Volume and power optimized high-performance system for UAV collision avoidance

Zoltán Nagy[*†], András Kiss[*†], Ákos Zarándy[*†], Bálint Vanek[*], Tamás Péni[*], József Bokor[*], Tamás Roska[*†]

[*]Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA-SZTAKI) Budapest, Hungary

[†]Pázmány Péter Catholic University, The Faculty of Information Technology, Budapest, Hungary,

***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. We report here a case study, where we implemented a many-core processor system, which can process a 100 megapixels/sec video flow, identifying remote airplanes, tracking flying objects by implementing computationally intensive Kalman filters. The introduced processor system is implemented in Spartan 6 FPGA, and consumes less than 1W.**

***Index Terms*— many-core architecture, collision avoidance, UAV, vision-based control, topographic operators, cellular processor arrays**

I. Introduction

IN the recent years, the development of the unmanned aerial vehicle (UAV) technology has reached a decent maturity level, which allows autonomous flights on predefined paths, or even seeking for ground structures or targets autonomously. However, collision avoidance system is not yet developed for small or medium sized UAVs; hence they may collide to each other or with manned aerial vehicles, which can lead to fatal accidents. To build this missing technology component, with the support of the Office of Naval Research (ONR) we have started to develop an on-board non-cooperating collision avoidance device. Such device has two major pieces, a sensory one to identify the other aircraft, and the control one, to design and execute an avoidance maneuver. In the sensory part, we consider visual sensors rather than radar, because the latter is too bulky and too expensive to fit on a small or medium sized UAV [1], [2].

The general requirement of an on-board collision avoidance system is the ability to perform Sense and Avoid functions at an "equivalent level of safety" (ELOS) to manned aircraft while not negatively impacting the existing infrastructure and manned Traffic Alert and Collision Avoidance System (TCAS) that create today's safe airspace [3], [4], [5] . To be able to achieve this strict requirement, we have to use 3 pieces of HD cameras as we have showed in [10], and the system should be able to perform real-time the image processing functionality, what we have described in [6]. To be able to handle this computationally heavy task, we needed to implement a many-core computer system in an FPGA. This paper introduces the architecture of the processor system, and gives the most important parameters, like computational performance, resource usage, and power consumption.

II. On-board image processing 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 task. Operation of the different units is synchronized by a Xilinx Microblaze soft processor core [7],[8]. The system can handle several cameras which are connected to the FPGA directly. The processing is done in two steps. First the suspicious objects which are considered to be candidate remote aircrafts are detected during the full-frame preprocessing step. Then, windows containing these objects are cut out from the high resolution images. These windows are called foveas. The foveas including the suspicious objects are further processed by the gray-scale and binary foveal processors in the second step. In this step, the most computationally intensive aircraft identification parts of the algorithm are executed on the foveas only. Block diagram of the on-board image processing and motion estimation system is shown in Fig. 1.

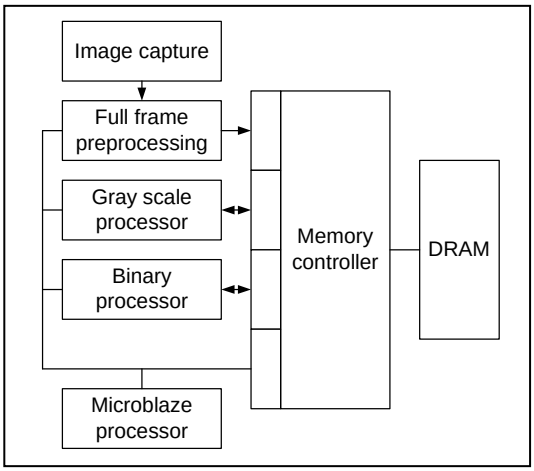

Fig. 1. Block diagram of the proposed image processing architecture

A. Image preprocessing system

The main task of the image preprocessor part is to provide a physical interface for the cameras, receive pixel data and identify suspicious objects. The main parts of the image preprocessor are shown in Fig. 2. The incoming pixel streams are processed immediately and the results along with the original pixel data are saved into the DRAM memory

connected to the FPGA. Depending on the size of the off-chip memory several input frames can be saved for further analysis. Resolution, frame rate and the pixel clock of the cameras do not typically match neither the clock frequency of the Microblaze processor's nor the clock of the memory interface. Therefore the image preprocessor works on three different clocks while synchronization between the clock domains are performed by using FIFO buffers.

The incoming frames are converted to binary images by using an adaptive threshold operation where the local average calculation can be achieved in either 3×3, or 5×5 or 7×7 windows.

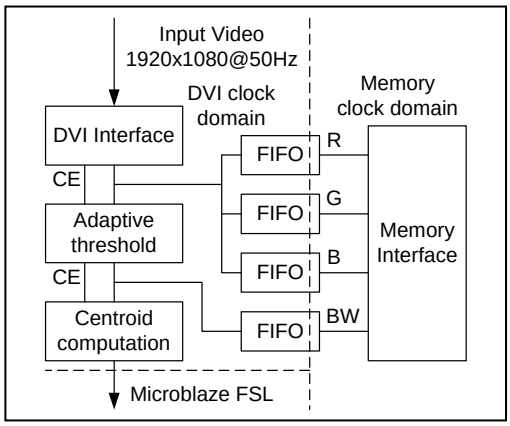

Fig. 2. Architecture of the full frame preprocessor

The next step is detection of the coordinates of the suspicions objects. First, the white pixels are counted in all non-overlapping 32×32 pixel sized parts of the thresholded image. The horizontal and vertical coordinates and the number of the found white pixels are sent to the Microblaze processor where the coordinates of the final 128 × 128 sized foveas are computed. These foveas are further investigated by the gray-scale and binary processors.

B. Grayscale processor

Grayscale operators of the algorithm are performed by the unit shown in Fig. 3. As opposed to the high resolution full frames, which were stored in external DRAM, the 128x128 sized foveas are stored in internal block RAMs (BRAM) of the FPGA. The number of foveas can be configured according to the requirements of the image processing algorithm. Fast offchip DRAM access is provided by a Direct Memory Access (DMA) engine which can cut out the 128×128 sized foveas from the input image. For efficient utilization of the available memory bandwidth, the top left corner of a fovea must fall on a 32 pixel boundary. The on-chip memories should also be accessed by the Microblaze processor to make decisions based on the results of the image processing algorithm. However, the grayscale image processing unit can run on higher clock frequency than the Microblaze processor, therefore the on-chip BRAM memories are used in dual-ported configuration where each port is connected to a different clock domain as it is shown in Fig. 3.

Foveas are processed by an architecture similar to the Falcon emulated digital CNN-UM processor [9]. However, the arithmetic unit of the processor here contains an array of highly optimized Processing Elements (PEs) 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 utilization of the partial reconfiguration feature of the modern FPGAs makes it possible to load a new mix of PEs to the FPGA without interrupting other functions of the device. This can be very useful for example to adapt to changing visibility conditions on-the-flight.

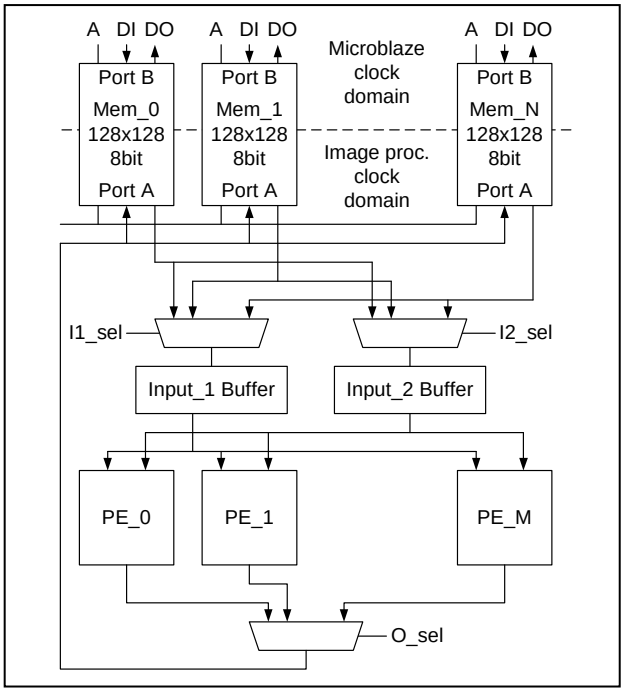

Fig. 3. Architecture of the grayscale processor

The supported operations are diffusion, local average calculation, orientation selective edge detection, thresholding, and arithmetic operations such as addition, subtraction, multiplication and absolute value computation.

Operation of the processor is initiated by the Microblaze processor, by sending an instruction word which contains the address of the three distinct memories (two sources and the target) and one processing element. The input memories provide data to fill up the input buffers where 3×3 sized neighborhood is generated for each pixel. These 9 neighboring pixels are handled in one single step by the processor elements. The result of the selected operator is saved into the third memory. Completion of the operation is indicated by generating an interrupt for the Microblaze processor.

Result of the operation can be quickly evaluated by the Microblaze processor by checking the global status signals. Completely white and black result is indicated by the global white and black signals while steady state of an iterative operation can be checked by the global change signal.

C. Binary processor

The architecture of the binary processor is similar to the grayscale processor. Here the internal BRAMs store the 128×128 sized binary foveas also. They are accessible both by the Microblaze and the binary processors as shown in Fig. 4.

However, the image processing algorithm requires more binary and morphological operators than grayscale operators, therefore the binary image processor is designed for higher performance. Port A of each BRAM is configured as a 128bit wide data bus and all the pixels in a row are computed in parallel here (Fig. 4). Due to architectural restrictions of the Spartan-6 FPGA data bus of the BRAMs, one BRAM can provide maximum 36 bits parallel only, therefore four pieces of 18kbit BRAMs are required for each memory blocks. In these blocks, four binary images can be stored at a time. The input buffers store three lines from the binary image and the processing is carried out by a linear array of 128 binary processors.

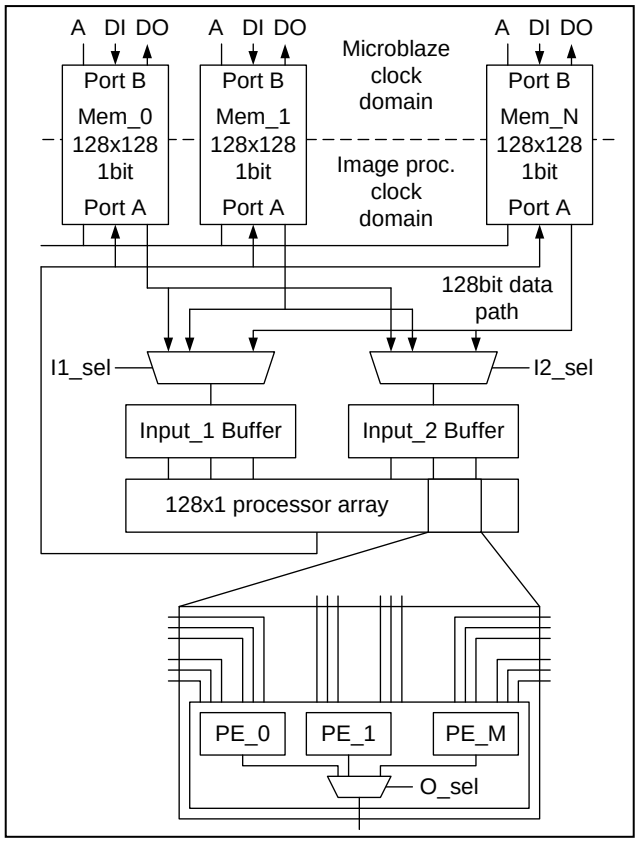

Fig. 4. Architecture of the binary processor

The supported operations are erosion, dilation, single pixel removal, reconstruction, and two input-one output logic operations such as AND, OR, and XOR. The global white, black and change signals are also implemented.

D. Double precision floating-point vector processor

From the results of the grayscale and the binary foveal operations, the Microblaze processor decides whether a candidate object was a real remote aircraft or not. Then it calculates its (x,y) coordinate and its wingspan. Using this information, motion prediction is performed by an unscented Kalman-filter. Majority of the operators during the solution of the Kalman-filter are matrix algebraic operators, matrix-vector, matrix-matrix additions and multiplications. In the last step of the Kalman filter calculation, a small matrix should be inverted where the matrix can be nearly singular, therefore double-precision floating point number representation is required to avoid numerical instability. Unfortunately, the Microblaze processor supports single precision floating-point operations. On the other hand its computing performance is not high enough to calculate the required operators for the unscented Kalman filter real-time. Therefore, a vector processor, shown in Fig. 5, was implemented.

The vector processor is build from a scratch-pad memory, several vector registers, and a floating-point adder and multiplier, because the majority of the required operations are multiplications and additions. The state of the filter and several matrices are stored in the scratch-pad memory where high-speed memory access by the Microblaze processor is critical, because it executes other operations like comparison, division and square root.

The vector processor can compute single addition, multiplication and multiply-add operation moreover one addition and one multiplication can be computed in parallel when distinct result registers are used. The length of the vectors is limited by the depth of the vector registers and can be configured on-the-fly to adapt to the requirements of the Kalman-filter. Using the six input LUT structure of the Spartan-6 FPGA, we found that the optimal depth of the vector registers is 64 elements.

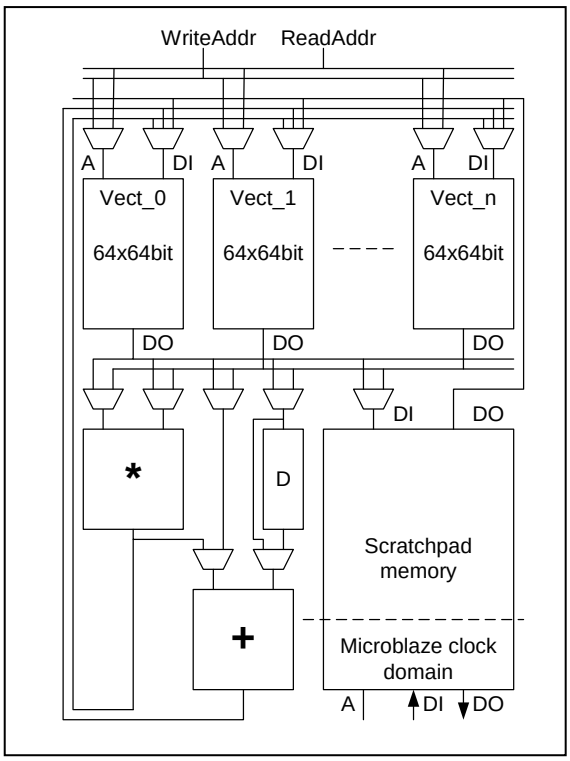

Fig. 5. Architecture of the floating point vector processor

III. IMPLEMENTATION AREA, PERFORMANCE

The prototype of the system is implemented on a Xilinx SP605 evaluation board which is equipped with a XC6SLX45T Spartan-6 FPGA, 128MB DDR3 DRAM emory, Gigabit Ethernet interface, one FMC-LPC connector and several other peripherals. Camera signals are connected via an FMC daughter board. Implementation details are described by using an example system configured to handle one HD resolution (1920×1080@50Hz) image flow. In the final system, 3 pieces of HD cameras running at 16.6Hz will

be used. Area requirements and power consumption of the full system are summarized on Table I.

The system use standard Xilinx IP cores to access devices on the prototyping board such as the on-chip hard memory controller block (MCB DDR3) and the soft Gigabit Ethernet IP core (Soft TEMAC) and other IPs for system level tasks such as clock management, interrupt controller, etc. The Microblaze processor part of the system is running on a moderate 66MHz clock frequency while the on-board DDR3 DRAM chip has a 400MHz clock frequency providing 1.6GB/s peak memory access bandwidth. The full frame preprocessing part is running on 165MHz pixel clock. The grayscale processor is configured to use all PE types described in Section II-B and grayscale foveas can be stored in four on-chip memories. The binary processor also contains all PE types described in Section II-C and four BRAM units where used to store 16 binary foveas. Both units are operating on 150MHz clock frequency which is limited by the BRAMs and the dedicated multipliers of the FPGA.

Each grayscale operation can be executed in 16,384 clock cycles which requires 110µs. This means that about 9100 operations/sec can be reached using 150MHz clock frequency. The grayscale processor consumes relatively small area therefore its performance can be improved by using four arithmetic units if required.

The binary operations can be executed significantly faster, only 0.86µs is needed for one operation. This equals to more than a million pieces of 128×128 sized binary operations in a second!

Assuming three pieces of 16.6Hz HD image flow, more than 180 grayscale and 23,000 binary operations can be executed on each frame. The current image processing algorithm executes 4 grayscale and about 50 binary operations, therefore approximately 45 foveas can be examined on each frame.

The system occupies about one half of the FPGA where the image processing system requires about one quarter of the chip and the Microblaze subsystem, the memory controller and the Gigabit Ethernet MAC can be implemented on one other quarter. The remaining free space makes it possible implement more functions on the FPGA such as collision estimation and trajectory generation.

Dynamic power consumption of the system is estimated by the Xilinx Power Analyzer and the results are summarized on Table I. Quiescent power consumption of the system is 0.6W which is more than half of the total power consumption. On the FPGA the clock generation and the memory interface require the majority of the power. Total power consumption of the system is around 1W which fits well into the power budget of a UAV.

IV. Conclusions

In this article a many-core image processing unit and a double precision floating point vector processor unit were introduced. 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 100Mpixel/sec video flow real-time, by applying multi-foveal processing approach. Their overall power consumption was under 1W.

Acknowledgment

The ONR Grants (N62909-11-1-7039, N62909-10-1-7081) is greatly acknowledged.

References

[1] Hutchings, T., Jeffryes, S., and Farmer, S. J., “Architecting UAV sense & avoid systems,” *Proc. Institution of Engineering and Technology Conf. Autonomous Systems*, 2007, pp. 1–8.

[2] Fasano, G., Accardo, D., Forlenza, L., Moccia, A., and Rispoli, A., “A multi-sensor obstacle detection and tracking system for autonomous UAV sense and avoid,” XX *Congresso Nazionale AIDAA, Milano*, 2009.

[3] Dempsey, M., “U.S. Army Unmanned Aircraft Systems Roadmap 2010-2035,” Tech. rep., U.S. Army UAS Center of Excellence, 2010.

[4] DeGarmo, M. T., “Issues Concerning Integration of Unmanned Aerial Vehicles in Civil Airspace,” Tech. rep., MITRE Center for Advanced Aviation System Development, 2004.

[5] Cox, T. H., Nagy, C. J., Skoog, M. A., Somers, I. A., and Warner, R., “Civil UAV Capability Assessment,” Tech. rep., NASA Dryden Flight Research Center, 2004.

[6] T. Zsedrovits, Á. Zarándy, B. Vanek, T. Péni, J. Bokor, T. Roska, “Collision avoidance for UAV using visual detection”, IEEE International Symposium on Circuits and Systems, ISCAS 2011, Rio de Janeiro.

[7] Z. Voroshazi, A. Kiss, Z. Nagy, and P. Szolgay, “Implementation of embedded emulated-digital CNN-UM global analogic programming unit on FPGA and its application,” International Journal of Circuit Theory and Applications, vol. 36, no. 5-6, pp. 589–603, 2008.

[8] “Xilinx products homepage,” [Online] http://www.xilinx.com, 2011.

[9] Z. Nagy and P. Szolgay, “Configurable Multi-layer CNN-UM Emulator on FPGA,” *IEEE Transaction on Circuit and* Systems *I: Fundamental Theory and Applications*, vol. 50, pp. 774–778, 2003.

[10] B. Vanek, T. Péni, T. Zsedrovits, Á. Zarándy, J. Bokor and T. Roska., "Vision only Sense and Avoid system for small UAVs”, *Am.Control Conference 2011*

Module	Slice Reg	LUTs	LUTRAM	BRAM/FIFO	DSP48A1	Power
MCB_DDR3	1978	2172	45	0	0	0,195
Soft_TEMAC	2636	2452	109	6	0	0,00548
image_proc_0	1737	3388	66	16	0	0,00973
image_proc_gray_0	604	622	100	32	2	0,00462
microblaze_0	1056	1363	113	0	3	0,00507
vga_in_ctrl_0	1751	1896	344	2	0	0,0223
other	1421	1356	48	4	0	0,211
System total	11183	13249	825	60	5	0,454
Spartan-6 XC6SLX45T (available)	54320	27288	6408	116	58	
used	20,59%	48,55%	12,87%	51,72%	8,62%	

Table 1. The resource requirements and the power consumption of the system.