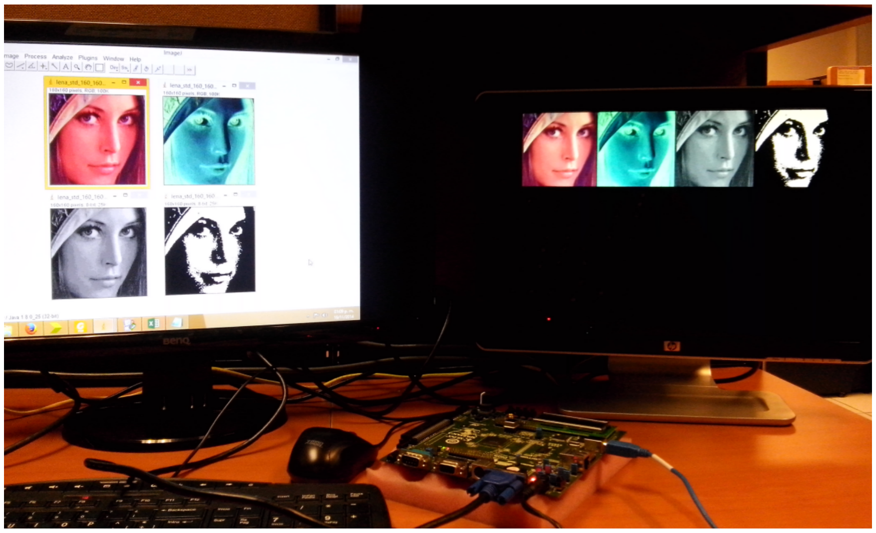


Fig. 5: Image processing ImageJ software (left monitor) and FPGA (right monitor)

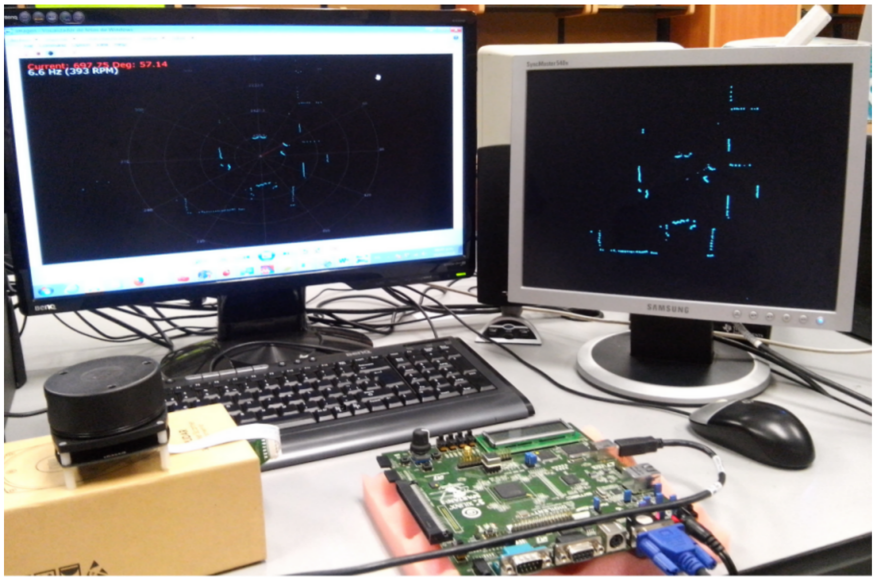


Fig. 6: LiDAR SDK (left monitor) and FPGA (right monitor)

```
  );
END COMPONENT;

 signal clkdiv   : std_logic := '0';
 signal clkdiv2  : std_logic := '0';
 constant hsyn   : integer :=  800;
 constant vsyn   : integer :=  521;
 constant pwh    : integer :=  96;
 constant pwv    : integer :=  2;
 constant bph    : integer :=  48;
 constant fph    : integer :=  16;
 constant bpv    : integer :=  29;
 constant fpv    : integer :=  10;
 constant x0     : integer :=  320;
 constant y0     : integer :=  240;
 signal hc       : integer range 0 to 1024;
 signal vc       : integer range 0 to 1024;
 signal hc0      : integer range 0 to 1024;
 signal vc0      : integer range 0 to 1024;
 signal hvc0     : integer range 0 to 32768;

begin
```

```
process (clk)
begin
        if clk' event and clk = '1' then
                clkdiv <= not clkdiv;
        end if;
end process;

process (clkdiv)

begin

 if clkdiv' event and clkdiv = '1' then
hc <= hc + 1;
if (hc = hsyn) then
vc <= vc + 1;
    hc <= 0;
    end if;

    if (vc = vsyn) then
    vc <= 0;
    end if;

          if (hc > pwh )
```

ISSN 2395-8618

```
        then        hs_out <= '1';

if (vc > pwv )
      then vs_out <= '1';
    else vs_out <= '0';
end if;

If (hc > (bph+pwh) ) and (hc < (hsyn - fph) )
and (vc >= (pwv+bpv)) and (vc < (vsyn - fpv))

        then
 if (hc < (bph+pwh+180) ) and (vc < (pwv+bpv)+180)
 then

        hc0 <= hc-144;
        vc0 <= vc-31;
        hvc0 <= vc0*180+hc0;
        addra1 <= conv_std_logic_vector(hvc0,15);
        red_out   <= douta1(7 downto 5);
        green_out <= douta1(4 downto 2);
        blue_out <= douta1(1 downto 0);
        else     red_out   <= "111";
        green_out <= "000";
        blue_out  <= "11";
 end if;
 else
                red_out   <= "000";
               green_out <= "000";
           blue_out  <= "00";
     end if;
 end if;

end process;

your_instance_name : ram
  PORT MAP (
    clka => clk,
    wea => wea1,
    addra => addra1,
    dina => dina1,
    douta => douta1
  );

with sw select

 wea1<= "1" when "11,
       "0" when others;

led <= wea1;

end Behavioral;
```

References

[1] Omar Rodríguez Zalapa, Antonio Hernández Zavala y Jorge Adalberto Huerta Ruelas. Sistema de medición de distancia mediante imágenes para determinar la posición de una esfera utilizando el sensor Kinect XBOX, Revista Polibits, Vol. 49, 2014, pp. 59–67.

[2] Hofer D. and Zagar B.G., Image processing for calibrating a coordinate measurement set-up, Measurement Science and Technology, Vol. 25, No. 11, 2014, pp. 115003-115017

[3] Hosseinpour Soleiman, Rafiee Shahin, Aghbashlo Mortaza and Mohtasebi Seyed Saeid, A novel image processing approach for in-line monitoring of visual texture during shrimp drying, JOURNAL OF FOOD ENGINEERING, Vol. 143, 2014, pp. 154-166.

[4] Lee Sang Hee, Lee Minho and Kim Hee Jin, Anatomy-based image processing analysis of the running pattern of the perioral artery for minimally invasive surgery BRITISH JOURNAL OF ORAL & MAXILLOFACIAL SURGERY, Vol. 52, No. 8, 2014, pp. 688-692.

[5] Gamarra Acosta, Margarita R., Velez Diaz Juan C., Schettini Castro Norelli, An innovative image-processing model for rust detection using Perlin Noise to simulate oxide textures, CORROSION SCIENCE, Vol. 88, 2014, pp. 141-151.

[6] Deyong You, Xiangdong Gao and Katayama, S. Monitoring of high-power laser welding using high-speed photographing and image processing, Mechanical Systems and Signal Processing, Vol. 49, No. 1, 2014, pp. 39-52.

[7] Lopez F., Maldague X., and Ibarra-Castanedo, Enhanced image processing for infrared non-destructive testing, OPTO-ELECTRONICS REVIEW, Vol. 22. No. 4, 2014, pp. 245-251.

[8] Charonko John J, Antoine Elizabeth and Vlachos Pavlos P., Multispectral processing for color particle image velocimetry, MICROFLUIDICS AND NANOFLUIDICS, Vol. 17, No. 4, 2014, pp. 729-743.

[9] Russ John C., The Image Processing Handbook, Sixth Edition, CRC Press 2011.

[10] Pinoli Jean-Charles, Mathematical Foundations of Image Processing and Analysis 1, John Wiley & Sons, Inc. 2014

[11] Bernd Jähne, Practical Handbook on Image Processing for Scientific and Technical Applications, Second Edition CRC Press 2004.

[12] Larobina Michele and Murino Loredana, Medical Image File Formats, JOURNAL OF DIGITAL IMAGING, Vol. 27, No. 2, 2014, 200-206

[13] Heymann J. Bernard and Belnap David M., Bsoft: Image processing and molecular modeling for electron microscopy, JOURNAL OF STRUCTURAL BIOLOGY, Vol. 157 No. 1, 2007, pp. 3-18.

[14] Skodras A, Christopoulos C. and Ebrahimi, T, The JPEG 2000 still image compression standard IEEE SIGNAL PROCESSING MAGAZINE, Vol. 18, No. 5, 2001, pp. 36-58.

[15] Wiggins RH, Davidson HC, Harnsberger HR, Lauman JR and Goede PA, Image file formats: Past, present, and future RADIOGRAPHICS, Vol. 21, No. 3, 2001, pp. 789-798.

[16] Lins RD and Machado DSA, Comparative study of file formats for image storage and transmission, JOURNAL OF ELECTRONIC IMAGING, Vol. 13, No. 1, 2004, pp. 175-181.

[17] Kim Do-Hyung, Jeon Sungbin, Park No-Cheol and Park, Kyoung-Su, Iterative design method for an image filter to improve the bit error rate in holographic data storage systems, MICROSYSTEM TECHNOLOGIES-MICRO-AND NANOSYSTEMS-INFORMATION STORAGE AND PROCESSING SYSTEMS, Vol. 28, No. 8-9,2014, pp. 1661-1669.

[18] Nadal J, Keeping the bits in place: A case study of raster image migration, SOC IMAGING SCI & TECHNOL, Final Program and Proceedings, 2005, pp. 249-252

[19] ZAMA C M S, System for converting word file into other format e.g. JPEG file format using e.g. HTML software, has CPU to convert individual characters from scanned image into comprehensible code in other format using optical character recognition, patent: ZA200803391-A.

[20] DARGELAS A M, Waveform image e.g. bitmap file, generating method, involves providing localized rendering of temporally organized waveform data based on user input and waveform viewer resolution, and loading waveform images into waveform viewer. patent: US2011234600-A1.

[21] Home page for Netpbm: http://netpbm.sourceforge.net/

[22] Blackwell Alan F., Palimpsest: A layered language for exploratory image processing, JOURNAL OF VISUAL LANGUAGES AND COMPUTING, Vol. 2, No. 5, 2014, 545-571.

[23] Wang Z. and Bovik AC. A universal image quality index, IEEE SIGNAL PROCESSING LETTERS, Vol. 9, No. 3, 2002, pp. 81-84.

[24] Pavel Surynek and Ivana Lukšová, Automated Classification of Bitmap Images using Decision Trees, Revista Polibits, Vol. 44, 2011, pp. 11–18.

[25] Minh N. Do and Martin Vetterli, The contourlet transform: An efficient directional multiresolution image representation, IEEE TRANSACTIONS ON IMAGE PROCESSING,Vol. 14, No. 12, 2005, 2091-2106.

[26] Manjunath BS and Ma WY, Texture features for browsing and retrieval of image data IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, Vol. 18,Issue. 8, 1996, pp. 837-842.

[27] Galizia, Antonella, D'Agostino, Daniele and Clematis, A Clematis, Andrea, An MPI-CUDA library for image processing on HPC architectures, JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS, Vol. 273, 2015, pp. 414-427.

[28] Ishigami Yuta, Waskitoaji Wihatmoko, Yoneda Masakazu, Takada Kenji, Hyakutake Tsuyoshi, Suga Takeo, Uchida Makoto, Nagumo Yuzo, Inukai Junji and Nishide Hiroyuki, Oxygen partial pressures on gas-diffusion layer surface and gas-flow channel wall in polymer electrolyte fuel cell during power generation studied by visualization technique combined with numerical simulation, JOURNAL OF POWER SOURCES, Vol. 269, 2014, pp. 556-564.

[29] Pakhira M.K. and Dutta A., Computing approximate value of the PBM index for counting number of clusters using genetic algorithm, 2011 International Conference on Recent Trends in Information Systems (ReTIS), 2011,mpp. 241-5

ISSN 2395-8618

[30] Shiva Shankar R., Mnssvkr Gupta V, Murthy K.V.S. and Someswararao C., Object Oriented Fuzzy Filter for Noise Reduction of PGM Images, Proceedings of the 2012 8th International Conference on Information Science and Digital Content Technology (ICIS and IDCTA), Vol. 3, 2012, pp. 776-82.

[31] Philippot E., Belaid A. and Belaid Y., Use of PGM for Form Recognition, Proceedings of the 10th IAPR International Workshop on Document Analysis Systems (DAS 2012), pp. 374-378

[32] Abdul-Jabbar I.A.-A, Jieqang Tan and Zhengfeng Hou, Face Recognition Enhancement Based on Image File Formats and Wavelet De-noising, International MultiConference of Computer Scientists (IMEC 2014). Proceedings, Vol. 1, pp. 441-445

[33] Radi H.R., Caleb W. W. K., M.N.Shah Zainudin and M.Muzafar Ismail, The Design and Implementation of VGA Controller on FPGA International Journal of Electrical & Computer Sciences, IJENS Vol. 12, No. 05, 2012, pp. 56-60.

[34] Ashish B. Pasaya and Kiritkumar R. Bhatt, Implementing VGA Application on FPGA using an Innovative Algorithm with the help of NIOS-II, International Journal Of Computational Engineering, Vol. 2, No.3, 771-775.

[35] Guohui Wang, Yong Guan and Yan Zhang, Designing of VGA Character String Display Module Base on FPGA, 2009 International Symposium on Intelligent Ubiquitous Computing and Education, IEEE, 2009, pp. 499-502.

[36] Ioan, A.D., Designing an optimal single chip FPGA video interface for embedded systems, Electrical and Electronics Engineering (ISEEE), 2010 3rd International Symposium on, pp. 58-63.

[37] Van-Huan Tran and Xuan-Tu Tran, An efficient architecture design for VGA monitor controller, Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on, pp. 3917-3921.

[38] Elliott D.G., Stumm M., Snelgrove W.M., Cojocaru C. and McKenzie R., Computational RAM: Implementing processors in memory IEEE DESIGN & TEST OF COMPUTERS, Vol. 16, No. 1, 1999, pp. 32-41

[39] Tessier Russell, Betz Vaughn, Neto David, Egier Aaron and Gopalsamy Thiagaraja, Power-efficient RAM mapping algorithms for FPGA embedded memory blocks, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, Vol. 26, No. 2, 2007, pp.278-290.

[40] Spartan-3A/3AN FPGA Starter Kit Board User Guide 11 UG334 (v1.1) June 19, 2008.

[41] T. Jian, C. Yuwei, A. Jaakkola, L. Jinbing, J. Hyyppa, and H. Hyyppa, NAVIS-An UGV Indoor Positioning System Using Laser Scan Matching for Large-Area Real-Time Applications, Sensors, vol. 14, no. 7, pp. 11805-11824, July, 2014.

[42] Y. Li, and Y. Ruichek, Occupancy Grid Mapping in Urban Environments from a Moving On-Board Stereo-Vision System, Sensors, vol. 14, no. 6, pp. 10454-10478, Jun, 2014.

[43] R. Wang, X. Li, and S. Wang, A laser scanning data acquisition and display system based on ROS, Proceedings of the 33rd Chinese Control Conference, pp. 8433-8437, 2014.

[44] Y. Yongtao, J. Li, G. Haiyan, W. Cheng, and Y. Jun, Semiautomated Extraction of Street Light Poles from Mobile LiDAR Point-Clouds, IEEE Transactions on Geoscience and Remote Sensing, vol. 53, no. 3, pp. 1374-1386, March, 2015.

[45] H. Yuqing, and M. Yuangang, An efficient registration algorithm based on spin image for LiDAR 3D point cloud models, Neurocomputing, vol. 151, pp. 354-363, 3, 2015.