Image Signal Processors on FPGAs

Di Wu and Andreas Moshovos
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
{*wudi7, moshovos*}*@eecg.toronto.edu*

***Abstract*— An Image Signal Processor (ISP) converts raw imaging sensor data into a format appropriate for further processing and human inspection. This work explores FPGA-based ISP designs considering specialized and programmable implementations and proposes an ISP using a programmable generic processing unit with comparable performance versus the dedicated implementations.**

This work considers the most commonly performed operations in an Image Signal Processor (ISP) including demosaicing, color correction, filtering, color space conversion, and gamma correction demonstrating that the 2-D filter is the bottleneck. Further, it compares the performance, area and power of (1) a highly specialized 2-D filter (SF) that exploits the symmetry of certain filter kernels, (2) a general 2-D filter (GF), and (3) a generic processing unit filter (GPUF) that is programmable and tailored for ISP operations.

We first implement the SF and the GF with on-chip BRAM-based row buffers followed by a 7×7 2-D FIR filter [1]. These specialized filters are rigid and support one type of operation only. Accordingly, we explore the possibility of a more general structure that can support different operations. The ISP operations generally have two stages, (1) selecting data and/or coefficients to compute intermediate results, and (2) summing up the intermediate results. We present the GPUF that implements this functionality via a vector of DSP blocks followed by a programmable reduction tree that sums up intermediate results at different reduction levels.

Fig. 1 shows the high level diagram of our generic ISP computation unit (GPUF). The GPUF uses a 2-D shift register file that provides simultaneous access to all of its elements. The 2-D shift register file supports row shift and column shift operations. A column shift operation shifts all the columns to the left in a circular manner. Similarly, a row shift operation shifts a new row of pixel data into the 2-D shift register, and all the rows currently in the register file are also shifted upwards. The DSP block units are fed by the 2-D shift register file and the coefficient register file which store pixel data and coefficients respectively. Data and coefficients are routed to their designated DSP block depending on the kernel type through the data switching interface. The results for the DSP blocks are sent to the reduction tree.

Qadeer et al. proposed the Convolution Engine (CE), a similar ASIC-based structure [2] specialized for the convolution-like applications. The CE focuses on post ISP algorithms such as H264 etc. Our generic computation unit uses a similar idea, but is tailored to process ISP related operations with an FPGA-friendly implementation.

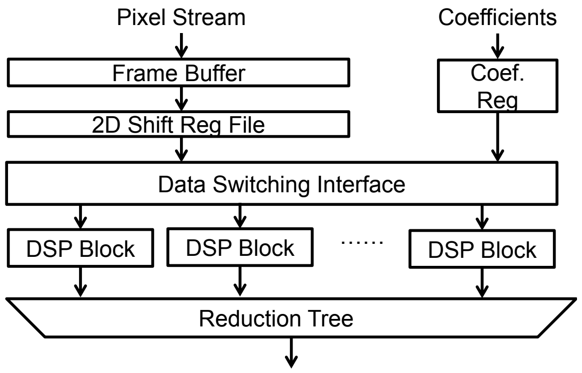

Figure 1: The generic processing unit.

The generic processing unit supports up to 7×7 kernels. We find that because the GPUF has a fixed number of multipliers, the number of filtering operation that can be processed concurrently drops for large kernels, hence affecting the frame rate. Table I summarizes Fmax, area and power of each filter implementation, as well as the corresponding frame rates processing 1080p images with 7×7 kernels.

Configuration		Frames per Second	Fmax	ALUT	DSP	Power
SF		191.85	401MHz	566	48	100.48mW
GF		186.10	378MHz	1224	150	562.79mW
GPUF	3x3	185.51	385MHz	4773	180	738.04mW
	5x5	132.49				
	7x7	82.44				

Table I: Results.

References

[1] D. Bailey, *Design for Embedded Image Processing on FPGAs*. Wiley, 2011.

[2] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: Balancing efficiency & flexibility in specialized computing," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*. ACM, 2013, pp. 24–35.