Reconfigurable Hardware Objects for Image Processing on FPGAs

Jan Kloub, Petr Honzík and Martin Daněk
Institute of Information Theory and Automation of ASCR
Pod Vodárenskou věží 4
182 08 Praha 8, Czech Republic
Telephone: (+420) 26605 2472
Fax: (+420) 26605 2511
Email: {kloub, peters, danek}@utia.cas.cz

***Abstract*—Embedded systems are getting more complex; that is why the high level of abstraction is required during the development process. High abstraction methods simplify implementation of complex computation systems and shorten the time to market. This paper presents an implementation of a graphic computing element (GCE) which can be used as a runtime parametrized building block in image processing applications in FPGAs. In terms of the object oriented model GCE encapsulates its internal data representation and rules for their manipulation. Several basic image processing operations have been implemented (Sobel edge detection, Gauss, mean, etc. filtering). These operations are called as GCE methods. Because of high spatial dependency of image data in image processing, an efficient image data reuse method has been implemented.**

I. Introduction

Image processing is getting widely used in embedded systems for many applications, such as object detection, security or video surveillance. Since application requirements can vary in time, resources should be efficiently reused. One way to reuse the same hardware resource is hardware reconfiguration.

In the case of general purpose image processing common computational constructs have to be found. Many image processing operations, e.g. edge detection, scaling, sharpening and filtering, can be implemented using a discrete 2-D convolution [1]. In general convolution is performed as a weighted sum of neighbouring pixels, and requires a significant number of operations which can be implemented in parallel on FPGAs [4]. A set of neighbour weight values (coefficients) is called a kernel. Discrete 2-D convolution is defined as:

$$f[x,y] = h[x,y] * g[x,y] = \sum_{i=-n}^{n} \sum_{j=-n}^{n} h[i,j]g[x-i,y-j]. \tag{1}$$

Here, $h[x,y]$ represents a convolution kernel and $g[x,y]$ represents an image. Convolution is a very general operator in image processing, and can be modified easily by changing its coefficient values; thus the convolution operator is suitable for reuse in image processing.

An image operation implementation should be hidden from the system point of view and only a set of image operation capabilities should be known. One way to do so is to use the object oriented model [2]. Systems should be composed of interacting objects where their internal functional implementation is hidden. Combining the object oriented model and the potential of a reconfigurable convolution core leads to an implementation of a new class of hardware objects [3].

II. Background and Related Work

The architecture of a graphic computing element (GCE) is based on a previous work on basic computing elements (BCEs) for acceleration of DSP operations. BCEs provide high performance for floating point matrix computations [7]. The graphic computing element described here uses a similar concept with a data flow unit and a simple control processor. The simple control processor controls the data flow unit to provide required functionality.

III. Memory Architecture

Memory hierarchy is usually limited by the performance and significant overhead of memory access transactions. It is necessary to avoid duplicate memory accesses as much as possible. In case of image processing high spatial data dependency usually leads to duplicate read operations. The following text describes the method to limit the number of memory accesses.

When the convolution core is in use, it is necessary to keep the number of image lines equal to the convolution matrix height (CMH). Image filtration is done over the last CMH image lines. It is necessary to keep the image line data in the local GCE memory and to reuse some lines in the next step when a new line is read. The last line (the oldest one) is released and the new one is stored instead.

Image lines are stored in memories connected in chain. While a new line is being stored in the first memory, old values are stored in the next one and so on. This way the last $CMH-1$ lines are reused. This results in a significant reduction of memory accesses. Without this reduction the number of memory read accesses would be $Image\_height * CMH$ and with the reduction $Image\_height + (CMH - 1)$. The

978-1-4244-6610-8/10/$26.00 ©2010 IEEE

reduction R is asymptotically equal to:

$$R = \frac{1}{CMH} \tag{2}$$

The convolution core accesses all image line memories in the chain in parallel. Parallel access to the memory chain enables fully pipelined processing.

It is possible to connect more memory chains together if more convolution operations are provided. Such a connection allows to share more image lines. Here a group of convolution cores which cover continuous interval in the vertical direction could share lines. Using this approach the number of memory accesses can be reduced. Reduction R_G can be expressed as:

$$R_G = \frac{1}{GH} \tag{3}$$

GH is length of the vertical continuous interval of convolution cores group.

IV. Results

For evaluation the Xilinx development kit ”XtremeDSP Development Platform Spartan-3A DSP 3400A Edition” was used. The graphic computing element was used in a design with MicroBlaze system based on a PLB bus as the control bus. A custom data bus provides high burst image data transfers. The system clock rate was 62.5 MHz.

The Sobel edge detector [6] implementation was chosen as a case study which is defined for image as a gradient magnitude $s(\mathbf{x})$ and can be computed as

$$s(x) = \sqrt{{\Delta_1}^2 + {\Delta_2}^2} \tag{4}$$

where Δ_1 and Δ_2 represents horizontal and vertical gradient operators. The gradient operators can be represented as convolution kernels which can be found in literature [1] and used in equation 1.

In current GCEs the square root functional unit is not presented and the Sobel is computed as $s(x) = {\Delta_1}^2 + {\Delta_2}^2$ and a multiplier is used for the square power operator.

Three variants of implementations were considered. The first and second implementations use GCE with only one convolution core (CC), and the third implementation uses GCE with two convolution cores. GCE with two convolution cores has all functional units doubled compared to GCE with one convolution core. The first implementation is decomposed in two separate steps driven by the MicroBlaze system. The horizontal edge detection is performed first, and then vertical edge detection is calculated and added with the previous result. The second implementation uses the same operations, but it is driven by a simple control processor (PicoBlaze) in GCE. In this case the Microblaze system drives only one operation. The operation implementation is hidden, and both convolution operations are driven by the PicoBlaze firmware (FW). The third implementation uses CGE with two convolution cores. Both the horizontal and the vertical detection is processed concurrently.

The performance results are summarized in Table I. The frame processing time was measured by an oscilloscope. Utilization of hardware resources is summarized in Table II.

Image dimension	**Frames per second**		
	a)	b)	c)
640x480	33.62	46.42	64.58
800x600	24.65	32.99	47.08
1024x768	17.20	22.21	32.64

TABLE I
Sobel image edge detection filter implementation performance (@62.5MHz) a) 1 x convolution core, b) 1 x convolution core with firmware support, c) 2x convolution core

FPGA Resource Type	**1x(CC, Add., Mul.)**	**2x(CC, Add., Mul.)**
Slices	1304/23872 (5.5%)	2157/23872 (9.0%)
FFs	1632/47744 (3.4%)	2761/47744 (5.8%)
LUTs	1831/47744 (3.8%)	2955/47744 (6.2%)
BRAMs	13/126 (10.3%)	20/126 (15.9%)

TABLE II
Graphic computation elements hardware resources utilization summary (Xilinx Spartan-3A DSP 3sd3400afg676-4)

V. Conclusion

This paper presented an implementation of a graphic computing element which supports image data reuse. Image applications require a lot of memory space to store image information. DDR memory can provide enough space, but in the memory hierarchy the DDR memory access has a significant overhead for access transactions. Image data reuse reduces the number of main memory accesses, thus it increases the performance. In our implementation image data are shared over a set of convolution cores, operations can be performed concurrently, and their results can be combined on the fly. Groups of elementary filters can be combined to form applications using the boosting algorithm [5] or multiple image pattern correlations.

Acknowledgment

This work was supported by the SCALOPES project; project number: Artemis JU 100029, MSMT 7H09005.

References

[1] Ballard D.H., Brown C.M.: Computer Vision, In *Englewood Cliffs , New Jersey 07632, PrenticeHall, Inc .*, 1982., ISBN: 0-13-165316-4.

[2] Craig I.D.: Object-Oriented Programming Languages: Interpretation, *Springer-Verlag London Limited*, 2007, ISBN: 1-84628-773-1.

[3] Abel N., Grüll F., Meier N., Beyer A., Kebschull U.: Parallel Hardware Objects for Dynamically Partial Reconfiguration In *FPL '08: Field Programmable Logic and Applications*, Heidelberg, 2008. pp. 563–566.

[4] Ballard D.H., Brown C.M.: Parameterized Convolution Filtering in a Field Programmable Gate Array, Electronic document available at *http://www.rgshoup.com/prof/pubs/FPGA93.pdf*

[5] Mian Zhou, Hong Wei: Face Verification Using GaborWavelets and AdaBoost, In *ICPR'06: Pattern Recognition, 18th International Conference*, Hong Kong, 2006., pp 404–407 .

[6] Pradabpet C., Ravinu N., Chivapreecha S., Knobnob B., Dejhan K.: An efficient filter structure for multiplierless Sobel edge detection, In *CITISIA'09: Innovative Technologies in Intelligent Systems and Industrial Applications.*, Hong Kong, 2009., pp 404–407 .

[7] Daněk M., Kadlec J., Bartosinski R., Kohout L., Increasing the Level of Abstraction in FPGA-based Designs, In *FPL'08: Field Programmable Logic and Applications, International Conference*, Heidelberg, Germany, 2008., pp 5–10.