Design of Real-Time Edge Detection Circuits on Multi-FPGA Prototyping System

Andrej Trost Baldomir Zajc
University of Ljubljana
Faculty of Electrical Engineering
Tržaška 25, 1000 Ljubljana, Slovenia
andrej.trost@fe.uni-lj.si

Abstract

The paper presents design and implementation of real-time edge detection circuits on a multi-FPGA reconfigurable system. A design approach for implementation of pipelined circuit structure on a modular reconfigurable system is described. The digital circuit design for two well known computer vision algorithms, the Canny edge detector and nonlinear Laplace operator, is presented. The circuit description starts with behavioral VHDL. The results of high-level functional simulation performed on sample images are compared to results of algorithm executed in C language. Next design steps are transformation to synthesizeable structural VHDL, circuit partitioning, synthesis and technology mapping. Each partition is implemented in a separate FPGA-SRAM module and hardware verification of the circuit is performed with the PC. Finally, the modules are connected together forming the real-time image processing circuit.

1 Introduction

Edge detection is frequently used low level computer vision operation with the goal of extracting geometric information from digital images [1]. In computer vision systems with real-time image processing requirements, we have to use specialized high speed processors or ASIC circuits [2,3]. ASIC circuits, however, require extensive designing and manufacturing efforts leading to long production time and financial risks. An alternative solution are FPGA integrated circuits [4]. Reconfigurable systems based on FPGA integrated circuits are used for fast prototyping implementations of the complete real-time designs [5,6].

In the edge detection process, we are searching for boundaries of objects in the image distinguished by quick changes of intensity (brightness). Most of the edge detection operators are local operators where the same procedure is repeatedly applied to all image pixels. Local operators are specifically suitable for implementation with digital circuits in pipeline architecture.

We will present design and implementation of the digital circuit for the two well known edge detection operators: Canny edge detector and nonlinear Laplace edge detector. A design flow on a multi-FPGA reconfigurable system is described first, then the edge detection algorithms and circuit architectures are presented.

2 Multi-FPGA Design Flow

Before we begin with the hardware design for the image processing algorithm, we have to study the algorithm and evaluate the results on sample images. The algorithm is simulated in the C environment and output images will be used later on for comparison with hardware simulation, as presented in Figure 1.

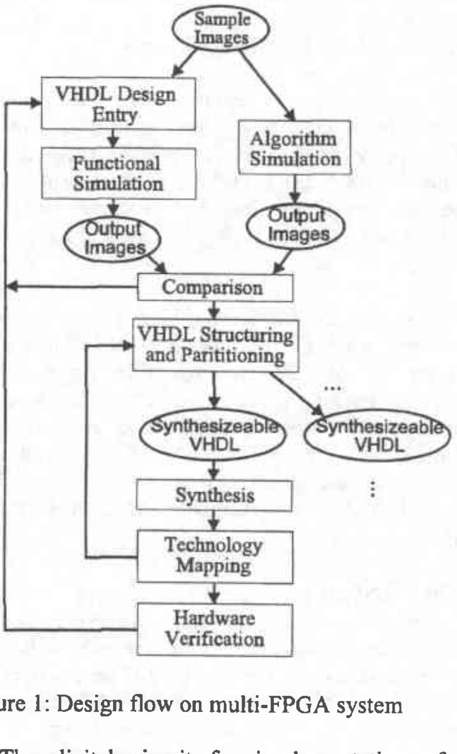

Figure 1: Design flow on multi-FPGA system

The digital circuit for implementation of the algorithm is first described and simulated in behavioral VHDL. The high level VHDL structures allow fast circuit design and simulation of the circuit on sample images. For the simulation purposes we created test bench which reads input pixels from an

ASCII image file and writes output pixels to output file. The ASCII image files are visualized with C programs and can be converted from/to standard image formats. This environment enables comparison of the images obtained by hardware simulation with the images obtained by algorithm simulation.

The next step is transformation of the behavioral VHDL to synthesizeable VHDL and in the same time, partitioning of the circuit according to the target implementation platform. The target platform consists of reconfigurable modules containing Xilinx FPGA devices (XC4008E and XC4010E) and static memory circuits, as presented in Figure 2.

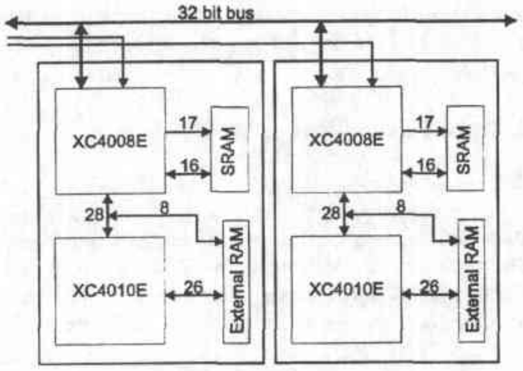

Figure 2: Modular multi-FPGA system

Image processing circuits in the pipeline architecture require a lot of local memory circuits for holding entire image row or several rows in the pipeline. Each module contains for this purpose 64k x 16bits of fast static memory. External memory connectors are provided if larger amount of memory is required.

The process of circuit partitioning, synthesis and technology mapping is repeated until the circuit partitions fit into the reconfigurable modules and timing requirements are satisfied. When the circuit is implemented in the reconfigurable module, we perform a hardware verification. A PC with a parallel port interface connected to the reconfigurable module is used for hardware verification of the emulated circuits.

The hardware verification procedure is similar to the simulation, but runs much faster since the operations are performed in hardware and PC is used only for data transfer. The reconfigurable modules are finally connected in series as presented in Figure 2, forming the real-time image processing system.

3 Canny Edge Detector

The Canny algorithm is frequently used for edge detection on digital images [7, 8]. The algorithm is based on computation of gradient on digital image and searching of local gradient maximums. Figure 3 presents the basic steps for edge detection.

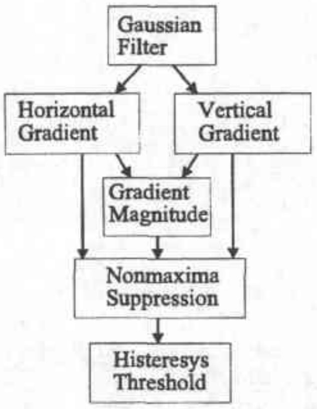

Figure 3: Canny edge detection algorithm

The first step is smoothing the image with Gaussian filter followed by computation of gradient in horizontal and vertical direction. Gradient magnitude is then obtained as a sum of squares of both gradients.

In the process of nonmaximum suppression only the local gradient magnitude maximums are preserved.

In the last step, we use threshold mechanism in order to remove smaller peaks of magnitude. Canny algorithm uses double threshold filter. Pixels with the magnitude above high threshold are classified as edge pixels. Pixels which have the magnitude above low threshold and are connected to edge pixels are also classified as edge pixels. Threshold filter uses a recursive procedure in order to find all edge pixels.

3.1 Canny Circuit Architecture

Figure 4 presents the circuit architecture for one-dimensional convolution with Gaussian vector (1, 3, 4, 3, 1). The presented circuit has minimal critical path through adders and multipliers.

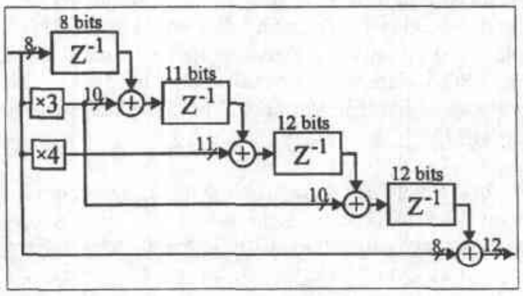

Figure 4: One-dimensional Gaussian convolution

The circuit accepts 8 bit input data. The exact amount of required bits for intermediate results were computed in order to decrease area and increase speed of the circuit. In order to normalize the Gaussian filter output to 8 bits, we have to add a divider by sum of coefficients at the end of the circuit. The complete Gaussian convolver consists of two equal one-dimensional convolvers and memory for holding intermediate image in the pipeline.

The circuit for computation of local gradient is presented in Figure 5. A FIFO memory in the size of one image line is required for computation of vertical gradient.

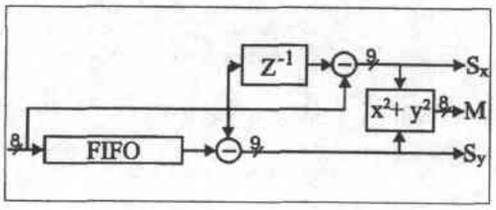

Figure 5: Circuit for gradient computation

The input pixels are 8 bit unsigned numbers, while the gradients are signed numbers. For obtaining the correct results, we have to extend the input data to 9 bits before computation of difference.

For the coding of magnitude which is a sum of squares of 8 bit numbers, we would need 17 bits. Experimentally we found out that using only lower 8 bits and saturation mechanism is sufficient most of the time. The results of square computation are cut to 8 bits and if more bits are required, the maximum 8 bit number (255) is used.

Nonmaximum suppression circuit uses both gradients and the gradient magnitude on its inputs. The gradients are used to determine closest neighbor in the gradient direction which is given in terms of a unit distance in x and y direction *(dx, dy)*. For example, gradient direction *dx=1, dy=0* is defined by:

$$|S_y| < tg22.5^\circ \cdot |S_x|$$

The other directions are derived in similar way. The conditions for direction were slightly modified in order to simplify the digital circuit. Instead of *tg22.5°*, which is *0.414*, we used constant *0.5*, which presents the angle *26.6°*. The conditions can be summarized in the following form:

$$(dx, dy) = \begin{cases} (1,0), & |S_y| < 0.5 \cdot |S_x| \\ (0,1), & |S_x| < 0.5 \cdot |S_y| \\ (1,-1), & S_x \cdot S_y < 0 \\ (1,1), & \textit{others} \end{cases}$$

For each magnitude pixel *M*[*i, j*], we choose between its eight neighbors the pixel closest to the gradient direction (∂x, ∂y) and mark it as *M+*. Similarly we find the neighbor pixel in the opposite direction and mark it as *M-*. The pixel *M*[*i, j*] is local maximum when:

$$(M > M^+ \wedge M \geq M^-) \vee (M > M^- \wedge M \geq M^+)$$

Each magnitude pixel, which is not local maximum, is given the value 0.

The circuit architecture for nonmaximum suppression is presented in Figure 6. The input pixels of two image rows are hold in a pipeline using two FIFO memories. The outputs of the FIFO memories and the current pixel input are further delayed in registers inside FPGA holding all 9 pixels of required 3x3 neighborhood.

The comparison is always performed between central pixel and one of its neighbors. The register outputs are connected through 3-state bus to the comparator. After two comparisons (in gradient and in opposite direction) the decision is made weather the output is central pixel or constant 0.

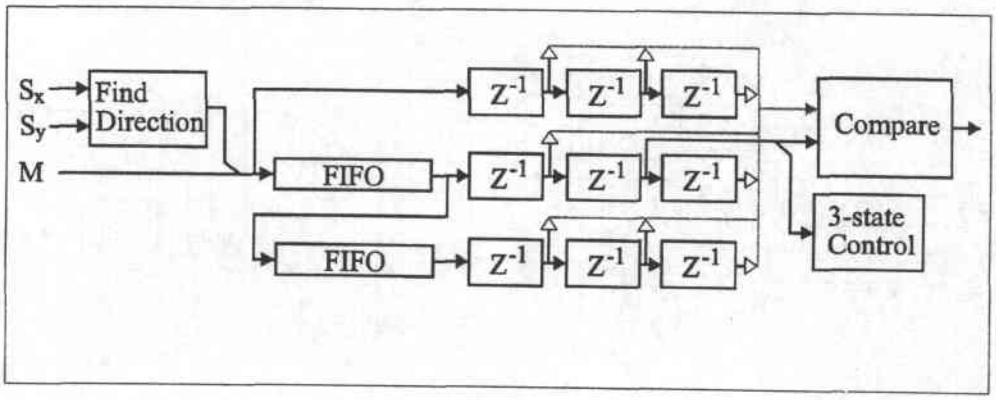

Figure 6: Nonmaximum suppression circuit

The double threshold filter uses recursive procedure which is not useful for pipeline circuit architecture. We use instead a pixel labeling and resolving procedure with two passes over each image. In the first pass, pixels are labeled and equivalences between neighbor labels are stored in an equivalence table. In the second pass, we use equivalence information in order to classify pixels as edge or background pixels. The circuit is explained in more details in [9].

Table 1 summarizes the CLB utilization and maximal clock frequency for each circuit partition.

Circuit Paritition	CLB Utilization	Max.Clock Frequency (MHz)
Convolution	38	39
Gradient	209	9.0
Nonmaximum suppression	40	11
Threshold	186	20

Table 1: Results of synthesis and technology mapping

4 Nonlinear Laplace Edge Detector

A nonlinear Laplace operator is efficient edge detector in noisy images [10]. Figure 7 presents the basic edge detection steps based on nonlinear Laplace algorithm.

Edge detection begins with a smoothing filter followed by discrete approximation of first derivative (edge strength detector) and second derivative (nonlinear Laplace operator). The edge positions are at zero crossings of the second derivative.

The edges obtained by zero crossing detector are multiplied by the gradient values from edge strength detector and a threshold is applied in order to produce final edges.

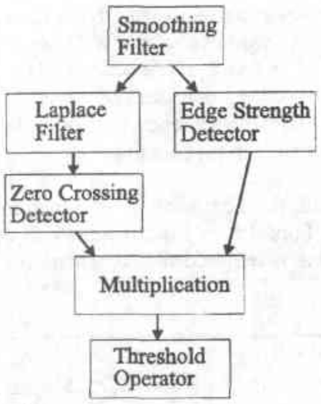

Figure 7: Nonlinear Laplace Algorithm

4.1 Nonlinear Laplace Circuit Architecture

The Gaussian filter can be used as a smoothing filter with the same circuit architecture as presented in the Canny edge detector section (Figure 4).

We used a simple form of nonlinear Laplace filter in a 3x3 neighborhood which is given by the following equations:

$$NL(x, y) = gradmax(x, y) + gradmin(x, y)$$

$$gradmax(x, y) = maximum(3x3) - I(x, y)$$
$$gradmin(x, y) = minimum(3x3) - I(x, y)$$

Local maximum and minimum values in the 3x3 neighborhood are subtracted from the central intensity value in order to obtain *gradmax* and *gradmin*. Both values are also used for edge strength detector given by:

$$E(x, y) = min\{gradmax(x, y), -gradmin(x, y)\}$$

The circuit architecture of nonlinear Laplace filter and edge strength detector is presented in Figure 8.

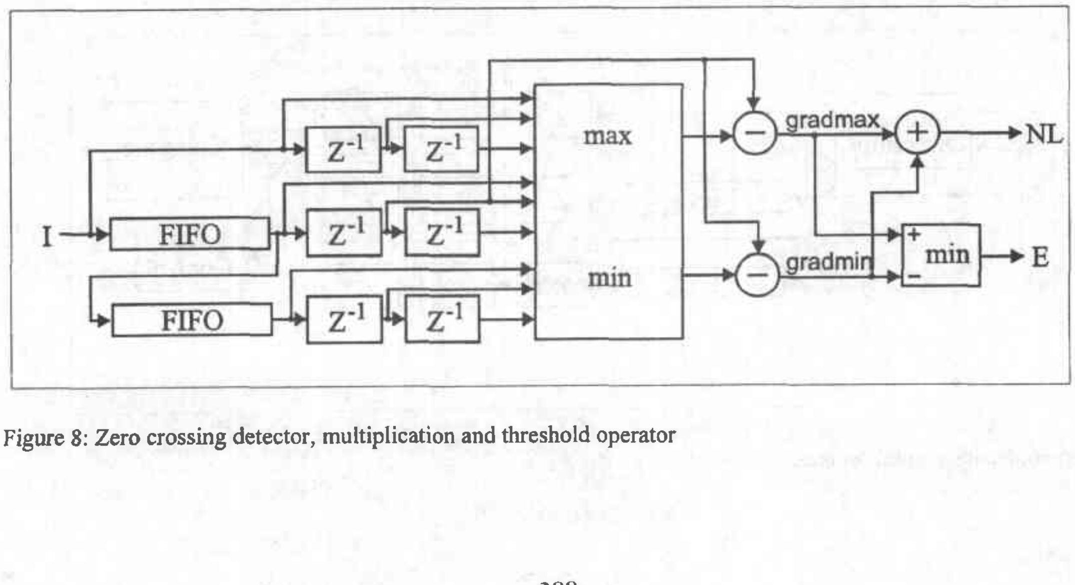

Figure 8: Zero crossing detector, multiplication and threshold operator

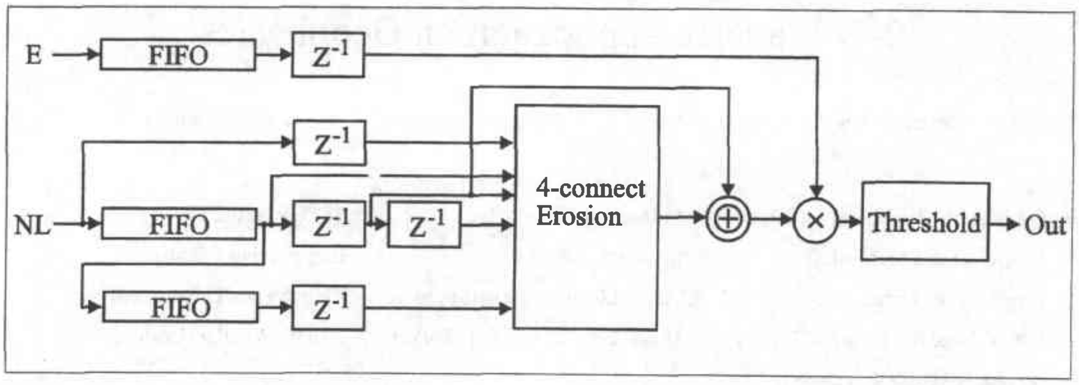

Figure 9: Second partition of nonlinear Laplace circuit

The process for obtaining local maximum or minimum intensity values requires 8 comparators and multiplexers and presents the critical part of the circuit. The throughput of the circuit can be increased with additional registers in this part. The circuit without registers occupies 167 CLBs and has a maximum frequency 3.2 MHz. If we add a register in the middle and at the end of this part of the circuit, we increase the frequency to 5.9 MHz and the circuit area to 174 CLBs. Two additional registers increase the frequency to 12.7 MHz and the area to 186 CLBs.

The input of zero crossing detector is a binary image of positive and negative regions. The input pixels are actually the sign bits of the nonlinear Laplace filter output. The zero crossing detector finds the 8-connected edge contour by an XOR operation of the input image with the 4-connect eroded image. The result is multiplied by the edge strength values and a threshold is finally applied, as presented in Figure 9. Note that the edge strength values have to be delayed in the FIFO buffers of the same length as the binary image in order to multiply the pixels from the same position. The implemented circuit occupies only 11 CLBs and achieves maximum frequency 35.6 MHz.

5 Conclusions

We presented design of two real-time edge detection circuits. The Canny edge detection algorithm and the nonlinear Laplace edge detector are typical algorithms used in low level computer vision systems. We have shown that the presented algorithms with slight modifications are suitable for implementation with digital circuits in the pipeline architecture. The circuits were carefully optimized during transformation from behavioral VHDL to synthesizeable structural VHDL in order to achieve real-time operation. The circuits were implemented in the modular multi-FPGA reconfigurable system. The hardware verification of each module was performed with the PC in order to test the operation of each circuit partition.

References

[1] W. K. Pratt, "Digital Image Processing", Wiley-Interscience Publ., 1978

[2] E. R. Davies, "Machine Vision: Theory, Algorithms, Practicalities", Academic Press, London, 1990

[3] C. Torras, "Computer Vision: theory and industrial applications", Berlin, Springer-Verlag, 1992

[4] S. D. Brown, R. J. Francis, J. Rose, "Field-Programmable Gate Arrays", Kluwer Academic Publishers, Boston, 1992

[5] P. M. Athanas, A. L. Abbott, "Real-Time Image Processing on a Custom Computing Platform", IEEE Computer, 1995, 14-24

[6] F.G. Lorca, L. Kessal, D. Demigny, "Efficient ASIC and FPGA Implementations of IIR Filters for Real Time Edge Detection", Proceedings ICIP 97, 1997

[7] J. Canny, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, št. 6, str. 679-698, november 1986

[8] W. Luk, T. WU, I. Page, "Hardware-Software Codesign of Multidimensional Programs", IEEE Workshop on FPGAs for Custom Computing Machines, str. 82-90, Los Alamitos, CA, april 1994

[9] A. Trost, "Prototyping System for Digital Image Processing", masters thesis (in Slovene), Ljubljana, 1998

[10] L. J. Van Vliet, I. T. Young and G. L. Beckers "A Nonlinear Laplace Operator as Edge Detector in Noisy Images", in Computer Vision, Graphics, and Image Processing 45, pp. 167-195, 1989