FPGA implementation of a foveal image processing system for UAV applications

Zoltán Nagy*†, Ákos Zarándy*, András Kiss†*, Máté Németh†* and Tamás Zsedrovics†
*Cellular Sensory and Optical Wave Computing Laboratory,
Institute for Computer Science and Control, Hungarian Academy of Sciences, H-1111
†Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Budapest, H-1083

***Abstract*—An on-board UAV high-performance collision avoidance system sets up drastic constraints, which can be fulfilled by using carefully optimized many-core computational architectures. In this demonstration we introduce a many-core processor system, which can process a 150 megapixels/sec video flow to identify remote airplanes. The introduced processor system is implemented on Xilinx Spartan-6 and Zynq SoC FPGAs, and consumes less than 1W.**

I. Introduction

Unmanned Aerial Vehicle (UAV) technology reached an advanced level, which enables them technically to fly autonomously a predefined paths and complete different missions. However, legally they are not allowed to fly fully autonomously, since flight authorities identified various safety shortcomings.

An on-board vision system for medium sized UAV should fulfill numerous tough specification criteria. Its resolution and field of view (FOV) should be high enough to identify intruder aircraft from large distance; it should be able to perform real-time processing; its size, weight, and power consumption parameters should satisfy on-board UAV operation requirements; and finally, it should be affordable. From functionality point of view, it is expected to calculate the attitude of the aircraft by calculating the differential orientation changes between consequent frames (yaw, pitch, roll angles) and detect intruder aircrafts, which are on a collision course; store all the acquired images in full resolution for archiving and for off-line testing purposes.

II. Overview of the system

The on-board image processing system should execute several parallel tasks. Each task of the algorithm has a dedicated execution unit designed for the specific functionality of the tasks shown in Figure 1. Operation of the different units is synchronized by either a Xilinx Microblaze soft processor core or an ARM Cortex-A9 hard processor core depending of the FPGA used during the implementation.

Detection of potentially dangerous intruder aircrafts in time requires the permanent monitoring of the field of view of $220^\circ \times 70^\circ$ in front of our UAV. To be able to perform robust visual detection of an aircraft, it should be at least 3 pixels large in the captured image. For a Cessna 172 class aircraft, with 10 meters wingspan, 0.1 degree/pixel resolution is minimum required. This means an overall minimum 2200×700 pixel resolution. The most straightforward solution would be to use one piece of high resolution camera with a low distortion ultra-wide angle optics. However, the problem with this setup is that the size and the weight of the camera and especially the ultra-wide view angle optics is way beyond the acceptable limits. Therefore we have decided to apply multiple small cameras.

978-1-4799-6007-1/14/$31.00 © 2014 IEEE

The system can handle several cameras which are connected to the FPGA directly. In the current setup 5 cameras are used which can be either low resolution WVGA (752×480) or a high resolution (1280×960) cameras.

Input of the cameras are combined by the Gather unit by concatenating the rows of the different cameras forming one large (3760×480) image. The concatenated image flow is saved into a SATA SSD for later analysis, saved into the on-board DDR memory and feed into the Adaptive threshold module.

During adaptive threshold a 5×5 sized neighborhood of the pixels is considered and the task is solved by a specialized Falcon processor. The resulting binary image is also saved to the off-chip memory and the binary objects are labeled and their centroids are computed by the Labeling Centroid Unit. Both units are running on 150MHz clock frequency.

The MicroBlaze/ARM then goes through the list of objects found. Based on the size and centroid of the objects suspicious locations are identified and fovea (region of interest, ROI) processing performs one after the other, by instructing the binary and the grayscale ROI processors to cut out the required windows, copy them into the internal block memories of the FPGA, and execute the program sequences.

A. Fovea Processor

The Fovea Processor contains grayscale and a binary window processor unit, and another unit, responsible for the calculation of the deviation of the pixel train on the perimeter of the objects. Grayscale operators, of the algorithm are performed by the unit shown in Figure 2. The 128x128 sized foveae is stored in internal block RAMs (BRAM) of the FPGA. The number of foveae can be configured according to the requirements of the image processing algorithm. Fast off-chip DRAM access is provided by a Direct Memory Access (DMA) engine which can cut out the 128x128 sized foveae from the input image. Loading of one fovea can be performed in 27.3μs while one processing step is carried out in 110μs.

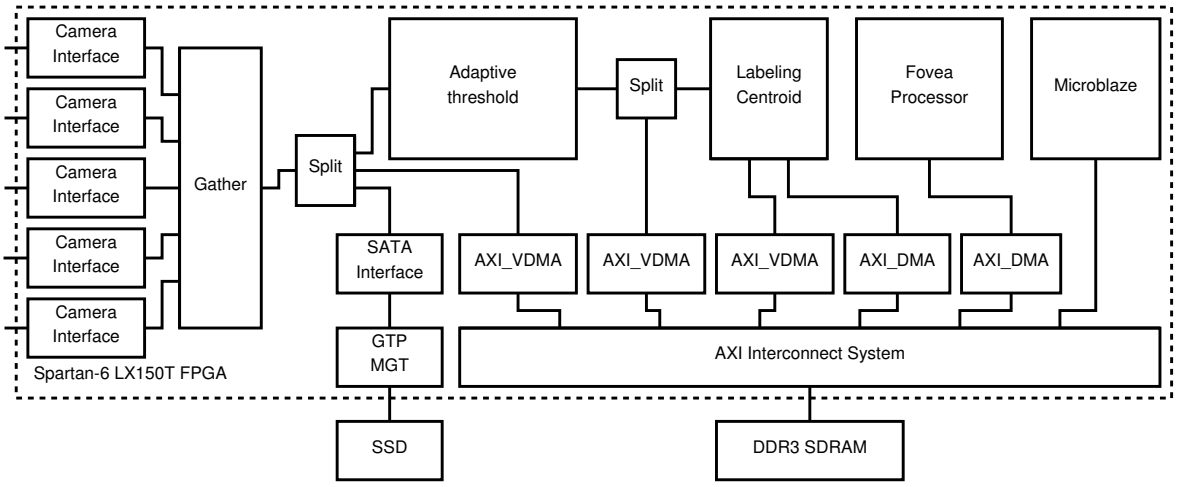

Fig. 1. Block diagram of the complete system

The arithmetic unit of the processor contains an array of highly optimized Processing Elements (PE) from, which only one is activated during an operation. The PE array has a modular structure where existing functions can be easily removed and new functions can be easily inserted before synthesizing the unit according to the requirements of the image processing algorithm. The supported operations are convolution, thresholding, and arithmetic operations such as addition, subtraction, multiplication and absolute value computation. These operations are used for matching areas around feature points and identifying displacements.

The architecture of the binary fovea processor is similar to the grayscale processor. Here the internal BRAMs store the 128x128 sized binary foveae also. However, the image processing algorithm requires more binary and morphological operators than gray-scale operators, therefore the binary image processor is designed for higher performance. Each BRAM is configured with a 128bit wide data bus and all the pixels in a row are computed in parallel here. The supported operations are erosion, dilation, single pixel removal, reconstruction, and two input-one output logic operations such as AND, OR, and XOR. According to the smaller amount of data loading of one binary fovea requires 3.4μs and its processing time is 866.6ns.

In addition to the spatial image processing operations the grayscale processor is extended with a specialized unit to accelerate tracing of the boundary of the objects, which is an important step to avoid false positive intruder detection near the edges of the clouds. The operation works on the original grayscale and the labeled image which is loaded into the BRAMs of the gray-scale processor. The boundary is traced counterclockwise and pixels of the object are searched in the 8 connected neighborhood. Computation of the mean value and deviance of the pixels of the traced path is also computed by this unit to reduce the load on the Microblaze/ARM processor.

For demonstration and test purposes previously recorded image sequences can be transferred to the system via Gigabit

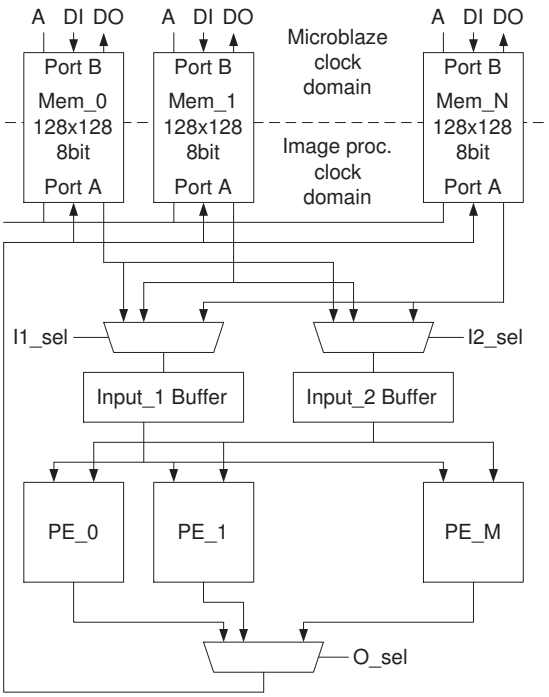

Fig. 2. Block diagram of the Gray-scale processor

Ethernet interface. Results of the image processing steps after threshold and labeling can be shown on a display.

III. Conclusion

A five-camera vision was introduced. The system was designed to be able to operate on UAV platforms. The processor units were implemented in FPGA, and they were designed for deliver the computational needs of an on-board UAV collision avoidance device. The processor units were capable to evaluate a 150Mpixel/sec video flow real-time, by applying multi-foveal processing approach.