

Hardware Software Co-Simulation and Real-Time Video Processing For Edge Detection Using Matlab Simulink Model Blockset

Purnawarman Musa and Nur Farida Irmawati

Lecture Gunadarma University, Jl. Margonda Raya No. 100, Pondokcina-Depok Indonesia

Abstract

FPGA can be useful in many different applications, such as developing more complex systems to be compatible. Using the blocks designed by Xilinx for its System Generator software, a simple algorithm can be designed and tested using Simulink and an FPGA development board, for example edge detection algorithm. Edge detection of image significantly reduces the amount of data and filters out unwanted or insignificant information and gives the significant information in an image. Implementing edge detection is not limited to software development but also hardware development. Edge detection implemented in hardware have emerged as the most viable solution for improving the performance of image processing systems.

This application note will make use of the Spartan-3A DSP development board to explain how to co-simulate Xilinx's highest level form of an Edgefilter on a still image and implementation sobel edge detection on real-time video based on FPGA board. This document begins by explaining how to convert a still image into a format useable by Matlab and Simulink, shows the construction of a basic algorithm using Xilinx's System Generator blocks, and also illustrates the procedure of implementing the design real-time video processing using the FPGA development board. The light condition and color contrast will affect the edge detection result. The result of this implementation is detecting edges from the real-time video that has been captured by the camera.

Keywords: Sobel Fiter, Edge Detection, Co-simulation, FPGA, Matlab, Simulink

1. Introduction

Hardware software using the blocks designed by Xilinx for its System Generator software, a simple algorithm can be designed and tested using Simulink and an FPGA development board for Gaussian, Edge, and Blur filters are used in many digital image and video processing systems as a morphological operation for improving the quality of an input image with respect to its ability to have more complex systems operate successfully on it.

Image processing is a wide range of discipline. The areas of application of image processing are so varied. There are many types of image processing algorithms, one of them is edge detection. Edge detection is a fundamental tool used in most image processing applications to obtain information from the frames as a step to feature extraction and object segmentation. The main goal of edge detection is to locate and identify sharp discontinuities from an image.

Usually edge detection algorithm is implemented using software-based, however, it is not limited to software development but also hardware development. Implementation of edge detection which is basic of image processing using software-based in computer is less effective because edge detection only use a low power in computer which has a high power. Implementing edge detection using FPGA is a solution to make the implementation more effective. The objective in this paper is to implement edge detection an image and on real time video based on FPGA board. The system used an input from a CMOS camera and output to a VGA display and verified the results video in real time.

A. Sobel Edge Detection

Edge detection is a process of locating an edge of an image. Detection of edges in an image is a very important step towards understanding image features. Edges consist of meaningful features and contain significant information. It significantly reduces the image size and filters out information that may be regarded as less relevant, thus preserving the important structural properties of an image. There are some edge detection algorithm method, such as Robert, Prewitt, Canny, and Sobel. The method that use in this paper is Sobel.

Sobel is gradient based edge detection algorithm which performs a 2-D spatial gradient measurement on the video data. It uses a pair of 3x3 convolution mask, one estimating gradient in horizontal, and other in vertical. This is shown in figure 1. Then the value of the gradient magnitude is computed from the above two gradients.

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(a) (b)

Figure 1. Sobel Edge detector – (a) and (b) Vertical

B. Hardware Software Co-Simulation for image static

A basic definition is manipulating simulated hardware with software to verify as much of the product functionality, hardware and software. System Generator provides accelerated simulation through hardware co-simulation. System Generator will automatically create a hardware simulation token for a design captured in the Xilinx DSP blockset that will run on one of over 20 supported hardware platforms. This hardware will co-simulate with the rest of the Simulink system to provide up to a 1000x simulation performance increase.

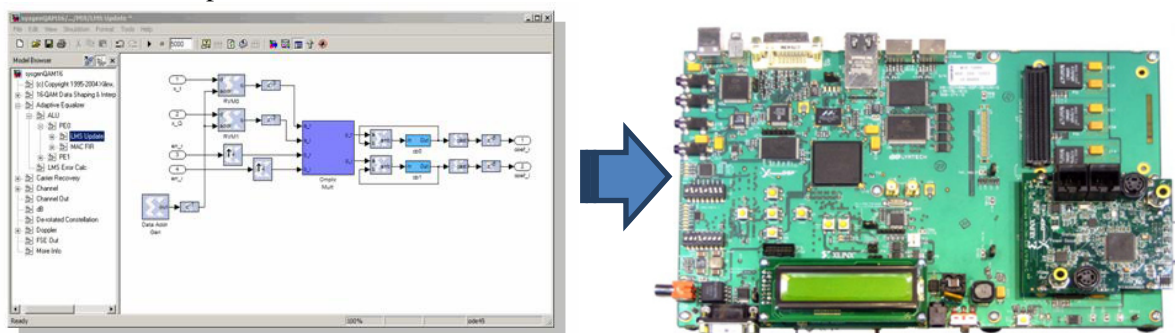


Figure2. Hardware Software Co-Simulation using Matlab Simulink Model on Spartan-3A DSP 3400 Development Platform

C. Real-Time Concept in Video Processing

Real-time is an elusive term that is often used to describe a wide variety of image/video processing systems and algorithms. Kehtarnavaz and Gamadia divides three main interpretation of the concept of real-time, namely real time in the perceptual sense, real-time in the software engineering sense, and real-time in the signal processing sense.

Real-time in the perceptual sense describes the interaction between a human and a computer device for a near instantaneous response of the device to an input by a human user. Real-time in the software engineering sense is based on the concept of bounded response time as in the perceptual sense. Real-time in the signal processing sense is based on the completing processing in the time available between successive input samples.

2. System Generator Flow Diagram

System Generator is a DSP design tool from Xilinx that enables the use of The Mathworks model-based design environment Simulink for FPGA design. Previous experience with Xilinx FPGAs or RTL design methodologies are not required when using System Generator. Designs are captured in the DSP friendly Simulink modeling environment using a Xilinx specific blockset. All of the downstream FPGA implementation steps including synthesis and place and route are automatically performed to generate an FPGA programming file.

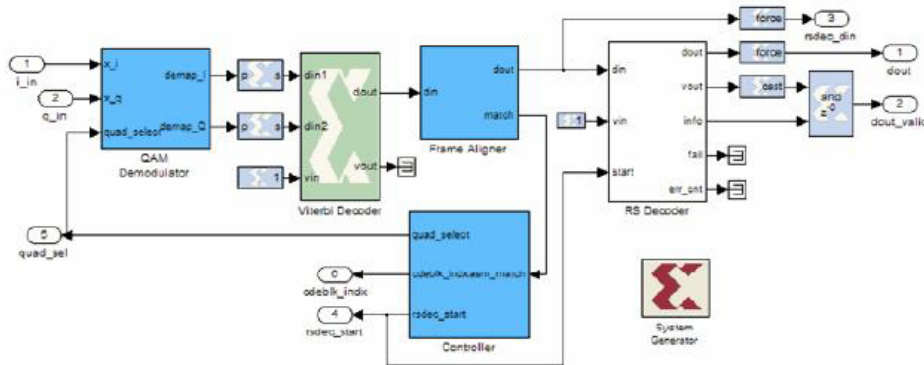


Figure 3. Simulink Model Blockset Design

In this chapter, designing flow diagram of system generator is explained using Simulink software. System Generator works within the Simulink model-based design methodology. Often an executable spec is created using the standard Simulink block sets. This spec can be designed using floating-point numerical precision and without hardware detail. Once the functionality and basic dataflow issues have been defined, System Generator can be used to specify the hardware implementation details for the Xilinx devices. System Generator uses the Xilinx DSP blockset for Simulink and will automatically invoke Xilinx Core Generator to generate highly optimized netlists for the DSP building blocks. System Generator can execute all the downstream implementation tools to product a bitstream for programming the FPGA.

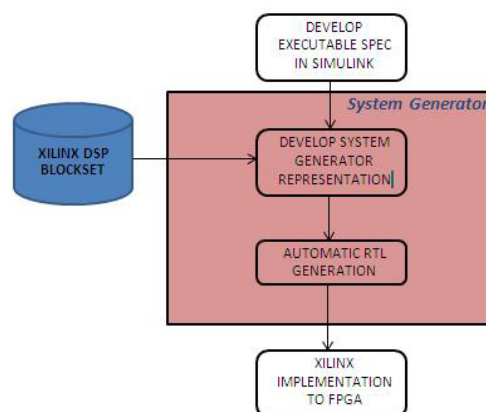


Figure 4. System Generator Design Flow Diagram

2.1 Xilinx DSP Blockset

The Xilinx DSP blockset is accessed via the Simulink Library browser which can be launched from the standard MATLAB toolbar. The blocks are separated into sub-categories for easier searching. One sub-category, "Index" includes all the block and is often the quickest way to access a block you are already familiar with. Over 90 DSP building blocks are available for constructing you DSP system.

2.2 Defining the FPGA Boundary

System Generator works with standard Simulink models. Two blocks called Gateway In and Gateway Out define the boundary of the FPGA from the Simulink simulation model. The Gateway In block converts the floating point input to a fixed-point number.

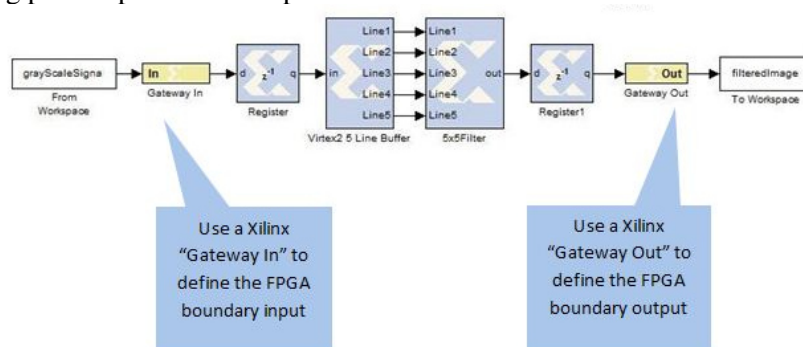


Figure 5. Defining the FPGA Boundary

2.3 Adding the System Generator Token

Every System Generator diagram requires that at least one System Generator token be placed on the diagram. This block is not connected to anything but serves to drive the FPGA implementation process. The property editor for this block allows specification of the target netlist, device, performance targets and system period. System Generator will issue an error if this block is absent.

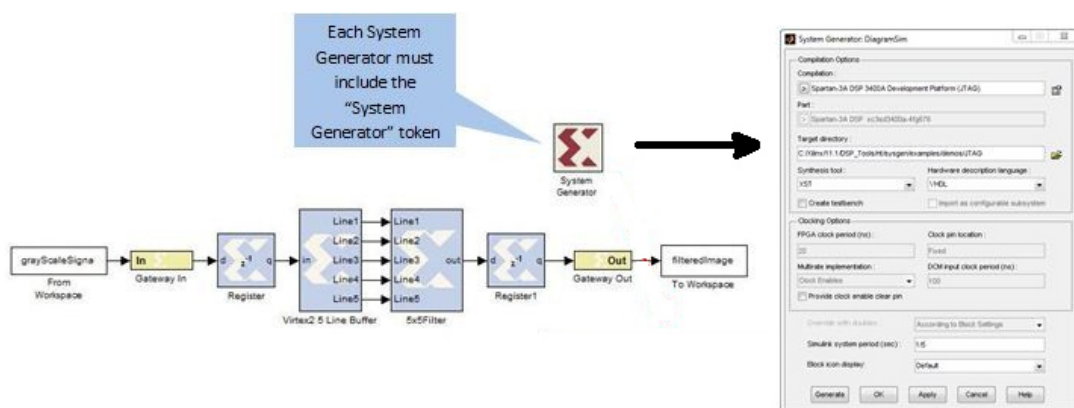


Figure 6. Adding the System Generator Token

2.4 Creating the DSP Design

Once the FPGA boundaries have been established using the Gateway blocks, the DSP design can be constructed using blocks from the Xilinx DSP blockset. Standard Simulink blocks are not supported for use within the Gateway In / Gateway out blocks. You will find a rich set of filters, FFTs, FEC cores, memories, arithmetic, logical and bitwise blocks available for use in constructing DSP designs. Each of these blocks are cycle and bit accurate.

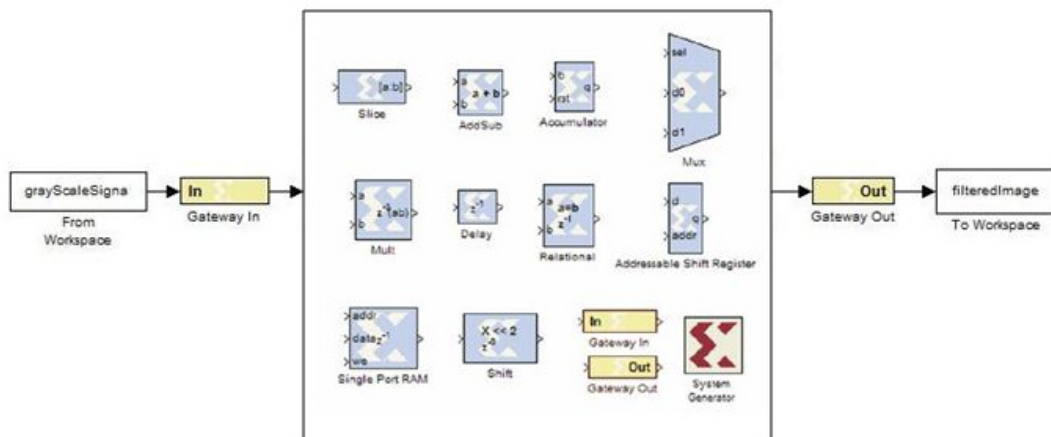


Figure 7. Creating the DSP Design

2.5 Automatic RTL Generation

Once the design is completed, the hardware implementation files can be generated using the Generate button available on the System Generator token properties editor. One option is to select Hardware Co-Simulation which allows the FPGA implementation steps of RTL synthesis and place and route to be performed interactively using tool specific user interfaces. Alternatively, choose Spartan 3A DSP Development Platform, and select JTAG. It will be created bitstream as the Compilation target and System Generator will automatically perform all implementation steps.

3. Design Co-Simulation and Real time video processing using Simulink for Sobel EdgeDetection Algorithm

3.1 Build hardware co-simulation of the edge detection filter model in Simulink

The first stage in the process for this simple filter example is to save the image of interest into a new directory where the project will be saved. It is recommended that the user chooses a square image for this tutorial so that no changes to the code provided by Xilinx will be necessary. Once an image is chosen, save it as a bitmap image.

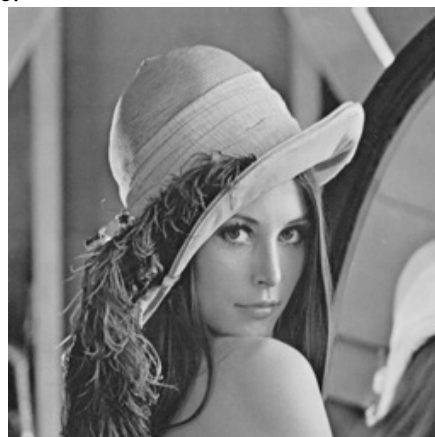


Figure 8. Input Image

For many image processing applications, the input image is first converted to grayscale. The following code can be used to convert the bitmap image into a grayscale signal that is a suitable data form for the FPGA hardware. The most important of the variables is grayScaleSignal, which will be used as the input to the hardware software co-simulation.

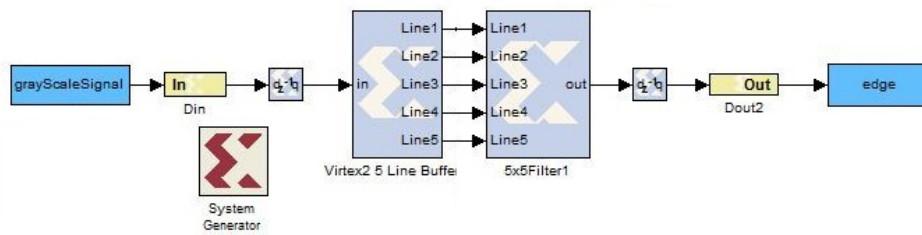


Figure 9. Filtering Design using System Generator Simulink

The figure above show the build of the edge filter model. "grayScaleSignal" is the block variable specifying the input for the co-simulation into the new model followed by the Gateway in blocks "edge" are the block variables specifying the output for the co-simulation. These blocks will receive the output from the FPGA co-simulation. The register blocks simulate D-Flip-Flops and should be used to synchronize the input and output with the FPGA clock. The filter will be implemented using Xilinx's 5x5Filter Block. The 5x5Filter block is a 5x5 mask that scans five lines of the image at a time to apply the chosen filter. Therefore, the system requires a buffer that accepts five lines from the image at a time to be fed into the filter. To begin the co-simulation process, the user must specify the amount of time that the system should run to successfully obtain all output data. To determine the run time for the system, use the following equation:

$$T = W \times W + 2 \times W + 30 \quad (1)$$

This equation will provide ample time for the system, assuming that the input image is a square image and 'W'. Represents the pixel width of the image.

2.2 Generate the co-simulation block

Model-based design refers the design practice of creating a high-level executable specification using the standard Simulink blocksets or MATLAB first to define the desired functional behavior with minimal hardware detail. This executable spec is then used as a reference model while the hardware representation is specified using the Xilinx DSP blockset. The picture below shows the result of generated hardware co-simulation.

The final addition to the co-simulation system is to add the System Generator block, which must be added to any System Generator design. For simplicity, copy and paste the System Generator block from the original system into the co-simulation system. This will allow the System Generator block to retain all settings necessary for this co-simulation tutorial.

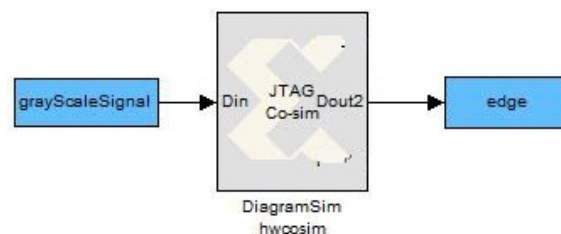


Figure 10. Co-simulation model

3.2 Build Real time video processing of the edge detection filter model in Simulink

The design will be created using Matlab/Simulink and generated using Xilinx System Generator to download into FPGA board. Figure11 is the model that showing how to design for the whole

system. The live video feed from camera input will be processed with the image processing algorithm in FPGA. Right after the process is done then the output will be display to the VGA monitor.

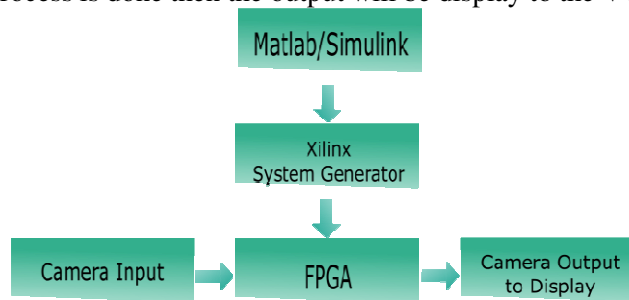


Figure 11. System Design

3.2.1 Matlab/Simulink

Matlab/Simulink is use for the design of the system. Figure12 is showing the full top system to implement edge detection.

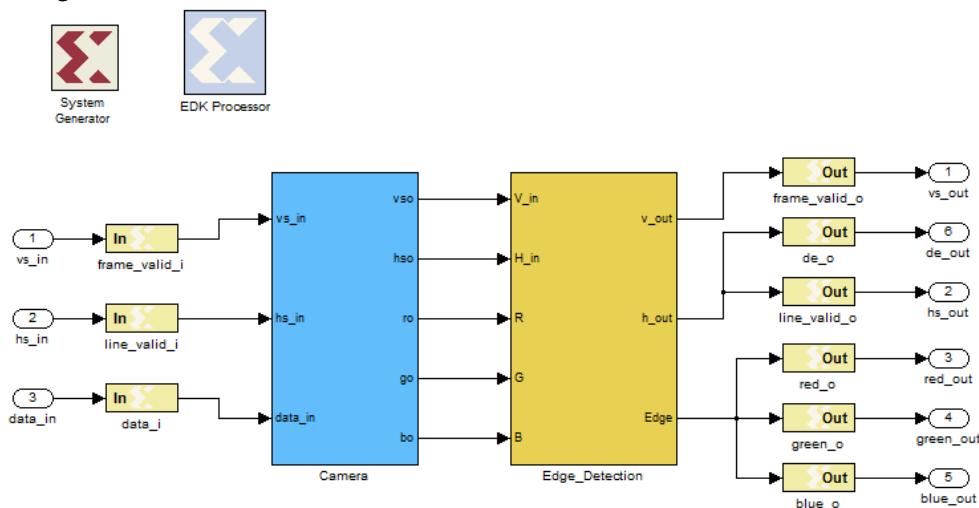


Figure 12. Top System

Camera block is borrowed from a demo system provided by Xilinx. In the camera block its set the dynamic range expansion, stuck pixel correction, brightness, Bayer filter, and color controls as shown in figure12.

The edge detection operation is implemented right after the camera block. Sobel calculation is described in figure 11, and from that calculation it can be used to build the sobel model in Matlab/Simulink as shown in figure14. Input from pixel_buffer will be processed in sobel calculation as neighbor matrix and it will be used for the horizontal and vertical differential calculation. As describe in previous chapter about sobel edge detection, this model is represent the process of that algorithm. After calculation with the mask matrix, the result from horizontal and vertical differential will be summed together to get the sobel value. Register block in this model is used to input threshold value as keyboard input to control the output.

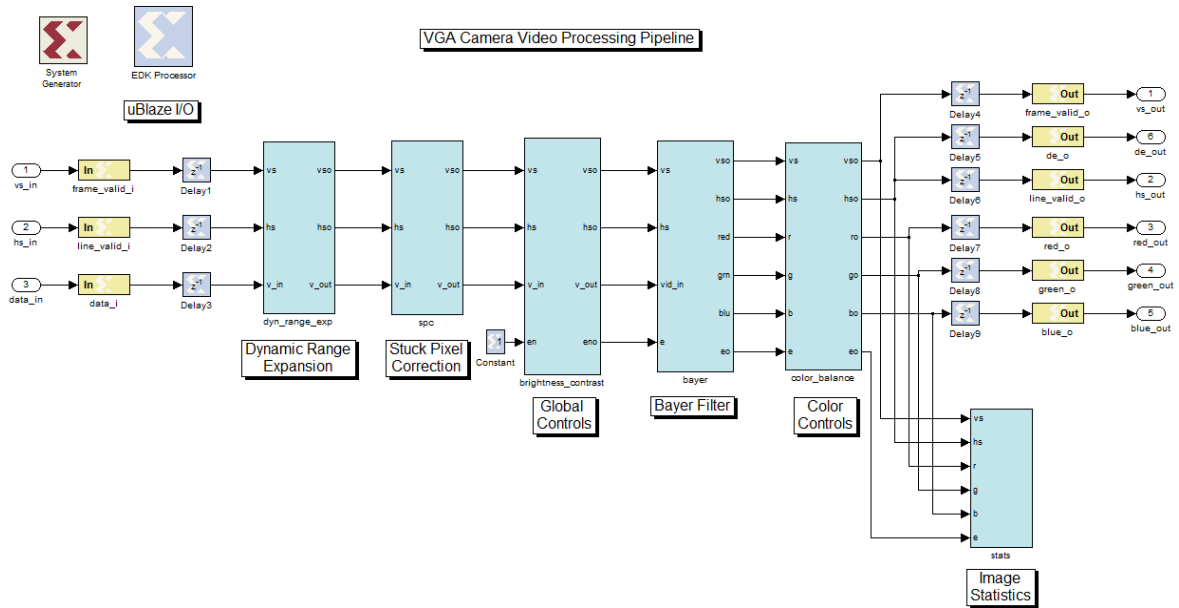


Figure 13. Camera Video Processing

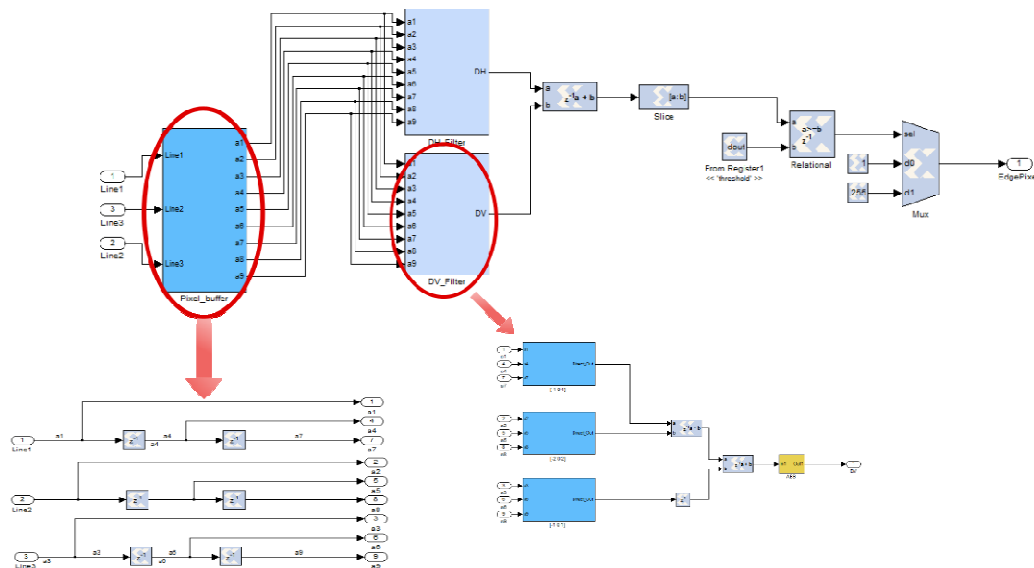


Figure 14. Sobel Edge Detection Block

3.2.2 Hardware Preparations

This hardware preparations should always be done at any time before running the system. This preparation applicable when wanting to reload the hardware to make sure there are no technical errors. This preparation is to set the board before download the program into FPGA board. The board preparations is shown in figure 15.

- A. Connect the VGA camera to the FMC-Video Card using the supplied CAT6 cable, first plug one end into the Camera #1 RJ45 connector on the FMC-Video Caard. Then plug the other end of the CAT6 cable into the RJ45 connector on the VGA camera.
- B. Connect a VGA monitor to the VGA output connector using using DVI adaptor on the Spartan-3A DSP FPGA Development Board
- C. Connect a notebook to the Spartan-3A DSP FPGA Development Board with a 9-pin RS232 serial Null Modem cable. Connect one end of the cable to the RS232 port on the Spartan-3A DSP FPGA Development Board and the other end of the cable to the serial port of the notebook
- D. Connect a notebook to the Spartan-3A DSP FPGA Development Board using JTAG cable. Connect one end of the JTAG cable to the JTAG port on the Spartan-3A DSP FPGA Development Board and the other end of the cable to usb notebook.
- E. Connect the power supply to the Spartan-3A DSP FPGA Development Board power supply input.

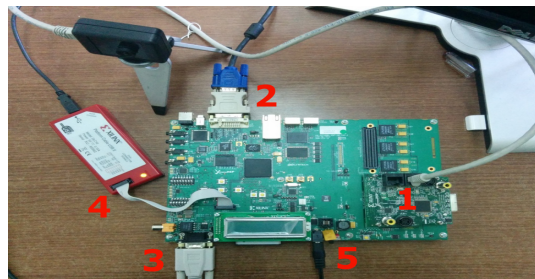


Figure 15. Board Preparations

3.2.3 Xilinx System Generator

The system design is developed using the Xilinx Embedded Development Kit (EDK) and System Generator for DSP. The Embedded Development Kit is a collection of Intellectual Property (IP) cores and tools for building FPGA based embedded systems. System Generator for DSP enables the use of the Matlab/Simulink modelling environment for FPGA design by providing a Simulink blockset of over 100 Xilinx optimized DSP building blocks.

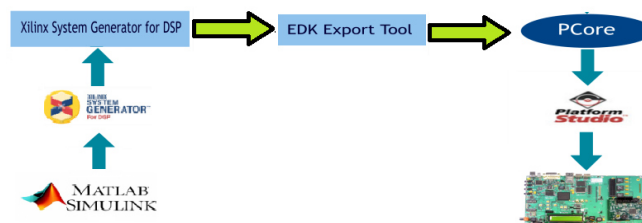


Figure 16. Sysgen Flow

Xilinx System Generator flow is shown in figure16. Matlab Simulink can be used to create a system design which is generate to EDK using Xilinx System Generator. In this research, the compilation target that author use is EDK Export Tool. The generated file is called pcores. The Simulink design that have been generated as pcores, will be used in Xilinx Platform Studio to complete the full system. Pcores used for including custom hardware peripherals. As shown in figure 16, the Xilinx Platform Studio will be download the bitstream into FPGA to implement the whole program into FPGA.

After generate Simulink using Xilinx System Generator, all of the algorithms will be processed in FPGA board. The setup for implementation consists of the Spartan-3A DSP FPGA Video Starter Kit (VSK), a development platform consisting of the Spartan-3A DSP 3400A FPGA, the FMC-Video daughter card, and a VGA camera as shown in figure17.



Figure 17. Spartan-3A DSP 3400 Development Platform, FMC-Video, and Camera

The Spartan-3A DSP 3400A Development Platform is built around a Spartan-3A DSP XC3SD3400A device that provides significant resources (for example, 126 embedded DSP blocks) for implementing high performance video processing systems and co-processors. Figure18 shows the process in the FPGA board. This video processing is designed as a system on a programmable chip with the help of Embedded Design Kit.

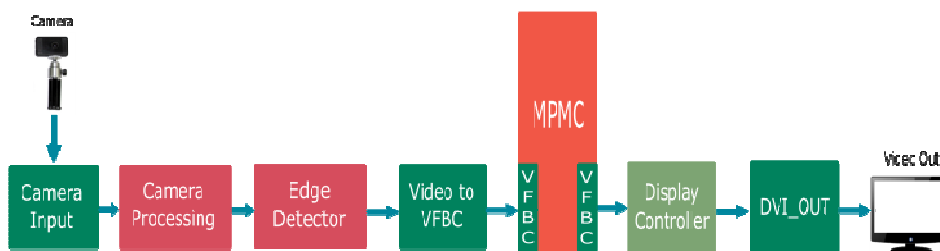


Figure 18. FPGA Design

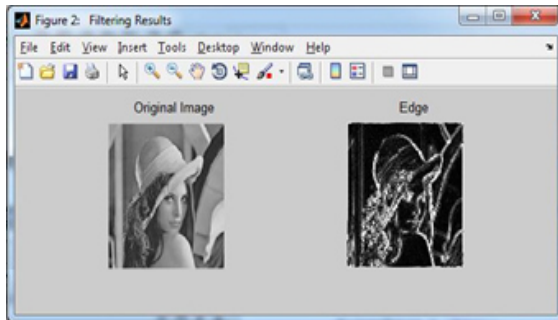
The serial video is de-serialized on the FMC-Video Card. The resulting parallel data stream is the input to the Camera In block. The camera core registers the signals and groups the video signals into a unified bus that is connected to the camera Processing block, which is included in the camera frame buffer reference design shipped with the VSK, control brightness, contrast and other parameters. The edge filter is applied on the input signal arriving from the Camera Processing block. The output signal is driven by Display Controller to the DVI output monitor. Video to VFBC and MPMC core helps to store the image data and buffer them to the output screen.

4. Experimental Result

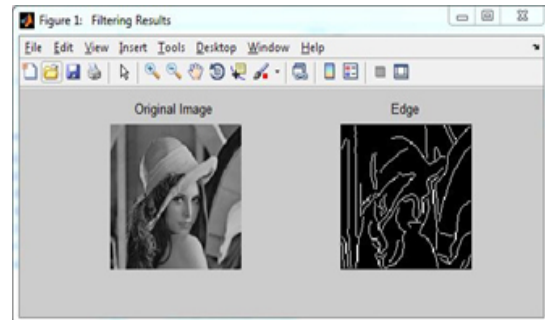
4.1. The output of the image from co-simulation and Matlab

The data received from the FPGA stored in the Matlab Workspace as a variable named "edge" for edge filter. The code translates the output data back into a two dimensional structure that can be read by Matlab. The output is then displayed as well as the original input image for comparison. Below is the output of the co-simulation process used in this tutorial with the original image compared next to the algorithm's output of image processed by an Edge filter.

The output above uses edge filter method using $BW = \text{edge}(I, \text{method})$ for detecting edges in image I, where method specifies the edge detection method used. The output using matlab is smoother than using co-simulation Simulink because there are any functions and parameters of edge that can be used for doing image processing like set up the method, threshold, direction, even sigma. But, using co-simulation is more accurate and more similar as the original than using matlab because the pixel used is more.



(a) Edge using Co-Simulation



(b) Edge using Matlab

Figure 19. Original Image vs. Filtered Image

Edge Filter using Matlab Based on Mathworks, there are a few of edge syntax that can be used for filtering the image.

```
SW = edge(I)
BW = edge(I,method)
BW = edge(I,method,threshold)
BW = edge(I,method,threshold,direction)
BW = edge(I,method,threshold,direction,'nothinning')
BW = edge(I,method,threshold,direction,sigma)
[BW,threshOut] = edge( __ )
[gpuarrayBW,threshOut] = edge(gpuarrayI, __ )
```

Figure 20. The Syntax of Edge Filter using Matlab Based on Mathworks

4.2. The output of the real time video processing

Output from camera video processing model is showing real-time video to the VGA monitor. The VSK includes a VGA camera based upon the Micron MT9V022 image sensor of resolution 720 x 480 pixels delivering serial frames at 60 fps through a FPGA Mezzanine Card (FMC) Daughter card which is an add-on card that augments the video capabilities of the Spartan-3A DSP 3400A Development Platform. Figure21 shows the original real time video using camera video processing model.



Figure 21. Original Real-Time Video

The edge detection algorithm is inserted after the camera processing model. So, the output will show the edge detection from the object that has been captured. The result for implementation of sobel edge detection is presented in figure22.



Figure 22. Sobel Edge Detection

Overall, the edge detection part of the system is very successful. It is impressive at detecting edges, and from the edge detection output, it can be seen what is the object's shape. The factor that affecting edge detection is light condition. When there is the large amount of lights that captured by the camera, then the detected edges is also growing up. In addition, it will be show some noise on the output. Compare the figure 22.a with figure 22.b. In figure22.b, there is more detected edges than detected edges in figure 22.a because of differences in amount of lights that captured by the camera.







Figure 23. Chess Piece

The other factor that affect the output is color. Look at the figure 23.a and figure 23.b. In figure 23.a, it can be seen there are some chess pieces in the chair clearly, but when the edge detection is implemented as shown in figure 23.b, the edge from dark brown chess piece (king) is not detected because the chess piece's color is almost the same as the background color (chair). It is different with the other chess piece in white color that is white king and white knight. The edges from both chess pieces is detected properly because their color is difference with the background.

3.1 The difference result between FPGA board and computerto implementing sobel edge detection in realtime video processing

The programming language that used to implement sobel in real time video processing in this research is Matlab. Sobel edge detection can be implemented using FPGA or Computer. The programming language that used in FPGA is Matlab Simulink, while Computer using Matlab programming. The result of Sobel edge detection from those media is difference. The difference can be seen in Table 1.

Table 1. Implementing Sobel Edge detection using Computer and FPGA board

Media	First Attempt	Second Attempt
Computer		
FPGA Board		

From Table 1, there are first attempt and second attempt column. Light intensity in both attempt was measured using a lux meter. In first attempt, light intensity that close to the camera is 5 lx, it can be seen that the object is detected more clearly than second attempt, it is because there are more light in the object than camera. In second attempt, light intensity that close to the object is 76 lx. It can be seen there are less object that detected than first attempt because there are more light that close to the camera than the object. From the image, it can be seen that computer's result has more noise than FPGA's result. Besides that, sobel edge detection result from computer has one until five second delay in the video result, while sobel edge detection result from FPGA board has 60 fps.

5. Conclusion

The image output by hardware software co-simulation using simulink are the presence of some differences with Matlab. Implementation of Edge Detection in Real-Time Video Processing that based on FPGA has been successfully developed. The overall implementation has been going quite well and running as expected. Matlab/Simulink can be used as a tool to develop the edge detection implementation using the Spartan-3A DSP FPGA Video Starter Kit.

Nevertheless, smooth running of the system depends on a few things. One of the factor is light condition. If there is large amount of light in the object, then the output will produce more noise in edge detection result. In addition, there is more detected edges in the output. However, if the light condition is normal, then the detected edges is quite well and produce less noise.

Another factor is the differences in color. If the object and its background's color is quite contrast, then the edge can be detected, but if the object and its background's color has a low contrast level, then it is hard to detect the edges from that object.

References

- Raman Maini, Himanshu Aggarwal. *Study and Comparison of Various Image Edge Detection Techniques*. IJIP, Volume (3): Issue (1).
- Ray, D. 2013. *Edge Detection in Digital Image Processing*.
- Said, Y., et al. 2012. *Embedded Real-Time video Processing System on FPGA*. Pages 85-92. ICISP, LNCS 7340.
- M. Iaj Chwalisz. 2011. *Xilinx fpga design using simulink with hardware cosimulation*,
- Adhyana Gupta. 2013. *Hardware Software Co-Simulation For Traffic Load Computation Using Matlab Simulink Model Blockset*, International Journal of Computational Science and Information Technology (IJCSITY) Vol.1, No.2, May.
- Kehtarnavas, N. and N., G. M. 2006. *Real-Time Image and Video Processing: From Research to Reality*. Dallas: Morgan & Claypool Publishers.
- Kehtarnavaz, N. 2004. *Real-Time Digital Signal Processing Based on The TMS320C6000*. Amsterdam: Elsevier Academic Press.
- X. U. Guide. 2008. *System generator for dsp*.
- T. Ganley. 2010. *Fpga co-simulation of gaussian filter algorithm*.
- C. Spandana. 2013. *Real time hardware co-simulation of sobel edge detection using fpga*, International Journal of Scientific Engineering and Technology Research.
- A. A. Ingle. 2014. *Hardware software co-simulation of edge detection for image processing system using delay block in xsg*, International Journal of Research in Engineering and Technology.