International Journal of Security and Its Applications
Vol. 10, No. 7 (2016) pp.345-354
http://dx.doi.org/10.14257/ijsia.2016.10.7.30

Research and Implementation of Sobel Edge Detection Using Model-Based Design

Erhu Xie

Department of Computer, Jining Normal University, Inner Mongolia, China
E-mail: 592257359@qq.com

Abstract

In order to satisfy the hardware implementation of the complex image algorithm, this paper gives a model-based design (MBD) workflow for an image processing algorithm applicable for edge detection of an image by Xilinx Spartan 6 FPGA. MBD provides a design flow that directly maps the design into FPGA using functional models of the design specification. Simulation results show that the proposed workflow is not only overcoming the defects of traditional development flow, but also to shorten the time-to-market, raise the quality of final the product.

Keywords*: model-based design; Sobel Edge Detection; FPGA; Image Processing*

1. Introduction

Image processing algorithms can be implemented on the reconfigurable hardware, which can quickly verify the correctness and applicability of the complex algorithms. Therefore, FPGA is an ideal choice for implementation of real time image processing. However, with the continuous increasing scale and complexity of image processing system, the number of code in image algorithm by FPGA is showing a trend of explosive increasing [1]. The traditional method of FPGA design for image processing includes successive stages of creating a specification of the future device, coding it on one of the hardware design languages (HDL) and extensive code tests for compliance with the specification. Using this method, it is hard to detect mistakes during the creation of the specification, and fixing them ay require partial or full repetition of following steps of design. If HDL code is written for image algorithm using traditional development flow in Xilinx FPGA, then it is too time consuming and bulky.

Over the last few years, a new development flow named Model Based Design (MBD) was put forward to solve the problem mentioned above [2]. In MBD workflow, a system model is at the center of the development process, from requirements development, through design, implementation, and testing. The code can be automatically generated for system and create test benches for system verification, which is saving time and avoiding the introduction of manually coded errors. Therefore, This workflow is elaborated to create hardware and software partitioning, automatically create hardware and software implementation code, and verify the hardware and software implementations in the context of the complete system. It is not only overcoming the defects of low efficiency and difficulty of meeting the requirements in traditional methods of development, but also to meet time-to-market and cost objectives.

The paper is organized as follows: In Section 2, we introduce key concepts regarding MBD and MBD workflow. Section 3 discusses Sobel edge detection algorithm is preferred which is most trustworthy and gives an efficient output in the field of image and video processing for the extraction of object edges. The main contribution of this work is presented in Section 4, where Sobel edge detection algorithm is designed and verified using MBD. Finally, Section 5 summarizes the main conclusions of this work.

2. Introduction of Model-Based Design

MBD technique is based on creating a model of a future system. Model-Based Design improves design quality and accelerates design and verification tasks by employing an executable specification [3][4]. This executable specification is elaborated to create hardware and software partitioning, automatically create hardware and software implementation code, and verify the hardware and software implementations in the context of the complete system. Significant advantages of Model-Based Design include the fact that it facilitates rapid design iterations and it moves the verification process all the way to the beginning of the design cycle. This helps detect system specification related errors, design errors, and implementation errors early.

In the MBD, a system model is at the center of the development process, from requirements development, through design, implementation, and testing. Model-Based Design integrates the world of system design engineers, FPGA designers, and verification engineers to increase productivity and produce correct by construction designs that match the executable system specifications [5]. MBD workflow is shown in Figure 1. The vertical arrows represent the different aspects of the design. The right side of the curved arrow indicates the validation and verification process. With use of MBD related tools, it can ensure correct evolution in the design stage by continuous testing and verification in the evolution process.

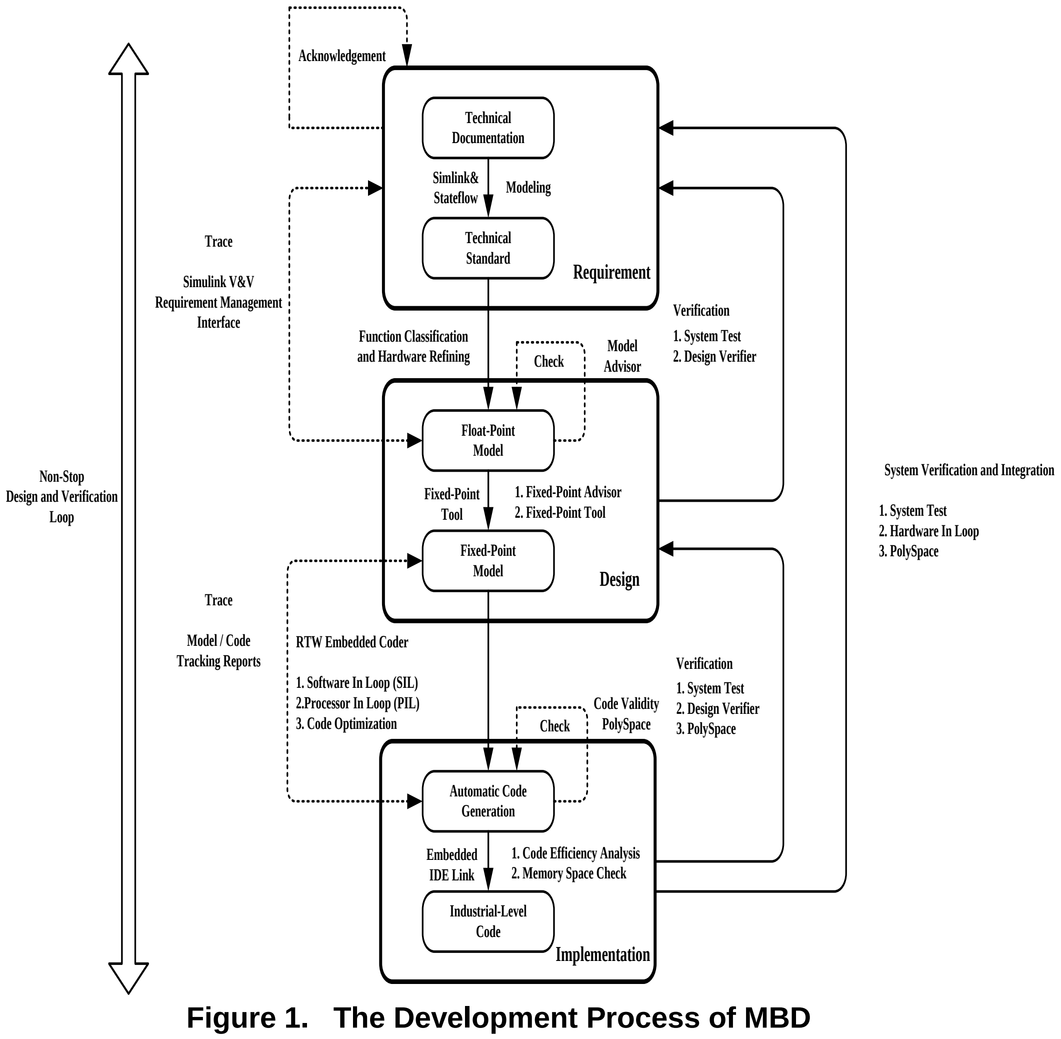


Figure 1. The Development Process of MBD

As illustrated in Figure 2, these high level requirements then drive the decisions of choosing hardware and software partitioning, hardware and software design and implementation by FPGA. MATLAB and Simulink are frequently used as the environments for capturing system level algorithms and design specifications. HDL tool provides a workflow advisor that automates the programming of Xilinx and Altera FPGA. It can control HDL architecture and implementation, highlight critical paths, and generate hardware resource utilization estimates. HDL tool generates VHDL and Verilog test benches for rapid verification of generated HDL code, automatically generates two types of co-simulation models. HDL co-simulation model is applied for performing HDL co-simulation with Simulink and an HDL simulator, such as Cadence Incisive or Mentor Graphics ModelSim. FPGA-in-the-loop (FIL) co-simulation model is applied for verifying design with Simulink and an FPGA board.

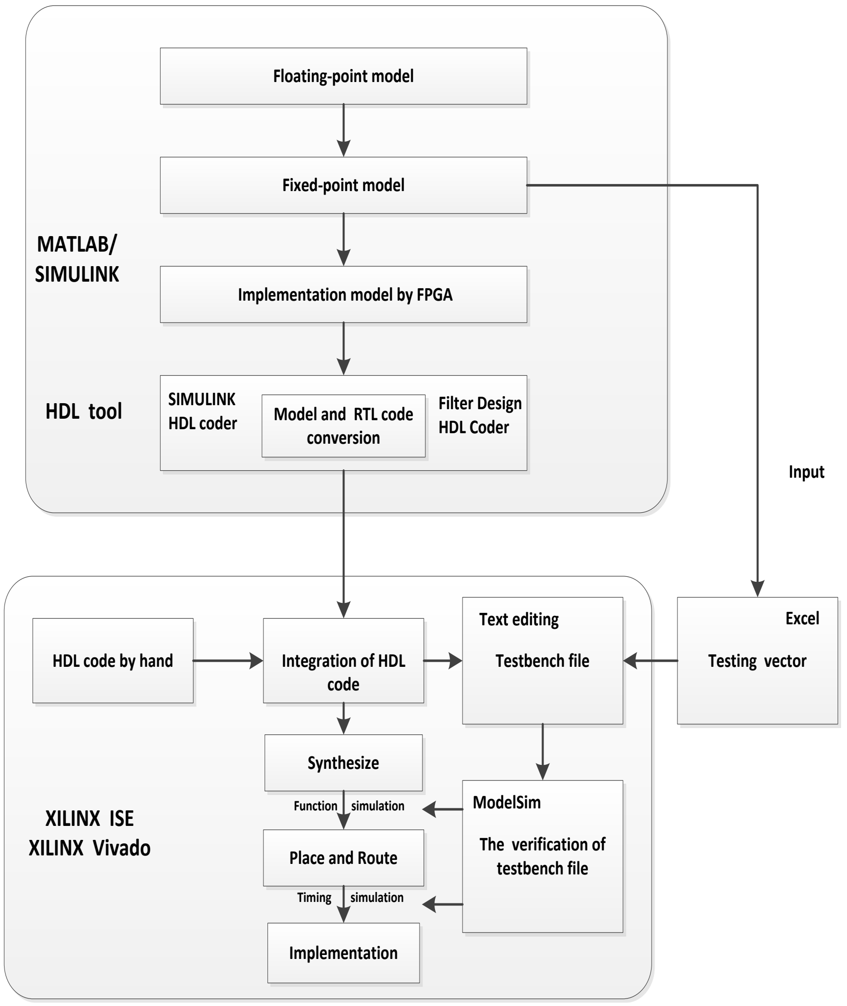


Figure 2. Implementation Diagram of MBD Workflow Using FPGA

3. Sobel Edge Detection Algorithm

In image processing, edge detection plays a very important role and a wide range of applications, such as image segmentation, scene analysis, focus region selection, object recognition. If we can find the boundary of an object by locating all its edges, then we have effectively segmented it. Superficially, edge detection seems a relatively straightforward affair. After all, edges are simply regions of intensity transition between one object and another. However, despite its conceptual simplicity, edge detection remains an active field of research.

The basic edge detection operator is a matrix gradient operator, which determines the level of variance between different pixels. The calculation method of the edge detection operator is to form a center of a matrix region of a pixel as the center of the pixel. If the value of this matrix region is above a given threshold, then the middle of the pixel is listed as the edge. Examples of edge detection gradient are Prewitt operator and Sobel operator.

Here Sobel operator based edge detection technique is used and is extended for real-time applications [6]. Due to the property of counteracting the noise sensitivity Sobel operator for edge detection over other gradient operators are chosen. The Sobel operator commonly known as Sobel filter is used for image processing and computer vision, which creates an image which focuses edges and transitions. It is discrete differentiation operator that calculates the gradient approximation of the image intensity function. The result of the Sobel operator is the corresponding gradient vector at that particular point. The Sobel operator convolves the image with an integer valued filter in vertical and horizontal direction so it is thus relatively cheaper in terms of computations. The gradient estimation that it produces is relatively simple for high frequency variations in the image.

The Sobel operator performs a 2D spatial gradient measurement on an image and emphasizes regions of high spatial frequency that correspond to edges. Typically, it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image [7]. The operator consists of a pair of 3×3 convolution kernels, which are shown in Figure 3.

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

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

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

Figure 3. Sobel Convolution Kernels and Image Segment

One kernel is simply the other rotated by 90°. Assuming an image segment of the same dimensions as that of the Sobel kernel, the result of convolution will be

$$|G| = |(f_1 + 2f_2 + f_3) - (f_7 + 2f_8 + f_9)| + |(f_1 + 2f_4 + f_7) - (f_3 + 2f_6 + f_9)| \tag{1}$$

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image to produce separate measurements of the gradient component in each orientation (call these G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{2}$$

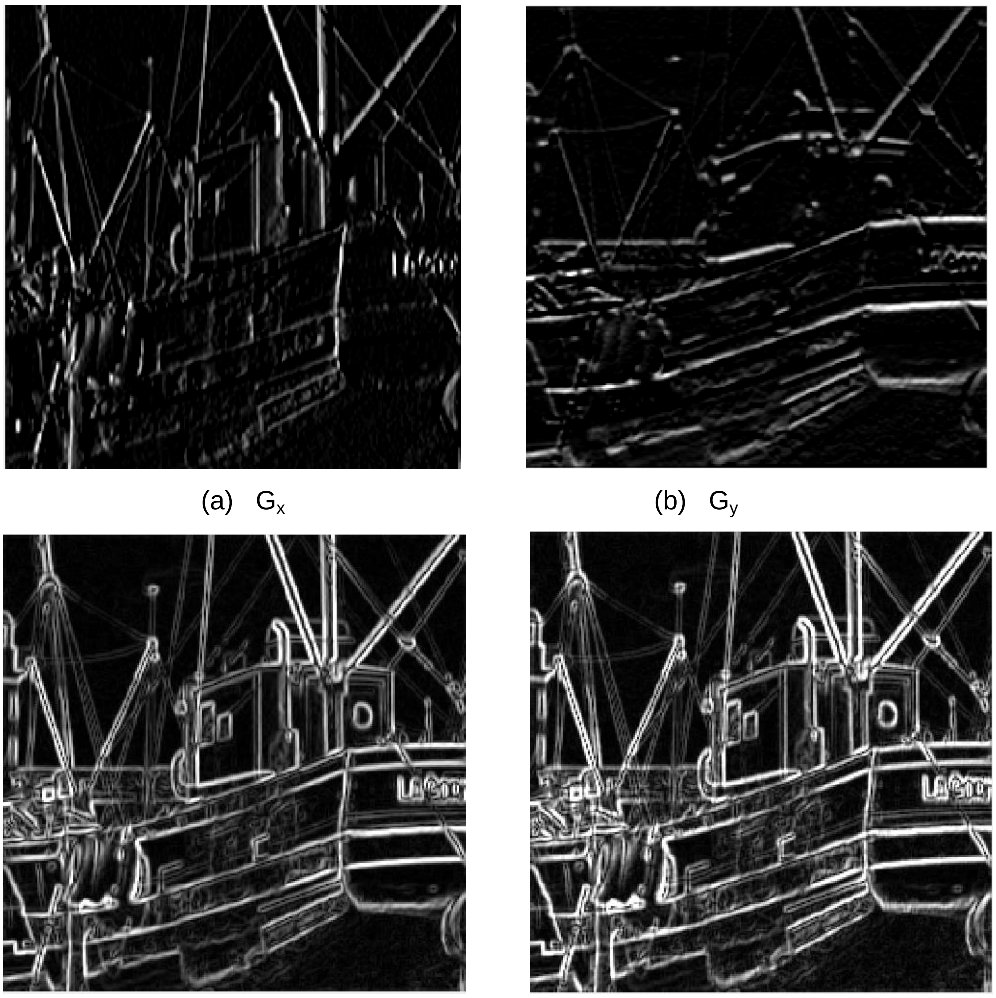
Typically, an approximate magnitude is computed for faster calculations using

$$|G| = |G_x| + |G_y| \tag{3}$$

The angle of orientation of the edge giving rise to the spatial gradient is given by

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{4}$$

In this case, orientation 0 is taken to mean that the direction of maximum contrast from black to white runs from left to right on the image, and other angles are measured counterclockwise from this. Figure 4 shows how MATLAB functions are used to implement the Sobel operators. The components G_x and G_y are also displayed separately, as well as the results for Equations (2) and (3). Usually, the MATLAB functions perform some type of morphological post processing that renders the edges in the image more distinct and sharper than with just using the aforementioned equations.



(a) G_x (b) G_y

(c) Image using Equation (2) (d) Image using Equation (3)

Figure 4. Sobel Convolution Kernels and Image Segment

Edge detection using Sobel operators works on the premise of computing an estimate of the first derivative of an image to extract edge information [8]. By computing the x and y direction derivatives of a specific pixel against a neighborhood of surrounding pixels, it is possible to extract the boundary between two distinct elements in an image [9]. Due to the computational load of calculating derivatives using squaring and square root operators, fixed coefficient masks have been adopted as a suitable approximation in computing the derivative at a specific point.

4. Sobel Edge Detection Algorithm via MBD

With the design becoming more complex, it has necessary to describe a high level for design. For the high level of description, it not only can allow designers to run faster simulation, but also be used throughout the development process validation. The resulting process allows developers to identify errors early and avoid costly errors found in the development direction. This high level design is usually done by the system engineer. To implement a DSP algorithm on hardware such as an FPGA, a system level engineer first designs the algorithm and verifies that the algorithm satisfies the project requirements. This design later becomes the golden reference for the engineers responsible for taking the algorithm to the hardware.

The Sobel edge detection algorithm has been implemented in Simulink. The algorithm is comprised of two 2D filters, one to calculate the gradient in the column direction (top filter) and one to calculate the gradient in the row direction (bottom filter), which is shown in Figure 5. Both filters use a 3x3 kernel.

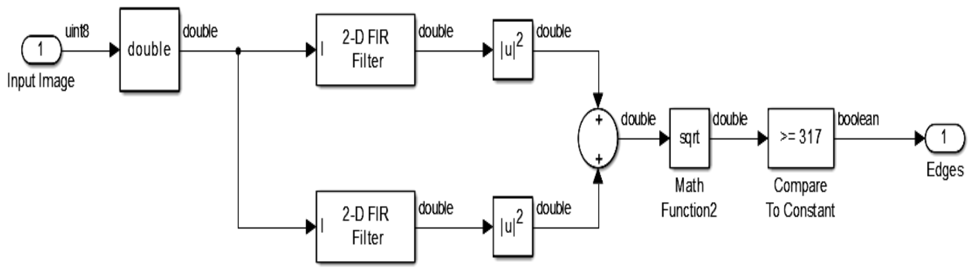
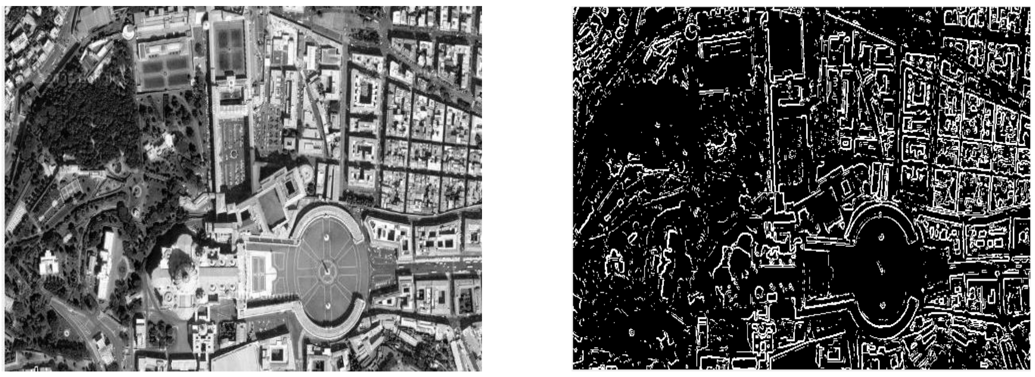


Figure 5. Sobel Edge Detection Using Equation (2)

The satellite image as the input to the edge detection algorithm. This image serves as an input test vector and is used throughout the design. Floating point simulation result is shown in Figure 6. If engineers who are responsible for the hardware implementation of the algorithm work in the Simulink environment as well, there is no need for extra overhead in porting the test vectors into different applications or creating test harnesses that are prone to human errors. The test harness that is used in the executable specification is used throughout the design.



(a) Original Satellite Image (b) Edge Detection Image

Figure 6. Implementation of Float Point for Sobel Edge Detection

When designing the executable model, the system engineer generally does not keep the implementation details in mind but rather implements the algorithm to match the behavioral requirements. Once the system engineer submits the executable specification to the development team, the development team may need to make modifications to the executable specification to fit the design into a real-time system that may have limited resources such as memory or processing power.

In this design, we may decide to eliminate the square operations and replace them with the absolute value operation for more efficient hardware implementation, and it is shown in Figure 7. This will cause a difference between the co-simulation result and the golden reference but for the sake of this example, we assume that the difference is acceptable. Open the edge detection design model. You can see the same test vectors as the previous model are still being used. The result can be easily verified against the golden reference. A numerical display shows the mean difference between the golden reference and the new design, and the simulation results are shown in Figure 8.

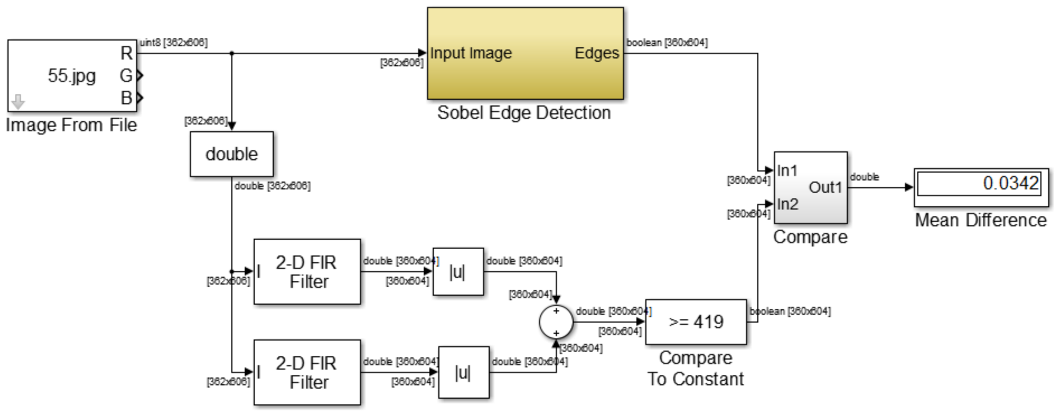
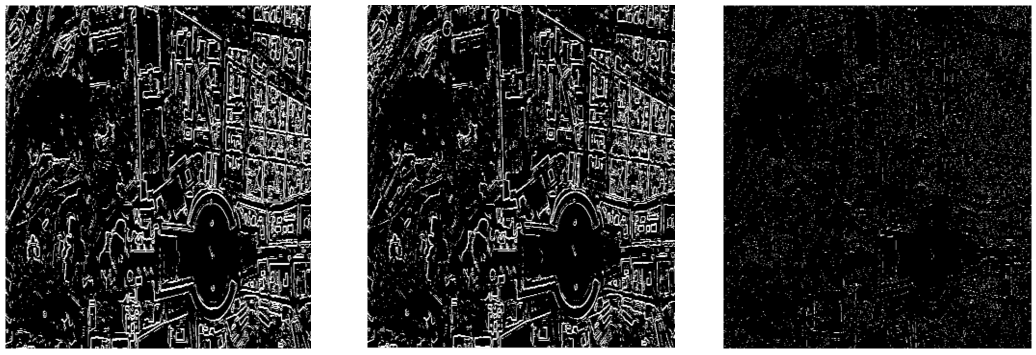


Figure 7. The Comparison of Float Point for Sobel Edge Detection



(a) Original algorithm (b) Improved algorithm (c) Comparison of Images

Figure 8. The Comparison of Float Point for Sobel Edge Detection

Since our ultimate goal is to implement the algorithm in an FPGA, we must convert our double precision design to a fixed-point design. This can be done easily using Simulink. We use the double precision model we developed in the previous section to directly develop a fixed-point model without introducing any new blocks. Simulink allows us to determine the number of bits and scaling for data as well as mathematical operations, and provides a great environment for analyzing the fixed-point operation of a system [10].

In this fixed-point design, the input to the filters is a signed 9-bit integer and the outputs of the filters are signed 11-bit integers. The structure diagram of fixed-point algorithm is shown in Figure 9. If you double-click on each computational block, such as the filters or the sum blocks, you can see that the developer can easily tune the bit width and scaling related to the internal computations of that block. This gives huge leverage to the designer to compromise between matching the output of the golden reference while using the least number of bits necessary to save area on the device. The simulation results are shown in Figure 10.

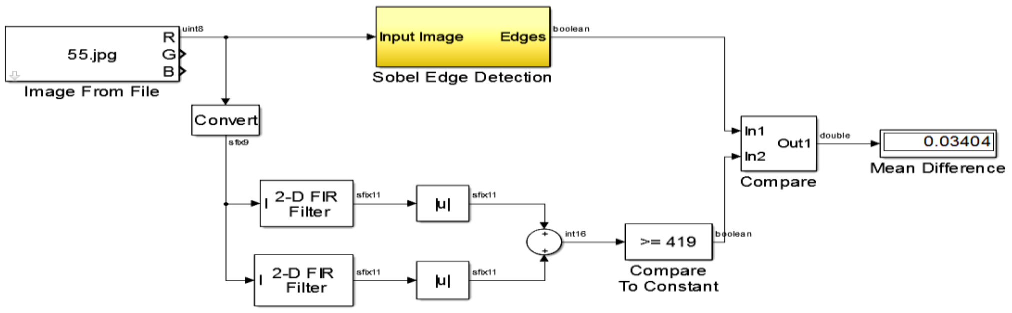


Figure 9. The Comparison of Fixed-point for Sobel Edge Detection

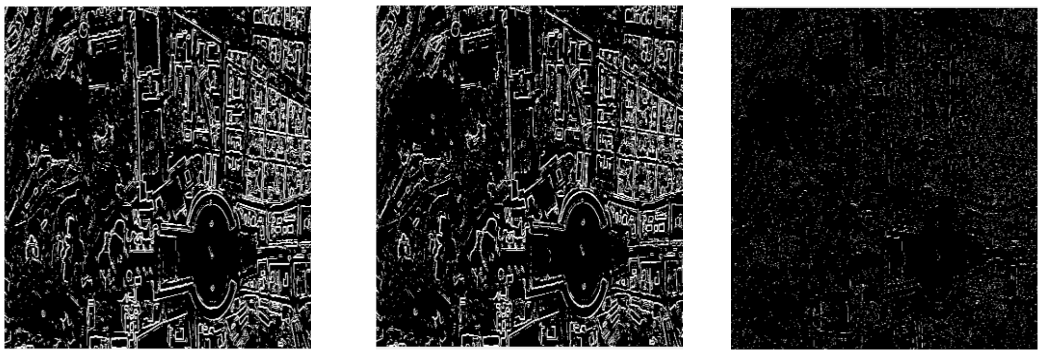


Figure 10. The Comparison of Fixed-point for Sobel Edge Detection

The model from the previous section can be passed to the HDL designer who can use the 2D filter designed in the last section to write the corresponding HDL code. Once the code is written, the HDL designer can use HDL Verifier to simulate the HDL design in the Simulink environment using ModelSim, and compare the output of the HDL design to the output of the executable specification. Note that in this process, there is no need for generating an HDL test bench. The Simulink model feeds the input test vector to ModelSim through HDL Verifier and extracts the data from ModelSim back to the Simulink environment. The HDL designer can readily verify whether the HDL code runs in accordance with the specifications. The Figure 11 shows a snap shot of the 2D filter displayed in ModelSim

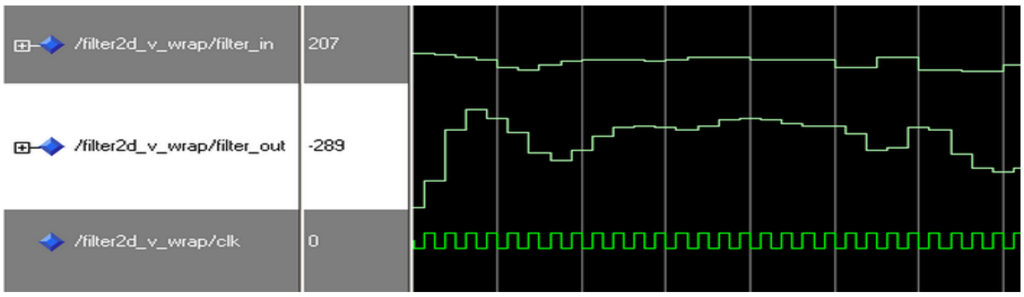


Figure 11. The Simulation of 2D Filter in ModelSim

5. Conclusion

A codesign strategy for sobel edge detection with MBD is presented in this paper. The design approach is supported by a parameterized model that provides optimized building blocks for a wide variety of modules, and an automatic synthesis tool for rapid description and implementation sobel edge detection. Combining these elements with a model-based design flow supported by different CAD tools running in the MATLAB environment eases the concurrent synthesis and verification of hardware and software components in every design stage. The described methodology allows accelerating the design process of image processing systems. This reduction in development time can be used to optimize a particular solution, as well as to compare different solutions in order to obtain the best tradeoff between cost and performance.

Acknowledgments

We would like to thank my mentor professor WANG for stimulating discussions with respect to the topic of this paper and laboratory equipment. Moreover, we greatly appreciate the reviewers' comments that lead to an improved presentation of the results.

References

[1] S. Wang and K. G. Shin, "Task Construction for Model-Based Design of Embedded Control Software", IEEE Trans. Software Eng, vol. 32, no. 4, **(2006)**, pp. 254 -265.
[2] J. Liu, "Model based design workflow for FPGA compliance with DO-254 standard", Proceedings of international symposium on information technology in medicine and education, **(2012)**, pp. 1026-1030.
[3] S. SanchezSolano, M. BroxJimenez, E. delToro, P. BroxJimenez and I. Baturone, "Model-based design methodology for rapid development of fuzzy controllers on FPGAs", IEEE Trans. Ind. Informat, vol. 9, no. 3, **(2013)**, pp. 1361-1370.
[4] V. Socci, "Implementing a model-based design and test workflow", Proceedings of 2015 IEEE International Symposium on Systems Engineering (ISSE), **(2015)**, pp. 28 -30.
[5] S. Sharma and W. Chen, "Using Model-Based Design to Accelerate FPGA Development for Automotive Applications", Proceedings of 2009 SAE World Congress, **(2009)**, pp. 1-15.
[6] N. Kanopoulos, N. Vasanthavada and R. Baker, "Design of an edge detection filter using the Sobel operator", IEEE J. Solid-State Circuits, vol. 23, no. 3, **(1988)**, pp. 358-367.
[7] S. Jin and W. Kim, "Fine directional de-interlacing algorithm using modified sobel operation", IEEE Trans. Consumer Electron., vol. 54, no. 2, **(2008)**, pp. 857-862.
[8] M. Petrou, V. A. Kovalev and J. R. Reichenbach, "Three-Dimensional Nonlinear Invisible Boundary Detection", IEEE Transactions on Image Processing, vol. 15, no. 10, (**2006**), pp. 3020 - 3032.
[9] T. Neuvo, P. Heinonen and I. Defee, "Linear-median hybrid edge detectors", IEEE Transactions on Circuits and Systems, vol. 34, no. 11, **(1987)**, pp. 1337 - 1343.
[10] P. K. Dash, S. S. Pujari and S. Nayak, "Implementation of Edge Detection Using FPGA and Model Based Approach", Proceedings of 2014 International Information Communication and Embedded Systems (ICICES), IEEE, **(2014)**, pp. 1- 6.

Author

Erhu Xie, He is associate Professor of Dept. of Computer Science, Jining Normal University, China. His current research interests include database development, image processing and software engineering.