An Efficient Design for Canny Edge Detection Algorithm using Xilinx System Generator

Avinash G. Mahalle
Department of Electronics Engineering
Government College of Engineering, Amravati,
Maharashtra, India
agmahalle10@gmail.com

A. M. Shah (*Assistant Professor*)
Department of Electronics Engineering
Government College of Engineering, Amravati,
Maharashtra, India
amshah.gcoea@gmail.com

***Abstract*— Digital Image consists of some informative pixels and some redundant pixels. Edge Detection is the process of preserving informative pixels and eliminating redundant data which improves bandwidth and storage efficiency. A sudden change in pixel intensity level is defined as an Edge. Theoretically, from the literature it is seen that Canny edge detection is the most accurate algorithm. It is also insensitive to Noise. The only disadvantage of Canny algorithm is its high computational complexity which limits its maximum frequency of application with high latency and low throughput. In this paper, tradeoff between Accuracy and Complexity is studied. An efficient Canny algorithm is designed using Xilinx System Generator which utilizes JTAG Hardware co-simulation approach. An accuracy is compromised in order to make it more efficient in terms of resource utilization than the conventional one. Spartan-3E FPGA is used as a reconfigurable hardware platform for the designed algorithm.**

Keywords*— *Image Processing; Edge Detection; Canny Algorithm; Xilinx System Generator; Spartan-3E FPGA

I. INTRODUCTION

Pixels are the fundamental unit of any Digital Image. Every pixel has some intensity value which varies in between 0-255. Edges are nothing but pixels which carry structural information of an Image. Edge Detection thus an important and essential step in any image processing application [1]. In recent years, Edge detection acquired a significant attention due to its role in the field of Computer or Machine Vision. Further, it has many applications such as Brain Tumor Detection, Satellite Imaging, Weather forecasting etc. Many edge detection techniques are used for this purpose which are mainly classified as Gradient based and Laplacian based [2]. Canny algorithm is also known as Optimal Edge Detection technique. It uses classical operator for gradient calculation. Canny algorithm has several steps which make it a complex algorithm than others. But, anyway Canny is the most accurate edge detection technique. It is also immune to Noise. Gradient based edge detection operators are Sobel, Prewitt and Robert. These are simple to design but sensitive to noise. Laplacian based edge detection techniques finds out derivative twice. A derivative mask acts like High Pass Filter. Thus, high frequency noise components can corrupt the whole image. A conventional Canny [3] has four steps. These are Image smoothing, Gradient Magnitude & Orientation Calculation, Nom-Maximum Suppression (NMS) and Hysteresis Thresholding. Xilinx System Generator allows common environment for MATLAB/Simulink and ISE Design Suit. XSG provides an efficient way of designing complex algorithms [4]. It automatically generates required Hardware Description Language (HDL) along with Test bench. Field Programmable Gate Array (FPGA) is used for prototyping purpose. Spartan-3E Starter Kit is programmed by downloading generated bit stream file. System Generator Token provides various types of Compilation.

II. CONVENTIONAL CANNY ALGORITHM

The Canny algorithm is also called as the optimal edge detector. It is proposed by John Canny in 1986. At that time, classical operators such as Robert, Prewitt and Sobel were known. His interest was to improve performance of these operators. All the mathematical formulation and analysis required to support his theory is given in [5]. Canny proposed few criteria for improving classical methods of edge detection. First one is Low Error Rate. It emphasizes on minimizing Errors. Second one is Localization of Edges. There should not be any difference between located and actual edges. Third and final criterion is Only One Response to Single Edge. It makes Canny the most accurate edge detection technique.

By considering all three criteria mentioned above, Canny came up with a four block model (as shown in Fig. 1). The first block is used to eliminate high frequency noise. Generally, Gaussian filter is used here as a Low Pass Filter. Due to this, Canny often known as Gaussian of Gradient. In second block, any classical operator is used for gradient magnitude and orientation calculation. As Sobel is more efficient than Robert and Prewitt, generally it is used for this purpose. Third block is called as Non-Maximum Suppression. In this block thick edges which are present in the gradient output due to smoothing step are made thin. Hysteresis Thresholding is the final block of Canny model. It performs Edge Linking by applying two thresholds.

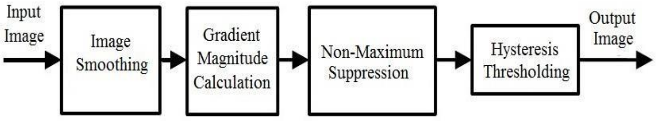


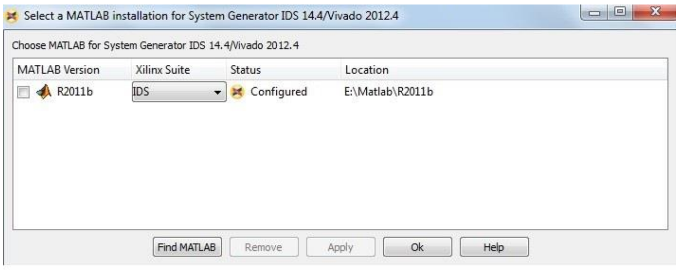
Fig. 1 Various Blocks of Canny Model

978-1-5386-2599-6/18/$31.00 ©2018 IEEE

III. System Development and Proposed Design

A. *System Development*

System used to design proposed Canny Edge Detection model should have licensed versions of MATLAB and Integrated Software Environment (ISE) Design Suit. Fig. 2 indicates generated configuring window when ISE Design Suit 14.4 is installed. Operating System used here is Windows7. System Generator provides a library of various Xilinx Blocks and Simulink Blocks. These Xilinx blocks use Fix Point Data Type which is required for hardware implementation whereas, Simulink blocks use Floating Point Data Type by default. Thus, Gateway Blocks are used for the connection between Simulink blocks and Xilinx blocks. System Generator can generate HDL code automatically from the created model. The automatically generated code in VHDL/Verilog can be synthesized along with its test bench and implemented on FPGA using bit stream file. System Generator Token allows user to perform various types of compilation. Out of these, Hardware Co-Simulation and Timing and Power Analysis compilations are used for this study.



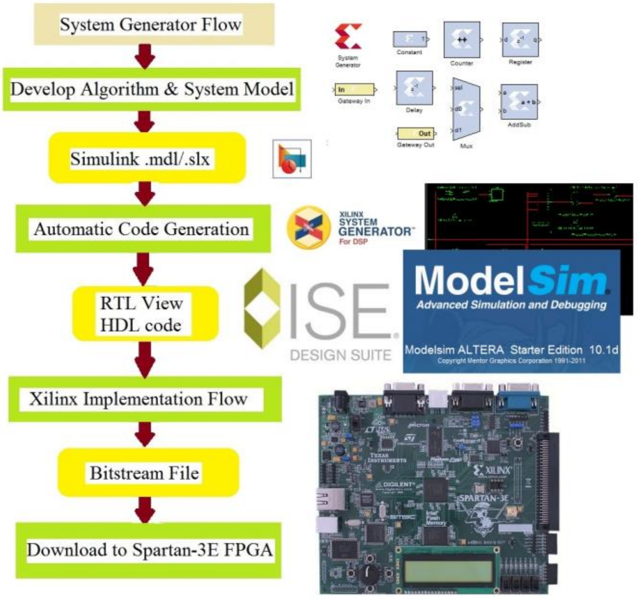
Select a MATLAB installation for System Generator IDS 14.4/Vivado 2012.4

Choose MATLAB for System Generator IDS 14.4/Vivado 2012.4

MATLAB Version	Xilinx Suite	Status	Location
□ R2011b	IDS	Configured	E:\Matlab\R2011b

Fig. 2 Configuring ISE Design Suit 14.4 with MATLAB R2011b

The Xilinx System Generator [6] is a plug-in to Simulink that enables designer to develop high-performance DSP systems for Xilinx FPGAs. Its flow is given as shown in Fig. 3.



System Generator Flow
Develop Algorithm & System Model
Simulink .mdl/.slx
Automatic Code Generation
RTL View
HDL code
Xilinx Implementation Flow
Bitstream File
Download to Spartan-3E FPGA
System Generator
Constant
Counter
Register
Gateway In
Delay
Mux
AddSub
Gateway Out
XILINX SYSTEM GENERATOR For DSP
ISE DESIGN SUITE
ModelSim Advanced Simulation and Debugging
Modelsim ALTERA Starter Edition 10.1d

Fig. 3 System Generator Flow

B. *Proposed Canny Algorithm Design*

Proposed design of Canny algorithm utilizes tradeoff between Accuracy and Complexity. Accuracy has been compromised in order to make conventional Canny less complex. Proposed design skips gradient direction calculation part and deals with magnitude calculation only.

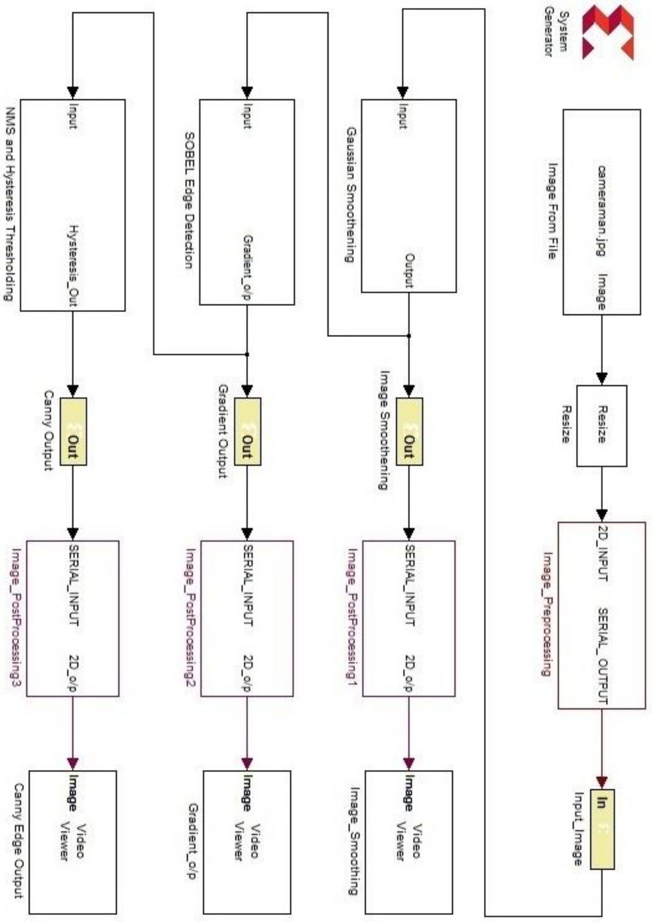


Fig. 4 Proposed design for Canny Algorithm Model

Fig. 4 shows design for proposed Canny algorithm. It consists of various blocks which are designed as follows:

Step 1. Image Smoothing:

Proposed design uses 5×5 Gaussian filter (shown in Fig. 5) for smoothing the image. This filter is already present in the Xilinx Reference Block set of Simulink library. It has total nine applications out of which we used Gaussian filtering. It acts as a low pass filter and thus removes Noise (i.e., high frequency components).

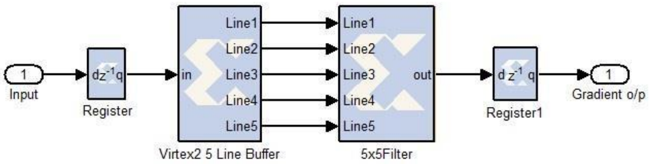


Fig. 5 Design for 5×5 Gaussian Filter

Step 2. Gradient Magnitude Calculation: Sobel operator is the most efficient classical operator and hence, generally used to find out gradient magnitude. Sobel uses 3×3 horizontal and vertical masks. Fig. 6(a) and 6(b) show horizontal and vertical gradient filters for Sobel operator respectively [7].

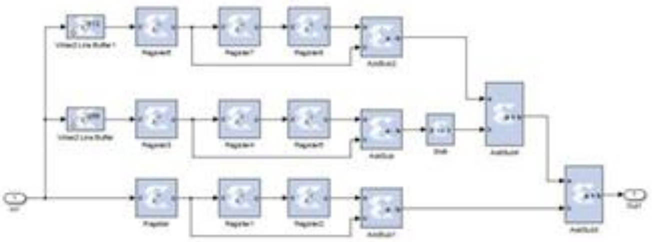


Fig. 6(a) Design for Sobel Vertical Gradient Filter

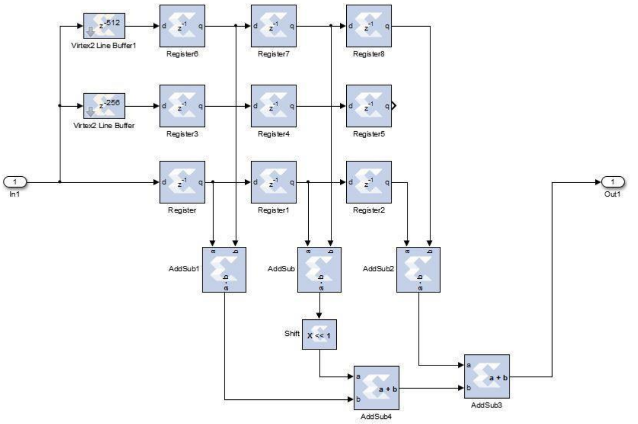


Fig. 6(b) Design for Sobel Vertical Gradient Filter

Step 3. Non-Maximum Suppression (NMS): Gradient output contains thick edges due to smoothing image. This block (designed as shown in Fig. 7) provides edge thinning. It compares all neighbouring pixel values to the central pixel and suppress it to zero value if it is not maximum. Otherwise, it is given to next block. In conventional Canny comparison is done with the pixels in the direction of gradient. This change in the design degrades accuracy of proposed Canny.

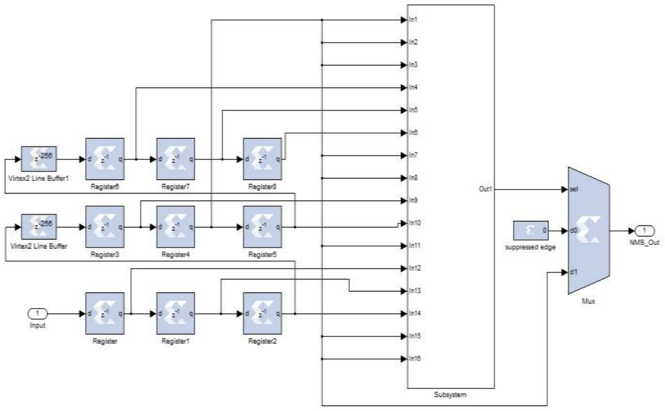


Fig. 7 Design for Non-Maximum Suppression Block

Step 4. Hysteresis Thresholding: It is the final block in canny algorithm design which uses two thresholds i.e., high and low. Also, it eliminates streaking effect caused due to single threshold when it is subjected to noisy image. If gradient value exceeds High Threshold then it is considered as Strong Edge (255) and when it is less than Low Threshold then it is considered as Weak Edge (suppressed to 0). Fig. 8 gives design of this block.

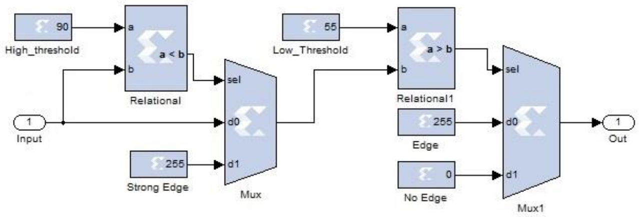
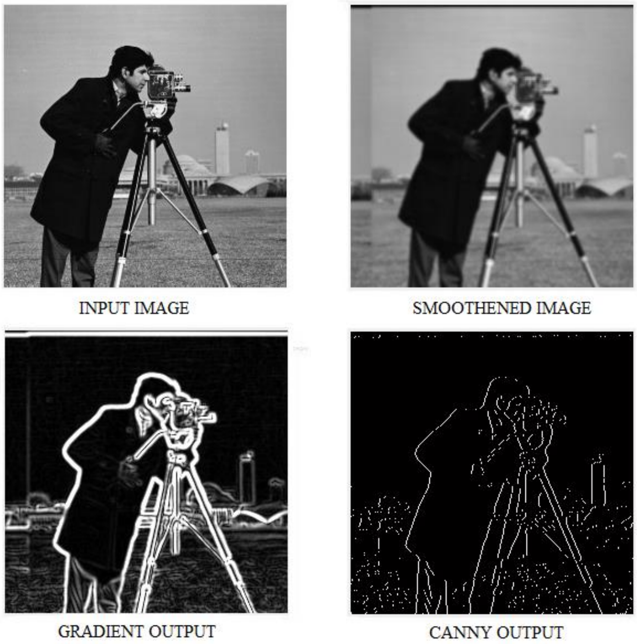


Fig. 8 Design for Hysteresis Thresholding Block

IV. Results and Performance Analysis

The proposed design is compiled using hardware co-simulation compilation given in system generator token. Output results are observed in MATLAB Environment using Hardware co-simulation compilation. Generated bit stream file is dumped to Spartan-3E Starter Kit [8] through JTAG cable.



INPUT IMAGE

SMOOTHENED IMAGE

GRADIENT OUTPUT

CANNY OUTPUT

Fig. 9 Output results for each step of proposed Canny

Fig. 9 shows outputs for various steps of Canny Algorithm. It can be observed that gradient output contains thick edges due to image smoothing. These edges are made thin using NMS block. Further it is given to Hysteresis thresholding which converts image into binary form as a final output edge map. Device utilization summary for the proposed Canny algorithm is shown in the table below:

TABLE I. DEVICE UTILIZATION SUMMARY FOR 256×256 IMAGE

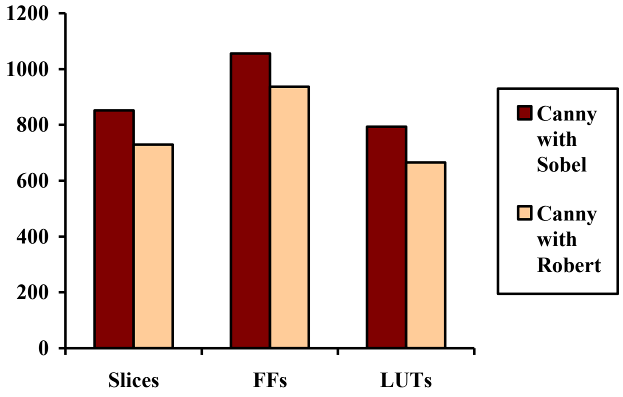
Device Utilization Summary			
Logic Utilization	***Used***	***Available***	***Utilization***
Number of Slice Flipflops	1039	9312	11%
Number of 4 input LUTs	704	9312	7%
Number of occupied slices	996	4656	21%
Slices containing related logic	996	996	100%
Slices containing unrelated logic	0	996	0%
Total Number of 4 input LUTs	799	9312	8%
Number used as logic	559		
Number used as a route-thru	95		
Number used as 16×1 RAMs	65		
Number used as Shift Registers	80		
Number of bonded IOBs	43	232	18%
Number of RAMB 16s	14	20	70%
Number of BUFGMUXs	1	24	4%
Number of MULT 18X 18SIOs	5	20	25%
Average Fanout of Non-clock nets	2.33		

As concluded in the earlier work [9], Robert operator is simpler in design than Sobel. Thus, in order to minimize resource utilization further, Sobel is replaced by Robert operator. But, it should be remembered that as the size of mask decreases it becomes more sensitive to noise. From Table II it is observed that proposed Canny with Robert not only reduces resources utilized but also it increases Frequency of operation than that of proposed Canny with Sobel.

TABLE II. CANNY ALGORITHM ANALYSIS FOR 512×512 IMAGE

	Canny with Sobel	*Canny with Robert*
No. of occupied slices	852	729
No. of Flip-flops	1056	937
Total number of LUTs	793	665
Total number of IOBs	38	36
Minimum Period (ns)	14.92	10.35
Maximum Frequency (MHz)	67.03	96.59

A better graphical view for comparison between proposed Canny with Sobel and with Robert on the basis of occupied Slices, Flip flops and Lookup Tables is shown below:



V. CONCLUSIONS

Proposed Canny algorithm is more efficient in terms of resource utilization than the conventional one. It is implemented on Spartan-3E FPGA by utilizing JTAG Hardware Co-Simulation compilation using XSG. An efficient Canny is designed by compromising the accuracy. It makes proposed design simpler and also enhances its limitation in terms of maximum frequency of operation. In future, proposed design can be utilized for real time applications.

REFERENCES

[1] R. C. Gonzalez, R. E. Woods, Digital Image Processing, 3rd edition, Prentice Hall, pp.187-190, 2007

[2] R. Maini, H. Aggarwal. "Study and comparison of various image edge detection techniques," *International Journal of Image Processing*, Vol.3, pp.1-12, Feb 2009

[3] K. K. Ahmed, A. Fuad, S. M. Rizvi, "Hardware software co-simulation of Canny edge detection algorithm," *International Journal of Computer Applications*, 122(19), pp.7-12, July 2015

[4] M. Ownby, W. H. Mahmoud, "A design methodology for implementing DSP with Xilinx System Generator for Matlab," *IEEE International Symposium on System Theory*, pp.404-408, March 2002

[5] J. Canny, "A computational approach to edge detection," *IEEE transaction on pattern analysis and machine intelligence*, vol. 8, no.6, pp. 679- 698, Nov 1986

[6] Xilinx System Generator Users Guide, www.xilinx.com

[7] Avinash G. Mahalle, A. M. Shah, "FPGA Implementation of Gradient Based Edge Detection Algorithms," *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, Volume 5, Issue 5, May 2017

[8] Spartan-3E Startar Kit Users Guide, www.xilinx.com

[9] Avinash G. Mahalle, A. M. Shah, "Hardware Co-Simulation of Classical Edge Detection Algorithms Using Xilinx System Generator," *International Research Journal of Engineering and Technology (IRJET)*, Volume 5, Issue 1, January 2018