# An Image Edge Detection Demo on the DE2i-150 Board

## Introduction

This demo is intended to help users gain a quick introduction to the DE2i-150 board using a demo example. This introductory demo shows how a simple edge detection algorithm can be performed using both on-board FPGA custom core and Atom general purpose CPU. A grayscale .bmp file is used as an input to the demo and a corresponding .bmp image file with only edges is the output of demo.

## Background Info

The main image processing (IP) algorithm used in this demo is Laplacian of Gaussian (LoG), which essentially is a simple matrix convolution. The algorithm, in this particular application, takes a predefined 3×3 kernel and convolves it with every image 3×3 window matrix in a sliding window fashion. As a result, single edge intensity values of raw image pixels are calculated, which then become pixels in the new edge-detected image.

$$\begin{Bmatrix} pixel6 & pixel3 & pixel0 \\ pixel7 & pixel4 & pixel1 \\ pixel8 & pixel5 & pixel2 \end{Bmatrix} \times \begin{Bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{Bmatrix}$$

**Figure 1: Convolution of matrices**

As shown in the figure above, the final convolved pixel result (edge intensity value) is:

$$convolvedpixel = pixel4 \times 4 - (pixel1 + pixel3 + pixel5 + pixel7)$$

Sample result is shown below:

**Figure 2: Original and Edge-Detected Images**

## Design Info

For the hardware design on the FPGA, two main project-specific blocks are present in this edge detection application: Top Level Design and User Logic State Machine. All arithmetic computations are programmed in Top Level Design block as shown in the following diagram. User Logic State Machine is the "brain" of the design, controlling Top Level Design to do certain operations and computations. It is also responsible for controlling Read & Write Master to read and write off-chip memory. The read/write masters are the interface to the system interconnect bus.

For the Software running on the Atom, the processor does memory reading and writing via PCIe bus. PCIe and Avalon communicate with provided bridging modules.

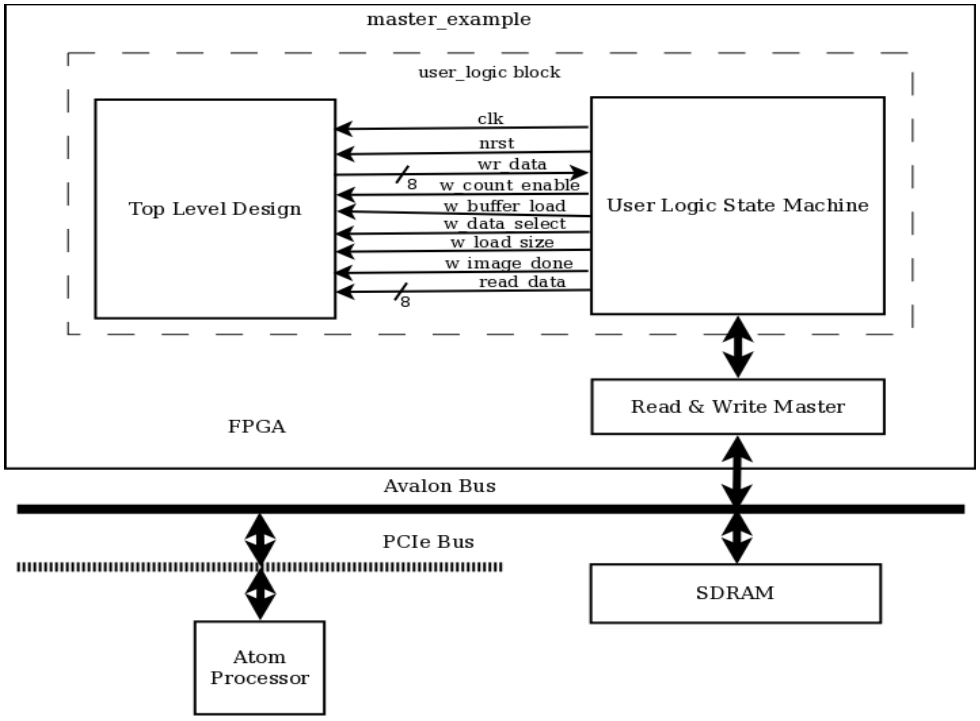


**Figure 3: Design Interface Diagram**

## Hardware and Software Review

| **Hardware (DE2i-150)** | |
|---|---|
| **Atom CPU** | An on-board general purpose processor that runs Linux OS. It is responsible for running relevant C code in this demo and exchanging various files between DE2i-150 board and user's ECN account. |
| **FPGA** | Programmed to be the design core of this demo. It is responsible for computing every edge intensity value. |

| | |
|---|---|
| **SDRAM** | An off-chip memory block that is mainly responsible for storing both original and processed images. |
| **SRAM** | An on-chip memory block that mainly serves as registers to bridge the non-image data communications between Atom and FPGA. Possible register data include start/stop byte, size of image, etc. |
| **Software (Quartus / C)** | |
| **Image Processing Design** | The main System Verilog code for the IP core. |
| **C Code** | This code runs on Atom and is written in C, with PCIe library. It is mainly responsible for writing to and reading from both SDRAM and SRAM. With specific input argument, it is used to demo the whole IP process in one setting. |
| **Others** | PCIe driver and required libraries |

## Demo Procedure

Please make sure to follow through all the steps. Ask your TA if something doesn't work.

1. Check all the connections on board. Make sure the Ethernet, mouse, keyboard, VGA to monitor and USB blaster to PC are all connected.

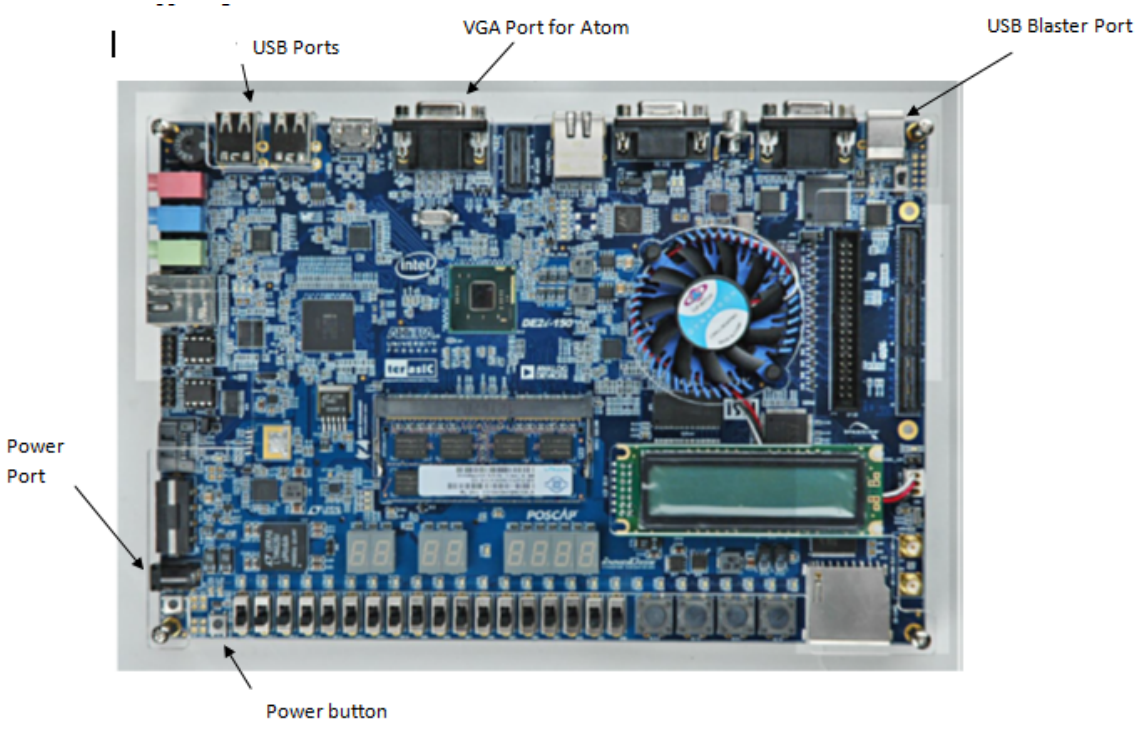


**Figure 4 : DE2i-150 board connections**

2. Power up DE2i-150 board by pressing the power button at bottom left of the board and wait for the OS to boot.
3. Now that you have the board setup. You need to program the design onto the FPGA. Invoke the quartus tool by typing “quartus &” in your terminal (of the host machine). In the Quartus tool go to Tools > Programmer. This will launch the Programmer tool to download the FPGA binary into the device. The programmer tool interface looks like figure 5 below.
4. In the Programmer tool, click on "Hardware Setup". The USB- Blaster should show up as in the figure 6 below. Note that the port number may not be exactly the same as in the figure below. Double click on the USB-Blaster entry.

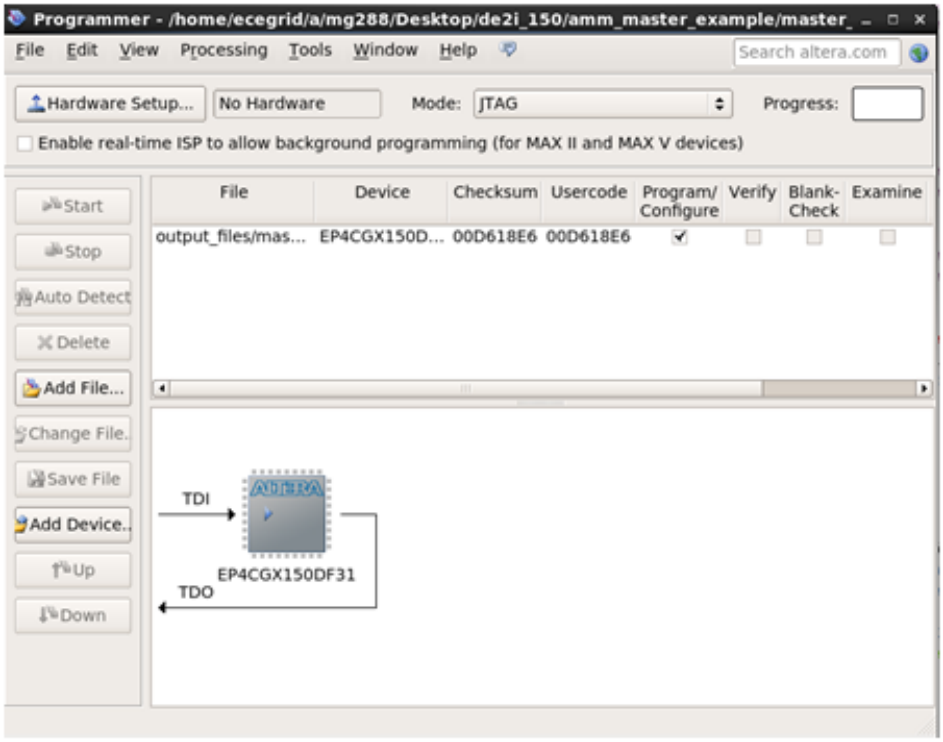


**Figure 5 : Programmer tool interface**


**Figure 6 : Hardware Setup for Programming the FPGA**

5. Now you should be back to the interface as in figure 5 above. Add the .sof provided to you as part of the demo files. This is the binary that gets loaded to the FPGA. Ensure the Program/Configure option is checked.

   NOTE : You can find "master_example.sof" file in the "hardware/output_files" folder in the tar ball provided to you.

6. Click on Start. The progress bar indicates the progress of writing the SOF file into the device. When it reaches 100% the device has been loaded with your hardware! Congratulations! You have now successfully programmed the FPGA with the demo design.
7. Now that the FPGA is loaded with the design we can run the demo on the Atom. Open up a terminal using the GUI. This is for the Linux distro that runs on the Atom. The Atom needs to be rebooted to recognize the modified FPGA hardware. Enter the command *sudo reboot*

   **NOTE**: in order to avoid any memory read/write issues, be sure to reboot the board every time after programming the FPGA.

8. Copy the "software" folder provided to you in the tar ball to the DE2i-150 board. Use

   *scp -r  software root@<ip_address of the board>:~/.*

   **NOTE:** use the command "ifconfig" to get the IP address of the board.

9. Open a terminal in the OS running on the Atom. Go to the demo folder. Type the command.

   *cd ~/software/*

10. Install drivers for PCIe (this needs to be done after every boot) by issuing:

    *cd drivers*

    *./load_terasic_qsys_pcie_driver.sh*

    A "*Matching device found*" message will be displayed if the drivers have been loaded correctly. Ask your TA if this step fails.

    **NOTE**: you may need to change the permissions of the above file to be executable. Type *chmod +x load_terasic_qsys_pcie_driver.sh* to enable executable permissions.

11. Go back to *linux_app_sample* folder and issue following command to start demo:

    *cd ..*

    *./app -d <imagefilename>*

    **NOTE**: there are two images present in the software folder. "dancing.bmp" and "lena512.bmp"

12. Follow the instructions on the screen and press corresponding buttons on the board if indicated by the program. Read the printed messages carefully. It will give you an idea as to what is happening in real time.
13. To view the output image, issue the following command to copy the output image to your ECN account:
    *scp out.bmp mgXX@ecegrid.ecn.purdue.edu:~/Desktop/*
14. Compare the input and the output image. The output image has been created with the hardware running on the FPGA.

This should give you a feel of what is possible using this board. Building a heterogeneous system is both a challenging and a rewarding experience. We hope this demo gives you an intuition of the design paradigms and possible challenges. We encourage you to use this platform to create applications of your own. Talk to your TA's to know more about how you can bring your ideas to life.