Smart Camera Design

Iván Olaf Hernández [1], Miguel Enrique Bravo Zanoguera[1], Guillermo Galaviz Yañez[1]

[1]Universidad Autónoma de Baja California, Facultad de Ingeniería
Blvd. Benito Juarez S/N, Mexicali, Baja California, México
ivan.olaf@yahoo.com, bravometric@yahoo.com, ggalaviz@uabc.mx

(Paper received on February 29, 2008, accepted on April 15, 2008)

Abstract. The design and implementation of a smart device for image capture, processing and display is presented. The architecture is based on a CMOS image sensor and a pipelined processing structure implemented in a single FPGA device. The embedded system includes: I2C serial communication with the sensor, interpolation processing for demosaicing the Bayer pattern of the sensor, structure for color correction and image display in VGA format. The design holds the on-chip sophisticated functions of the image sensor. It can be used for low-cost vision systems that require real-time processing and portability .

Keywords: FPGA, pipelined processing, VHDL, CMOS image sensor, Smart Camera.

1 Introduction

The design of a smart camera architecture based on a FPGA and a CMOS image sensor is presented. A Cyclone II FPGA from Altera is used as data pixel processor. A Micron's MT9T001 CMOS image sensor is used as the image capture element. Low level image processing operations were implemented using a pipelined architecture, obtaining a high data rate. After an initial latency, every pixel is computed at input data frequency, which gives real-time results.

The design was specified using VHDL. Since it is a standard language the design can be implemented in any FPGA, regardless of the manufacturer or the EDA tool used. This work was done from the scratch: the image capture stage is not based on a commercial camera with an analog or digital video output, instead of that a pipelined processing architecture was implemented to obtain color image from the sensor's raw digital output (Bayer patterned).

The whole system for image capture, processing and display was developed in a single FPGA. Different digital subsystems were implemented to do the following tasks:

- A component to write commands to the image sensor and control its functions. The I2C communication protocol was implemented
- A component to apply bilinear interpolation to extract the three color components for each pixel of the image sensor (Bayer patterned)
- A component that allows the display in the three color channels in VGA format.
- A component to apply a color correction matrix for suitable color perception image

Special Issue in Electronics and Biomedical Informatics,
Computer Science and Informatics
Research in Computing Science 35, 2008, pp. 97-106

A block diagram of the system is presented in Figure 1. The image processing block implements the pipelined structure to perform bilinear interpolation and color correction in real time using minimal memory resources. It is also possible to implement any other image operation that has the same structure. Furthermore, the architecture presented can be reconfigured since is implemented in a FPGA

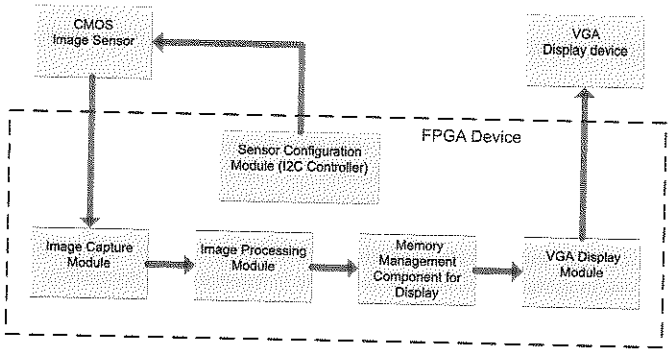

Fig. 1. Simplified scheme of the developed architecture.

2 MT9T001 Image Sensor

The MT9T001 is a CMOS active-pixel digital image sensor with an array of 2,048 horizontal by 1,536 vertical (QXGA format). It has an on chip analog-to-digital converter that provides 10 bits per pixel. It has a maximum pixel data rate of 48 megapixels per second. The sensor can be operated in its default mode or programmed by the user for frame size, exposure, gain setting, and other parameters. It is programmable through a simple two wire serial interface following the I2C protocol [1]The MT9T001 uses a color filter array with the Bayer pattern, in which every pixel has a filter of one of the primary colors. The MT9T001 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking (not valid data). The amount of horizontal blanking and vertical blanking is programmable. It is possible to modify the values of the control internal registers of the MT9T001 in order to change the window size and its location within the pixel array.

3 Image System Design in a FPGA

3.1 Control Interface for the MT9T001 Registers

The MT9T001 sensor can be programmed through a two wire serial interface that controls the reads and writes of the internal registers of the sensor using the I2C

master-slave protocol. In this work the sensor works as the slave. A design was done in a FPGA to implement the write sequence. Component I2C_ESCRITURA was realized to establish the write sequence to the sensor registers, it is subdivided in the components listed in table 1. Figure 2 shows a block diagram of component I2C_ESCRITURA.

Table 1. Components list of I2C_ESCRITURA design

Components	Brief Description
I2C_WRA	State machine to realize the 16-bit write sequence
I2C_CNTR	Provides registers number and data to be written in them
	Obtains clock frequency of 100 kHz for I2C protocol

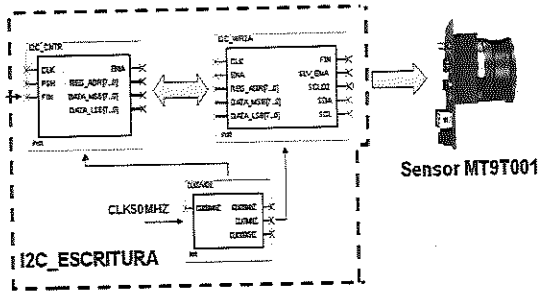

Fig. 2. Block diagram of the component I2C_ESCRITURA and its subcomponents.

3.2 Design to Obtain the Three Color Components of Every Pixel

The MT9T001 sensor uses a color filter array with the Bayer pattern, thus it is necessary to process this raw image in order to obtain the two missing color components for every pixel, using the color data of the adjacent pixels. The bilinear interpolation algorithm was selected for this task, since it is an algorithm that allows implementation using a regular and repetitive structure. The bilinear interpolation algorithm is widely used due to its low computation cost and because it offers an acceptable quality [2].

If an image sensor has the Bayer pattern as shown in figure 3b), the bilinear interpolation obtains the values of the missing colors by taking the average of the neighbor pixels; for instance, if it is located on a blue pixel it takes the average of the four red pixels and the four green pixels to obtain the red and green values respectively, in figure 3a) this idea is presented. From figure 3b), some examples for calculating pixel values are shown below:

If the pixel to work with is the blue pixel B_8, the red and green values are obtained from:

$$R_8 = (R_2 + R_4 + R_{12} + R_{14})/4 \text{ and } G_8 = (G_3 + G_7 + G_9 + G_{13})/4$$

If the pixel to work with is the red pixel R_{12} the blue and green values are obtained from:

$$B_{12} = (B_6 + B_8 + B_{16} + B_{18})/4 \text{ and } G_{12} = (G_7 + G_{11} + G_{13} + B_{17})/4$$

If the pixel to work with is the green pixel G_7 the red and blue values are obtained from:

$$R_7 = (R_2 + R_{12})/2 \text{ and } B_7 = (B_6 + B_8)/2$$

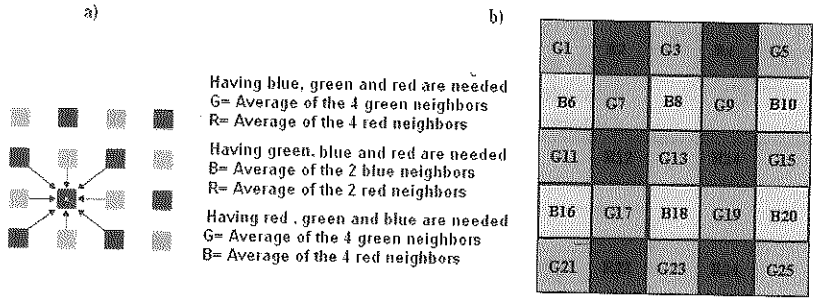

Fig. 3. a) The bilinear interpolation algorithm, b) shows the Bayer pattern

To implement these equations on hardware it is necessary to use adders and dividers. Divisions in powers of two can be implemented with right shift operations. VHDL allows these arithmetic and logical operations. The bilinear interpolation is a neighborhood operation similar to convolution filters with kernels of size 3x3.

The architecture of a general 2-D convolver presented in [3] was used. This architecture is shown in figure 4 and it can be observed that it only requires two line buffers to form the pipeline and three shift registers of three elements (pixels) to perform the 3x3 convolution window, which is used to have the necessary pixels for the operations with the eight neighbors.The use of two line buffers shouldn't be seen as a limitation, but as a design condition to use the minimal memory resources.

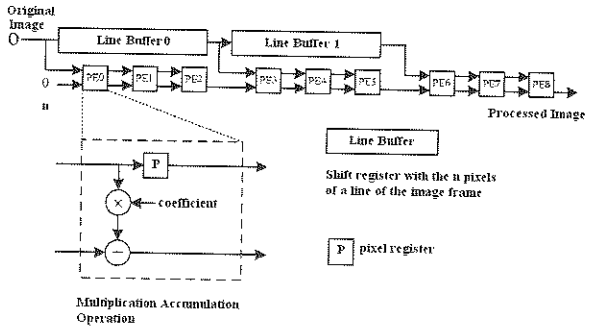

Fig. 4. Architecture to implement a general 2-D Convolver.

Subcomponents were created as a part of the major component PIPEBILINEAL, to implement the bilinear interpolation in a pipelined architecture. The component PIPE

creates the shift registers to form the pipeline. These registers must have the same number of elements as pixels in a line of the image frame. Since every pixel has 10 bits a considerable amount of memory is required. The VHDL coding style used forced the EDA software QUARTUS II to implement these shift registers in the dedicated memory blocks of the FPGA instead of misusing the available generic logic elements of the device (chapter 7 of [4] shows the VHDL coding styles). Component CONTPOS2 implements the counters that bring the pixel position relative to the center point of the convolution window. This position refers to the coordinates (row, column) of the corresponding pixel in the image frame and it is used to know if the pixel is located on a red, green or blue filter and to know which missing colors must be estimated. Component PIPECONV2 implements the registers to form the filter window and the hardware required to perform the arithmetic operations to obtain the bilinear interpolation; depending on the pixel position, it takes the required pixels from the window registers and performs the necessary operations. Component FRAMEBEGIN accepts the data and synchronization signals of the MT9T001 sensor. In figure 5 a block diagram of PIPEBILINEAL is shown.

Figure 6 shows the architecture to apply the pipelined bilinear interpolation, with an example of a 4x4 image frame size, where the line buffers for the pipeline have four pixels (as the four pixels in a line of the image frame). The necessary data to carry out the processing is stored in the three lower registers. These registers form the neighborhood window that is represented by a dashed line over the data frame in figure 8. The data stream is carried out pixel by pixel, line by line. In this architecture the pipeline orders the pixels in such a way that the operations can be performed in parallel. After a latency time of two lines and two pixels, the processed data is obtained at the input pixel data rate. Component PIPECONV takes into account the conditions where the window is on the edges of the image frame, for those cases it is considered that the missing pixels have a null value.

Table 2. Components list of PIPEBILINEAL design

Components	Brief Description
PIPE	Creates the shift registers to form the pipeline.
CONTPOS2	Brings the pixel position in the image frame.
PIPECONV2	interpolation operations
FRAMEBEGIN	Accepts the sensor's data and synchronization signals

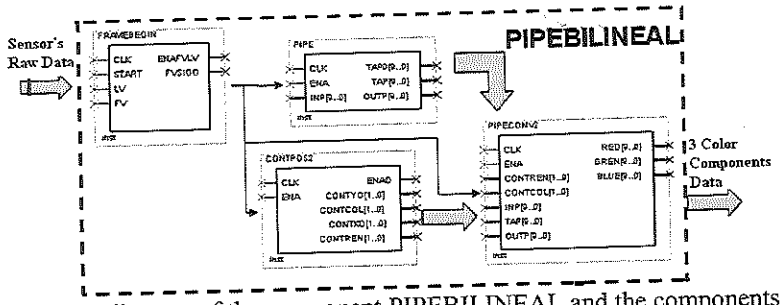

Fig. 5. Block diagram of the component PIPEBILINEAL and the components that conforms it

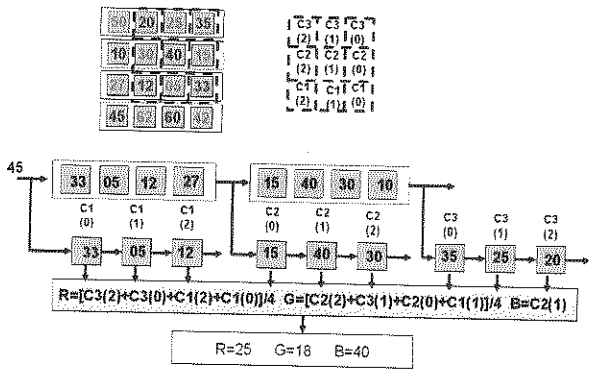

Fig.6. Pipelined architecture to implement bilinear interpolation, showing an example with a 4x4 image frame size.

3.3 Real Time Display of the Interpolated Data in Three Color Channels.

To carry out the real time display of the interpolated pixels (the obtained pixels from applying the bilinear interpolation process), alternatives where looked to utilizes the minimum of memory and it was concluded that with three memory arrays, every one of them with two line buffers, it would be sufficient to capture and display in real time. For major comprehension, in figure 7 are represented two arrays (RGB1, RGB2) of three buffers each one. The capture and display process is explained next:

After the latency of the bilinear interpolation is taken into account (figure 7a), the first interpolated data line is stored in the RGB1 array (figure 7b). Then, the second interpolated data line is stored in the RGB2 array while the content of RGB1 is read out for display (figure 7c); these steps are repeated nonstop. In figure 7, the signal "capture of line" represents the line valid signal from the sensor and the signal "display of line" represents the valid display line period of the VGA format. The data begins to be displayed after the interpolation latency and after the first interpolated data line is stored (figure 7c). For this process to run, the capture cycle must have the same duration as the display cycle and be synchronized. The 48Mhz clock signal from the sensor acts as the master clock and a derived 24Mhz clock signal is used for the display cycle in VGA format; therefore, synchronizing the capture and display cycles. The 24 MHz clock signal generates a frame rate of 53 frames/s for the display in VGA format. The I2C_ESCRITURA component was used to configure a 640x480 image frame size (VGA) and to modify the horizontal and vertical blanking duration accordingly, so the capture and display cycles were synchronized.

The DESP_REAL4 module was created for the real time display and it is subdivided in the components shown in figure 8 and in table 3. The BUFDESP4 component accepts the interpolated data coming from the PIPEBILINEAL component generates the counters and provides the address and enable signals to write and read to each of the color channels memory. The components BUFRED, BUFGREEN and BUFBLUE are memory blocks containing each one a two line buffers, each component stores on of the three color channels. These memory blocks have separated port, enable and clock

signals for write and read, and a VHDL coding style was used in order to implement dual port and dual clock RAM memory utilizing the dedicated memory blocks of the FPGA, as it is recommended in chapter 7 of [4]. The component VGA5 generates the synchronization signals for the VGA format and accepts the three data ports coming from the components BUFRED, BUFGREEN and BUFBLUE, providing the VGA display in the three color channels. The component CLKDIVIDE divides the frequency of the master clock from 48 MHz to a 24 MHz clock signal.

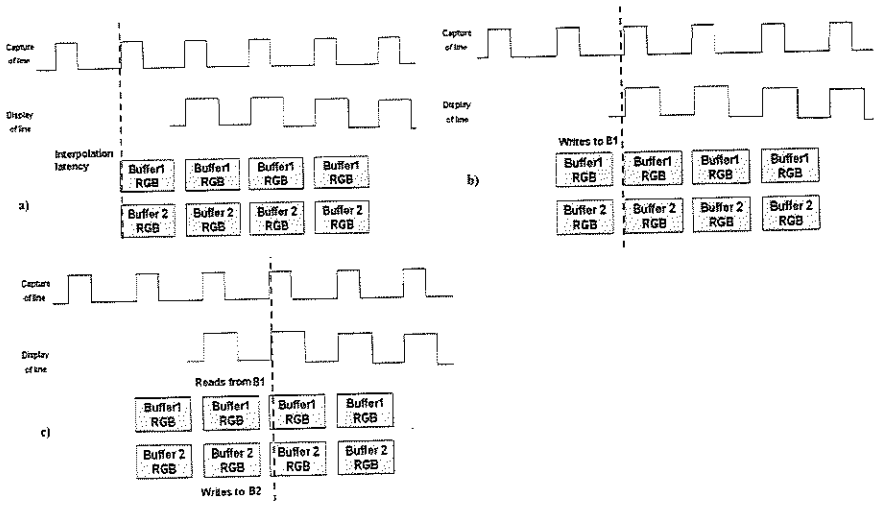

Fig.7. Management of memory to carry out the capture and display of the interpolated data in the three color channels

3.4 Color Correction

A color correction is applied since the spectral response of the CMOS image sensor is different that the response of the human eye, and also different from the response of the display device [5]. The color correction multiplies a 3x3 matrix with the vector formed of the red (R), green (G) and blue (B) values of every interpolated pixel, as it is shown next: $\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ g_1 & g_2 & g_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$, or $\begin{aligned} R' &= r_1R + r_2G + r_3B \\ G' &= g_1R + g_2G + g_3B \\ B' &= b_1R + b_2G + b_3B \end{aligned}$, where $\begin{bmatrix} R \\ G \\ B \end{bmatrix}$ = RGB values of the interpolated pixel, $\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$ = RGB values of the corrected pixel and $\begin{bmatrix} r_1 & r_2 & r_3 \\ g_1 & g_2 & g_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$ is the correction matrix.

Typically the coefficients have values $r_1 > 1, r_2 < 1, r_3 < 1$, $g_1 < 1, g_2 > 1, g_3 < 1$, $b_1 < 1, b_2 < 1, b_3 > 1$. Where the sum of the values of each coefficient set must be equal to one to maintain color balance [6]. For color correction implementation, the product of the coefficient fractional part by the interpolated data (R, G or B), and the product of the coefficient integer part by the interpolated data, are obtained separately, and then these products are added resulting an integer number (disregarding the fractional part). This is done for every interpolated data, these results are added and limited to a possible maximum or to zero if it is a negative result. Figure 9 shows the structure to obtain the equation for the corrected red (R'), for the equations of the corrected green and blue (G', B'), the same structure is used with the corresponded coefficients. The component MULTADD5 was designed to have a structure required to apply the correction matrix, and the values of the coefficients will depend of user application.

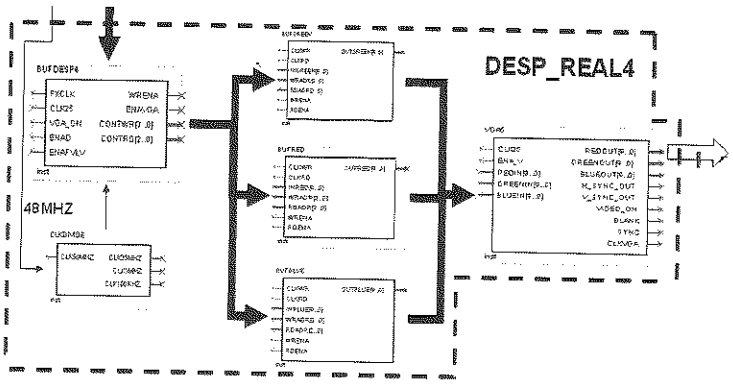

Fig.8. Block diagram of DESP_REAL4 showing its components

Table 3. List of components of the design DESP_REAL4

Components	Brief Description
BUFDESP4	Accepts interpolated data and generates write/read signals for the RAM arrays.
BUFRED, BUFGREEN & BUFBLUE	Each one creates a memory block containing two line buffers, the blocks are dual port and dual clock RAM, each component stores one of the three color channels.
VGA5	Generates the synchronization signals for VGA format in three color channels
CLKDIVIDE	Obtains the 24 MHz clock signal for the VGA display

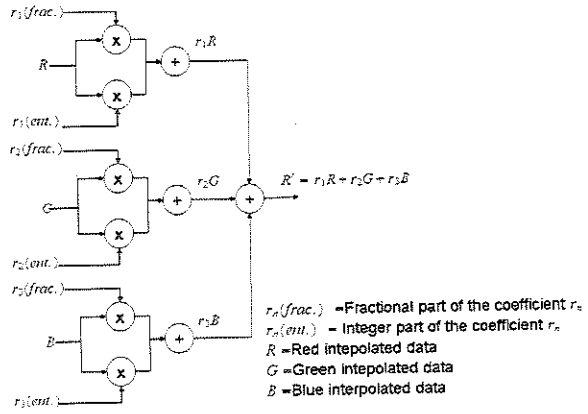

Fig.9. Structure to obtain the equation for the corrected red R'.

4. Results and Discussion

The top level design called DESP_BAYER3 joins the components to carry out the sensor configuration (I2C_ESCRITURA), the bilinear interpolation (PIPEBILINEAL), the color correction (MULTADD5) and the real time display in the three color channels (DESP_REAL4.vhd). Figure 10 shows a block diagram of DESP_BAYER3.

In this system the initial latency time t_l, is the sum of the latencies provoked by the interpolation processing $t_i = (n+2)1/f$ (time of capturing two lines and two pixels), the real time display process $t_d = (n)1/f$ (time of capturing one line) and the color correction structure $t_c = 1/f$. After a latency time t_l described by equation 1, the system begins to display image from the sensor in VGA format.

$$t_l = t_i + t_d + t_c = (2n+3)1/f \tag{1}$$

Where n is the number of pixels in a line of the image frame and f is the frequency of the clock.

The resources utilized from the FPGA Cyclone II to implement the system DESP_BAYER3 are shown in table 4, and represent an optimized design, leaving plenty of free resources, especially for memory.

Table 4. Utilized resources of the FPGA Cyclone II

FPGA used resources	Used elements /total elements	Percentage
Total logic elements	930/33,216	28%
Total memory bits	51,160/483,840	11%
9-bit embedded multipliers	10/70	14%

It is important to mention, that within the VHDL codes, exist a list of parameters used by the components of the system in such a way that values of number of pixels in a line and number of bits in a pixel can be configured, also the pipelined interpolation processing architecture can be configured in the number of line buffers, the number of register for the filter window.

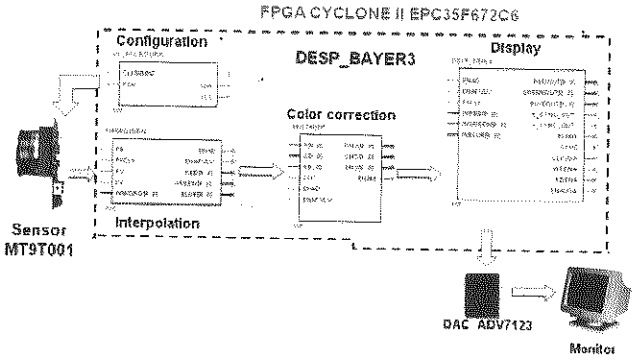

Fig. 10. System that performs the sensor configuration, the bilinear interpolation, the color correction and the real time display in the three color channels.

5. Conclusions

The smart camera system developed speeds up image processing of the interpolation and color correction algorithms, obtaining results at the pixel input data rate after an initial latency required to store two lines of the image frame. This system can be used in software based vision systems as a subsystem to speed up processing of low level image operations or it can be used as stand alone system. The resources utilized by the implemented system are just a low percentage of the full FPGA device used, thus the free resources can be used to implement other processing architecture, or to integrate other connectivity capabilities using the same Cyclone II device.

6. References

1. 1/2-Inch 3-Megapixel CMOS Digital Image Sensor MT9T001P12STC, Micron Technology, Inc (2004)
2. A Study of Spatial Color Interpolation Algorithms for Single-Detector Digital Cameras. Stanford University, http://scien.stanford.edu/class/psych221/projects/99/tingchen/index.htm
3. Chi-Jeng Chang, Zen-Yi Huang, Hsin-Yen Li, Kai-Ting Hu, and Wen-Chih Tseng.: Pipelined Operation of Image Capturing and Processing. In: 5th IEEE Conference on Nanotechnology, Nagoya, Japan (2005)
4. Quartus II Handbook, Volume 1, 7 Recommended HDL Coding Styles. Altera Corporation (2007)
5. Color Correction for Image Sensors Application Notes. Image Sensor Solutions. Kodak (2003)
6. Color Correction Matrix Application Note. Lumenera Corporation (2005)