FOREGROUND DETECTION AND IMAGE SEGMENTATION IN A FLEXIBLE ASVP PLATFORM FOR FPGAS

Roman Bartosinski, Martin Danek, Jaroslav Sykora, Lukas Kohout

Department of Signal Processing
Institute of Information Theory and Automation (UTIA) of the ASCR, v.v.i.
Pod Vodarenskou vezi 4, Praha, Czech Republic
{sykora, kohoutl, bartosr, danek}@utia.cz

Petr Honzik

CIP plus s.r.o.
Milinska 130, Pribram, Czech Republic
petr.honzik@cip.cz

ABSTRACT

This demonstration shows an early prototype of low-level image processing to be used in an embedded smart camera, that is foreground detection and image segmentation. The example uses camera with resolution 640x480 pixels for input images processed at 100MHz in the FPGA. The input can be easily extended to higher resolutions. The processed output is displayed on LCD screen.

1. PLATFORM

The implementation platform used in this demo is the Application-Specific Vector Processor (ASVP) described in [1] and [2]. In traditional work-flows based on direct implementation in hardware description languages hardware accelerators must be recompiled and re-synthesized each time to generate a new FPGA configuration (bitstream) that can be verified. The drawback of the traditional approach is the long synthesis time caused mainly by a slow place&route process of the low-level tools. This also limits the end user to minor changes of the implemented algorithm such as tuning of coefficients. In our approach we abstract the custom accelerator into a specialized programmable architecture based on a network of programmable data-streaming computing nodes with local memory. To design for the platform, in the first step the high-level source code is analyzed and domain features are extracted. Based on the required domain features new computing kernels are implemented in the computing nodes or the existing ones are modified. The architecture is programmable by firmware to the extent that the sequencing of basic processing kernels can be changed without re-synthesizing the hardware, hence source code modifications that do not change the problem domain can be tested quickly for they require only fast firmware recompilation.

The system-level view is presented in Figure 1. Similar to the streaming architectures the execution control is hierarchical:

This work has been supported from project SMECY, project number Artemis JU 100230 and MSMT 7H10001.

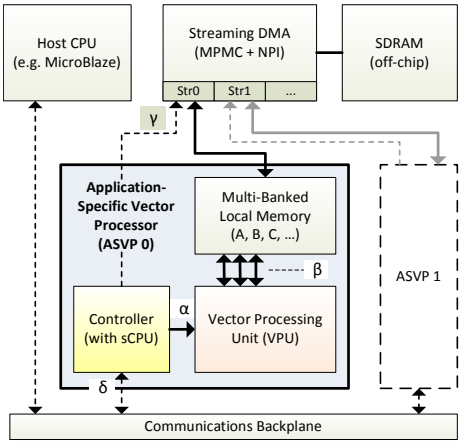

Fig. 1. A system-level organization of an ASVP-based core.

1. *Task scheduling* is executing in the Host CPU (e.g. MicroBlaze). Optional inter-core synchronization is handled by the Communications Backplane (δ).
2. *Scheduling of the vector instructions* is realized in a simple scalar control processor (sCPU) embedded in each ASVP core .The sCPU forms and issues wide instruction words (α) to the Vector Processing Unit (VPU).
3. *Data path multiplexing* and vector processing is realized autonomously in the VPU. The unit handles both the vector-linear and vector-reduction operations, as well as local memory banks access scheduling (β).

The whole video processing chain is developed and implemented on a Xilinx SP605 development board with a low cost, low power Xilinx Spartan 6 FPGA. The MicroBlaze soft-core processor is employed as the host CPU. Each algorithm is implemented in a separate ASVP. The entire system runs at 50MHz except the data flow units that run at 100MHz. All accelerators are synthesized with four memory banks with 1024x32bit words.

Table 1. Memory required to store contextual data - Modified Mixture of Gaussians (4 Gaussian models per pixel).

Format	Resolution	Contextual data
VGA	640x480	23.4 MB
HD 720	1280x720	72 MB
HD-DV	1440x1080	118,65 MB
Full HD 1080	1920x1080	162 MB

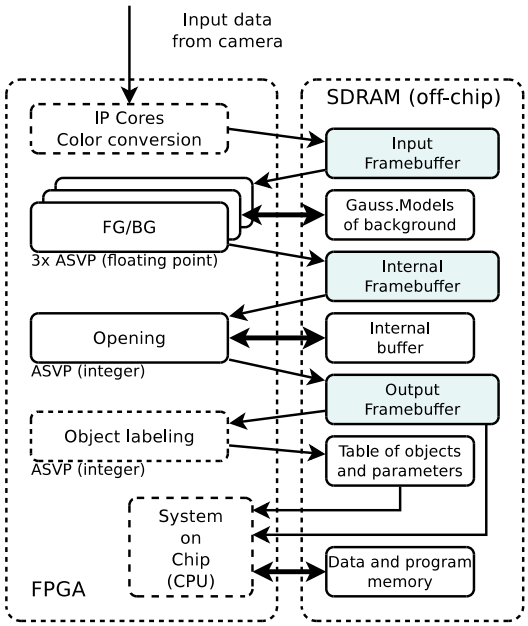

Fig. 2. The data path of the video chain.

2. RESULTS

The foreground detection is derived from the common Mixture of Gaussian Models algorithm that has been modified to eliminate operations difficult to implement in hardware such as division and square root while maintaining the convergence properties of the original algorithm (normalization). The memory requirements of the algorithm used in the demonstration is shown in Table 1. The demonstration uses image resolution of 640x480 as we think it is sufficient for performing initial analysis of the scene; the higher resolutions are to be used for subsequent image processing.

The data path of the implementation is shown in Figure 2. The labelling process is implemented in software executed in MicroBlaze in the current version.

As mentioned above the video application has been implemented in the system on a chip with hardware compute cores on a Xilinx SP605 development board. The board contains power management chip accessible through PMBus. Power consumption has been measured by this chip on power rail that supplies power to the FPGA internal logic.

The average power consumption of the entire system on a chip with disabled accelerators is $Pavg = 433.2mW$. Table 2 contains performance data for the motion detection algorithm executed on one, two and three floating-point compute cores for an input video stream with resolution of 640x480 pixels. It is shown that the time required to process one frame is a multiple of the number of used compute cores, but the consumed energy is the same. The initial requirements for the minimal frame rate 5 FPS can be reached with three cores running in parallel; each core computes one-third of each frame.

Table 2. Motion detection. Implementation parameters measured on Xilinx SP605.

Parameter	Number of used ASVP		
	1	**2**	**3**
$FPS[s^{-1}]$	2.07	4.14	6.21
$P_{SoC}[mW]$	460.26	487.58	509.68
$P_{ASVP}[mW]$	27.061	54.38	76.48
Computation of one frame			
$t[s]$	0.436	0.218	0.146
$E[mJ]$	11.81	11.87	11.13

Table 3. Morphological opening process. Implementation parameters measured on Xilinx SP605.

Parameter	Value
$FPS[s^{-1}]$	11.51
$P_{SoC}[mW]$	452.419
$P_{ASVP}[mW]$	19.219
Computation of one frame	
$t[s]$	0.0868
$E[mJ]$	1.67

Table 3 shows average of consumed energy for processing one frame in opening function. The current implementation of opening function reaches frame rate 11.51 FPS. Consumed energy is 1.67 mJ for one frame and it is only one tenth of energy consumed by motion detection operation.

3. REFERENCES

[1] Jaroslav Sykora, Lukas Kohout, Roman Bartosinski, Leos Kafka, Martin Danek, and Petr Honzik, "The Architecture and the Technology Characterization of an FPGA-based Customizable Application-Specific Vector Processor," in *Proceedings of the 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. 2012, DDECS '12, pp. 62–67, IEEE.

[2] Jaroslav Sykora, Lukas Kohout, Roman Bartosinski, Leos Kafka, Martin Danek, and Petr Honzik, "Reducing Instruction Issue Overheads in Application Specific Vector Processors," in *Proceedings of the 15th Euromicro Conference on Digital System Design*. 2012, DSD '12, pp. 600–607, IEEE Computer Society.