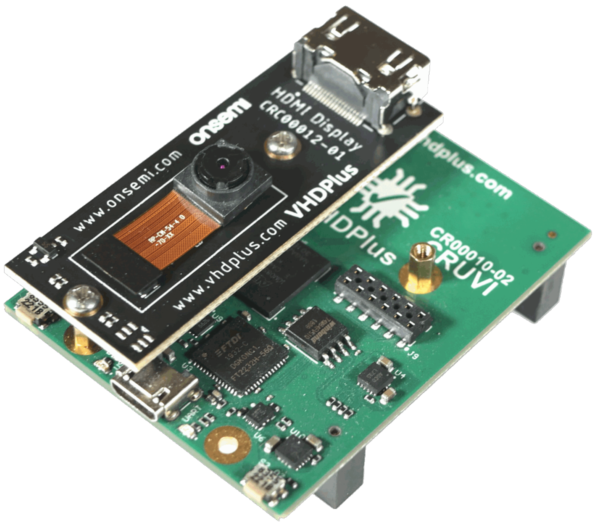
Image processing with minimal delay on MAX10 FPGA

Leon Beier

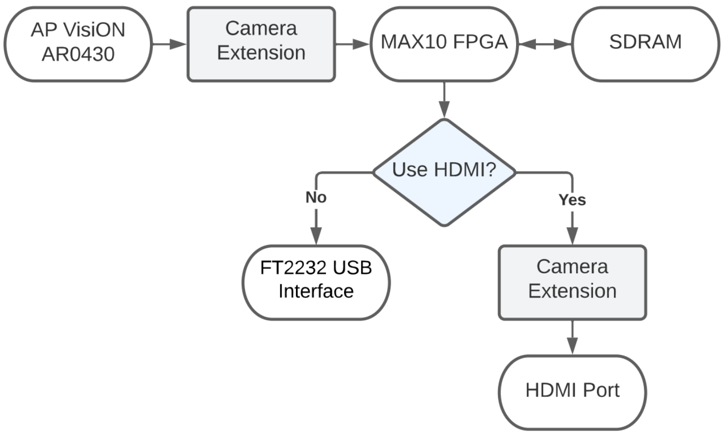
March 9, 2022



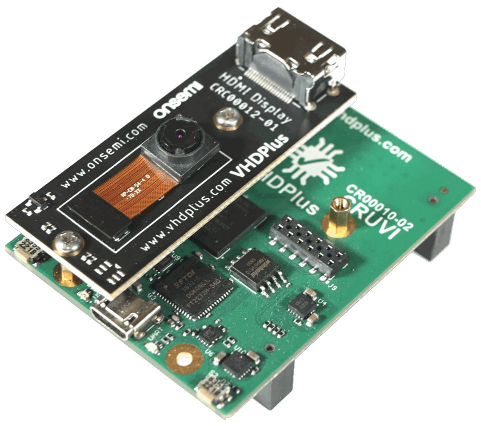
1 Abstract

In image recognition, it is often important to analyze as many images as possible per second and with as little delay as possible. FPGAs offer many advantages over processor-based solutions, which will be discussed further in this article. However, implementing image recognition algorithms on an FPGA is often more complicated. Therefore a simpler, Plug&Play solution was developed with the VHDPlus IDE and the VHDPlus hardware. Thus the simple programming of the processor and the high performance of the FPGA are combined.

2 Overview

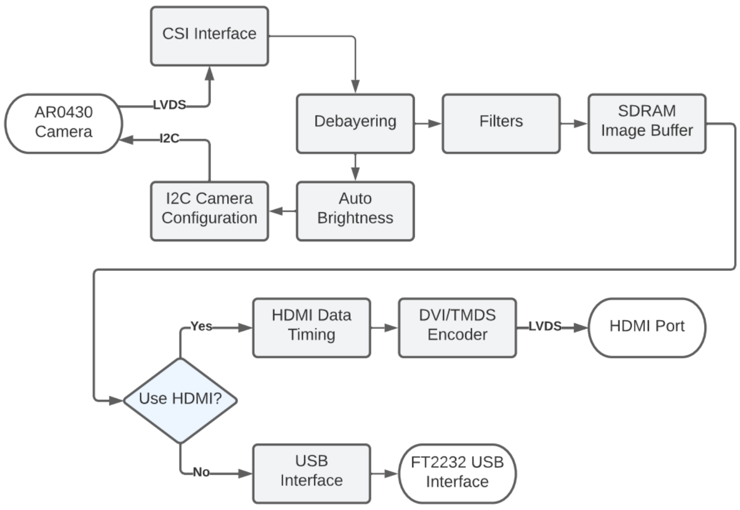


The system consists of a VHDPlus OnSemi IAS camera extension with Appletec AP VisiON AR0430 and a VHDPlus Core MAX10. The AR0430 camera can capture 4MP images at up to 120fps and has a special feature that depth information can also be extracted from the image at 30fps. This OnSemi image sensor is used in the Appletec AP VisiON camera module and connected to the adapter board with the IAS connector. The board has the necessary power supply for the camera. With a 4-lane MIPI CSI interface the images are transferred to the FPGA via a CRUVI High Speed connector. With the CRUVI High Speed connector, this extension can be easily installed on the Core MAX10. An Intel FPGA MAX10 (10M08SAU169) is used as the FPGA on the VHDPlus Core MAX10. This FPGA was chosen mainly to offer a solution as cheap as possible. In the following, we will analyze how complex image recognition algorithms can be implemented with this FPGA as a minimal solution. To evaluate the results, the camera extension offers an HDMI connection and the Core MAX10 an USB interface. Thus, the images can either be viewed directly or sent to a PC. In addition, the Core MAX10 offers a 64Mb SDRAM with which the images can be stored temporarily.



Despite so many components, the system is only 5.6×4.5×2.2cm in size. It is ideal for testing the performance of FPGAs in image processing, and the AR0430 camera offers much more capability than used here. But with the CURVI connector the camera can also be connected to faster FPGAs to use the full potential.

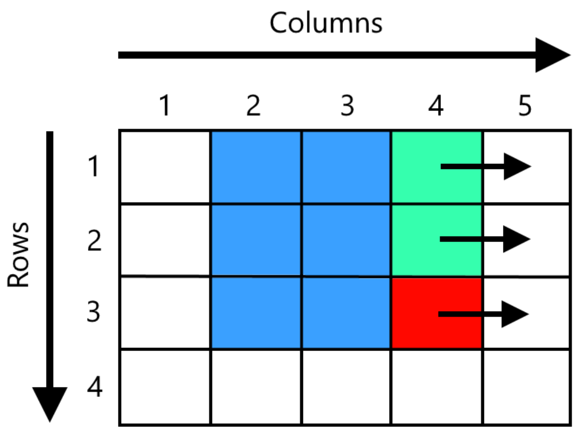
3 FPGA implimentation



One advantage of the FPGA is that no additional peripherals are required as an interface for the camera and HDMI. With the LVDS I/Os of the MAX10 with single supply 200mbps per differential transmitter and receiver are possible. With Dual Supply or with other FPGAs higher data rates could be used to receive the camera data or send the HDMI data. But also the 4 LVDS inputs for the camera and the 3 LVDS outputs for HDMI are sufficient to receive a 1280x960 RAW image at 40fps and output the processed image. Since many monitors expect 60fps, the 1280x960 pixel image is downsized to 640x480 while debayering. With a pixel clock of 28MHz the image can then be displayed with the HDMI output. Alternatively, you have more freedom with the USB port. Here the 2304x1728 RAW image is received with 15fps and stored as 1152x864 image in the SDRAM after debayering. However, the transfer via USB is very slow with UART. This can be improved by a separate USB interface, so the speed like with HDMI and the maximum resolution can be achieved.

4 Image processing with FPGA

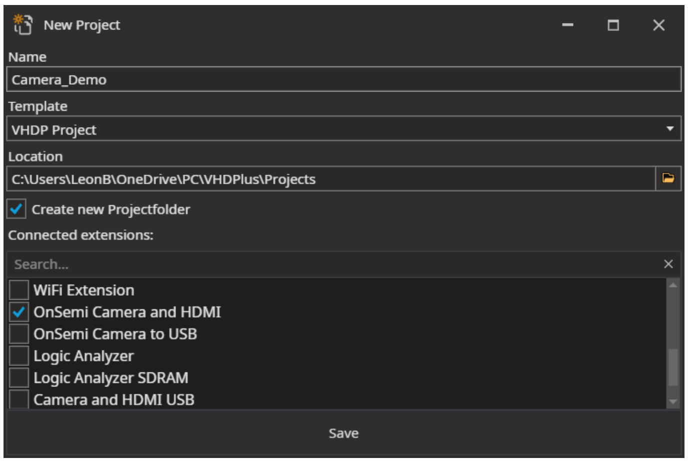
The main advantage of the FPGA is the performance of the image processing. With a processor, all pixels would first be written to RAM. If then, for example, a simple color filter is to be applied to recognize certain colors, each pixel would have to be read out of the RAM again. Then the filter would be applied and the pixel would be written back into RAM. Even with several fast processors doing this in parallel, this can take a long time. With the FPGA, however, while the pixels are being received by the FPGA with the CSI interface, they are already being passed to the internal logic for the filter and calculated without delay.



Even if several pixels have to be considered, as for example with CNNs or a blur filter, the FPGA offers a clear time advantage. Only the required number of pixel rows is stored in the SRAM of the FPGA and for each new pixel a value can be calculated that depends on several pixels from different rows. For exact implimentation you would create a RAM for each row. In the picture you can see in green the addition of the 2 RAMs for the rows, in red the current pixel and in blue the last 2 values that are buffered. So overall the delay would be negligible with the FPGA since the filters are applied directly and the output, in frames per second, is the same as the input from the camera.

5 Implementation with the VHDPlus IDE

FPGAs can be used to implement very efficient and high-performance camera systems. Compared to processors, however, FPGAs have the disadvantage that implementation is more difficult. For processors, there are already ready to use libraries and drivers for camera and display. With the VHDPlus IDE, however, image recognition with FPGAs is plug & play and even feasible for beginners. As an example we look at a simple recognition of objects of one color. Neural networks, edge detection and various other filters are just as possible to implement with FPGAs.



New Project

Name: Camera_Demo

Template: VHDP Project

Location: C:\Users\LeonB\OneDrive\PC\VHDPlus\Projects

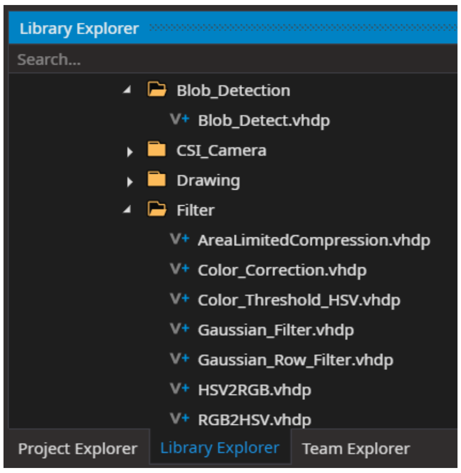
[✓] Create new Projectfolder

Connected extensions:

Search...

- [ ] WiFi Extension
- [✓] OnSemi Camera and HDMI
- [ ] OnSemi Camera to USB
- [ ] Logic Analyzer
- [ ] Logic Analyzer SDRAM
- [ ] Camera and HDMI USB

When creating a new project, an example for the camera can be selected directly. Thus, the implementation of the camera and output interface is already created and the user only needs to implement the filters depending on the application.

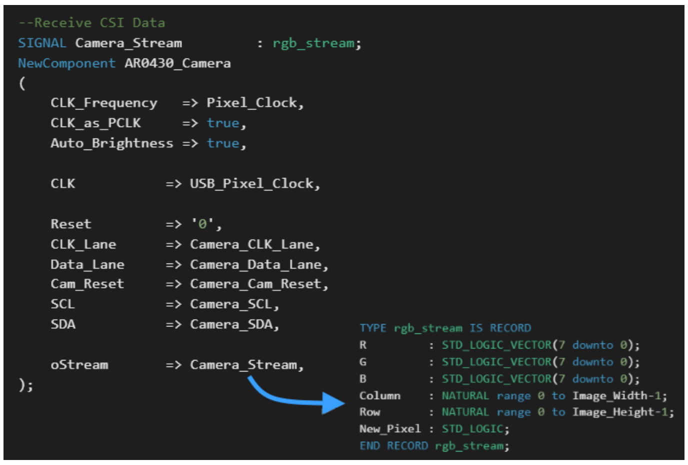


Library Explorer

Search...

- Blob_Detection
  - Blob_Detect.vhdp
- CSI_Camera
- Drawing
- Filter
  - AreaLimitedCompression.vhdp
  - Color_Correction.vhdp
  - Color_Threshold_HSV.vhdp
  - Gaussian_Filter.vhdp
  - Gaussian_Row_Filter.vhdp
  - HSV2RGB.vhdp
  - RGB2HSV.vhdp

Here you can then look in the Library Explorer for filters that we have already tested. Or on Github there are examples in VHDL, which can also be easily connected together with the implimentation in the IDE. So for image recognition you could first convert the camera data into the HSV color space, filter certain colors with the Color Threshold filter, minimize the noise with the Gaussian filter and then determine the positions and sizes of the objects with the Blob Detector.



```
--Receive CSI Data
SIGNAL Camera_Stream         : rgb_stream;
NewComponent AR0430_Camera
(
    CLK_Frequency   => Pixel_Clock,
    CLK_as_PCLK     => true,
    Auto_Brightness => true,

    CLK            => USB_Pixel_Clock,

    Reset          => '0',
    CLK_Lane       => Camera_CLK_Lane,
    Data_Lane      => Camera_Data_Lane,
    Cam_Reset      => Camera_Cam_Reset,
    SCL            => Camera_SCL,
    SDA            => Camera_SDA,

    oStream        => Camera_Stream,
);
```

```
TYPE rgb_stream IS RECORD
R         : STD_LOGIC_VECTOR(7 downto 0);
G         : STD_LOGIC_VECTOR(7 downto 0);
B         : STD_LOGIC_VECTOR(7 downto 0);
Column    : NATURAL range 0 to Image_Width-1;
Row       : NATURAL range 0 to Image_Height-1;
New_Pixel : STD_LOGIC;
END RECORD rgb_stream;
```

But also own filters are easy to realize. With oStream the received data of the camera can be used easily. In R, G and B the color of the pixel is stored. New Pixel is a clock signal, which indicates a new pixel with the rising edge. With Column and Row the current row and column of the pixel in the image is stored. With the VHDPlus IDE you can easily add a NIOS II processor to the design. This could make then further computations with the calculated values. And finally the VHDPlus IDE also offers tools to receive and visualize images via USB. For the creation of the camera project and to the implementation of processors, there are also simple tutorial videos in which this is explained. Thus, depending on the requirements, the FPGA solution can also keep up with processors in terms of development time and ease of use.

6 Performance and resources

It is also important to know how many logic elements and other resources the FPGA must have for the actual application. With the MAX10 on the Core MAX10 it is very easy to use it in own projects and it offers with 8000 logic elements, 387Kb internal RAM and an internal Flash already enough resources for small to medium sized image recognition projects. So often the solution with FPGA is not only more performant, but also cheaper than with a processor, which would also comply with the needed limits for delay and frames per second. In the following, the resources for the example of color detection are listed.

Component	Logic Elements	RAM Bits
CSI Interface + Debayering	300	31888
I2C Camera Configuration + Auto Brightness	921	0
SDRAM Image Buffer	645	41344

For HDMI output:

Component	Logic Elements	RAM Bits
HDMI Data Timing	83	44
DVI Encoder	368	0

And for the USB output:

Component	Logic Elements	RAM Bits
USB Interface	332	0

Thus, with an USB interface 73% and with HDMI interface 71% of the logic elements would still be free for image processing.

For the filters:

Component	Logic Elements	RAM Bits
Color Correction	194	102
RGB to HSV	803	45
HSV Threshold	9	0
Gaussian Filter	304	16384

The University of Bonn also offers the so-called "FPGA Vision Remote Lab". Here you can find an example for a small neural network and for an edge detection. The VHDL code can also be used and needs the following resources.

For 3 inputs, 2 hidden neurons and one output neuron:

Component	Logic Elements	RAM Bits
Neuronal Net	686	102400

For edge detection:

Component	Logic Elements	RAM Bits
Edge Detection	880	126976

The RAM is needed for the Signum function and could be reduced by another activation function.

7 Summary

If data is to be processed quickly, as in image processing, it is worth taking a look at FPGAs. FPGAs of the MAX10 series from Intel FPGA are inexpensive and can be used without a lot of peripherals. Together with the easy development with the VHDPlus IDE and CRUVI hardware, FPGA programming is also not as complex as before. Thus, the best performance can be achieved with minimal development effort and price. FPGAs also are beneficial for future-oriented technologies such as neural networks. Here, significantly higher data rates can be achieved through parallel calculations.