embeddedworld2022
Exhibition&Conference
... it's a smarter world

Real-time Image Signal Processor for SoC/FPGA With Direct GPU Communication

Richard Weiss
Institute of Embedded Systems
Zurich University of Applied Sciences (ZHAW)
Zurich, Switzerland
wesr@zhaw.ch

Hans Gelke
Institute of Embedded Systems
Zurich University of Applied Sciences (ZHAW)
Zurich, Switzerland
gelk@zhaw.ch

***Abstract*—Many video applications require to capture images in a dynamic environment. Embedded cameras can be configured to adapt to these environments. However, the algorithm that controls these settings is often left up to the user. Additionally, no tools for preprocessing and calibration are provided as the implementation of these tools is time-consuming, especially in an FPGA environment. In this paper, an Image Signal Processor (ISP) for Xilinx FPGAs is discussed. The goal was to create a versatile toolset to improve image quality regarding the needs of typical video applications. Thus, the following features were implemented. First, automatic white balancing and automatic gain control were added to ensure that the color tone and the exposure of the captured video adapt to the environment in which the camera is used. Additionally, non-linearities and offsets of each color channel caused by the image sensor can be corrected by lookup tables. Moreover, classic image processing algorithms, such as blurring or edge detection, can be applied using the specifically designed configurable convolution filter. Furthermore, the perceived sharpness of the video stream can be increased by the sharpening feature. Finally, for grayscale videos, a histogram equalization can be used to increase the overall contrast, which is interesting for machine learning or medical applications. A glass-to-glass demonstration was built to test the ISP implementation. The demonstration consists of a CSI (Camera Serial Interface) camera connected to a video capturing card featuring a Xilinx Ultracale+ MPSoC FPGA on which the ISP was implemented. The capture card is connected to the PCIe port of a host PC, which displays the video. For the data transfer between the FPGA and the host computer, a frame-based DMA was developed. This DMA has the capability to transfer the data directly into the memory that is accessible from the GPU. The tests revealed that all developed features perform as intended, verifying the proper implementation. The various tools provided by the ISP are a solid basis for any video application. In addition, the modular design of the video pipeline and the used Xilinx AXI-Stream video interface allows adding future features with minimal effort.**

I. CONCEPT

A. General Concept

The ISP should be capable to process an incoming video stream in real-time. Each element of the ISP must therefore have a constant delay and a throughput that matches the data rate of the input video stream. To fulfill the real-time requirement, the data stream through the ISP never leaves the FPGA fabric until it is completely processed.

The ISP is divided into two parts. One part is the video pipeline in the FPGA fabric which is responsible for the actual video processing. The other part controls the pipeline by applying the current user settings and calculates the algorithmic tasks needed by some of the pipeline elements. Because of this division, each element of the SoC is used regarding its strength. Furthermore, offloading some of the algorithms to the processor allows to quickly improve them or adapt to different situations.

A camera application not always needs the complete feature set of the ISP. Additionally, available resources are often limited. That's why, the ISP has a modular design that allows to exclude unused features. The reduced number of elements in the pipeline additionally comes with a reduced latency. The modularity is achieved by packing the features in blocks which are connected using the industry standard AXI-Stream interface [1]. Using this interface further allows to use other IPs that are for example developed by Xilinx itself.

An ISP is never the end consumer of a video pipeline. That's why it must be easy to include in a new or existing design. The most time-consuming part of including an IP into a design is to understand how it works and how it must be configured. Features, such as the automatic white balancing or automatic gain control further need an algorithm that updates camera setting according to statistics that were extracted from the video stream. It can be seen, that as a consumer it is unpractical and unnecessary to control each register of the pipeline. For that reason, an abstraction layer is provided to reduce the amount of interaction between the consumer and the ISP.

II. FEATURES

There are many features that could have been included into the ISP. The following ones were selected for implementation, as they are most used and provide a significant benefit. However, due to the modularity, further features could be later added.

A. Convolution Filter

The convolutional filter is often used in classical video processing. It allows to manipulate the pixels while also taking their surrounding pixels into account. The application of such a filter could be blurring the image or detecting edges.

B. Sharpening

Sharpening is used to enhance brightness transitions in a video or image. This technique can be found in TVs, cameras, or image manipulation software such as Photoshop. As every classic image processing tool, it cannot create details that were not present in the original image. However, existing details are enhanced in a way that results in a sharper image.

C. Automatic white balancing

White balancing is used to adapt to the current lighting. Various light sources do have different color temperatures. Human eyes are constantly adapting to the lighting such that a white object (completely reflect the incoming light of all frequencies) appears white. This adaptation can for example be observed while a colored ski goggle is worn. First, the white snow has the color of the goggle which then slowly changes into a perfect white. Camera sensors, however, don't adapt to the lighting leading to a color shift that can look odd for somebody that looks at the image because the eyes are not necessarily adapted to the same lighting conditions.

Compensating for the color temperature is a simple task that can be achieved with a separate gain for each color channel. However, the exact values for the gains and therefore the profile of the lighting must be known. An automatic white balancing algorithm uses the recorded frames to estimate the color temperature of the lighting. For an observer it is under certain circumstances impossible to determine the color characteristics of the light source. If a white wall illuminated with a certain color is filmed, no algorithm can tell the difference to a filmed wall with the said color illuminated with a white light source.

D. Automatic gain control

The gain of a camera determines how bright the recorded image gets given a specific brightness of the scene. In many video applications, the scene has various brightness. If the gain of the camera is not adapted to the current brightness of the scene, the captured video would be either too dark or too bright with clipping. The correct gain can be determined by analyzing the recorded frames regarding the mean brightness and the number of clipping pixels.

E. Histogram equalization

This technique is used to rearrange the brightness values in an image to utilize the whole available value range and increase the contrast.

F. Lookup tables

Lookup tables map an incoming value to a predefined output value. This allows to correct non linearities of the camera, cut off a black floor, performing gamma correction, or adjust the contrast in certain brightness ranges.

G. Demosaicing

The RAW data from a camera sensor has only one color channel per pixel. In each patch of 2x2 pixels, there are two green, one red and one blue pixels. Video applications, however, most of the time use the color formats RGB or YCbCr. Furthermore, image processing algorithms often need the complete color information for every pixel. That's why the mosaic pattern has to be transformed into the RGB color format by interpolating the color information of the surounding pixels. This transformation is called demosaicing.

III. OVERVIEW OF THE FINAL PIPELINE

Figure 1 provides a conceptual overview of the ISP implemented entirely in the FPGA. The video stream busses are complaint with the AXI-Stream Video interface. Additionally, the ISP is separated into a processor part and programmable logic part.

A. Controling

The processor controls the programmable logic via the AXI-Lite interface that the AXI Interconnect distributes to every block in the pipeline. The reverse information flow from the programmable logic to the processor occurs via interrupts.

The Control Register Bank, located in the programmable logic, stores the basic configuration of the ISP. It also serves as an abstraction layer to the consumer of the video stream, denoted as External Control Unit. The control application running on the processor constantly polls the current status of the Control Register Bank. If the configuration has been updated, the control application reconfigures the pipeline. The control application stops and resets the pipeline before it reconfigures and restarts it, for example, when switching the video source or changing the video dimensions.

The Camera Control block offers the hardware interface for the communication between the FPGA and the connected camera. An I2C interface and GPIOs are used to control the cameras.

B. Hardware blocks in the pipeline

The pipeline selects from multiple video sources due to the Receiver and Input Selection block at the beginning of the pipeline. Additionally, this block includes the CSI hardware receivers and the demosaicing functionality.

After the video stream is transformed into the RGB color format, the Lookup Table block corrects non-linearities and black offsets caused by the image sensor.

The convolution filter then convolves the video stream with a user-defined filter. The processor can reconfigure the specific coefficients of the filter. Therefore, the user can implement any filter that he desires. The only limitation is the size of the applied filter, which must be defined before compiling the hardware design.

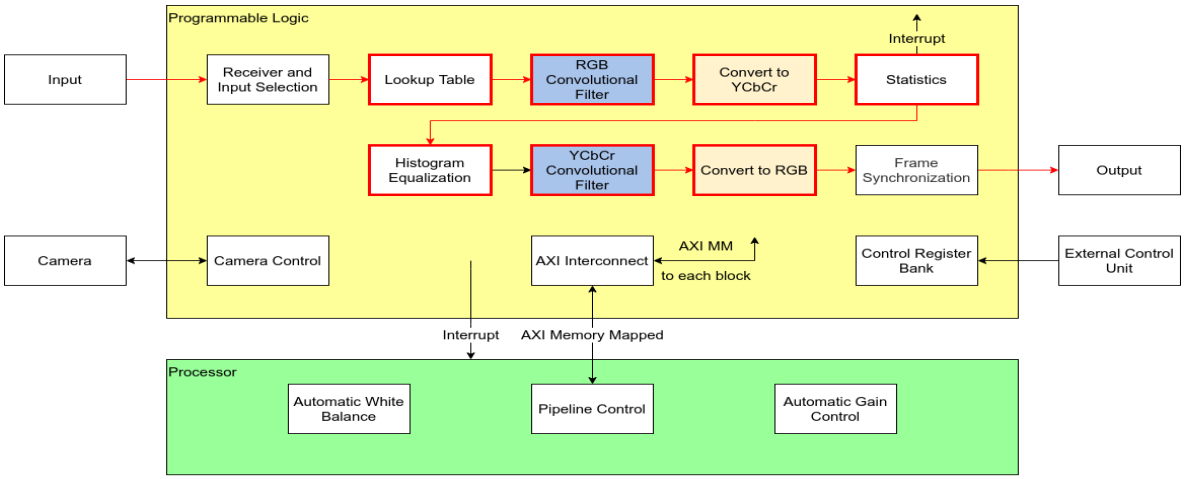

Figure 1, Conceptional overview of the ISP

The subsequent Color Space Converter transforms the RGB video stream into the YCbCr color format to be able to manipulate the luminance of the video stream.

The Statistics block collects information about the video stream content. As soon as the block has analyzed a new frame, it notifies the processor via an interrupt. The algorithmic processes of the Automatic White Balancing, Automatic Gain Control, and Histogram Equalization require information collected with the Statistics block.

The subsequent Histogram Equalization block enhances the contrast of the image if it is activated. However, histogram equalization should only be used for grayscale videos.

The following Color Space Converter transforms the YCbCr video back into the RGB color format. However, the video can also bypass the converter such that the output video format is YCbCr. The pipeline can, therefore, also provide a grayscale or YCbCr video format at the output.

The Frame Synchronization block at the end of the pipeline acts as an interface between the pipeline itself and the PCIe bus. However, following video pipeline parts that fully support the AXI-Stream Video protocol do not need this synchronization block.

C. Algorithms to update the pipeline

The processor handles the algorithmic part of the Automatic White Balance, the Automatic Gain Control, and the Histogram Equalization. These algorithms do not require parallelization but demand complicated calculations. Therefore, the processor is better suited for this task. Offloading these algorithms to the processor increases the system's flexibility and reduces the resource usage in the programmable logic.

The functional units implemented in the processor use the collected data of the Statistics block and then control other parts of the pipeline accordingly. First, the Automatic White Balance algorithm uses the RGB Convolution Filter to apply the adjusted channel gains. Next, the Automatic Gain Control updates the configuration registers of the camera via the Camera Control block. Finally, the Histogram Equalization is applied via the block of the same name.

IV. FDMA

A. What is FDMA

The Frame Based Direct Memory Access (FDMA) IP is designed by the Institue of Embedded Systems (InES) to autonomically transfer data between an FPGA and the GPU of a host PC over a PCIe interface. The IP is designed to work independently without any interaction with the host processor after the system has been initialized. Another central characteristic of the FDMA IP is the ability to handle the incoming data as frames by including a signaling interface. A frame in this context can but doesn't have to be an image of a video stream. It can be any data. The term frame is used because this DMA engine is designed for repetitive transfers of the same size. The framed data is then sent using the AXI to PCIe Bridge IP from Xilinx [2]. FDMA can handle multiple buffers in each direction, which allows a double or tripple buffer design.

Figure 2 gives a graphical overview of the FDMA based co-design concept. In the GPU memory, there are multiple RX and TX buffers and a flag buffer. The flags are for controlling the access to the RX and TX buffers and mark the data as valid. The FDMA IP has an input and an output AXI stream interfaces to connect to the data source or sink.

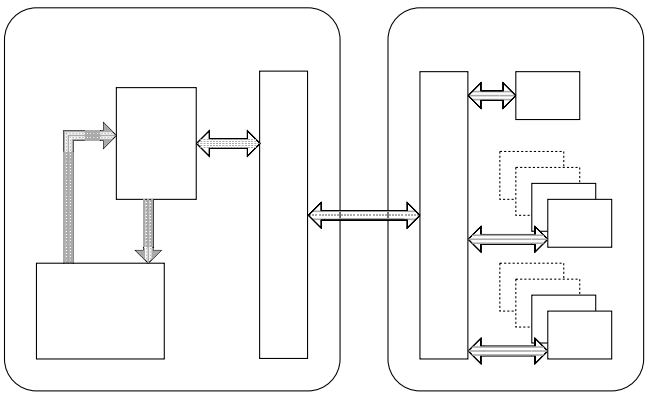

Figure 2, FDMA concept

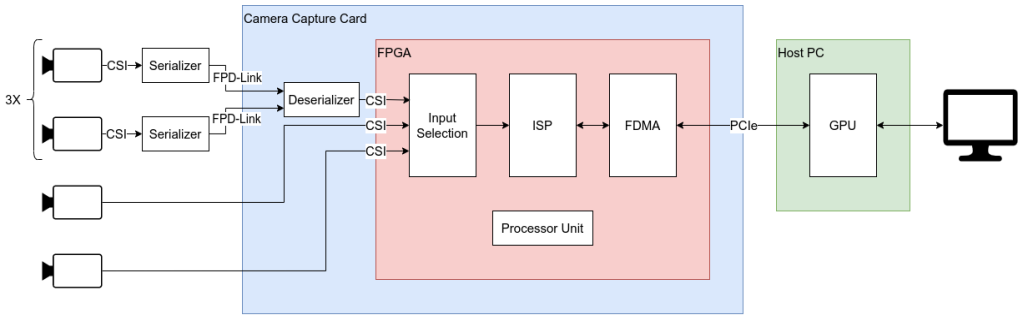

Figure 3, Complete glass to glass example design

B. Configuration

The FDMA IP has a register bank accessible through an AXI lite interface which is for simplicity not shown in Figure 2. Through this register bank, the IP can be configured and started. This register bank has to be available to the host for the setup process.

C. FPGA to GPU transfer

After enabling the FDMA IP, it starts polling the GPU flag for the first TX buffer. As soon the flags indicate that the TX buffer is accessible for the FPGA it starts writing the data through the AXI PCIe bridge and the PCIe bus directly into the GPU RAM. After the transfer, the FDMA IP sets the current TX buffer flag to indicate that the GPU can access this buffer and the data is valid. The FDMA IP starts immediately polling the flag for the next TX buffer and starts the next transfer if the buffer is ready. The sequence diagramm displayed in Figure 4 depicts this mechanism.

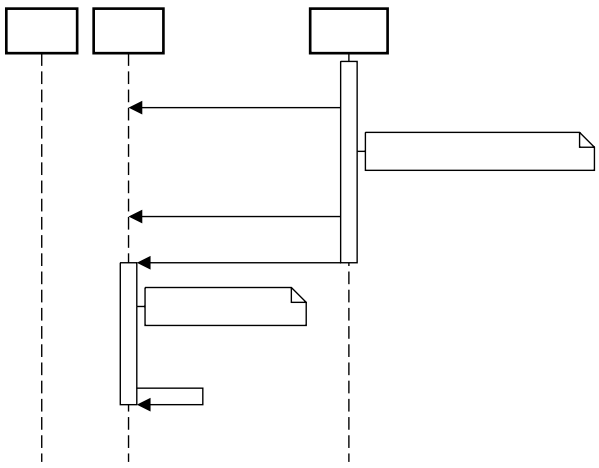

Figure 4, Sequence diagram of an FPGA to GPU transfer

V. FINAL SETUP

The combination of the proposed ISP and FDMA allows to create a camera capture card that handles all the required preprocessing while minimizing the workload for the host PC.

Figure 3 shows the concept of a complete pipeline that was developed to demonstrate the capabilities of the combined ISP and FDMA. The developed capture card features 6 FPD-Link and 2 Camera Serial Interface (CSI [3]) ports to connect cameras. Cameras send video streams to the FPGA via CSI.

In this configuration, the captured video is streamed through the FPGA fabric with a constant delay. The delay only varies if the video sink is too slow and stalls the pipeline. In this case, the video sink is not apropriately dimensioned for the desired application and there is no way to uphold a realtime pipeline.

VI. REFERENCES

[1] Amba AXI4 interface protocol. https://www.xilinx.com/products/intellectual-property/axi.html. Accessed: 2022-05-20.

[2] DMA for PCI Express (PCIe) Subsystem. https://www.xilinx.com/products/intellectual-property/pcie-dma.html#overview. Accessed: 2022-05-20

[3] MIPI CSI interface. https://www.mipi.org/specifications/csi-2. Accessed: 2022-05-20.