2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology

1

May 6th - 9th, 2009

Ambassador City Jomtien

ISBN 978-1-4244-3388-9
IEEE Catalog Number: CFP0906E
Library of Congress: 2008910219

ECTI

50 ANNIVERSARY KMUTNB

ELECTRICAL

NECTEC a member of NSTDA

IEEE THAILAND SECTION

Hardware/Software Co-design for Line Detection Algorithm on FPGA

W. Kayankit W. Suntiamorntut
Department of Computer Engineering, Faculty of Engineering
Prince of Songkla University, Hatyai Songkhla 90112 Thailand
{warut.k@hotmail.com, wannarat@coe.psu.ac.th}

Abstract- **This paper presents the design methodology of hardware/software co-design on FPGA. In this research work uses line detection algorithm to be our case study. C-based programming named ImpulseC has been employed. By using this, we could reduce the time of the design. Two main components are involved. Edge detection module is the first component. Various algorithms, Robert, Sobel and Prewitt, are compared. The best algorithm in term of performance and area is selected. The second component is hough transform. This module is divided into hardware and software due to the limit of the available logic block. At the end, the system both in hardware/software for line detection on FPGA is simulated.**

I. INTRODUCTION

FPGA-based reconfigurable computers have become viable computing device. It could compute as super-computer by performing both in directly on FPGA hardware and on processor[1][2][3]. In [4], they present the operating system designed specially for reconfigurable computers. The concept of hardware process designed on FPGA hardware is introduced and replaced on software program. The OS kernel employs to maintain the hardware/software interface.

The line detection was mainly performed on software due to its large hardware requirement. By using hardware/software co-design, line detection system could be implemented on FPGA.

In this paper a FPGA implementation of the line detection has proposed. The choice of Robert edge detection is selected among Robert, Sobel and Prewitt, due to its low area and high speed. While a Hough transform will be divided into software and hardware process. Software platform uses ISE Xilinx 8.2i with simulation on ModelsimSE6.2.

II. EDGE DETECTION[5] AND HOUGH TRANSFORM

The line detection contains two major image processing routine. One is edge detection. Another is hough transform.

A. Robert edge detection

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. The operator consists of a pair of 2×2 convolution kernels as shown in Figure 1. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator. The input to the operator commonly uses a grayscale image.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Figure 1 2×2 convolution kernels in Robert

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan(Gy / Gx) - 3\pi / 4$$

B. Sobel edge detection

The operator consists of a pair of 3×3 convolution kernels as shown in Figure 2. One kernel is simply the other rotated by 90°. Sobel filter is a simple approximation to the concept of gradient with smoothing.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2 3×3 convolution kernels in Sobel

978-1-4244-3388-9/09/$25.00 ©2009 IEEE

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

Typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute.

The angle of orientation of the edge (relative to the pixel grid)

giving rise to the spatial gradient is given by:

$$\theta = \arctan(Gy / Gx)$$

C. *Prewitt edge detection*

Prewitt filter is very similar to Sobel filter. The difference with respect to Sobel filter is the spectral response. It is only suitable for well-contrasted noiseless image. Prewitt operator is used for detecting vertical and horizontal edges in images as shown in Figure 3.

+1	+1	+1
0	0	0
-1	-1	-1

Gx

-1	0	+1
-1	0	+1
-1	0	+1

Gy

Figure 3 Prewitt mask

D. *Hough transform*

The Hough Transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough Transform is commonly used for the detection of regular curves such as lines, circles, ellipses, etc.

The Hough Transform is an algorithm that can provide a significant solution to the problems associated with line detection and definition. It is defined by the parametric representation used to describe lines in the picture plane. It was introduced by Paul Hough in 1962 and patented by IBM. The transform parameterizes a description of a feature at any given location in the original image space. Hough used slope-intercept parameters exclusively.

Hough transform is commonly used to establish whether pixels line on a curve of a specific shape. A point with coordinates (xi,yi) can have an infinite number of straight line passing through it. These line can be described in polar coordinates in the (ρ,θ) space by the equation:

$$\rho = xi \cdot \sin\theta + yi \cdot \cos\theta$$

III. SYSTEM DEVELOPMENT BY IMPULSEC

E. *ImpuseC*

With the advantage of ImpulseC based FPGA design tools, the complete hardware/software design is easily compiled, simulated or debugged with a single tool set named ImpulseC CoDeveloper as shown in figure4.

The ImpulseC tools include a software-to-hardware compiler that converts individual ImpulseC processes to functionally equivalent hardware descriptions and that generates the necessary process-to-process interface logic.

C language applications, image processing algorithms are compiled and optimized to FPGA hardware represented by hardware description language (HDL). Compiler also generates hardware/software interfacing and the link between host processor and FPGA hardware.

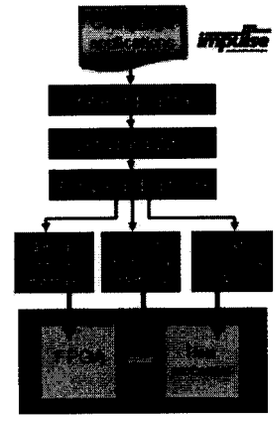


Figure 4 ImpulseC CoDeveloper

The standard C language does not contain parallel processing and the programming of parallel systems. The parallelism can be exploited at two distinct levels: at the application system level and the level of statements within a specific subroutine or loop.

ImpulseC programming model are processes and streams. Processes are independently synchronized. This makes it possible to create high-performance, mixed hardware/software applications for FPGA-based platforms without the need to write low-level VHDL or Verilog.

F. *Hardware/Software co-design process*

There are many different methods to design and implement hardware and software co-design. This research chooses ImpulseC design process.

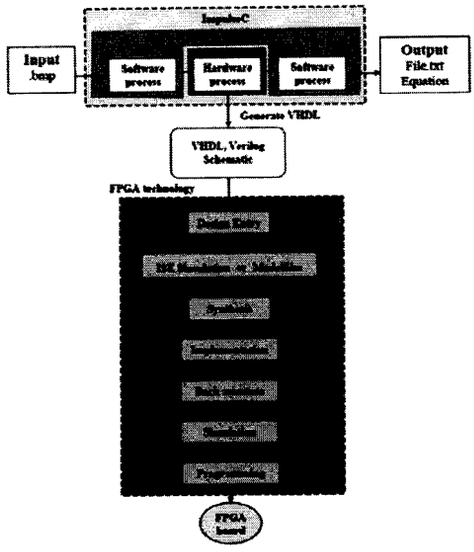


Figure 5 Hardware/Software co-design process

As shown in Figure5, we could separate the design process to be two sections: the system developed by ImpulseC and hardware implementation on FPGA.

IV. Experimental Results

The edge detection and hough transform algorithm were modeled in ImpulseC on Spartan3E16000 based platform. The bmp file format is brought to the system as the input file. Then later, the image file is transformed into the gray scale image.

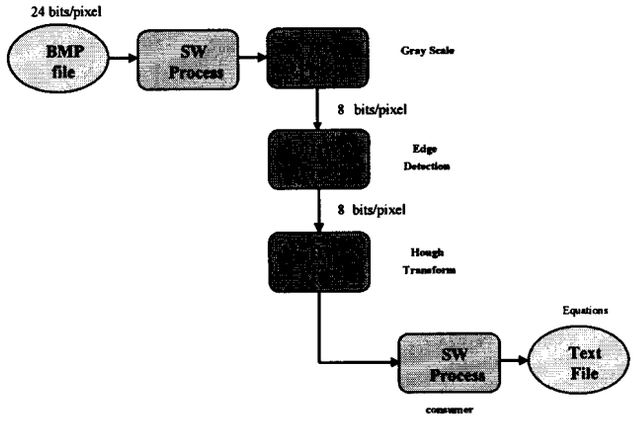


Figure 6 Line detection design process using ImpulseC

When the gray scale image is completed, edge detection is activated. After the edge detection process is done in one row, the result will forwarded directly to hough transform process. This technique will increase the throughput of line detection system. However, we cannot implement sine or cosine function onto the hardware because it consumes a large area of memory. Therefore, this function will be implemented on software process instead. The final output is generated as a text file showing the (Xmin, Ymin) and (Xmax, Ymax) results.

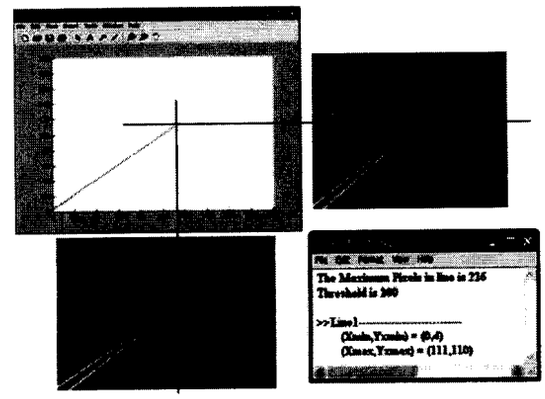


Figure 7 Line detection result compared to Matlab7.0

The FPGA resource usage from simulation is presented in Table1. We found that the line detection algorithm could be solved within a second.

TABLE I FPGA resources for line detection

FPGA board / Resources	Spartan3E (XC3S1600E)
Number of External IOBs	16.0%
Number of BUFGMUXs	4.2%
Number of MULT18X18SIOs	77.7%
Number of RAMB16s	8.3%
Number of Slices	64.0%
Number of SLICEMs	52.2%
Clock net clk_BUFGP	34.507 ns

V. Conclusion and Future Work

The hardware/software co-design is useful and reasonable to improve the performance of intensive computing such as image processing. The line detection is the case study of using C-language based programming to implement the hardware without the logic design knowledge.

References

[1] http://www.cray.com/products/xd1/.
[2] C. Chang, J. Wawrzynek and R Brodersen. BEE2: A high-end reconfigurable computing system. *IEEE Des. Test. Comput.*, pp. 114-125, March, 2005.
[3] T. Hamada et al. Prograpе-1: A Programmable special-purpose computer for many-body simulations. In 6th *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'98)*, 15-17 April 1998, Napa Valley, CA, USA, pp. 256-257, 1998.
[4] Hayden Kwok-Hay So. *BORPH: An Operating System for FPGA-Based Reconfigurable Computers*, PhD thesis, University of California, Berkeley, 2008..
[5] http://www.uweb.ucsb.edu/~shahnam/.