NTNU
Norwegian University of
Science and Technology

Defective Pixel Correction

Henrik Backe-Hansen

Master of Science in Electronics

Submission date:	June 2010
Supervisor:	Einar Johan Aas, IET
Co-supervisor:	Christian Stephansen, Aptina Norway

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Implementing a system for detection and correction of defective pixels onto an FPGA

Assignment given: 05. January 2010
Supervisor: Einar Johan Aas, IET

NTNU
Norwegian University of
Science and Technology

Defect Pixel Correction

by

Henrik Backe-Hansen

Masters Thesis

A thesis submitted in partial fulfillment for the requirement for the degree of Master of Technology

Supervisors: Prof. Einar Johan Aas and Christian Stephansen

June 23, 2010

Norwegian University of Science and Technology

Department of Electronics and Telecommunication

Assignment

This assignment specifies that the candidate must find a method of detecting defective pixels in a data stream from an image sensor and correcting them. All the processing is to take place in hardware and the system should be completed to such a degree that it can function with the constraints imposed when operating in a real time environment. This master's thesis is a continuation of a project [22] with the goal of implementing algorithms for detecting and correcting defective pixels in an image,with the processing done in software. The median, conservative smoothing, gradient and adaptive algorithm were programmed in software and then compared with each other based on their peak signal to noise ratio, mean square error, mean absolute error, simulation time, and survey results.The survey was issued by showing the resulting images from the various algorithms with the same amount of noise added pre correction. The results show a correlation between the statistical values and what the survey showed to be the best image. In other words, the image with the highest levels of the measured values was also the image most participants thought was the best image. It was therefore concluded that the median algorithm produced the best results in terms of measured values and visual inspection, and as, such would be a good choice to implement as part of a master's thesis project.
The median algorithm will be implemented in hardware using a development board with an Xilinx virtex 5 fpga.Once the vhdl programming is complete, an embedded system will store the values from the algorithm onto a flash memory.The data can then be inspected by measuring values and be subject to visual inspection. Assignment was given 05.January 2010.

Supervisors:	Einar Johan Aas, NTNU
	Christian Stephansen, Aptina Norway

Abstract

When using CMOS technology for image sensors, there is a possibility that any given pixel is defective and will thus produce a value that does not correlate to the amount of light it was subject to. As such, the processing unit will calculate a value that differs from the value produced if the transistor was working correctly. Having a pixel with a defective value can manifest itself as a light spot or a dark spot depending on whether the transistor for that pixel is on or off. In some areas where the value of the defective pixel does not differ greatly from its neighbors, the image will not appear as degraded in the eyes of the viewer as if the defective value was in great contrast to its surroundings.The ability to compensate for the defective pixels with an algorithm will result in a more robust device that is not required to function perfectly in order to produce an image. It also translates into profit as a company can sell image sensors that would otherwise have been discarded by testing procedures.

This report is organized with chapter 1 providing the introduction to the assignment in terms of the nature of defective pixels and also creating a context with explanation as to why it is an important aspect of manufacturing image sensors .Chapter 2 describes the development board that is utilized and how an embedded system can utilize a vhdl peripheral. It also shows what components will go into making an embedded system with the required functionality. The theory behind components and techniques used in this project is in chapter 3. The vhdl files to be added to a peripheral so that they can be accessed by the cpu, and the architectures of the vhdl files and microblaze are placed in chapter 4. Chapter 5 contains the simulations of the input images with different noise levels and threshold levels in addition to tests designed to determine the embedded systems functional ability.The vhdl files and the microblaze systems are synthesized with the resulting numbers revealed in chapter 6. The tools used in this project are listed in chapter 7 with their version number. Chapter 8 contains discussions regarding the results and techniques in this project. The concluding remarks and the further work for the project are in chapter 9 and 10, respectively. A list of terms will explain abbreviations used in this report.

Preface

This report along with the system produced are the final products of a project that aimed at implementing a defective pixel correction algorithm with all the processing done in hardware. It also concludes the master's degree with specialization in design of digital systems from the Norwegian University of Science and Technology in Trondheim, Norway. The project spans a wide area with designing both hardware and software to communicate with the hardware. As such, it has been necessary to consult with a few key persons in order to get the project completed, and they all deserve recognition for their contributions. Christian Stephansen at Aptina Norway was instrumental in offering in-depth information regarding the demands on an image sensor. He also gave input to what the end goal of the thesis should be. Professor Einar Johan Aas was the supervisor at the university who was also a part of determining the goal of the thesis along with valuable input as to the organization of the report and serving as a technical consultant. Jan Anders Mathisen was kind enough to lend me the Xilinx evaluation board and point out places to learn about the board that facilitated getting into programming at an early stage. He has also been a technical consultant for helping me problem solve building the microblaze system at times where I was at a stand still. My American relatives Teri Venker and Terry Jacobson graciously opened their home to me and allowed me to live with them in Madison, Wisconsin for the duration of the thesis so I could experience another culture while doing the research.

Madison, Wisconsin, U.S.A. June 23, 2010

Henrik Backe-Hansen
henrik.backe.hansen@gmail.com

Contents

List of Figures

List of Tables

List of Terms

FPGA	Field Programmable Gate Array
FLASH	Memory that can retain data even when not powered
CPU	Central Processing Unit i.e. processor
NTNU	Norwegian University of Science and Technology
CMOS	Complimentary Metal Oxide Semiconductor
MATLAB	Mathematical tool for technical computing
I/O UNIT	Input and Output unit
VHDL	Hardware Descriptive Language
INTERRUPT	Signal to the cpu that an IO unit needs attention
TEXTIO	Vhdl package that enables write to file and read from file
BAYER SENSOR	A way of representing colors from an image sensor
MICROBLAZE	Xilinx soft core cpu on the fpga
BSB	Base system builder to help make embedded design
XPS	Xilinx platform studio, a tool for working with the cpu
RS232	Serial communication protocol
LCD	Liquid Crystal Display
LED	Light Emitting Diode
HARVARD ARCHITECTURE	Separate storage for instruction and data
DDR2 MEMORY	Double Data Rata Synchronous Memory
DSLR	Digital Single Lens Reflex
FPS	Frames per Second
BSB	Base System Builder
PLB	Processor Local Bus
ILMB	Instruction Local Memory Bus
DLMB	Data Local Memory Bus
BRAM,BRAM	Storage elements inside the fpga
DSP	Digital Signal Processing
NETLIST	Physical description of the system
BITFILE	A filetype used to program the fpga
BITSTREAM	The information inside a Bit file
ISR	Interrupt Service Routine
BAUDRATE	Rate at which new symbols arrive on a serial line
PARITY	Error checking mechanism for serial communication
START BIT	Bit that starts a serial communication
STOP BIT	Bit that signifies stop of data
UART	Universal Asynchronous Receive Transmit
FIFO	First in First out stack memory
IP	Intellectual Property
IPIC	Intellectual Property Interconnect
IPIF	Intellectual Property Interface
HEADER FILE	File that contains the method of a c file with the same name
WRAPPER FILE	Vhdl representation of a system component
MS	Milli Second
NS	Nano Second

MHZ	Mega Hertz
HZ	Hertz
KHZ	Kilo Hertz
LUT	Look Up Table
SLICE	Programmable fpga elements that contains LUTs
KB	Kilo Byte
GB	Giga Byte
PGM	Portable Gray Map
CF	Compact Flash
HEXADECIMAL	Number representation with base 16
MHS	Microprocessor Hardware Specification
MSS	Microprocessor Software Specification
UCF	User Constraint File
dB	Decibel
GB	Giga Byte (10^9 byte)
F-stop	1 f-stops is Decreasing / Increasing the amount of light by a factor of two
.vhd	VHDL FILE POSTFIX

Chapter 1

Introduction

This thesis is the continuation of a project that was aimed at creating, comparing and evaluating different methods for correcting defective pixels in an image with all the processing done in software. The projects aims at implementing the best method from the software project into hardware. The assignment is specified by Aptina Norway in conjunction with the Norwegian University of Science and Technology.

1.1 Background

Defective pixels are caused by an error in the fabrication of the image sensor that translates into a transistor being either static on or static off and will thus translate into the cpu interpreting false value from the pixel. The next sections will describe how defective pixels can manifest themselves on an image and variables involved in the detecting and correcting algorithm.

1.1.1 Static Defect

Static defective pixels in an image are types of defects that will contribute towards degrading the image on every exposure and in the same way every time.That means that a defective pixel in position (a,b) in the image will show the same value on every image regardless of the amount of light it is subjected to. Static defective pixels can further be divided into two categories of "always-on" and "always-off" where an always off transistor never has any current flowing through it and with an always on transistor having current flowing through it at all times. In a correctly working pixel, the current that flows through a transistor is correlated with the exposure time of the image. A transistor with no current flowing through it will always be interpreted by the cpu as the value 0, corresponding the white color using an 8- bit gray scale color representation. Having a transistor that has current flowing through it at all times will be interpreted by the cpu as the value 255 using the same 8-bit gray scale color representation and it will result in a black color. The idea behind defective pixel correction is to replace defective pixels with the median of the neighboring pixels of the same color according to the Bayer image filter pattern found in section 3.3.3. The end result will have new pixels in the positions of the defective pixels going into the algorithm. The different correcting algorithms mostly differ in the way they calculate the value of the pixel that was defective.The median algorithm utilizes the value of the nine neighboring pixels and replaces the defective pixel with the median of the neighboring pixels.The task becomes even more complex as it is advantageous to have a new pixel value not being based on values that are themselves defective. An even more complex issue is the demands put on an image senor from a camera application when operating in real time and making sure that the system can cope with the demands.

1.1.2 Pixel Defect

Single defective pixels in an image are situations where the surrounding neighboring pixels are giving off the correct value. Multiple defective pixels, however, can have defective pixels as neighbors, thus complicating the search for working pixels on which to base the new values.

Individual Pixel Defect

With only a single defective pixel in an area, the corrective measure is to assign a new value to the position of the defective pixel based on the nine neighboring pixels of the same color. The algorithm designed adds a two pixel border of the value 127 (for 8 bit image). The border will enable searching for defective pixels at edges and corners all with the same algorithm. The downside to using a border with fixed values is that defective pixels towards corners and edges gets a value that is not based on the values in the vicinity but rather a fixed value, which can lead to a somewhat strange color for that pixel.

Multiple Pixel Defects

An image exposed to high levels of noise is prone to have situations where a defective pixel has a defective pixel as its neighbor of the same color. The adaptiveness of the proposed algorithm will detect this and treat it accordingly by ensuring that no defective values are used in the calculation of a pixel.That is accomplished by having the detection and correction done in separate steps by searching through the image and marking all the positions that are defective. The correction algorithm can then correct values based for every position that is defective while still not utilizing values that are flagged as defective. In the case of a defective value being a target for calculation of another defective pixel, the first defective is replaced by the number 127, which is the middle value for 8-bit values.

1.1.3 Programmable Threshold

In order to classify any pixel as defective they need to be compared with their neighbors of the same color.The detection algorithm, therefore, needs to know the highest and lowest value of the neighboring pixels, excluding the value of the pixel in question, in order to ascertain whether or not the position should be classified as defective. The threshold [30] is the value that is added to the highest value and subtracted from the lowest value in order to produce an acceptance range of values that are considered to be not defective. Having a value falling outside this range will require the pixels position to be classified as defective. Different images may require different threshold in order to produce the best result. That is why the threshold variable can be altered in the vhdl file package. An image with high contrast has large differences between closely related pixels without the pixels being defective while a low contrasting image does not have such large differences.The programmable threshold is designed to account for different images being processed with different thresholds.

1.2 Motivation

Being able to correct defective pixels from an image sensor with defective values has the benefit of being able to sell image sensors with defective values that would otherwise have been rejected by test procedures. It will also enable the correction of defects that occur only after some time and so make a more robust device as far as errors are concerned.

1.3 Objective

This project aims at producing a system that can detect and correct defective pixel values from an image with all the signal processing is done in hardware. A software system is required to read values produced by the processing unit. The resulting image will have to be interpreted visually by a human eye and by the means of statistical values in order to ascertain the algorithm's ability to correct pixels. The resulting image will be a more esthetically pleasing version of the noise degraded input image with fewer visual pixel defects.

Chapter 2

System

2.1 Xilinx ML501

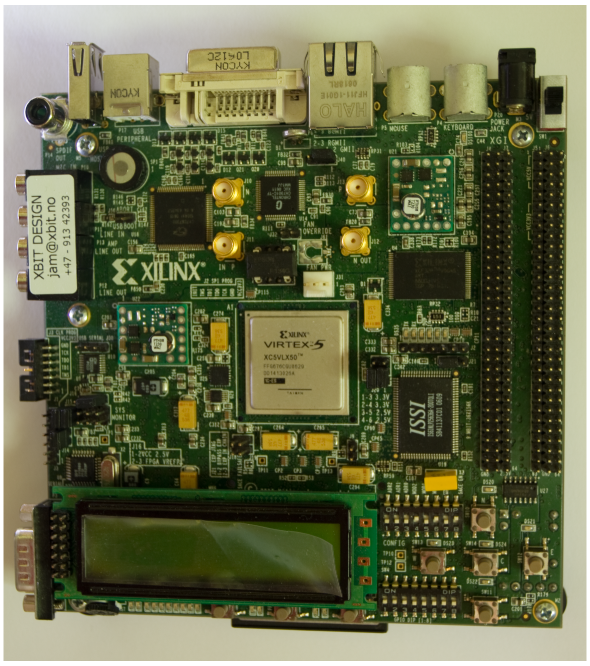


Figure 2.1: Xilinx ML501 development board

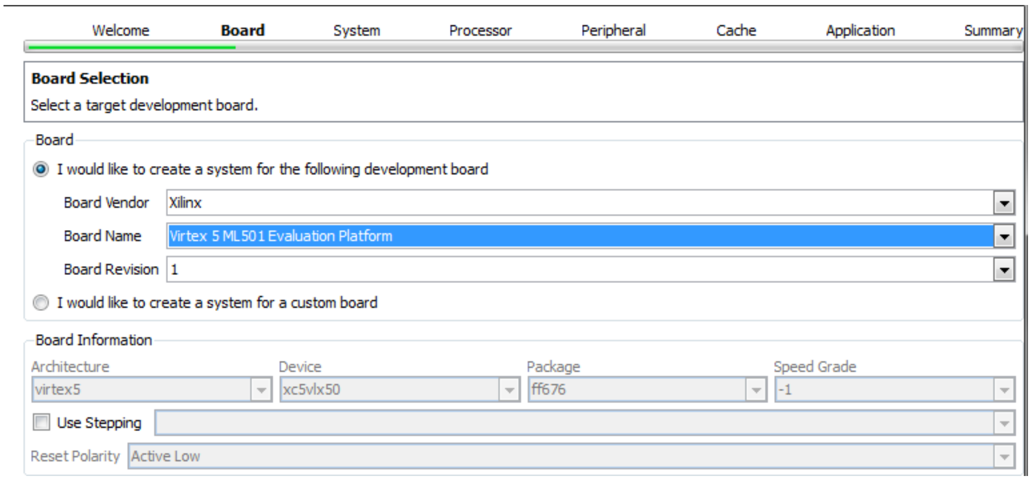
Xilinx ML501 is an evaluation board with a vertex 5 fpga and a soft core microblaze processor. It has built in DDR2 memory and system-ace to communicate with the compact flash card, and serial communication using the rs232 protocol. It has an LCD display and pushbuttons and LED lights as general purpose I/O. It also has input and output capabilities as far as audio and video are concerned. This project merely utilized a few of its features.The name of the physical pins on the board are specified in [26].

2.2 Microblaze

Microblaze is a soft core cpu using Harvard Architecture on the evaluation board. It contains separate data local memory bus (dlmb) and instruction local memory bus (ilmb) in addition to a processor local bus (plb) for connecting the cpu to peripheral components. The system can be customized by adding peripheral units with different functionality in order to tailor a system to specific needs. The cpu model used in the system includes some peripheral units like the system ace controller, vhdl design, rs232 and interrupt controller. The system ace controller is an interface for communication with the compact flash card using read and write commands.The vhdl peripheral contains the vhdl files that actually do the correction processing of defective pixels. Those values are then stored in a register that is accessible by software functions.The next section will explain the methods in the system ace interface and the storing of values from the vhdl peripheral. The interrupt controller relays any interrupt from a peripheral to the cpu so it can execute its interrupt service routine. The rs232 is the default standard input and output for the processor, but it will not be utilized in this project as a debugging tool.

2.2.1 Creating an embedded system with platform studio

The Xilinx platform studio has a feature called the base system builder wizard that greatly simplifies the task of creating an embedded system to interact with the fpga as it asks for the different parameters in the system and then produces templates and drivers for the system [31, 32].The wizard will ask where the project is to be placed on the disk and the name of the system. Based on those inputs it creates a folder with all the necessary files for the project.The various postfixes for the files in project directory can be found in [20]. The wizard then asks if the designer would like to specify a custom board or to use an already made development board in the list. The project used a Xilinx ml501 as is evident in figure 2.2.



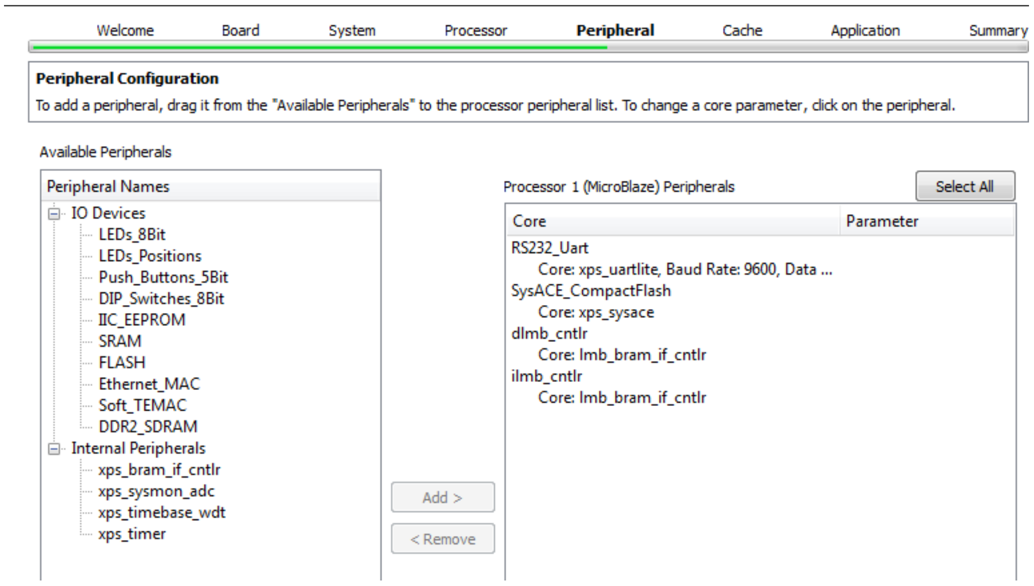
Board Selection
Select a target development board.

Board
(●) I would like to create a system for the following development board
Board Vendor: Xilinx
Board Name: Virtex 5 ML501 Evaluation Platform
Board Revision: 1
( ) I would like to create a system for a custom board

Board Information
Architecture: virtex5
Device: xc5vlx50
Package: ff676
Speed Grade: -1
[ ] Use Stepping
Reset Polarity: Active Low

Figure 2.2: Choosing a board

The wizard will offer the option to make the processor as a stand alone system or as one of many processors in a system. The next step in the wizard is specifying the clock frequency at which the cpu is to operate.There may be some limitation for the frequency for certain peripherals as they may need a certain frequency. Here, the option to add some memory to the processor is also available that can be turned into cache memory for the processor.This project will stay with the default 125 MHz frequency and 8 KB of local memory for instructions and data.



Peripheral Configuration

To add a peripheral, drag it from the "Available Peripherals" to the processor peripheral list. To change a core parameter, click on the peripheral.

Available Peripherals

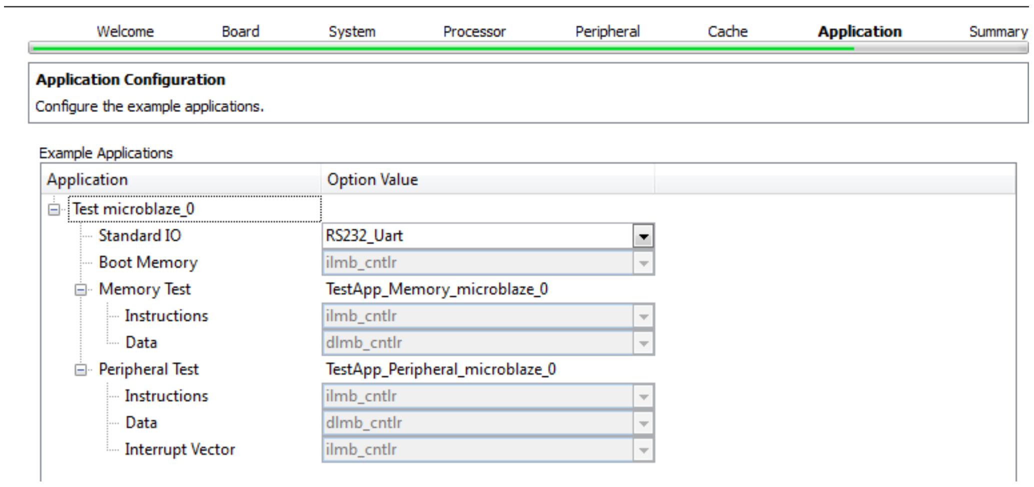
Peripheral Names
- IO Devices
  - LEDs_8Bit
  - LEDs_Positions
  - Push_Buttons_5Bit
  - DIP_Switches_8Bit
  - IIC_EEPROM
  - SRAM
  - FLASH
  - Ethernet_MAC
  - Soft_TEMAC
  - DDR2_SDRAM
- Internal Peripherals
  - xps_bram_if_cntlr
  - xps_sysmon_adc
  - xps_timebase_wdt
  - xps_timer

Processor 1 (MicroBlaze) Peripherals

Core	Parameter
RS232_Uart	
Core: xps_uartlite, Baud Rate: 9600, Data ...	
SysACE_CompactFlash	
Core: xps_sysace	
dlmb_cntlr	
Core: lmb_bram_if_cntlr	
ilmb_cntlr	
Core: lmb_bram_if_cntlr	

Figure 2.3: Choosing peripherals

Figure 2.3 shows how different I/O units can be added to the plb bus and also internal peripherals can be added to the cpu.The external I/O devices communicate with recourses on the board like the LED and the push buttons, in addition to resources off the chip like ethernet, and rs232. The application screen shown in figure 2.4 lets the designer choose what memory to run the built in tests off of and also the standard input and output for the system. This project does not have any external memory, so all instructions for the cpu are stored in local block ram inside the fpga.The RS232 serial connection is default as the standard input and output. It will remain in the system to fulfill its duties but it will not be used for printing status messages as the computer used does not have an RS232 serial port.



Application Configuration

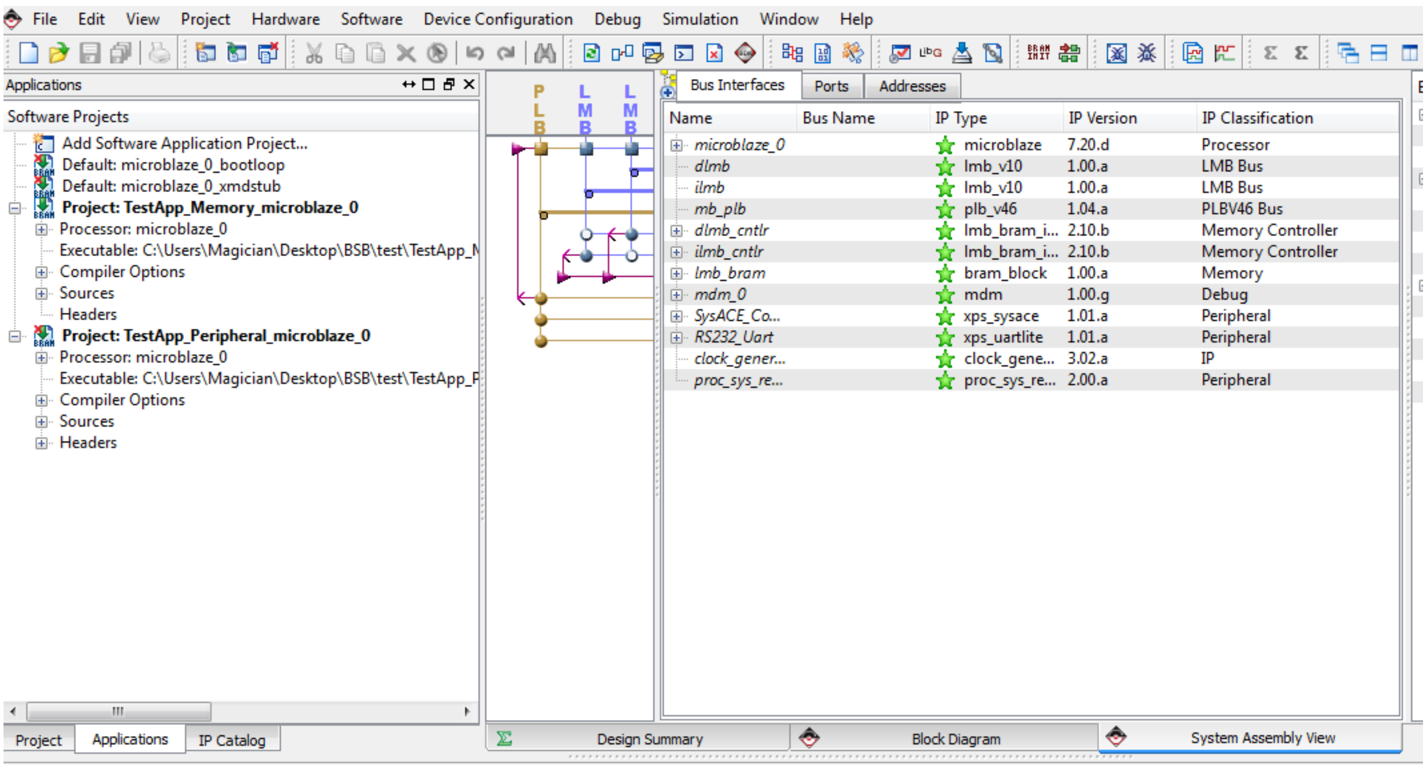
Configure the example applications.

Example Applications

Application	Option Value
Test microblaze_0	
Standard IO	RS232_Uart
Boot Memory	ilmb_cntlr
Memory Test	TestApp_Memory_microblaze_0
Instructions	ilmb_cntlr
Data	dlmb_cntlr
Peripheral Test	TestApp_Peripheral_microblaze_0
Instructions	ilmb_cntlr
Data	dlmb_cntlr
Interrupt Vector	ilmb_cntlr

Figure 2.4: Applications

The wizard is now done with all the information it needed to set up the system and after some waiting time the screen will show what is in figure 2.5, which is the home screen of the system with information on the hardware connection as well as the software files.The applications tab shows the two default projects TestApp_Memory and TestApp_Perppherial that were created by the wizard. These two projects can be used to run a test on the system so to ensure that it the hardware can communicate with the software correctly for both the memory and the connected peripheral units. The "Ports" tab in the figure shows all the internal and external ports associated with the cpu's address space while the "Addresses" tab shows the address range of all the peripherals connected plb bus.



Software Projects
- Add Software Application Project...
- Default: microblaze_0_bootloop
- Default: microblaze_0_xmdstub
- **Project: TestApp_Memory_microblaze_0**
  - Processor: microblaze_0
  - Executable: C:\Users\Magician\Desktop\BSB\test\TestApp_M
  - Compiler Options
  - Sources
  - Headers
- **Project: TestApp_Peripheral_microblaze_0**
  - Processor: microblaze_0
  - Executable: C:\Users\Magician\Desktop\BSB\test\TestApp_P
  - Compiler Options
  - Sources
  - Headers

Name	Bus Name	IP Type	IP Version	IP Classification
microblaze_0		microblaze	7.20.d	Processor
dlmb		lmb_v10	1.00.a	LMB Bus
ilmb		lmb_v10	1.00.a	LMB Bus
mb_plb		plb_v46	1.04.a	PLBV46 Bus
dlmb_cntlr		lmb_bram_i...	2.10.b	Memory Controller
ilmb_cntlr		lmb_bram_i...	2.10.b	Memory Controller
lmb_bram		bram_block	1.00.a	Memory
mdm_0		mdm	1.00.g	Debug
SysACE_Co...		xps_sysace	1.01.a	Peripheral
RS232_Uart		xps_uartlite	1.01.a	Peripheral
clock_gener...		clock_gene...	3.02.a	IP
proc_sys_re...		proc_sys_re...	2.00.a	Peripheral

Figure 2.5: Base System Builder Complete

It is now time to download the two created test projects to the fpga, one by one, to make sure that the cpu can access the connected peripherals. The first step is to go to the software tab in the main window to "generate libraries and dsp". That will create the libraries with functions that are associated with the peripherals in the system. It is then time to make this project the default project for initializing the bram on the fpga, which means that the data in this project will be placed in the block ram for instructions so that the cpu will execute those instruction when it is turned on. The software source code has to be compiled with the "build project" command.The result of compiling the source is an executable file. The "generate netlist" command reads the executable file and generates an vhdl description of the components in the system and then runs synthesis on those files while the "generate bitstream" takes the netlist files and creates a bit-file that contains the entire embedded system.The bit file is the same format that is used to program an fpga.The software source files needs to be injected into the bitstream in order to get the cpu to execute those instructions. The bit file is complete when the software has been injected into the bit file, and can be downloaded to the fpga with the Impact tool.The Impact tool can program the fpga by creating an ace file or by using a cable to the fpga. For this project it will create an ace file from the bit file which in turn is copied into the compact flash card at address 6. All that is left is to insert the compact flash card into the card reader on the board, power it up, and select the ace file at position 6 as the source file for the cpu. Running the TestApp_peripheral will do a test of the connected peripherals e.g. printing strings to a computer connected to the rs232 serial port or flash the LEDs if they are connected.

2.2.2 System Ace Controller

The system ace controller is an interface between the compact flash card and the processor local bus. It makes sure that the plb bus and the cf card can communicate and that the timing is correct.The controller and all its specifications are given in [29]. The library generated for the system ace controller contains functions to read from the device and write to the device. There is, however, a library called xilinx xilfatfs [28] that further simplifies the interaction with the cf card by the means of functions to open, read, write and close a file. The functions available using the Xilfatfs library, for interacting with the compact flash card, are:

sysace_fopen	(file, mode)
sysace_fread	(buffer,size,count,file)
sysace_fwrite	(buffer, size, count,file)
sysace_fclose	(file)
sysace_mkdir	(path)
sysace_chdir	(path)

Interaction with the cf card requires that certain steps are done in the correct order. The first step is to make a directory where the file should be located and then open the file in read or write mode, depending on what is the desired operation. The directory is specified using the sysace_mkdir command while opening a file in write mode is done with the sysace_fopen(file, 'w') function. It is now ready for communication with the cf card using the sysace_fwrite function (assuming that the file was opened in write mode). Upon completion of writing the data, the file has to be closed using the sysace_fclose function. The function sysace_fread is used when the target file is opened in read mode, and the sysace_chdir is a function to change a working directory.

2.2.3 RS232

The rs232 protocol is a communication protocol for serial communication with one transmitter and one receiver. The signals used in the communication can be seen in [24] along with the parameters used to setup communication between two participants. Platform studio offers a version of the rs232 called uart lite that only contains the receive and transmit pin as external pins from the chip.This has the added benefit of not requiring knowledge of the entire rs-232 protocol in order to initiate communication. Uart lite is also the default std_in and std_out for the cpu. This module can be used for printing status messages to a connected computer so it is in fact a debugging tool for the software platform.A serial communication contains several parameters that must have the same values on the transmitter and the receiver side for the receiver to decode the message correctly. Parameters that need to be specified are the stop bit, start bit, baud rate and parity. The start bit is a message to the receiver that the first bit of the message is being transfered.This is based on the protocol being asynchronous and therefore used the start bit to synchronize with the transmitter. The start bit will therefore synchronize receiver and transmitter. The stop bit denotes the end of the message and acts as a resynchronizations tool if the receiver, for whatever reason, did not receive the start bit and therefore not the message. The parity bit is an error checking tool where the transmitter generates a bit value that the receiver can use to determine whether it received the same message that was sent.

2.2.4 Interrupts

There are two basic methods for a peripheral to let the processor know that it needs attention;polling and interrupts. Polling is a method where the processor will ask the various peripherals if they have any data for the processor. It will ask each peripheral every time period. This method can work great for systems that only operate on data from peripherals, but will not be such a great choice if the data from the peripherals is not essential for the processing at all times.This system will only act on the data from the interrupts, but it was chosen to use an interrupt controller in order to gain experience with setting up and handling interrupts. Xps offers an interrupt controller that relays interrupts from the peripherals to the cpu. It is connected to the cpu's interrupt port, on one side, and the interrupt signal from the peripheral on the other side. Interrupt based systems are methods where the peripherals will tell the processor when it needs attention and thus only take up processing time when it needs to communicate with the cpu. Good tutorials for creating interrupt based systems can be found in [14, 27].
The system implemented for this project utilizes an interrupt based processor so that it receives an interrupt from the vhdl peripheral that it wants attention, and the processor will go into its interrupt service routine (isr) to execute the instructions for that interrupt.When an interrupt "fires", the processor will store its current values and go into its isr. When the processor completes its isr, it will load the values it had prior to the interrupt as to continue execution where it left off before the interrupt.

Interrupt Controller

The vhdl peripheral will set the interrupt flag of the interrupt controller when it needs to communicate with the processor. One interrupt line can be directly connected to the processor's interrupt port, but having multiple interrupt lines requires the implementation of an interrupt controller that can act as an arbiter to make sure every interrupt gets serviced. It will act as a decoder for interrupt lines for various peripherals as the processor has only one interrupt port.The interrupt controller can relay information to the cpu of the peripheral that triggered an interrupt so the correct isr is executed. When a peripheral sets the interrupt flag high, that will signal to the interrupt controller that it wants to communicate with the processor and the interrupt controller in turn sets the processor's interrupt line high.This is the message to the cpu to go into the isr of the source of the interrupt. The specifications of the interrupt controller can be found in [33].

Interrupt enabling

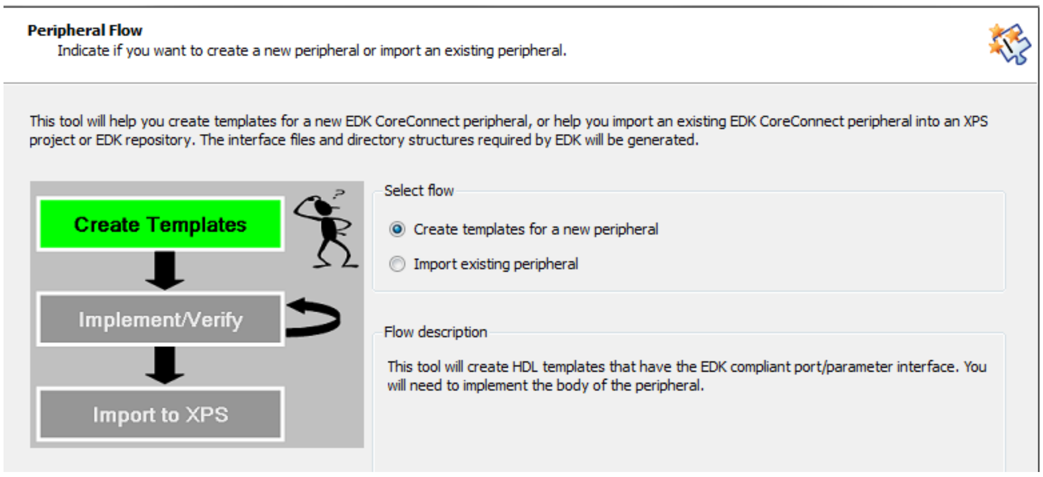
When setting up interrupts on a processor it is essential that they are set up correctly on all devices that set, handle and relay the interrupt. It needs to be enabled on the peripheral, interrupt controller and in microblaze in this situation. It is also important that the interrupts are acknowledged on the peripheral and the interrupt controller before de asserting the processor interrupt line to low.This is to avoid a situation where an interrupt is never acknowledged and is therefore blocking all other instruction to the cpu as the cpu will think the peripheral is asking for a new interrupt and go into its isr again.

Interrupt service routine

Isr is the instructions that are executed every time a given peripheral triggers an interrupt. Different peripherals have different isr that are executed. It is therefore imperative that the source of the interrupt is found so as to ensure the correct isr is being executed. On this project the vhdl peripheral will assert the interrupt signal on the bus and then it is sent to the interrupt controller that will relay the interrupt to the processor.

2.2.5 Creating an intellectual property

Xps has a wizard for creating peripherals that can be connected to the cpu in such a manner that the designer does not need to know the specifics of the bus protocol in order to access data on a peripheral but can rather communicate on the bus using a few signals.Tutorials on how to create an custom IP and utilize its features can be found at [1, 25]. Figure 2.6 shows the start of the guide where the designer can choose to create a template for a custom peripheral or import an already-made peripheral. Choosing to create a template results in the wizard creating the vhdl wrapper files that enable communication between the custom ip functionality and the plb bus. Once a peripheral is created and the default vhdl files created by the wizard altered, the peripheral has to be imported back into xps with the new changes in order to get a custom ip with the desired functionality.


Peripheral Flow
Indicate if you want to create a new peripheral or import an existing peripheral.

This tool will help you create templates for a new EDK CoreConnect peripheral, or help you import an existing EDK CoreConnect peripheral into an XPS project or EDK repository. The interface files and directory structures required by EDK will be generated.

Select flow

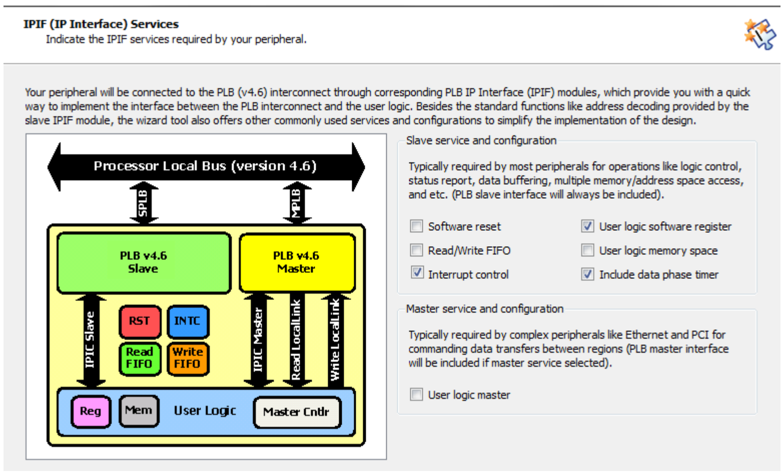
[● Create templates for a new peripheral]
○ Import existing peripheral

Flow description

This tool will create HDL templates that have the EDK compliant port/parameter interface. You will need to implement the body of the peripheral.

Figure 2.6: Create Custom Peripheral guide step 1

The ipif screen shown in figure 2.7 is where the designer will choose what modules to be implemented into the peripheral. The ability to have read and write fifo, interrupts, memory space or register space can all be specified in the ipif section.This project will utilize an interrupt mechanism for the peripheral to communicate to the cpu that it needs attention and it will include the default checked boxes in the "user logic software register" and "include data phase timer".


IPIF (IP Interface) Services
Indicate the IPIF services required by your peripheral.

Your peripheral will be connected to the PLB (v4.6) interconnect through corresponding PLB IP Interface (IPIF) modules, which provide you with a quick way to implement the interface between the PLB interconnect and the user logic. Besides the standard functions like address decoding provided by the slave IPIF module, the wizard tool also offers other commonly used services and configurations to simplify the implementation of the design.

Slave service and configuration

Typically required by most peripherals for operations like logic control, status report, data buffering, multiple memory/address space access, and etc. (PLB slave interface will always be included).

☐ Software reset
☑ User logic software register
☐ Read/Write FIFO
☐ User logic memory space
☑ Interrupt control
☑ Include data phase timer

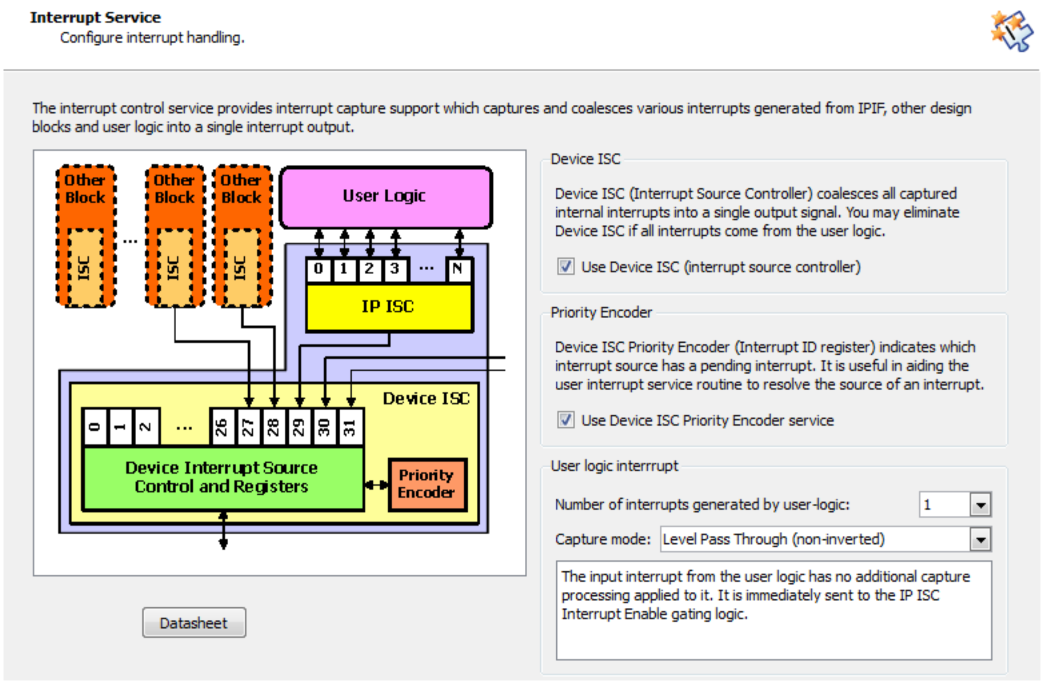
Master service and configuration

Typically required by complex peripherals like Ethernet and PCI for commanding data transfers between regions (PLB master interface will be included if master service selected).

☐ User logic master

Figure 2.7: Peripheral interface

The user logic software register was chosen by the designer so a determination has to be made as to how many software accessible registers the peripheral should contain. These registers are named slv_reg0–slv_reg(N-1) for N specified registers. It can been seen from the vhdl file user_logic that it specifies how to read and write to any of the accessible registers.


Interrupt Service
Configure interrupt handling.

The interrupt control service provides interrupt capture support which captures and coalesces various interrupts generated from IPIF, other design blocks and user logic into a single interrupt output.

Datasheet

Device ISC

Device ISC (Interrupt Source Controller) coalesces all captured internal interrupts into a single output signal. You may eliminate Device ISC if all interrupts come from the user logic.

[✓] Use Device ISC (interrupt source controller)

Priority Encoder

Device ISC Priority Encoder (Interrupt ID register) indicates which interrupt source has a pending interrupt. It is useful in aiding the user interrupt service routine to resolve the source of an interrupt.

[✓] Use Device ISC Priority Encoder service

User logic interrrupt

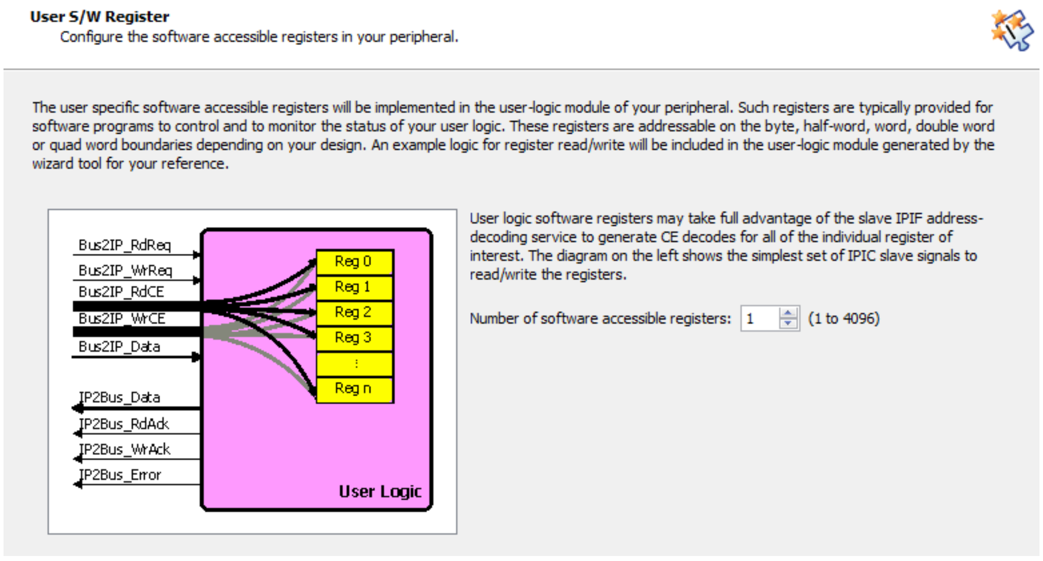
Number of interrupts generated by user-logic: 1

Capture mode: Level Pass Through (non-inverted)

The input interrupt from the user logic has no additional capture processing applied to it. It is immediately sent to the IP ISC Interrupt Enable gating logic.

Figure 2.8: Interupt service

The next step is specifying the number of interrupts needed in the system and also if the interrupts should be captured as inverted, low level or high level. The wizard will create signals in the user_logic file that is connected to the interrupt signal on the plb bus. The interrupt mechanism can therefore be accessed by assigning a signal within the vhdl module. By opting for the user logic register in figure 2.7, the number of registers is specified in figure 2.9. The number of registers chosen can be seen in the user_logic module with each register having a signal assigned, in order for functions in software to read from the registers and write to the registers.


User S/W Register
Configure the software accessible registers in your peripheral.

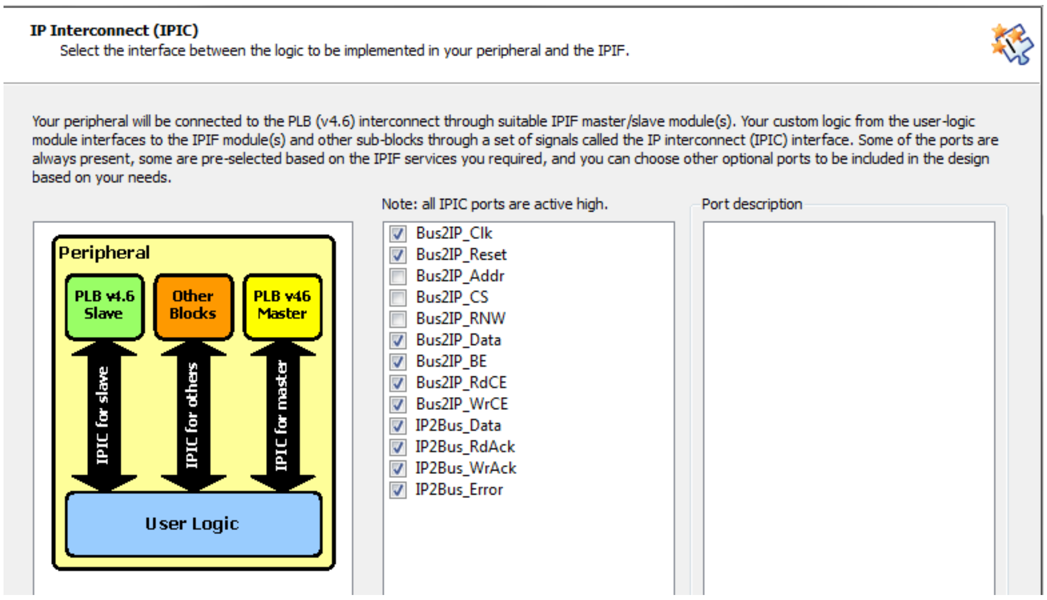
The user specific software accessible registers will be implemented in the user-logic module of your peripheral. Such registers are typically provided for software programs to control and to monitor the status of your user logic. These registers are addressable on the byte, half-word, word, double word or quad word boundaries depending on your design. An example logic for register read/write will be included in the user-logic module generated by the wizard tool for your reference.

User logic software registers may take full advantage of the slave IPIF address-decoding service to generate CE decodes for all of the individual register of interest. The diagram on the left shows the simplest set of IPIC slave signals to read/write the registers.

Number of software accessible registers: 1 (1 to 4096)

Figure 2.9: Peripheral registers

The ipic is the tab where all the signals from the ipif module to the user_logic module are specified, also with the option of checking a new signal that the designer may opt to use. The signals in the tab are shown in figure 2.10.These signals represent a simplified version of the plb bus signals. The signals will be utilized in the peripheral in order to communicate with the ipif that in turn communicates with the plb bus.



IP Interconnect (IPIC)
Select the interface between the logic to be implemented in your peripheral and the IPIF.

Your peripheral will be connected to the PLB (v4.6) interconnect through suitable IPIF master/slave module(s). Your custom logic from the user-logic module interfaces to the IPIF module(s) and other sub-blocks through a set of signals called the IP interconnect (IPIC) interface. Some of the ports are always present, some are pre-selected based on the IPIF services you required, and you can choose other optional ports to be included in the design based on your needs.

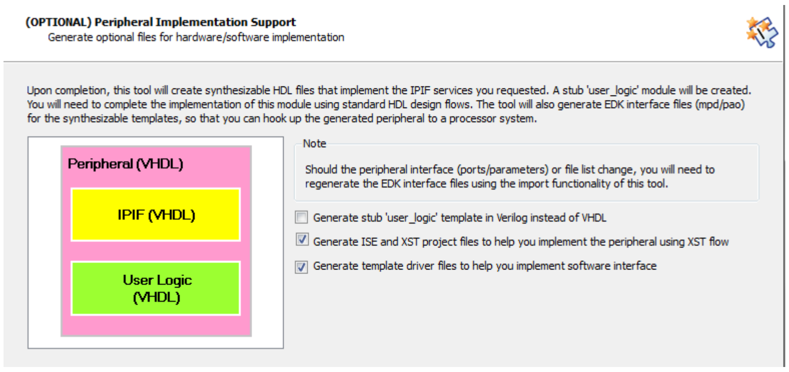
Note: all IPIC ports are active high.

- [✓] Bus2IP_Clk
- [✓] Bus2IP_Reset
- [ ] Bus2IP_Addr
- [ ] Bus2IP_CS
- [ ] Bus2IP_RNW
- [✓] Bus2IP_Data
- [✓] Bus2IP_BE
- [✓] Bus2IP_RdCE
- [✓] Bus2IP_WrCE
- [✓] IP2Bus_Data
- [✓] IP2Bus_RdAck
- [✓] IP2Bus_WrAck
- [✓] IP2Bus_Error

Port description

Figure 2.10: Peripheral Interconnect

The hierarchy of the custom peripheral is shown in figure 2.11 with the peripheral being the top level module with the ipif module on the level below and the user_logic module on the lowest level.There are options for the wizard to create ISE project files for the peripheral and also to make templates for the driver of the peripheral. The created ISE project facilitates easier importation into platform studio as the files can be synthesized prior to being imported.



(OPTIONAL) Peripheral Implementation Support
Generate optional files for hardware/software implementation

Upon completion, this tool will create synthesizable HDL files that implement the IPIF services you requested. A stub 'user_logic' module will be created. You will need to complete the implementation of this module using standard HDL design flows. The tool will also generate EDK interface files (mpd/pao) for the synthesizable templates, so that you can hook up the generated peripheral to a processor system.

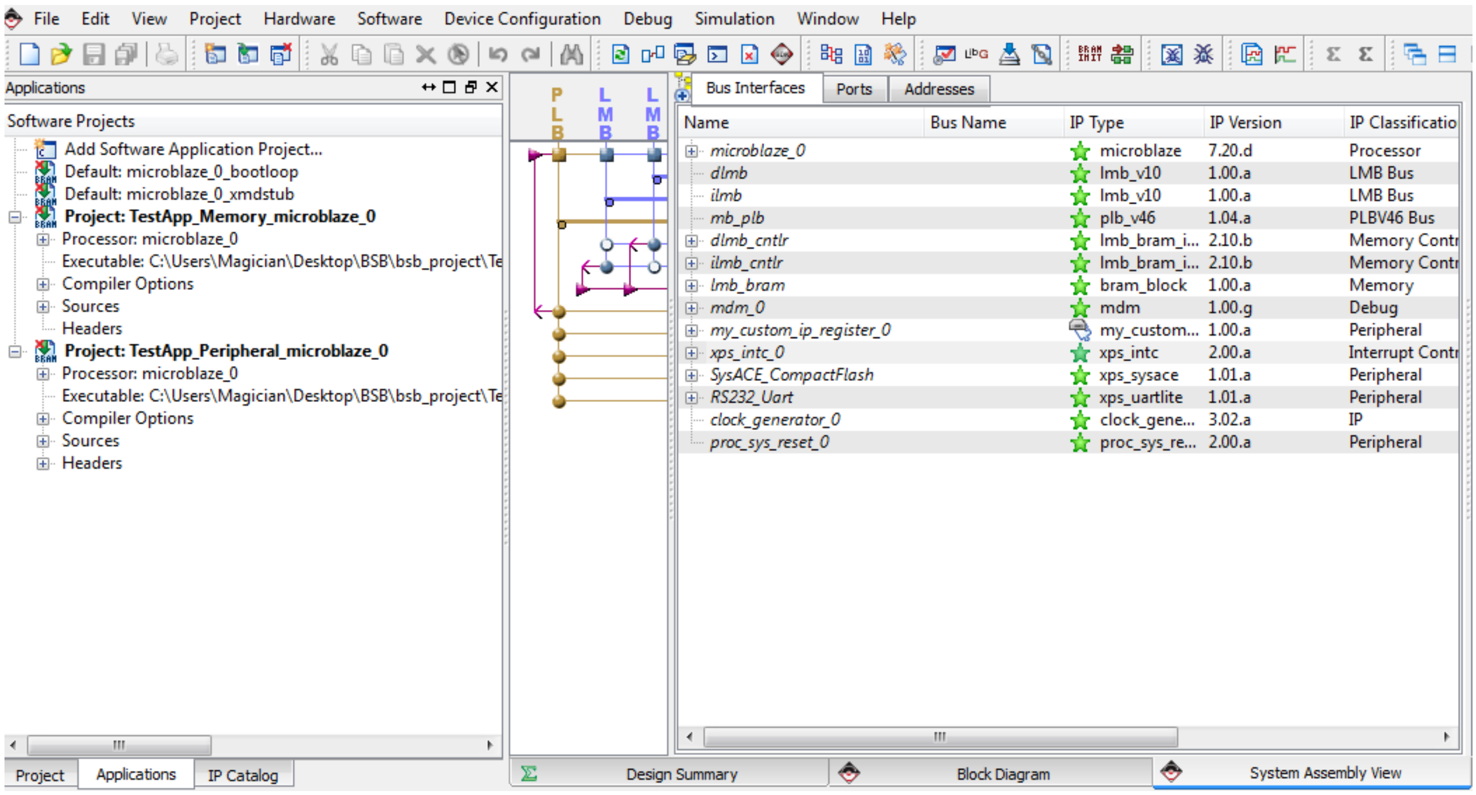
Note

Should the peripheral interface (ports/parameters) or file list change, you will need to regenerate the EDK interface files using the import functionality of this tool.

- [ ] Generate stub 'user_logic' template in Verilog instead of VHDL
- [✓] Generate ISE and XST project files to help you implement the peripheral using XST flow
- [✓] Generate template driver files to help you implement software interface

Figure 2.11: Peripheral hierarchy

The custom peripheral wizard has now created the two vhdl files as a wrapper for the peripheral. The files are called my_custom_ip_register.vhd and user_logic.vhd where my_custom_ip_register.vhd is the wrapper for the ipif while the user_logic is the file where the designer can add functionality given the set of signals shown in figure 2.10. These files are located in the pcores folder in the working directory. The folder contains all custom ip modules created by the wizard.



Software Projects
- Add Software Application Project...
- Default: microblaze_0_bootloop
- Default: microblaze_0_xmdstub
- **Project: TestApp_Memory_microblaze_0**
  - Processor: microblaze_0
  - Executable: C:\Users\Magician\Desktop\BSB\bsb_project\Te
  - Compiler Options
  - Sources
  - Headers
- **Project: TestApp_Peripheral_microblaze_0**
  - Processor: microblaze_0
  - Executable: C:\Users\Magician\Desktop\BSB\bsb_project\Te
  - Compiler Options
  - Sources
  - Headers

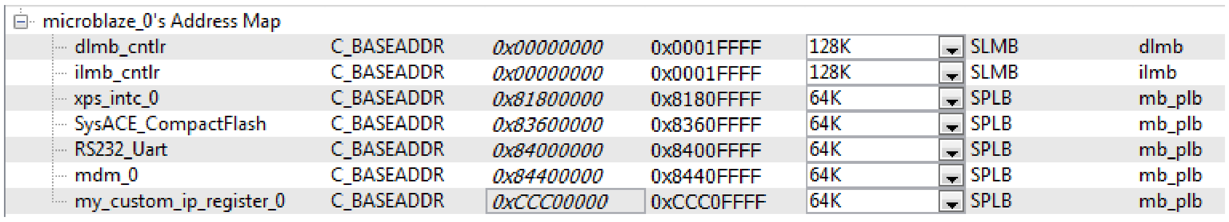
Name	Bus Name	IP Type	IP Version	IP Classificatio
microblaze_0		microblaze	7.20.d	Processor
dlmb		lmb_v10	1.00.a	LMB Bus
ilmb		lmb_v10	1.00.a	LMB Bus
mb_plb		plb_v46	1.04.a	PLBV46 Bus
dlmb_cntlr		lmb_bram_i...	2.10.b	Memory Contr
ilmb_cntlr		lmb_bram_i...	2.10.b	Memory Contr
lmb_bram		bram_block	1.00.a	Memory
mdm_0		mdm	1.00.g	Debug
my_custom_ip_register_0		my_custom...	1.00.a	Peripheral
xps_intc_0		xps_intc	2.00.a	Interrupt Contr
SysACE_CompactFlash		xps_sysace	1.01.a	Peripheral
RS232_Uart		xps_uartlite	1.01.a	Peripheral
clock_generator_0		clock_gene...	3.02.a	IP
proc_sys_reset_0		proc_sys_re...	2.00.a	Peripheral

Figure 2.12: Import peripheral complete

The newly created custom ip is located in the IP catalog under user cores and must now be added to the project and have a connection to the plb bus established. The only step left then is to go to the address tab in order to assign an address range for the peripheral so that the processor can communicate with it. The hardware part of connecting the peripheral is now complete but the designer still needs to run the "generate libraries and dsp" command from the software tab in order to create the new libraries with the functions that are associated with the custom peripheral. When the libraries are generated the software will recognize the functions associated with the peripheral.

2.2.6 Access Peripherals

The cpu has a memory map that indicates which addresses belong to which peripheral. The memory map is a division of the total memory of the cpu divided by the different peripherals and their memory needs so that every unit can be accessed by using those addresses.The various peripherals may need different space allocated in the memory so they will thus have narrower address ranges. The Xilinx Platform Studio makes this memory map an automated feature so the user does not need to specify anything other than the memory needed for each peripheral. There are two variables that are important when a peripheral is to be accessed; the base address and the high address. The base address is the starting address for the peripheral and the high address is the end address of the peripheral. Both addresses are given in hexadecimal numbers.The example given in figure 2.13 illustrates the different peripherals in the system along with their base address, high address, memory size and connection to the bus. The modules all have header files that describe the methods that are associated with that peripheral and also the offset to the different registers within a peripheral. Within the custom peripheral there are methods for reading from a register, writing to a register and setting up interrupts.



microblaze_0's Address Map						
dlmb_cntlr	C_BASEADDR	0x00000000	0x0001FFFF	128K	SLMB	dlmb
ilmb_cntlr	C_BASEADDR	0x00000000	0x0001FFFF	128K	SLMB	ilmb
xps_intc_0	C_BASEADDR	0x81800000	0x8180FFFF	64K	SPLB	mb_plb
SysACE_CompactFlash	C_BASEADDR	0x83600000	0x8360FFFF	64K	SPLB	mb_plb
RS232_Uart	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb_plb
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb
my_custom_ip_register_0	C_BASEADDR	0xCCC00000	0xCCC0FFFF	64K	SPLB	mb_plb

Figure 2.13: Microblaze address map

The cpu can access the system ace controller using the address range from base address to high address. The size of the address space can be determined by subtracting the base address from the high address. As an example the high address of the system ace controller is 0x8360FFFF and the base address is 0x83600000. Subtracting the base address from the high address leaves us with the address space in bytes. The address space of the system ace controller is 65536 or 64k.The amount of memory is also shown in the address map in order to simplify the calculation of the amount of memory for a given peripheral.The header file of a given peripheral specifies what the different addresses of the peripheral will accomplish.

2.2.7 Adding c source file

Creating a new c source file for a project is done by clicking on the sources tab and the select "add existing file or add new file". This will create a new file to one of the two processes created by the wizard.The c code will be executed by the cpu if the process it belongs to is set to initialize the block ram.The file is accessible from the "sources" menu in that process. Utilizing functions from a peripheral requires that its header file is included at the start of the c file and also added to the "header" menu in the process.

Chapter 3

Supporting Theory

This chapter will contain the information needed to understand the techniques used in the project, and evaluate the results of the algorithm for detecting and correcting defective pixels in an image.

3.1 Median Filter

This thesis employs a median algorithm for correcting pixels that are classified as defective. The algorithm will then gather the values of the eight neighbors of the same color, and assign the median of those values as the new value at that position. Figure 3.1 shows some values that will be used to explain the operating procedure of the median filter. Let's assume that the threshold for detection is 10. That means that any value higher than threshold + the highest value, or lower than threshold - lowest value is classified as defective.

$$\begin{bmatrix}
 & & \vdots & \vdots & \vdots & \vdots & \vdots & & \\
\dots & \dots & 114 & 115 & 102 & 92 & 72 & \dots & \dots \\
\dots & \dots & 140 & 115 & 128 & 111 & 121 & \dots & \dots \\
\dots & \dots & 100 & 101 & 127 & 103 & 113 & \dots & \dots \\
\dots & \dots & 141 & 201 & 90 & 131 & 61 & \dots & \dots \\
\dots & \dots & 117 & 119 & 99 & 97 & 120 & \dots & \dots \\
 & & \vdots & \vdots & \vdots & \vdots & \vdots & &
\end{bmatrix}$$

Figure 3.1: Truncated Original Image Values

The first step with this algorithm is to calculate the maximum and minimum values not including the pixel in question.The values chosen for the middle pixel in ascending order are {72 99 100 102 113 114 117 120 127}. The extreme values, not including the middle pixel, are 72 and 120. The middle pixel will be classified as defective when using a threshold larger than 7 and classified as a working pixel with a threshold lower than 7. In this instance it is assumed that the threshold is such that the middle pixel is classified as defective. Next step in to replace the defective value with the median of the neighboring pixels of the same color. In this example the median is 113.

$$\begin{bmatrix} & & \vdots & \vdots & \vdots & \vdots & \vdots & & \\ \ldots & \ldots & 114 & 115 & 102 & 92 & 72 & \ldots & \ldots \\ \ldots & \ldots & 140 & 115 & 128 & 111 & 121 & \ldots & \ldots \\ \ldots & \ldots & 100 & 101 & 113 & 103 & 113 & \ldots & \ldots \\ \ldots & \ldots & 141 & 201 & 90 & 131 & 61 & \ldots & \ldots \\ \ldots & \ldots & 117 & 119 & 99 & 97 & 120 & \ldots & \ldots \\ & & \vdots & \vdots & \vdots & \vdots & \vdots & & \end{bmatrix}$$

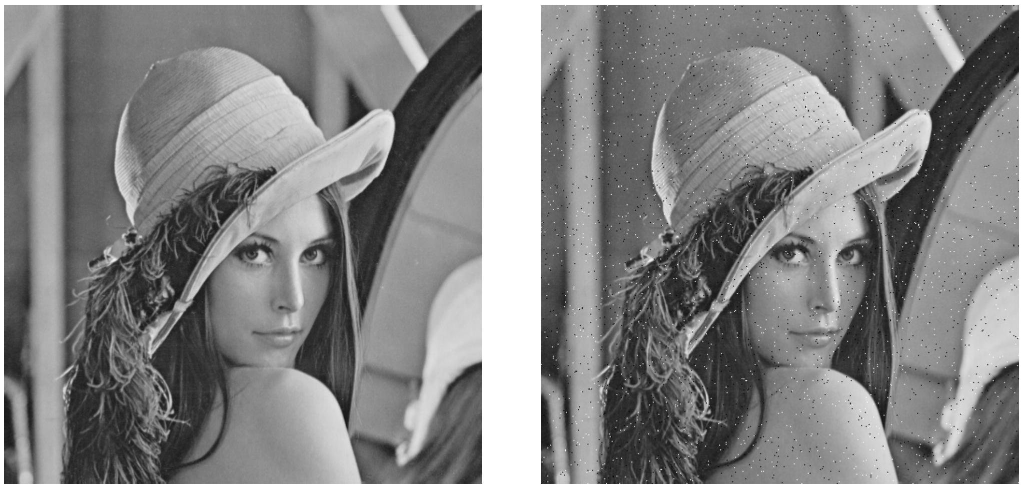
Figure 3.2: Resulting Truncated Image Values

3.2 Types of Noise

In the previous sections the main focus has been on detecting and correcting defective pixels. The appearance of a defect pixel can be seen as noise when put under scrutiny from an observer. It is therefore possible to model a defective pixel as noise on a test-image in order to conclude which algorithm produces the most esthetically pleasing image when removing the noise. There are two dominating sources of noise in an image; salt and pepper noise and Gaussian noise. The next section will describe them in detail.

3.2.1 Salt and pepper noise

Salt and pepper noise is a somewhat typical noise seen on images with defective pixels. It manifests itself as white spots (salt) or black spots (pepper) on the image [19]. A white spot is caused by a false saturation on the image sensor while the black spot is caused by a failed response of the same image sensor. These defects are usually the result of an error on the fabrication of the image sensor. It can be modeled in matlab were a value (typically 0.01)will determine the probability that a given pixel is corrupted with noise and so interpreted as either a black or white pixel. Statistically this kind of noise has zero mean value and is, thus, described completely by its variance. This kind of noise can be successfully removed based on the understanding that the corrupted pixels will have a very high or very low value. As such it will stand out in comparison to its neighboring pixels, assuming that they work correctly, and the removing filter does not need to be very complex, although a very good reconstruction of an image still requires a degree of complexity in the algorithm to deal with different contrast in the same image. Figure 1 shows an original Lena image [15] and 3.4(b) shows the same image corrupted with salt and pepper noise. The algorithm presented in this thesis will start with an image corrupted by noise and then manipulate it to get as close to the original images as possible.



(a) Lena (b) Lena corrupted by salt and pepper noise

Figure 3.3: Impact of salt and pepper noise

3.2.2 Gaussian Noise

Gaussian noise is a noise type that can corrupt pixels with any value within the range of values [11]. Some pixels may be slightly altered while others may be severely altered, thus making the job of detecting and correcting them far more difficult. This type of noise can be added to an image when it is transferred over a noisy information channel. As it is not the scope of this project to remove Gaussian noise, it is only serves the purpose of adding to the understanding of how different noise sources corrupt an image. Gaussian noise has a probability density function (pdf) that describes the probability of a value being within a certain range (see equation (3.1)).The Gaussian noise also has a zero mean value and is described by its variance.

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.1}$$

In equation (3.1) μ is the mean value and σ is the standard deviation. For a normal distribution the μ is zero and σ is 1.The fact that pixels corrupted by Gaussian noise can have any value makes them harder to detect and correct in the time domain. They may not stand apart from the neighboring pixels, like with the salt and pepper noise, but rather be assigned random values to random pixels.Figure 3.4 shows the impact of Gaussian noise on an image alongside the original image.

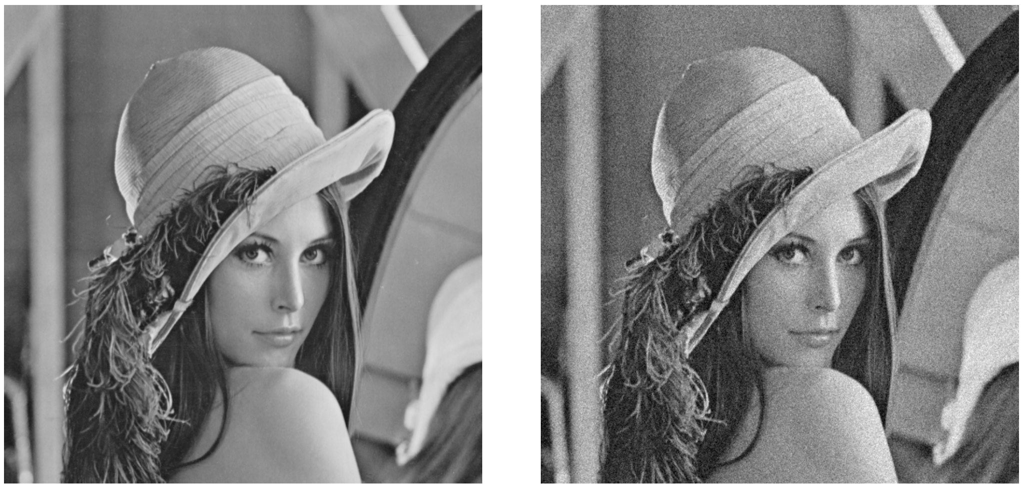

(a) Lena (b) Lena corrupted by Gaussian noise

Figure 3.4: Impact of Gaussian noise

3.3 Images

3.3.1 Histogram

Most modern digital cameras have the option of displaying the histogram of the image along with the image itself on the screen of the camera. The histogram is a graphical representation of the exposure with regards to the balance of highlights, mid-tones and shadows [13]. Having few components in the shadow-or-highlights will result in an image that is over exposed or under exposed. A perfect exposure will have components in all the three regions in a histogram. Using an 8-bit image will result in a histogram in the range of 0–255(sometimes expressed as 8 F-stops). Figure 3.5 shows the different tones of an image with its corresponding grey-level range. Looking at the histogram it can be concluded that the image has all its components within the mid-tone and shadows area and as such does not have any highlight components. The rectangular bar at the bottom represents the grey scale with color level and corresponding pixel count.

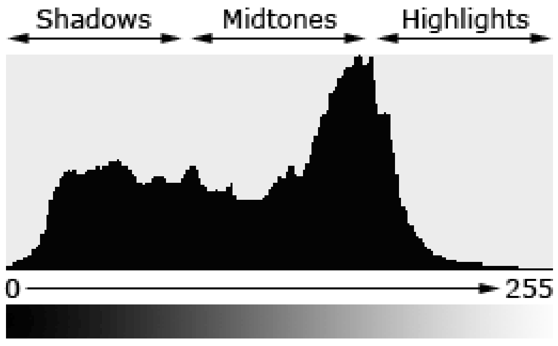


Figure 3.5: Image histogram

3.3.2 Contrast

A histogram contains several important aspects that will tell a photographer how the different components within the image relate to one another. There are, however, other measures that can be read from the histogram such as the overall contrast of an image and exposure. The contrast is the ratio of the largest component and the smallest component of the histogram. A small ratio, corresponding to a narrow set of values on a histogram, will cause the image to have a low contrast. Having a large ratio will result in a high contrast image.

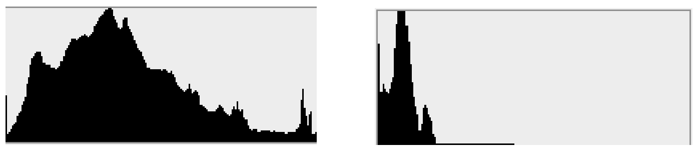

(a) High Contrast (b) Low Contrast

Figure 3.6: High and Low contrast histograms

3.3.3 Bayer Color Filter

The Bayer color filter [4] is used for arranging colors on a square image sensor. It consists of 50% green, 25 % blue and 25% red.The green color is usually divided into green for red and green for blue so that it really is composed of four different colors. Since it has more than twice as much green values as red and blue combined, it is sometimes called RGBG or RGGB. The correction algorithms need to evaluate only pixels with the same color for any given position which is done by looking at pixel (i± 2, j± 2) assuming that the middle pixel is at position (i,j). The Bayer values are light intensities of red, green,

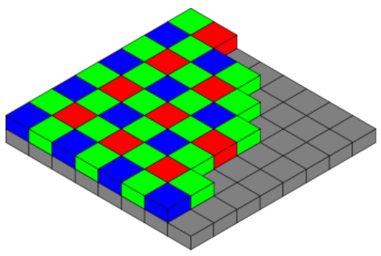


Figure 3.7: Bayer Filter [5]

and blue from the image sensor. It is the job of the processor to convert these light intensities into pixel values in order to obtain a RGB image, which is done by looking at the intensity of the neighboring pixels. Each intensity can range from 0 to 255 in an 8 bit image thus resulting in 16.7 million possible combinations. It can be observed that the next neighbor of the same color is two pixels up, down, left and right so that will dictate how the search for neighbors will go on.This pattern will be utilized in the vhdl part in order to find the next neighbors of the same color to be able to classify a pixel as defective.

3.3.4 PGM Image Format

PGM image formats [18] is an abbreviation for *P*ortable *G*ray *M*ap where the pixel values indicate a light intensity. An image with 8 bits per pixel will result in 256 different levels from black to white. The interesting aspect of PGM image format is that there is no data compression so the pixel values can be read directly from the image. The file format consists of a file header and the image data. The header consists of a number P1-P5, the size of the image in the x and y direction, and the maximum value. An example PGM file showing the number eight looks like figure 3.8.

```
P2
5  7
15
0  0   0   0   0
0  15  15  15  0
0  15  0   15  0
0  15  15  15  0
0  15  0   15  0
0  15  15  15  0
0  0   0   0   0
```

Figure 3.8: Example PGM file format

3.4 Number representation

There are several ways of representing a number with the decimal system begin the predominant system. In scientific research there are situations where using the decimal system would be inefficient, e.g representing large memory spaces.The hexadecimal scale is used when dealing with cpu address spaces while the logarithmic scale is used in measuring decibel in an audio visual system.

3.4.1 Hexadecimal scale with base 16

The hexadecimal number system [12] is a positional system with a radix or base of 16. There are then sixteen distinct values in the hexadecimal system where $0-9$ is the same as the decimal $0-9$ and $A-F$ represents decimal $10-15$. The hexadecimal number 3CB is equal, in decimal, to $(3*16^2)+(12*16^1)+(11*16^0) = 971$. A hexadecimal number is often denoted by a $0x\ number$ as a prefix or as a postfix $number_{16}$. The use of an identifier is to avoid ambiguity with the number as the decimal number 140_{10} is not equal to the hexadecimal number 140_{16}. Figure 3.9 shows the different hexadecimal numbers along with their decimal value and its binary value.

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Figure 3.9: Hexadecimal number representation

Hexadecimal numbers can be broken down to a binary representation by replacing each number by its binary representation according to figure 3.9. Hexadecimal numbers are used in this project for representing addresses in the address space of the cpu. An address can have the value 0x2F, where the prefix denotes that it is a hexadecimal number. The number 2 is replaced by the binary representation of 2 (0010) and the F is replaced by the binary representation of F (1111).The binary representation of 0x2F is therefore 00101111 but the leading zeros in a binary representation can be skipped so the resulting number is 101111.The hexadecimal representation can be used to represent large numbers with fewer digits than its binary and decimal representation.

3.4.2 Decibel scale with base 10

The logarithmic scale [16, 9] used the logarithm of a value instead of the value itself. The value 100 is replaced by 3, as $3 * log_{10}$ is 100. There are different bases, or radix for the logarithmic scale but for measuring decibel (dB) the base 10 is used for the calculations. Large values become more manageable when using the logarithmic scale. Humans hear in a logarithmic scale, so that is why it is extensively used to measure the quality of an audio systems. The decibel values will be used when measuring the peak signal to noise ratio in the simulation chapter. The formula for calculating a decibel value is in equation (3.2), where p_0 and p_1 are the measured values from inp ut to output.

$$[H]dB = 10 * log_{10}(\frac{P_o}{P_1}) \tag{3.2}$$

An increase in decibel from 4 dB to 5 dB represents an increase from 2.51 to 3.16 in real value. Having zero decibel from a system means that the output and the input have the same physical value. If the fraction P_0/P_1 is larger than one, the resulting decibel value is positive while it is negative if the fraction is less than one [8].

3.5 Text representation using ascii

Ascii is the abbreviation for American standard code for information interchange [3, 2] and is a numerical representation of the alphabet.This is useful as a computer cannot interpret a character but rather the ascii value of the character.The initial ascii setup contained 127 characters and thus required seven bits in order to represent all possible combinations. Most of these characters are printable characters like letters and numbers while some characters are non-printable but these are mostly now being considered obsolete.An upper case letter and a lower case letter do have different representation in the ascii coding scheme. Having more bits available introduces more possible symbols to be represented in order to include special letters from different languages. [3] shows the 256 symbols that are represented by 8 bits.The word **t-e-s-t** is interpreted by a computer as **116-101-115-116** using an 8 bit ascii table.

3.6 Measured Values

When comparing two images, there are several values that describes the image. The next section will show the measures used in this project.

3.6.1 Mean Square Error (MSE)

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(I(i,j) - L(i,j))^2 \tag{3.3}$$

Equation (3.3), used for images, measures the squared difference between the reconstructed image and the original image. The reconstructed image will not be pixel wise identical to the original image and this method measures how far they are apart in squared value [21]. I(i,j) is the value of the reconstructed image while L(i,j) is the value of the original image. Lower values of mse will translate into a better quality image (see equation 3.5).

3.6.2 Mean absolute error(MAE)

The mean absolute error [17] shows how the values of the reconstructed image compare to the values of the same pixel in the original image and averages that value. Equation

(3.4) shows the formula for calculating the mean absolute error. 3.4

$$MAE = \frac{1}{mn} \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} |I(i,j) - L(i,j)| \tag{3.4}$$

3.6.3 Peak signal to noise ratio (PSNR)

Peak signal to noise ratio indicates how the maximum signal power compares with the maximum noise power for two images. It is a measure to tell how much an image is pixel wise altered from its original. It is therefore a good measure to determine the quality of the reconstructed image compared to the noise free image [34]. The higher value of psnr translates into a better image quality when comparing two images.

$$PSNR = 10 \log_{10} \left[\frac{MAX_i^2}{MSE} \right] = 20 \log_{10} \left[\frac{2^{k^2}}{\sqrt{\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |I(i,j) - L(i,j)|^2}} \right] \tag{3.5}$$

In the equation above the m and n are the size of the image matrix, I and J are the two images to be inspected and k is the number of bits used to represent the image.

3.6.4 Real time constraints

Modern DSLR cameras have the ability to expose multiple frames in a sequence within a short time frame, and those cameras are said to have high frames per second (fps) rate. The task of correcting defective pixel's from those types of camera introduces a time constraint criteria in order to keep up with the new frames shot. Exposing 30 frames per second means that each frame takes 33 milliseconds to complete. The correction algorithm should therefore utilize a fraction of the time of the frame itself in order to be efficient. Utilizing a correction algorithm will inevitably lead to a slower fps rate but that may be a tradeoff worth doing if the images come out with a noticeable lower noise level.

3.7 TextIO vhdl package

The textIO is a vhdl package that enables reading from a file and writing to a file when running a test bench in simulation in software. Values from the simulation can be stored in a file in a pre-determined location and the simulation can get its values from a file and then use them in a test bench. This is a useful tool in order to specify multiple input values without the hassle of having to write every value in a vhdl file.The package reads one line with the "readline" command. The line is then stored in a variable that has to be split up if there are multiple values in a line.The "writeline" command writes a value from a variable to a line. That line is written to the file with the "write" command. The directory of the pre and post fix of the file are specified using the "FILE" command along with the operating mode (read / write).

Chapter 4

Architecture

This chapter will introduce the modules that makes up the system for correcting defective pixels in a vhdl peripheral and also the modules used in conjunction with the cpu. The top level module is microblaze with the vhdl files that does the processing as a peripheral to that cpu. The cpu will read a value from the register inside the vhdl peripheral each time an interrupt fires and store the value on the cf card.

4.1 VHDL Architecture

Figure 4.1 shows the hierarchy of the vhdl files that does the processing with the ctrl module as the top level module but it should be noted that the user_logic.vhd is on a higher hierarchical level than the ctrl module and the my_custom_ip_register.vhd as the top level vhdl file for the entire peripheral. The my_custom_ip_register.vhd is the interface from the bus to the user_logic.vhd where the signal mapping of the ctrl, image_core, median and size modules takes place.

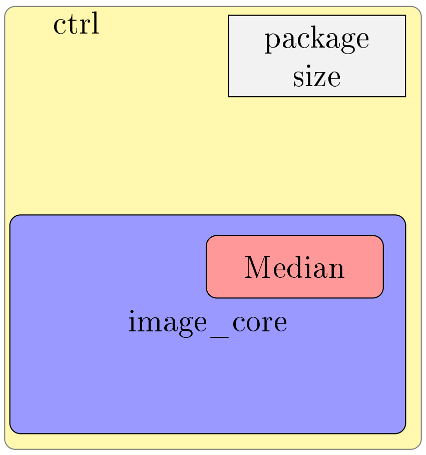


Figure 4.1: Vhdl file hierarchy

4.1.1 Size Package Module

The size package contains variables that are used by the various modules of the system. The reason for using a package is to avoid having to declare the same variables in all the modules, in addition to making it easier to change a value as it only needs to be changed one place. The values are accessible for the modules that implement the package. These values are shown in figure 4.2 where the "size_x" and the "size_y" are the size of the image to be corrected with the "th" being the threshold value used in the classification of a defective value. The "bits" variable is the number of bits per pixel. Eight bits per pixel is equivalent to 256 levels. "Bit_depth" is merely two to the power of bits. ($2^8 = 256$). The "data" is an array that contains the values of the image and those values are converted from the image to an array of the values of the image in such a way that vhdl will recognize the array of values and be able to read those values.

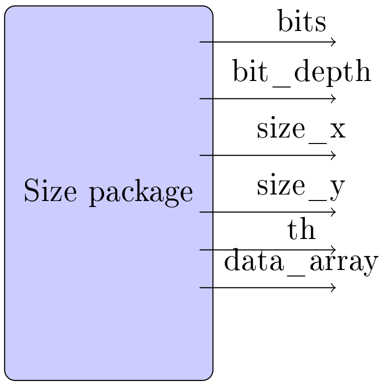


Figure 4.2: Size package

4.1.2 Median module

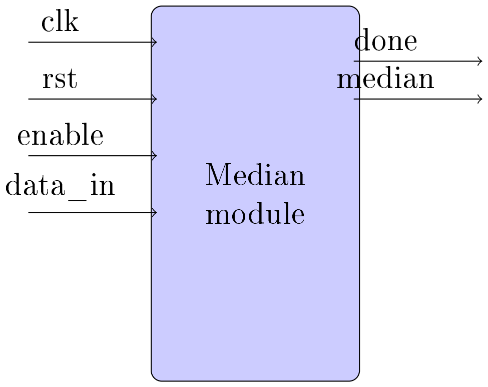


Figure 4.3: Median Module

The median module is the module that calculates the median of any given array, with its input and output signals are given in figure 4.3. The enable signal will cause the state machine to make a transition from "idle" state to the "read_values" state and the signal also denotes that a new input value is present at the input. That value is stored in an array called "data" at the position indicated by the "packet_counter" variable. The next state is the "idle" state where the state machine will wait for more values. When all the values have been stored in the "data" array, indicated by the "packet_counter" = 8, there will be a state transition to the "start" state. This state is the starting state for the double for-loop that will go through all the values in the array and swap two values if the value at "index" is larger than the value at ("index" +1).The "count" variable is that is incremented by one each time a value at "index" is smaller than the value at "index +1 and so does not need to swap place. When the for loop can go through the entire array without swapping any values, that is an indication that the array is in ascending order.This happens when the "count" variable is equal to 8. At that point the next state is "three" where the done flag is set high and the median value set to the value of position 4 in the nine value array. Figure 4.4 shows a graphical representation of the state machine built for this median module along with the signals that will cause a state change.The actual vhdl code for the median module can be found in section .1.1

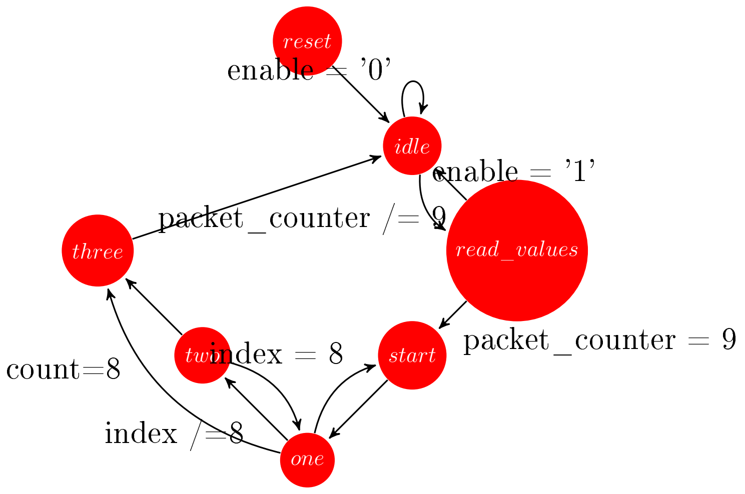


Figure 4.4: Median module state machine

4.1.3 Image_core Module

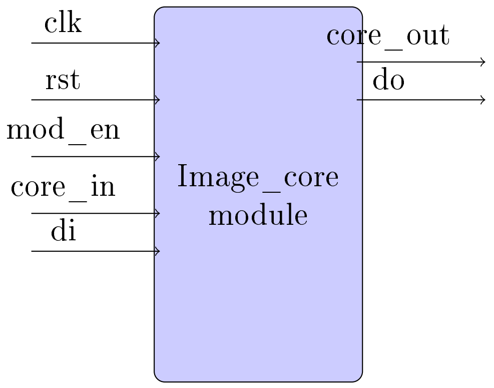


Figure 4.5: Image_core Module

The image core is the module where all the processing on the image is done. Its input and output signals are shown in figure 4.5.The state machine for the image_core is starting in the "idle" state where it will assign the data out (do) flag to low while waiting for the mod_en signal to be high. Then the next state is "read_values" where the values are inputted serially and stored in an array called bayer_array. The values are stored as if they where from an image sensor in order to duplicate the reading pattern [23]. These are the values from an image so the size of the bayer_array is the same as the size of the pixels within the image. The values are stored in such a manner that it corresponds to the way data are read from a bayer image sensor.The transition to the next state happens when all the values are in their correct place in the array, and the next state is "padding127". This is a state where a new array is created, with the sizes in the x and y direction being four pixels larger than the bayer_array. All the positions are assigned with the value 127. The next state is the data_mapping state where the original values in the bayer_array are mapped into the new array in such a way that the values in the bayer_array are surrounded by a two-pixel border of the value 127. The reason for adding this border is to be able to use a single algorithm to go through the entire image including edges and corners as corners and edges are lacking some neighboring pixels of the same color. The next step is to find the defective values in the image one-by-one, and that is done in the "find_defects" state. The way a pixel is classified as defective is that its value is higher than the threshold value plus the highest of its neighbors or lower than the threshold value minus the lowest of its neighboring values. A defective pixel will be assigned a '1' on that position in the pixel_map, which is an array with the same size as the img_array but where all the positions have the value 0 or 1.The reasoning for doing detection and correction in two different steps is to avoid using the value of a pixel at a defective position as a basic for a new value. The correction takes place in the "correct" state by going through the pixel_map and selecting the positions that have a '1' entry. Then an array is composed with the values of the neighboring pixels of the same color. If one of those values are at a position that is classified as defective, the value 127 is assigned in the array instead of the defective value.The array of the neighboring values are sent to the median module serially by assigning the enable signal to high and assigning values of the array to the data_in port of the median module. The above mentioned procedure is done on all the locations that are classified as defective. The last step in the "correct" state is to wait for the median module to produce its result and assign the new value to the correct location in the array. Since the two pixel border is there just for aiding the algorithm with searching for defective values in corners and edges, the border values are removed in the "de_mapping" state. The corrected values are mapped into the bayer_array where they will be assigned to the output of the module in the "read_out" state. The values are assigned to the core_out port along with the data out (do) signal set high when a new value is assigned. Upon completion of assigning all the values to the output the state machine is done with the image in question and set into the "idle" state.The graphical representation of the state machine can be found in figure 4.6 and the vhdl code is in section .1.1.

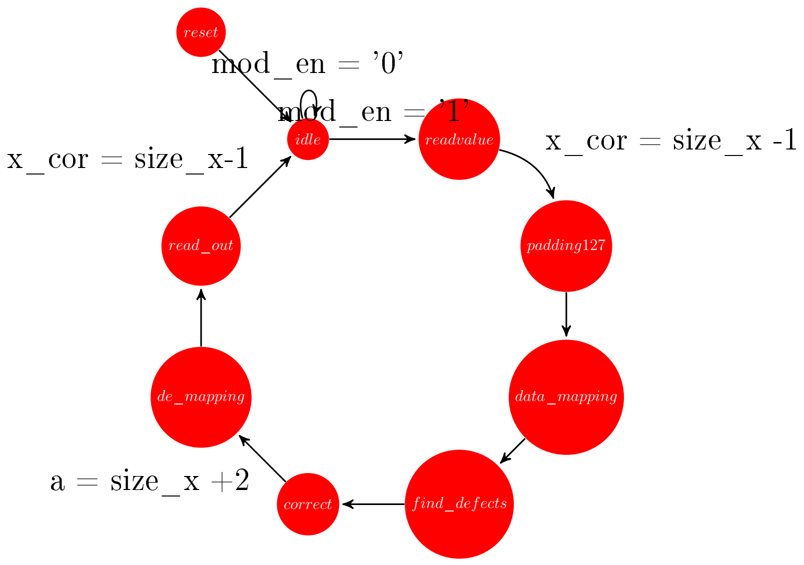


Figure 4.6: Image_core module state machine

4.1.4 Ctrl Module

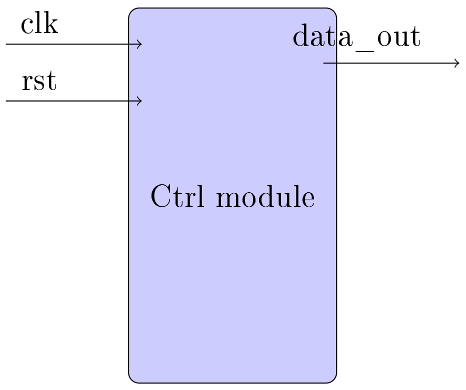


Figure 4.7: Ctrl Module

The crtl module is the top level module in the processing part of the peripheral with its ports being shown in figure 4.7. It only needs the clock signal to start it's instruction and its responsible for assigning the correct flags in order to read and write values to and from the image_core. It consists of a process that assigns values to the image_core and a process that reads the values from the image_core and outputs them to the output of the ctrl module.The process that reads values from the array within the size package contains a small state machine. The "idle" state of the state machine assigns the mod_en flag high and then makes the transition into the "start" state which is nothing more than a delay state to get the timing right. The variable sent_count keeps track of the number of values sent to the image_core, and this value is checked in the state "one" so it can be ascertained when all the values are sent.The mod_en flag is de asserted in the "one" state and then it makes a transition to the state "two" where the various values are assigned to the core_in port of the image_core.The state machine described here can be found in figure 4.8. The second process of the ctrl module reads the values from the image_core and assigns them to its output.This is triggered when the data out (do) signal from the image_core is assigned to '1'. The vhdl code for the ctrl module can be found in section .1.1.

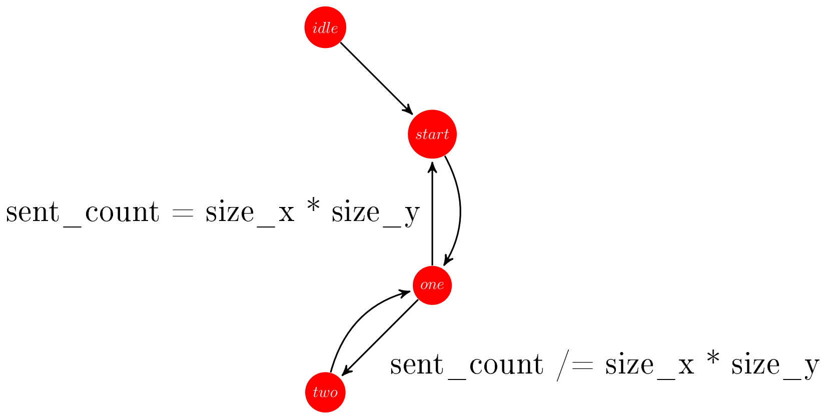


Figure 4.8: Ctrl module state machine

4.2 Note

Even though the vhdl files, shown in figure 4.1, showed correct behavior when simulated it seems impossible to run synthesis on the files as the ISE synthesis tool runs out of memory before completing the synthesis. The decision was therefore made, in accordance with the supervisor, to proceed without running synthesis on the vhdl files. This will make it impossible to test the system on the fpga so the simulation of the vhdl files will serve as an indications of the system's ability to detect and correct defective pixel values. The custom ip module is still implemented in the system but with limited functionality. It will contain a process for generating interrupts and test the ability to write to a peripheral register and read from a peripheral register.
The peripheral will show how the ctrl.vhd module can be mapped in the user_logic.vhd file with its mapping of signals. A successful test will provide evidence that the embedded system can handle reading a value from a register and store it on a cf card.

4.3 Microblaze

The microblaze system contains both hardware and software that enables communication between the cpu and its peripherals. The next section deals with the hardware part of the microblaze system while the preceding section deals with the software part of the system.

4.3.1 Hardware Architecture

Vhdl Peripheral

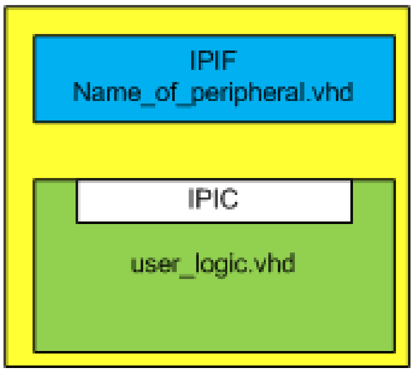


Figure 4.9: Vhdl Peripheral hierarchy

The xps wizard created the template for the peripheral to be connected to the cpu by the plb bus. It also created the interface necessary for the peripheral to communicate with the plb bus signals. The ipif communicates with the plb bus using the entire set of plb bus signals. The ipic is the interface between the ipif and the user modules and custom functionality can be added here or custom vhdl module can be mapped from this module. The vhdl file user_logic.vhd created by the wizard needs to be altered in order to get the correct functionality where a value is read from a register and stored in a file on the cf card. The user_logic file has been altered by adding a process that will generate an enable signal and also the component mapping of the ctrl module. It should also be mentioned that the user_logic file has six software accessible registers, represented by the signals slv_reg0–slv_reg5. The file also contains both a process for reading the values of a register and a process for writing values to a register. The process that generates the interrupt also has a reset signal that will reset the interrupt on the peripheral. The reset signal reads its value from bit 0 in the slv_reg0 register so writing a 0 to that position in slv_reg0 resets the interrupt. The file also contains a component mapping for the ctrl module where the clock and reset signals are mapped from the Bus2IP_Clk and Bus2IP_Reset that is a part of the plb bus signals. The output signal of the ctrl module is mapped to a signal called output so the value of the output can be read by reading slv_reg1.

Connections

The Microblaze system is made up of the components described in section 2 that will now be connected to the plb bus so their functions can be utilized by some software on the cpu. The base system builder wizard and the create peripheral wizard described in section 2 made a system with a block diagram that is given in figure 4.10. The custom peripheral was given the address range from 0xCCC00000 to 0xCC0FFFF when the tool generated the addresses. Since there is an interrupt controller in the system it is important that it is connected correctly from the vhdl peripheral all the way to the processor. The peripheral's interrupt signal is connected to the interrupt controllers *Intr* port, and its *Irq* port is connected to the cpu's interrupt port. The block diagram of the hardware connections in Microblaze is seen in figure 4.10 where it is apparent that the interrupt controller, my_custom_ip_register and the cpu share a common yellow mark on the top right corner that indicates the handler, source and target, respectively, of the interrupt mechanism.

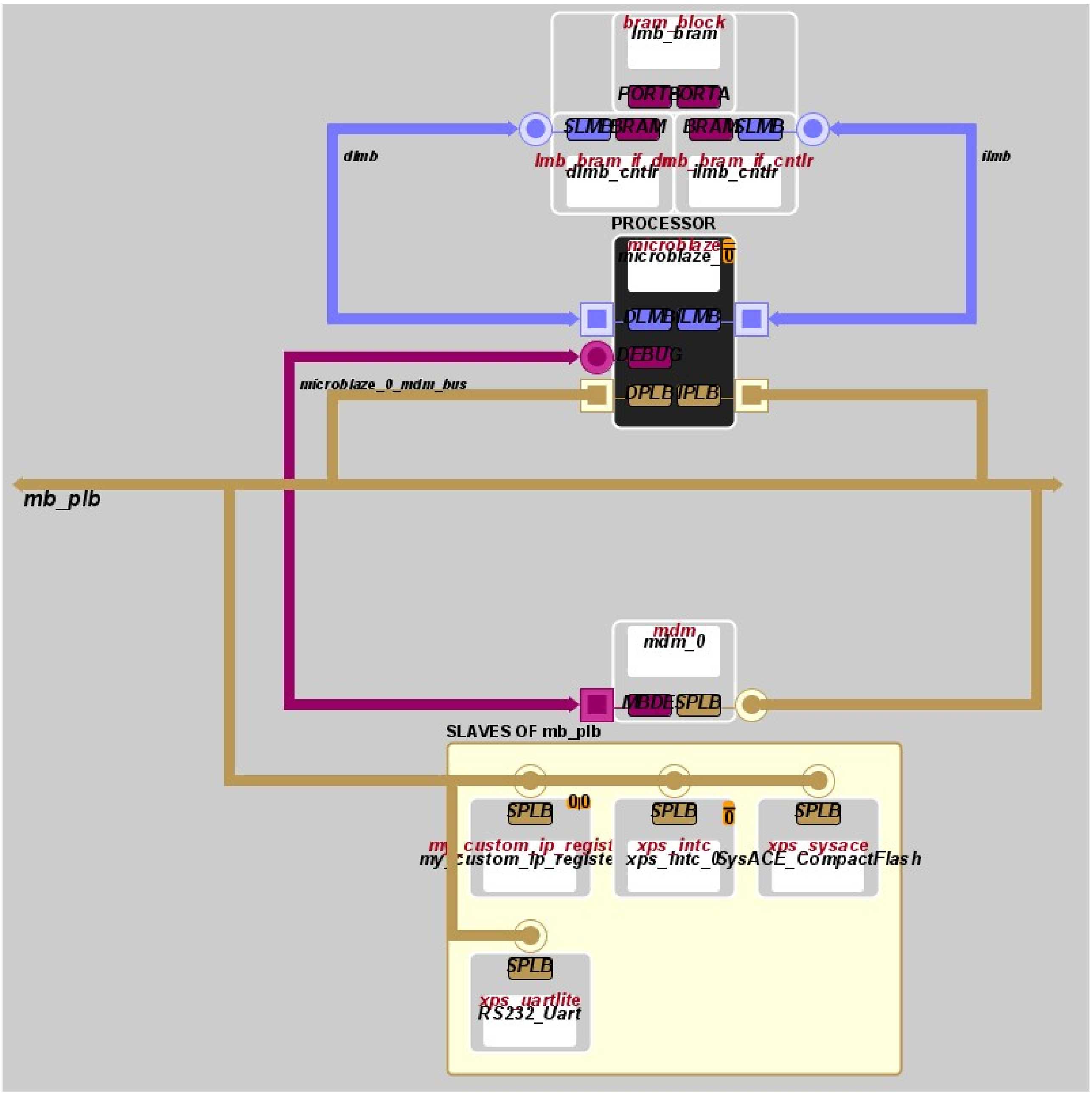
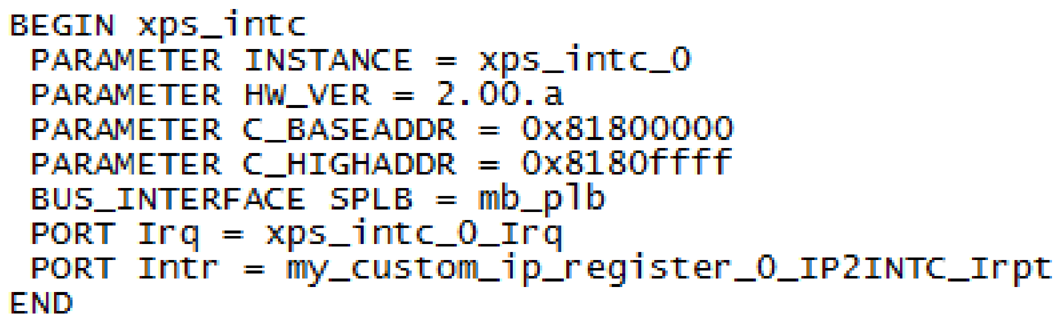


Figure 4.10: Microblaze Block Diagram

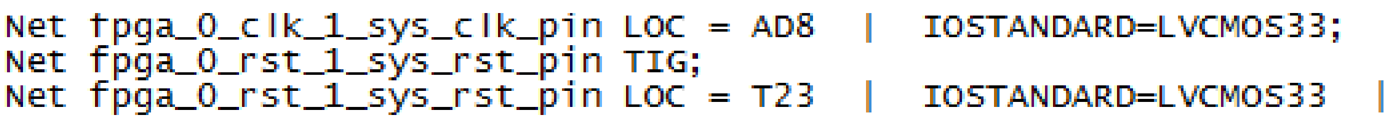
The microprocessor hardware specification*.mhs* file for the system defines the hardware components used along with the bus architecture, peripherals, processor ,system connectivity and address space [20]. The file is located in the main program directory and can be found in section .2.2.



```
BEGIN xps_intc
 PARAMETER INSTANCE = xps_intc_0
 PARAMETER HW_VER = 2.00.a
 PARAMETER C_BASEADDR = 0x81800000
 PARAMETER C_HIGHADDR = 0x8180ffff
 BUS_INTERFACE SPLB = mb_plb
 PORT Irq = xps_intc_0_Irq
 PORT Intr = my_custom_ip_register_0_IP2INTC_Irpt
END
```

Figure 4.11: Microprocessor hardware specification example

Figure 4.11 shows the hardware settings for the interrupt controller with its name, hardware version, high address, base address, bus interface and ports with the signals connected to the ports. The user constraint file *.ucf* for the system specifies how the signals inside the fpga are connected to external pins on the fpga. It can be specified which clock on the chip that is used in a design with the ucf file. The file is stored in the data folder in the program directory and can be found in section .2.3.



```
Net fpga_0_clk_1_sys_clk_pin LOC = AD8  |  IOSTANDARD=LVCMOS33;
Net fpga_0_rst_1_sys_rst_pin TIG;
Net fpga_0_rst_1_sys_rst_pin LOC = T23  |  IOSTANDARD=LVCMOS33  |
```

Figure 4.12: User constraint file example

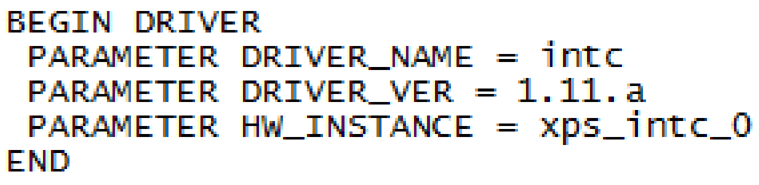
Figure 4.12 show how the vhdl signals fpga_0_clk_1_sys_clk_pin is connected to the external pin of the fpga called AD8.

4.3.2 Software Architecture

The hardware description and connections of the peripheral are now complete so the designer can utilize the functions of the peripheral in order to achieve the desired result. First, let's look at the header files that are of interest for this project, starting with the xparameters header file (section .2.4) which is the file with the address ranges of all the components connected to the cpu. The sysace_stdio.h provides functions to read, write, open and close the compact flash card in addition to specifying the directory of the file to be interacted with. The my_custom_ip_register.h contains functions for reading data from the register in the peripheral and functions to write data to the register. The header files mb_interface.h and xintc.h contains all the functionality to setup interrupts, handle interrupts and acknowledge interrupts on the interrupt controller and on microblaze. The functionality of the c source file as based around interrupts so the file consists of an interrupt service routine and the setup of the interrupts in addition to writing data to the cf card and reading data from registers within the peripheral.

Setting up the interrupts requires that an interrupt handler is registered and the enabling of interrupts in all modules that process, send or receive interrupts. Registering an interrupt handler is done with the "XIntc_RegisterHandler" function found in the xintc_i.h file which is accessible through the xintc.h file. This is the connection between the interrupt signal from the peripheral and the interrupt service routine for that peripheral. Interrupts needs to be enabled in the custom ip, interrupt controller and microblaze so that it can function correctly. It is set up in the interrupt controller by using "XIntc_mMasterEnable" function and the "XIntc_mEnableIntr" function found in the xintc.h file. The custom ip enables interrupt by using the " MY_CUSTOM_IP_REGISTER_EnableInterrupt" command in the my_custom_ip_register.h file. The last step is to enable interrupts on the processor, and that is done with the "microblaze_enable_interrupts();" command found in the mb_interface.h file.

Every time there is an interrupt, the cpu will execute the same piece of code over and over again. That code includes functions to setup a file directory, open the file, read the data from a register and write that value to a file on the compact flash card, close the file and acknowledge the interrupt on the peripheral and in the interrupt controller. The specification of a directory is completed by using the "sysace_mkdir" command from the sysace_stdio.h file. The same file contains the functions sysace_fopen,sysace_fclose, sysace_fwrite to open, close and write to the card, respectively. A data value is read from a software accessible register by using the "MY_CUSTOM_IP_REGISTER_mReadReg" command. This is a general function to read from a register so it is important to get the register offset value correct as not to read from another register. The offset for all the five registers within the peripheral are defined in the my_custom_ip_register.h header file.The base address of the peripheral is defined in the "xparameters.h" file. The c source files used to test the different aspects of the peripheral in this project can be found in sections .5.1,.5.2,.5.3.
The microprocessor software specification file *.mss* specifies the driver name, driver version and the hardware instance to be used on. The .mss file for the system can be found in section .2.1.



```
BEGIN DRIVER
 PARAMETER DRIVER_NAME = intc
 PARAMETER DRIVER_VER = 1.11.a
 PARAMETER HW_INSTANCE = xps_intc_0
END
```

Figure 4.13: Microprocessor software specification

Figure 4.13 shows the driver setup for the interrupt controller.Notice how the hardware instance name is the same name as the instance name specified in the mhs file.

4.4 Matlab scripts

Matlab is a great tool for working with images in software as it has the ability to read an array of values from an image and it can write an array of values to an image file. These abilities will be utilized for this project in order to obtain the image data and to make an image from the output values. The next subsections will go into how this is done. This is a good way of evaluating the results of the simulation as the processing vhdl files cannot be synthesized.

4.4.1 Reading Image

The vhdl modules would normally, in a camera environment, be fed data from an image sensor. This will be duplicated with the aid of Matlab by writing a script, called image_in.m that reads an image from a file, adds noise and then stores the resulting array as a format that vhdl can recognize as an array of values.The script can be found in section .4.1.This script reads all the values in an image and stores them in a two-dimensional array. The desired noise is then injected into the array of values and then it is written in a format $(value1, value2)$ and stored in a text file. The content of the file can then be pasted into the package vhdl file and read by the vhdl modules.

4.4.2 Write Image

A Matlab script is also required to read the values in the file written by the textIO package and then convert it to an image. This is done in the image_out.m file that can be found in section .4.2.The script reads the values in the file produced by the textIO package and then makes a two dimensional array that is written to a PGM image file and stored. Upon completion the images needs to be inverted in a photo editing software before receiving the results in the simulation chapter.

4.4.3 Measuring Statistical Values

The values used to make the graphs in chapter 5 were gathered using a Matlab script. It reads the original image along with the restored images and stores them in two different two dimensional arrays. A double for-loop searches through the two arrays in order to calculate the mean absolute error, mean square error and the peak signal to noise ratio.The script can be found in section .4.3.

Chapter 5

Simulation

All the modules used in this system have been tested to ensure correct behavior on every level of the hierarchy both in terms of vhdl files and software modules. The next sections will show how the various modules have been tested and the results. All the testing has been done at the clock frequency of 100MHz in Active HDL.

5.1 VHDL module testing

The vhdl files that makes up the vhdl peripheral have been tested using a test benches created in activeHDL, in order to ensure that they will exhibit correct behavior. The following sub sections will show results from simulation of the various modules and then bring it all together with a simulation of all the processing files in the peripheral.

5.1.1 Median Testing

The median module was tested by inputting an array of values and then make sure that the output median value was indeed the median of the input values. Figure 5.1 shows an input sequence where all the values are inputed serially and then stored in an array, called data, within the module.

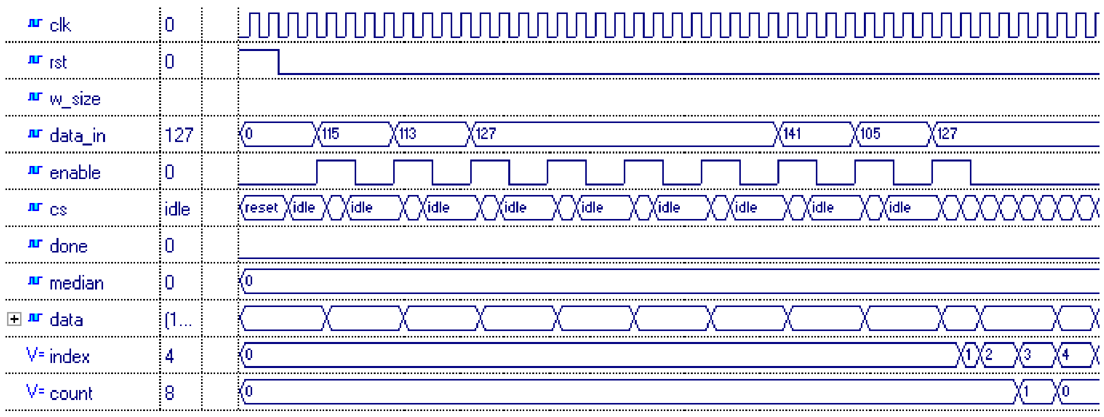


Figure 5.1: Input Sequence

The input array for the median module is {115,113,127,127,127,127,141,105,127}. The same array in ascending order is {105,113,115,127,127,127,127,127,141}, and that corresponds to the data array in the median module. Thus, the module produced the correct median value for that array. Figure 5.2 shows the end of the execution of the median module, for these values, where the done signal is assigned high and the median value is assigned to the median port.

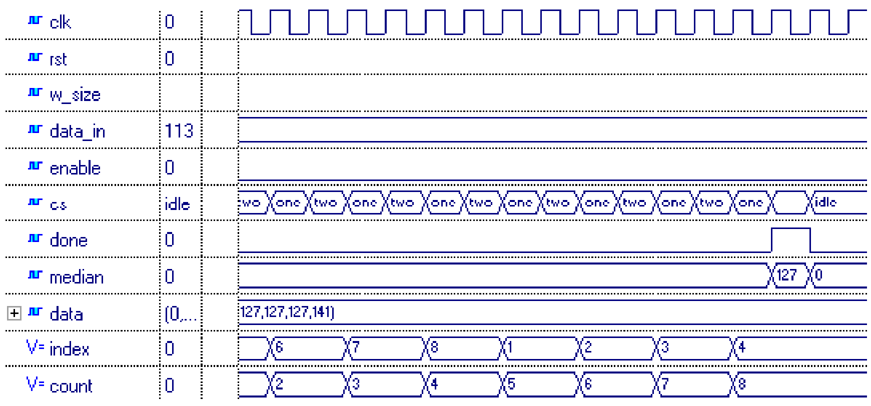


Figure 5.2: Output Sequence

5.1.2 Image_core Testing

The image_core is on a higher hierarchical level than the median module. Its job is to search through a two dimensional array of image values in order to determine the positions of the defective pixels. It will input the correct neighboring pixels to the median module and wait for it to produce the median value. The best way to test this module is to input the values from an image with noise and then make an image of the output values in order to visually confirm that the image does in fact have lower levels of noise.

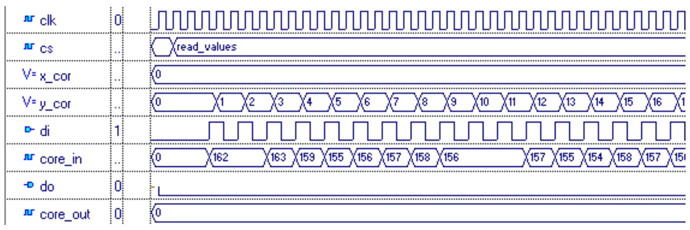


Figure 5.3: Image_core input sequence

Figure 5.3 shows the input signals that are utilized in the image_core module when values from an image are inputed to the image_core module. They are then stored in an array called bayer_array. Figure 5.4 shows that the state machine in the image_core module returns to idle state when it's done with the processing of the input image.

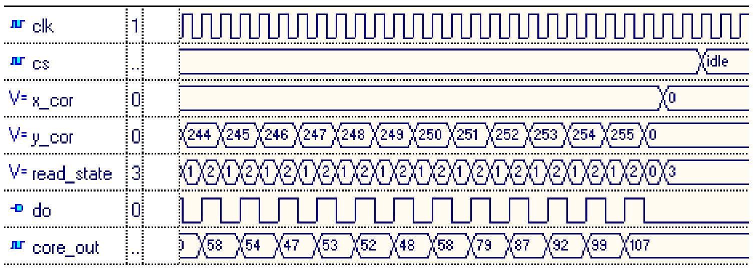


Figure 5.4: Image_core end of execution

The output values from the image_core module are stored in a file using the textIO package. Those values are then run through the Matlab script in order to get an image from the values. The values that are sent into the image_core module are from the Lena [15] image that has been corrupted by noise with the probability of 0.01(1%). The input image is shown in figure 5.5.

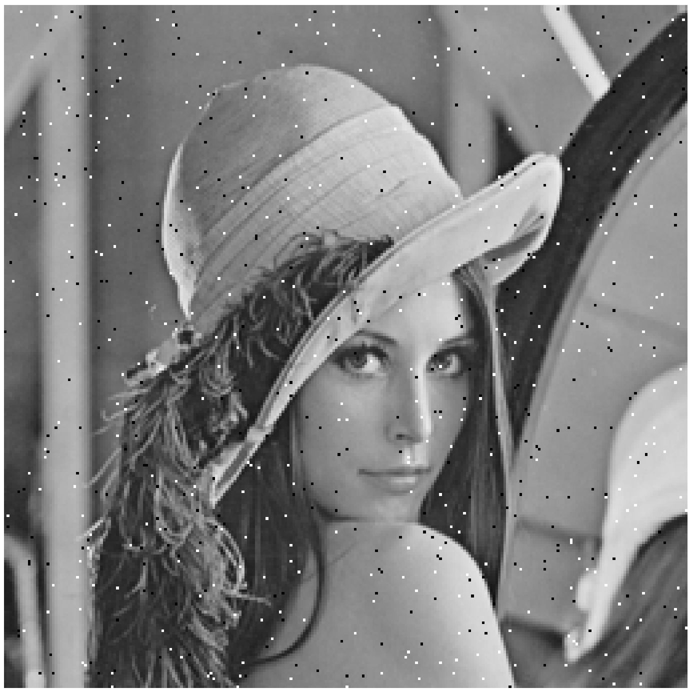


Figure 5.5: Input Image with noise

Figure 5.6 shows the resulting output image from the image_core and stored in the textIO based file. It is clear that the resulting image is a great improvement over the noise corrupted input image.

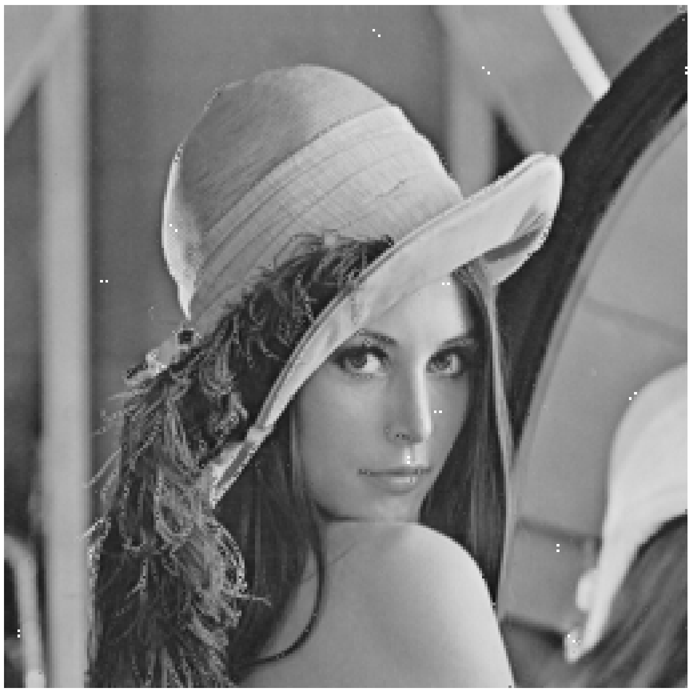


Figure 5.6: Output image

5.1.3 Ctrl Testing

Testing the ctrl module is done by using the same input values as when testing the image_core, but this time the only output signals will be the new image values as that is the only signal declared as output signal in the ctrl module.

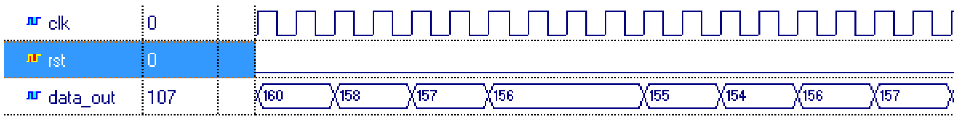


Figure 5.7: Ctrl output sequence

Figure 5.7 shows how the output values of the image are assigned to the output of the ctrl module. This will occur until all the values have been assigned to the data_out signal. All the values are then stored in a file using the textIO package so the content of the file can be made into an image with the image_out.m matlab script. The image is identical to figure 5.6 which is only to be expected as there is no added processing happening in the ctrl module. Testing these modules individually proved that they exhibit the correct behavior as intended.

5.2 Test of the whole VHDL system

The whole system will now be simulated using the values obtained from the textIO vhdl package in order to visually and statistically verify that the resulting image is an image with less noise quantities than the image corrupted with noise. The Lena image(256x256) [15], Field image(500x500) [10] and the canyon image(200x300) [7] will be used as input images with different noise levels and threshold levels. These images have different contrast levels and different numbers of pixels so it will be a test of the algorithm's ability to correct pixels from different contrast images and also the scalability of the algorithm. In addition to putting the image under visual scrutiny, a few key values will be measured that are relevant when comparing images. The next sections will show the resulting graphs from the resulting images with different noise levels and different threshold values.

5.2.1 Different noise levels

The input images were simulated with different noise levels in the interval $0.001 - 0.1$. The mean square error, mean absolute error, peak signal to noise ratio and the simulation time were recorded for every noise level and all the simulation took place with the threshold value at 10. The resulting images can be found in sections .6.2, .6.4, .6.6 along with the input image corrupted by noise. It is therefore possible to view and compare the input image and the output image in order to ascertain the algorithm's ability to remove defective pixels.

Peak Signal to Noise Ratio

The typical values for peak signal to noise ratio for an image with lossy compression is between 30 − 50. That range can be a guideline to evaluate at which noise levels the resulting image will be of sufficient quality. The starting value for the peak signal to noise ratio for the Lena image is 38 at the noise level of 0.001 while the end value at the noise level of 0.1 is 22. That means that the algorithm produces acceptable quality with peak signal to noise ratios for noise levels from 0.001 − 0.02, as can be seen from figure 5.8. It shows the trajectory as a function of noise density, and it is evident that there is an almost inverse exponential relationship between the two except from a few values near the beginning of the graph. The field image does have a lower starting value for the peak signal to noise ratio than the Lena image while it follows a similar trajectory. It will have sufficient quality in the noise range 0.001 − −0.03. The canyon image has its highest peak signal to noise ratio of 23 and will thus not qualify for comparison as it never has a value over 30.

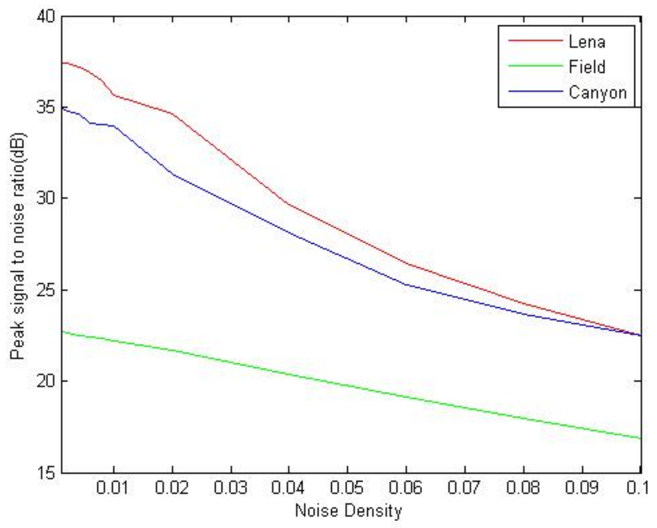


Figure 5.8: Peak signal to noise ratio for different noise levels

Mean Square Error

Figure 5.9 shows how the mean square error relates to different noise densities. There is an exponential growth, for the Lena image, in the mean square error with the starting value of 0.043 at 0.001 noise level and ends at 1.33 at 0.1 noise level. The mean square error relates to the peak signal to noise ratio (see section 3.6.1) so that a lower mean square error translates into a higher peak signal to noise ratio. As such, optimizing either one of the values will result in a higher peak signal to noise ratio. The canyon image has a similar trajectory with a higher peak than the trajectory for the Lena image. The trajectory for the field image had a much lower peak signal to noise ratio and thus a higher mean square error trajectory.

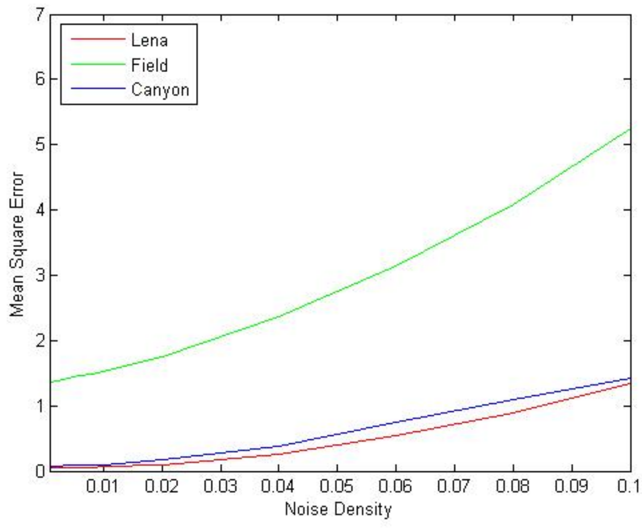


Figure 5.9: Mean square error for different noise levels

Mean Absolute Error

The mean absolute error for the three input images with different noise levels is shown in figure 5.10. The trajectory of the Lena image follows an exponential path with a starting point of 0.68 at 0.001 noise level and ends at 3.03 at 0.1 noise level. Comparing figure 5.9 and figure 5.10 shows a similar trajectory only with the mean absolute error having a higher values than the graph of the mean square error. The canyon image has a similar trajectory while the trajectory of the field image has a higher value at all noise levels.

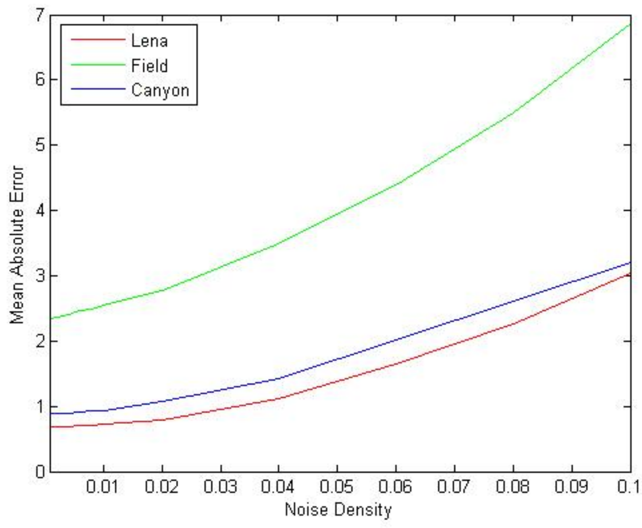


Figure 5.10: Mean absolute error for different noise levels

Simulation times

As with any system required to operate in real time it does have certain demands as far as time is concerned. These simulation times will, therefore, prove or disprove the systems ability to meet those demands that would be imposed in a real time environment.

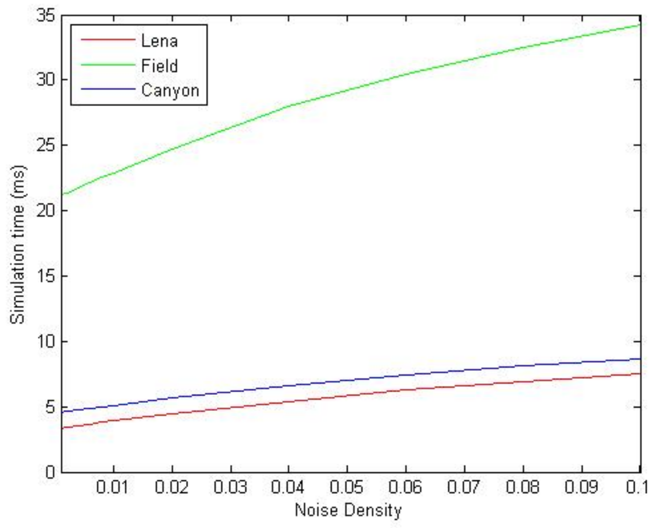


Figure 5.11: Simulation times for different noise levels

Figure 5.11 shows the time it took in order to complete a simulation at the different noise levels for the three input images. All the values of time on the y axis are measured in milliseconds. At low noise densities the simulation for the Lena image time is less than 4 milliseconds.That is equivalent is 250 times per second, which would be fast enough for most camera applications.The highest simulation time comes, not surprisingly, at the highest noise level, results in up to 7.5 milliseconds. That is the same as 133 times per second. In terms of simulation time, this algorithm has sufficient capabilities to be placed in a real time camera environment. The canyon image has a simulation time of 5 milliseconds for small quantities of noise and 9 milliseconds for high noise quantities. Those numbers represents 200 and 111 times per second. The field image does have the highest simulation times of the three images with 21 milliseconds for low noise quantities and 34 which is equivalent to 47 times per second.The images uses 34 milliseconds for high noise quantities which is 29 times per second. Another important aspect of the algorithm is its scalability in terms of simulation time per pixel. An image with a higher number of pixels will require a longer simulation time. It is therefore a trade off for the designer as to how large pixel wise images are required and the time limitation imposed by the environment.

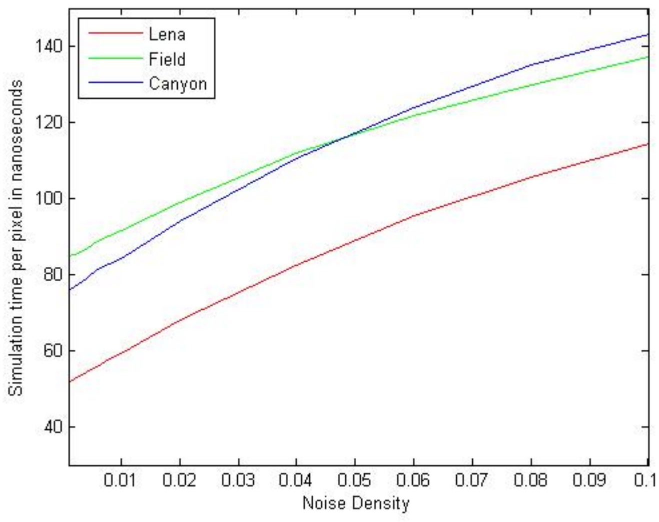


Figure 5.12: Simulation times per pixel

Figure 5.12 shows the simulation times per pixel for the different noise levels. The Lena image causes the lowest simulation time despite containing more pixels than the canyon image. The field image has many more pixels than the field image but still has about the same simulation time. Given the values from the three images, it's possible to interpolate a trajectory in order to estimate the simulation time any image with a given number of pixels.

5.2.2 Different threshold values

This section contains data that was gathered with a constant noise level of 0.02 while the threshold value has been changed within the range $2 - 50$ for all the images. The goal of these simulations is to determine whether altering the threshold for an image will result in higher peak signal to noise ratio. The resulting images from the simulation are placed in sections .6.3, .6.5, .6.7.

Peak Signal to Noise Ratio

Figure 5.13 reveals that a threshold of $20 - 30$ is the best level in order to obtain the highest peak signal to noise ratio for the Lena image. Choosing to use the optimized threshold level of 25 as opposed to the default level 10 does only increase the peak signal to noise ratio from 34.5dB to 36dB. Some situations may have this as a trade off worth doing, while others may opt for a shortened simulation time and a smaller peak signal to noise ratio.The canyon image does have a spike in the peak signal to noise ratio in the $10 - 15$ range but it is otherwise pretty much has a constant decline at all threshold levels. The value of the spike is so small that in most cases it will not amount to anything but increased simulation time.
The field image does have the highest peak signal to noise ratio for the lowest possible

threshold levels. It is clear that this image will require a low threshold in order to produce the highest peak signal to noise ratio. The higher threshold levels also means that areas with low contrasts and defective pixel may not be corrected because th threshold level is too high. As such, it is best to go for the lowest threshold that produces the desired peak signal to noise ratio.

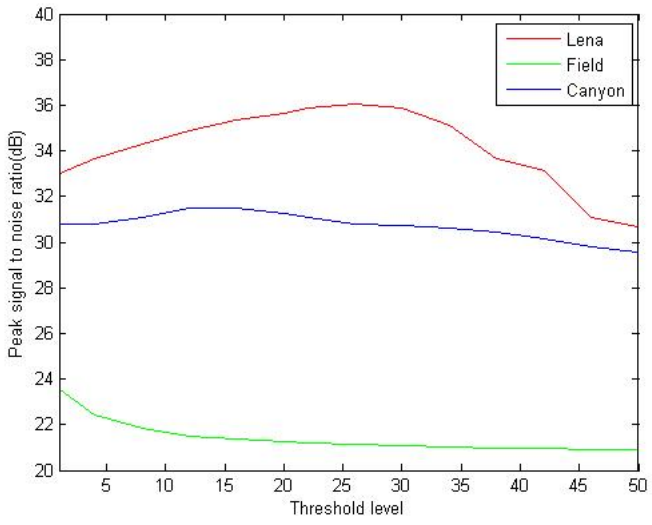


Figure 5.13: Peak signal to noise ratio for different threshold levels

Mean Square Error

Figure 5.14 shows that mean square error for the three images. The Lena image does have an optimum level that corresponds to the optimum level of the peak signal to noise ratio in the range 20 − 30. The canyon image does have its lowest mean square error in the range 10 − 15 and the field image has its lowest mean square error at low noise quantities.

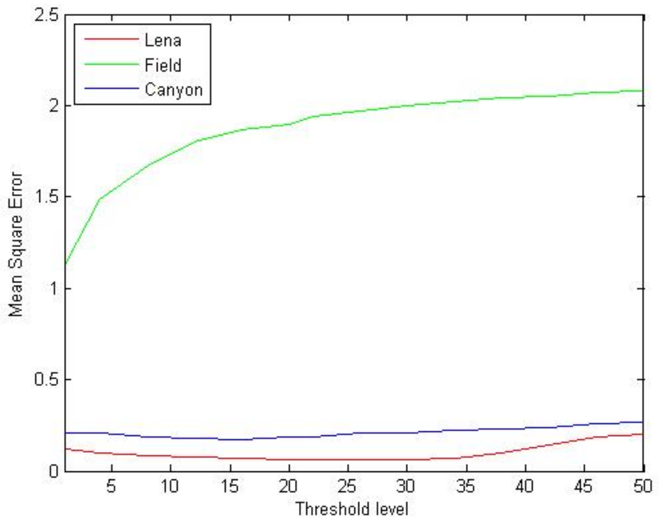


Figure 5.14: Mean square error for different threshold levels

Mean Absolute Error

Figure 5.14 shows the relationship between the mean square error and the noise density. The Lena image still produces the least amount of mean absolute error with a low point in the range $15 - 35$. The canyon image has its low point in the range $15 - 30$ and the field image has its lowest values towards the lowest threshold value.

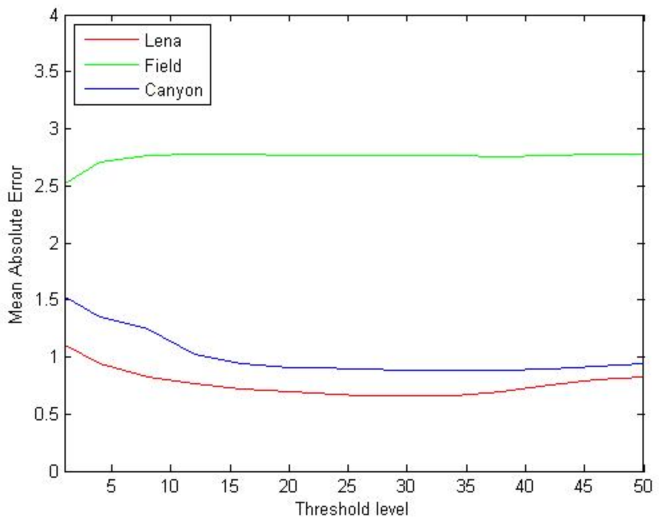


Figure 5.15: Mean absolute error for different threshold levels

Simulation times

The simulation times are shown in figure 5.16. The filed image has the highest simulation time with 40 milliseconds for low threshold levels and 20 milliseconds for high threshold levels. Both the canyon image and the Lena image have a simulation time of 10 milliseconds for low noise levels and 5 milliseconds for high threshold levels. The simulation time should, however, be considered of less importance when tailoring the algorithm to produce best results for a given image as long as it meets the constraints imposed on it by its real time environment.

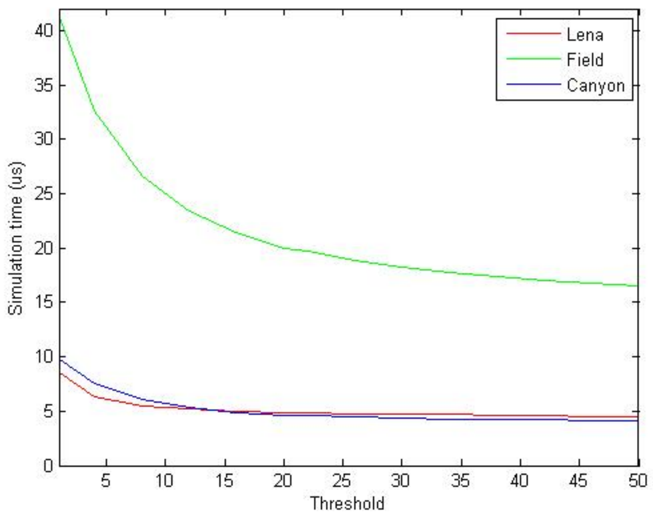


Figure 5.16: Simulation times for different threshold levels

5.3 Software algorithms

This section will show the results of the simulations of the median algorithm that were designed in software in my last semester's project with the three images as input. The matlab script .4.4 was used for the simulations with the threshold of 10 and with the noise level of 0.02.The resulting images from the simulation can be found in section .6.8.

	Lena	Field	Canyon
psnr(dB)	35.1	31.6	32.75
time(s)	3.03	6.5	1.28

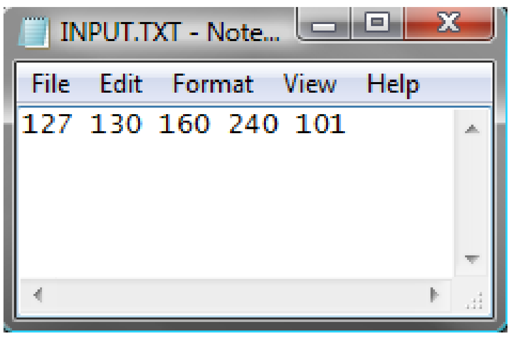
Table 5.1: Software algorithm results

5.4 Microblaze simulation

The vhdl peripheral in the microblaze system is a module to test how values can be read to and from the peripheral and how to use interrupts and interacting with the cf card. Values are read from the compact flash card into the peripheral in the first test of the system ace controller. The resulting output value from the peripheral is stored in a file on the compact flash card by using interrupt. The testing will start with the interaction with the cf card, followed by the interrupt mechanism and reading and writing values to and from registers.

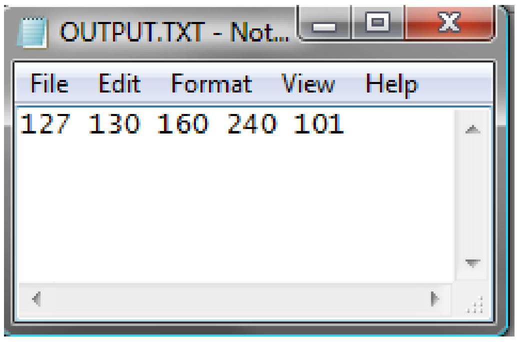
5.4.1 System Ace write to card

The simulation of the compact flash card will verify that files can be read from a file on the cf card and stored back into another file on the card in the same directory. The input file directory is in the root directory of the cf card with a file named "input.txt". The file will contain five integer values, and the file opened in notepad is shown in figure 5.17(a). The read data is stored in a character buffer called "readBuffer" before it is written to a file called "output.txt". Figure 5.17 shows that the output file contains the same data as the input file and thus verifying that the cf card can be written to and read from. The c code used to test the read and write capabilities of the cf card can be found in section .5.1.



```
127 130 160 240 101
```

(a) Input file



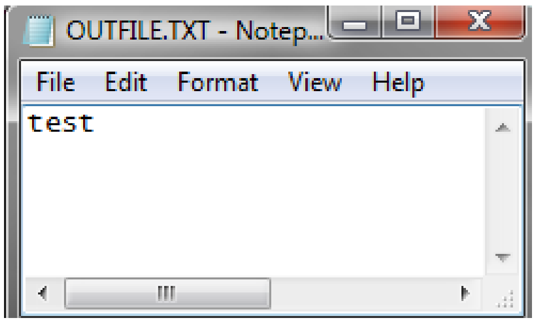
```
127 130 160 240 101
```

(b) Output file

Figure 5.17: Compact flash read and write simulation

5.4.2 Interrupt testing

Setting up an interrupt based system requires that the interrupts are enabled on all the components that deal with an interrupt. It is also necessary to register a handler that is invoked whenever the interrupt "fires". One interrupt handler is required for every device that communicates with the cpu by the means of interrupts. The source file requires one function to enable the setup the interrupt on microblaze, the interrupt controller and the my_custom_ip_register and a function to be executed when an interrupt fires. The init() function in the source code(section .5.2) contains all the required functions for the interrupt mechanism to work correctly. It will register an interrupt handler, enable interrupts on the interrupt controller my_custom_ip_register and microblaze. The init() function only needs to be run once when the cpu is turned on. The function intr_handler() is the interrupt service routine that is executed on every interrupt. A simple test of the interrupts is to write some data to the cf card in the isr. Inspecting the output file will determine if the correct data has been written to the card. For this test the data written to a file on the cf card contains the characters "test", which means that it will be written to the card if the isr is executed. Interaction with the cf card proves to work correctly unless there is an absence of data, in a file with the correct name, written to the card. Figure 5.18 shows the data written to the card by the isr, and it is the same data as the isr was instructed to write. The file is therefore evidence that the interrupt was set up correctly and that it was assigned by the custom ip module.

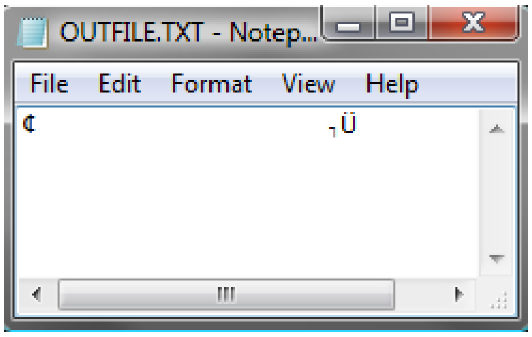


```
test
```

Figure 5.18: File written by the isr

5.4.3 Read value from peripheral register

This section was the last to be tested as it requires both the interrupt mechanism and the read and write methods for the cf card to function correctly in addition to adding functions to write data to a peripheral register and to read data from the register. A hexadecimal value of A2 (162 in decimal value) is written to register 3 in the peripheral unit before the init() function sets up the interrupts. The isr is altered so that it reads the value of register 3 from the peripheral unit and stores it in a variable before writing it to the predetermined file on the cf card. Having the file contain the same value as the one written to the card will be evidence that interaction with the peripheral is working.



```
¢                                   ┐Ü
```

Figure 5.19: Test Read from a Register

Figure 5.19 shows the file produced by the isr and stored on the card. The output file contains values that are coded in ascii so that it will not show $A2_{16}$ or 162_{10} but rather the ascii symbol with the decimal value of 162. The resulting file produced by the isr, in figure 5.19, shows the ascii symbol of the values it read from the register. Using the ascii table at [3] reveals that the symbol written is at the 162nd position and thus confirming that reading from the register and writing to the register works correctly.

Chapter 6

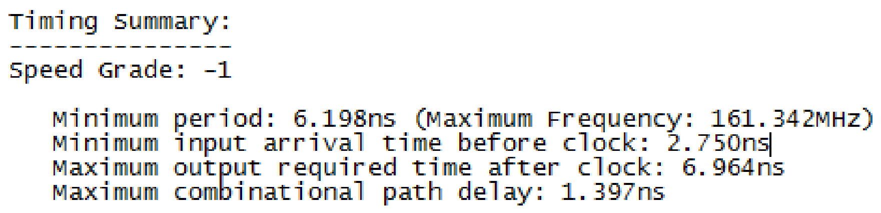
Synthesis

6.1 VHDL Synthesis

The vhdl files that do the processing part of this project cannot be synthesized in the vhdl hierarchy they are built as the Xilinx ise synthetization tool eventually runs out of memory without giving any error message. In accordance with the supervisor at the university, the vhdl files will not be synthesized as part of this project. The vhdl files created by the wizard for the custom ip will, however, be synthesized as part of running synthesis on microblaze. The area report will thus only be for the vhdl files created by the wizard.

6.2 Microblaze Syntethization

The embedded system created in platform studio are synthesized by the "generate netlist" command and the synthesis report is stored in the report folder in the project directory. The synthesis report for the design can be found in section .3. The maximum delay through the system is 6.18 ns which gives a clock frequency of the system of 161.34 MHz. The clock frequency and variable related to the clock are found in figure 6.2.

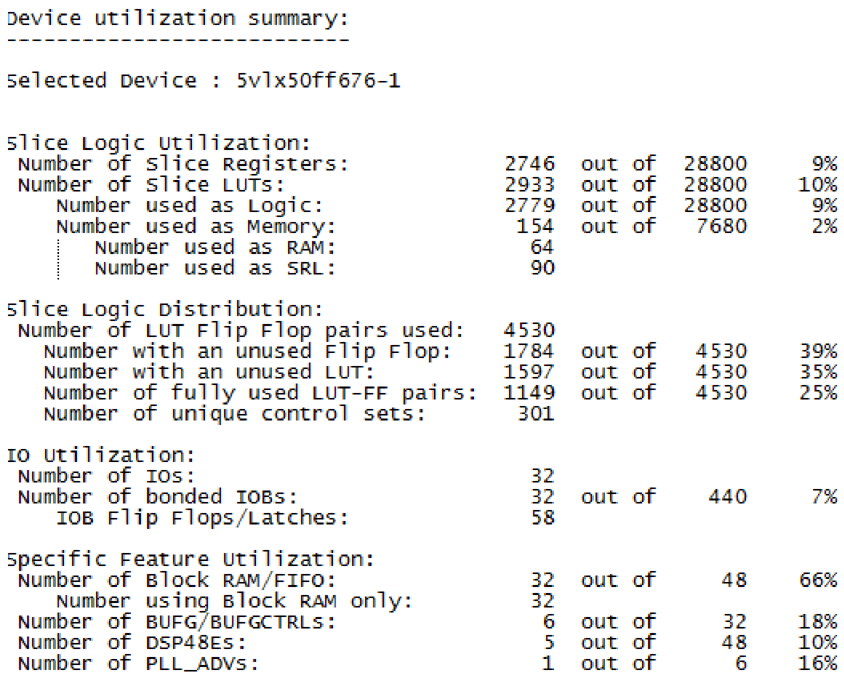


```
Timing Summary:
---------------
Speed Grade: -1

   Minimum period: 6.198ns (Maximum Frequency: 161.342MHz)
   Minimum input arrival time before clock: 2.750ns
   Maximum output required time after clock: 6.964ns
   Maximum combinational path delay: 1.397ns
```

Figure 6.1: Synthesis delay

Figure 6.1 shows a part from the synthesis report regarding the resources utilized for the system. The design utilizes 9% of the available registers and 10% of the available LUTs. These numbers shows that the design only utilizes a small amount of resources on the chip as can be expected for a such a small system. Many systems of this kind will have area limitations so that utilizing a small percentage of the resources is always great news. The I/O utilization is at 7% and with 66% of ram/fifo utilized. The reason for having such a large percentage of ram/fifo utilization is a combination of the requirement of storing values within the peripheral that can be read by the cpu, in addition to only having 64 kb of memory to begin with.



```
Device utilization summary:
---------------------------

Selected Device : 5vlx50ff676-1


Slice Logic Utilization:
 Number of Slice Registers:             2746  out of  28800     9%
 Number of Slice LUTs:                  2933  out of  28800    10%
    Number used as Logic:               2779  out of  28800     9%
    Number used as Memory:               154  out of   7680     2%
       Number used as RAM:                64
       Number used as SRL:                90

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    4530
   Number with an unused Flip Flop:     1784  out of   4530    39%
   Number with an unused LUT:           1597  out of   4530    35%
   Number of fully used LUT-FF pairs:   1149  out of   4530    25%
   Number of unique control sets:        301

IO Utilization:
 Number of IOs:                           32
 Number of bonded IOBs:                   32  out of    440     7%
    IOB Flip Flops/Latches:               58

Specific Feature Utilization:
 Number of Block RAM/FIFO:                32  out of     48    66%
    Number using Block RAM only:          32
 Number of BUFG/BUFGCTRLs:                 6  out of     32    18%
 Number of DSP48Es:                        5  out of     48    10%
 Number of PLL_ADVs:                       1  out of      6    16%
```

Figure 6.2: Synthesis resource utilization

Chapter 7

Tools

This project has required a handful of tools in order to deal with the different aspects of the design from converting an image in software to adding peripheral, to the cpu to synthesizing the complete design. The computer used in this project has a 32 bit Windows Vista Home Basic operating system with Service pack 2, dual processor where each processor has a frequency of 2.26GHz, and the computer has 3.0 GB ram. Matlab was the tool used to convert an image from a file to an array in the vhdl package file and to convert values from the simulation to an image. The hardware programming and simulation were done in Active HDL software, and when the simulation showed correct behavior it was synthesized in ISE project navigator, which is part of the Xilinx web pack. The creation of the embedded design took place in platform studio which also did the synthezation of the embedded design. The design could then be implemented to the fpga with the Impact tool which is a part of the webpack from Xilinx. The various software and version number type can be seen from table 7.1.

Development board	Xilinx ML501
Matlab	Matlab Version 7.4.0.287(R2007a) Copyright 1984-2007 The MathWorks, Inc
Active HDL	Active HDS version 7.2.1644 Student Edition, Copyright(c) ALDEC, Inc All rights reserved
ISE	Project Navigator version 11.5, Copyright (c) 1995-2009
ISE	Project Navigator version 12.1, Copyright (c) 1995-2009
Platform studio	Xilinx Platform Studio 11.5, Copyright (c) 1995-2009 Xilinx, Inc
Platform studio	Xilinx Platform Studio 12.1, Copyright (c) 1995-2009 Xilinx, Inc
Impact	ISE impact4 version 11.5, copyright (c) 1995-2009, Inc
Impact	ISE impact4 version 12.1, copyright (c) 1995-2009, Inc

Table 7.1: Software utilized

Chapter 8

Discussion

The discussion of the results from the hardware and software part of this project are given separately in order to put the results into context.

8.1 Vhdl

The vhdl file median.vhd is the only file that will synthesize in the project navigator tool because it is at the lowest level in the vhdl hierarchy. The other vhdl files cannot be synthesized so the simulation of the files in conjunction with the textIO package and Matlab scripts will serve as the bases for determining this design's ability to operate in a real time system. It is clearly not desirable not to be able to synthesize the vhdl files as it introduces uncertainties as to whether the system will have the same behavior in hardware as it exhibited in the simulation in software. It also introduces no evidence as to the physical size of the vhdl files that will inevitably require a higher utilization of resources as the majority of all the processing is done there. The only area values that are given is the area of the two vhdl wrapper files created by the import custom ip wizard in order to set up communication between the custom ip and the plb bus. It is also assumed that adding the other vhdl files will introduce a longer delay path in the design, which will lead to a lower clock frequency.

8.2 Microblaze

The embedded system was created and synthesized in the platform studio. The design shows that it has a high clock frequency and a low resource utilization except for ram usage. This can be explained by the fact that the system really does very little processing. The other peripherals show higher clock frequencies than the processor so the cpu is the weakest link when calculating the maximum clock frequency. The code written to interact with the cf card, set up interrupts and reading from a register and writing to a register are shown to work as desired. Testing the system by writing values to a file on a cf card would prove to be a better way of testing the system if the output data were written in decimal number as opposed to ascii code as the data could be written into matlab with a simple script.

8.3 Simulation

The graphs based on the data from the simulations data shows that a high peak signal to noise ratio leads to a low mean square error and mean absolute error. This is also obvious from looking at equations (3.4), (3.3), (3.5) that optimizing for one of the variables will result in an optimization for all the variables.Simulating the images with different thresholds introduced the concept of being able to optimize an image in order to achieve higher peak signal to noise ratio and lower mean square error and mean absolute error on an image-by-image basis. As an illustration, the peak signal to noise ratio for 0.02 noise level with the threshold at 10 is 34.59dB. The simulation of the different thresholds revealed that the optimum threshold for the highest peak signal to noise ratio was $20-30$. The peak signal to noise ratio for the threshold of 26 with 0.02 nose level is 36.04dB. Those numbers show that there is something to gain by optimizing the threshold, but that is at the expense of higher simulation times as the optimum threshold level has to be determined based on simulations on different threshold levels. The simulation for correcting the defective pixels can only be run after the threshold level has been found. Having the threshold value at 10 seems to be a good general purpose value as the peak signal to noise ratio is close to the peak signal to noise ratio at the optimized threshold level.

8.4 Real time operation

Modern cameras like the Canon 7d [6] have the ability to expose multiple images at a fast rate of frames per second. The Canon 7d can shoot 8 frames per second at continuous shooting. That introduces a limit as to how long a correction algorithm can use and still keep up with new data arriving from the sensor. Shooting 8 frames per second translates into 125 milliseconds per image. The simulation times for the 256x256 pixel test image vary between 3.5 and 7.5 millisecond depending on the noise level. Assuming an average simulation time of $(3.5+7.5)/2 = 5.5$ milliseconds the time frame for a single exposure increased from 125 millisecond to 130.5 millisecond. This results in the frames per second being reduced from 8 to 7.6. The time it takes to correct an image is a small fraction of the time it takes to expose an image, which is what is desirable as the main function of the system is to expose images. The field image used an average simulation time of 28 milliseconds and thus increased the time for each exposure to a total of 153 milliseconds, which is equivalent to 6.5 exposures per second with the 8 fps initially. The canyon image has an average simulation time of 6.5 milliseconds which makes the total exposure time 131.5 milliseconds. That enables the camera to have a fps rate of 7.6 which is the same as for the Lena image.

8.5 Comparison with 2009 fall project

The simulation of the algorithm designed in my 2009 fall project was re-simulated with the Lena, field and canyon images in order to compare it with statistical values and simulation times of the same algorithm designed in hardware. The comparison values were both gathered during simulation using a threshold of 10 and a noise level of 0.02. The simulation times and the peak signal to noise ratios for the software algorithms are shown in figure .6.8.The Lena image took 3.03 seconds to simulate in software and 4.46 milliseconds in hardware so a tremendous amount of execution time is saved when doing the processing in hardware.The corresponding peak signal to noise ratio from the software algorithm is 35.1 dB while it is 34.5 dB for the version where all the processing is done in hardware. The field image took 6.5 seconds to simulate in hardware while it took 24.76 milliseconds. The peak signal to noise ratio was 31.6 dB for the software version and 21.66 milliseconds for the hardware version.The canyon image took 1.28 seconds to simulate in software with a corresponding peak signal to noise ratio of 32.74 dB while the hardware version was simulated in 5.64 milliseconds with a peak signal to noise ratio of 31.33 dB. The above data shows that doing the processing in hardware for this algorithm executes much faster with a somewhat lower peak signal to noise ratio.

8.6 Interpreting images

The values resulting from the vhdl simulation have been interpreted by a Matlab script in order to get the values, stored by the textIO package, into a two dimensional array that in turn can be written to an image. There is, however, a phenomena where an image may be inverted when the Matlab script image_out.m (.4.2) is done executing. These images therefore needs to be entered into a photo editing software where the images have to be inverted. I have found no clue to the nature of this phenomena in the PGM standards so it is still an unsolved mystery.

8.7 Border pixels

Defective pixels near an edge or a corner will use the border-pixels in order to find the new value for a pixel. As the border-pixels all have the value 127 it may replace a pixel with a value that does not represent its neighbors. A better way of making a border might be to create the border as a mirror of the values in the first two rows and columns in such a way that the first border row will have the same values as the second row of the image, looking at the upper edge. The new pixel values will then have a value corresponding to the neighboring pixels.

8.8 Tools

Xilinx released the 12.1 version of the ise webpack and the platform studio as the project are ongoing so that is the reason that both the 11.5 version and the 12.1 version was listed in the tools chapter.

8.9 Appendix

The package vhdl file contains an array with all the values from the input images and it was not possible to get Latex to add that file to the appendix as the pdflatex compiler kept shutting down when attempting to compile. The file is therefore added to the appendix without the array with the image values. The values in the array are specified in the signal the data_array.

Chapter 9

Concluding remarks

This project aimed at making an algorithm to detect and correct defective pixels in images, and with all the processing done in hardware. The simulations show that the algorithm is exhibiting correct behavior with the output values represented as an image showing less quantity of noise than the input. The project has met the goals that were associated with the assignment even though simulation had to take place in several steps as opposed to if the vhdl file could have been synthesized.

Chapter 10

Further Work

The project has been simulated in software but has not been synthesized due to ISE software is unable to synthesize the design as it runs out of memory. Further work for the project can then be divided into getting the algorithm to a working stage in an fpga and improving the design once it is working. The embedded system has a peripheral that outputs values to a register in the same manner as the vhdl peripheral was intended to do. The user_logic.vhd file and the ctrl.vhd can be altered to accommodate the ability to read values from a file into the ctrl module. The ctrl.vhd module needs to add in input signal for data. It would then read the input signal, and relay them to the image_core module. The value can be read from a slv_reg in the peripheral. The output values from the ctrl module can be mapped to the output signal of the user_logic. The value can now be read by software by reading slv_reg1. The border pixels should be given values that represents the values of the images as opposed to having a fixed value for all images. This can be done by having the border rows mirror the value of the edges of the image. The values on the border would then ensure that a better replacement value for the pixel was used. That would make the system more functional as it can operate on any set of values given that the size of the image is the same. The task of improving the system once operational is about making tradeoffs between the variables in the design like execution time and resource utilization. It is therefore imperative that the system is operational and that the limitation of the target device is known before attempting to optimize the system.

Bibliography

[1] Adding a custom ip , http://www.youtube.com/watch?v=DkjVXeqRKjE&feature=related.

[2] Ascii table , http://en.wikipedia.org/wiki/ASCII.

[3] Ascii table , http://www.web-source.net/symbols.htm.

[4] Bayer filter, http://en.wikipedia.org/wiki/Bayer_filter.

[5] Bayer image filter , http://en.wikipedia.org/wiki/Bayer_filter.

[6] Canon 7d , http://shop.usa.canon.com/webapp/wcs/stores/servlet/product_10051_10051_230851_-1.

[7] Canyon image , http://bluemesaphotography.com/hye/wp-content/uploads/2009/07/IMGP8596Crop6by9BW-200x300.jpg.

[8] Decibel measuring , http://www.phys.unsw.edu.au/jw/dB.html.

[9] Decibel scale , http://en.wikipedia.org/wiki/Decibel.

[10] Field image , http://www.thephotoargus.com/wp-content/uploads/2009/11/bw47.jpg.

[11] Gaussian noise , http://en.wikipedia.org/wiki/Gaussian_noise.

[12] Hexadecimal nubers , http://en.wikipedia.org/wiki/Hexadecimal.

[13] Histogram , http://www.cambridgeincolour.com/tutorials/histograms1.htm..

[14] Interrupt in microblaze tutorial, http://www.fpgadeveloper.com/2008/10/timer-with-interrupts.html.

[15] Lena image, http://www.cs.cmu.edu/~chuck/nsipg/nsi.html.

[16] Logarithmic nubers , http://en.wikipedia.org/wiki/Logarithmic_scale.

[17] Mean absolute error http://www.fmi.uni-sofia.bg/vesta/Virtual_Labs/freq/freq6.html.

[18] Pgm file format, http://netpbm.sourceforge.net/doc/pgm.html.

[19] Salt and pepper noise , http://en.wikipedia.org/wiki/Salt_and_pepper_noise.

[20] Xilinx file extensions, `http://www.xilinx.com/itp/xilinx8/help/platform_studio/html/ps_r_gst_project_files.htm`.

[21] Mood A., F Graybill, and D Boes. *Introduction to the Theory of Statistics.* McGraw-Hill, 3rd edition, 1974.

[22] Henrik Backe-Hansen. Defective pixel correction. Technical report, 2009.

[23] Nokia Coroporation. Smia 1.0 part 1:functional spesification. Technical report, 2004.

[24] Granite Island group. Rs232 interface. Technical report.

[25] Rod Jesman, Fernando Martinez Vanilla, and Jafar Saniie. Microblaze tutorial creating a simple embedded system and adding custom peripheral using xilinx edk software tools. Technical report.

[26] Xilinx. Ml501 evalutaion platform. Technical report.

[27] Xilinx. Using and creating interrupt-based systems. Technical report, 2005.

[28] Xilinx. Libxil fatfile system(fatfs). Technical report, 2006.

[29] Xilinx. System ace compactflash solution. Technical report, 2008.

[30] Xilinx. Defective pixel correction v1.0. Technical report, 2009.

[31] Xilinx. Edk concepts, tools, and techniques. Technical report, 2009.

[32] Xilinx. Embedded system tools reference guide. Technical report, 2009.

[33] Xilinx. Logicore ip xps interrupt controller (v2.01a). Technical report, 2010.

[34] Wang Yuanji, Li Jianhua, Lu Yi, Fu Yao, and Jiang Qinzhong. Image quality evaluation based on image weighted separating block peak signal to noise ratio, 2003.

.1 VHDL

.1.1 VHDL Source Code

Median Module

```
---------------------------------------------------------------
--
-- Title          : median.vhd
-- Design         : defect_pixel_corr
-- Author         : Henrik Backe-Hansen
-- Company        : NTNU / Aptina Norway
--
---------------------------------------------------------------
-- Description : this module will take the w_size, enable and data_in
     as the primary input signals.
--the enable signal will denote that a new value is ready at the
   data_in port.
--the w_size will determine the size of the internal data array as
   the size is w_size*w_size-1
--The values are first inputed into the data array and then sorted
--the median of the array is then put on the median output port and
   the done flag is set to '1';
---------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use work.size.all;

entity median is
  port(
  clk    : in     std_logic;                  --clock
  rst    : in     std_logic;                  --reset signal from top level
      module
  data_in  : in     integer range 0 to bit_depth; --integers from the
      sliding window
  enable   : in     std_logic;                  --signal to enble the
      median module
  done     : out  std_logic;                  --signal denotes the median
      values is found
  median   : out  integer range 0 to bit_depth  --the corresponding
      median value
  );
  end entity;

architecture med_arch of median is

--------------------------------------------------------------
--Signal declaration
```

```
----------------------------------------
type state_type is(idle,read_values,start,one,two,three,reset);
signal cs,ns:state_type;

type data_format is array (8 downto 0) of integer range 0 to
   bit_depth;   --input data array size setup
signal data:data_format:= (others => 0);

begin
----------------------------------------
--This process resets the module in case of the rst signal being
   evaluated to '1'.
--It changes the value of the ns(Next State) signal to the cs(Current
    State) signal if rst is '0'
----------------------------------------
   sync_proc:process(clk,rst)
    begin
        if(rst = '1')then
      cs <= reset;
      elsif(rising_edge(clk))then
      cs <= ns;
      end if;
    end process;


----------------------------------------
--temp is a variable used when two values changes places
--index is used to loop through the array
--median_counter points to the position of the median in a sorted
   array
--count counts the number of instruction that can be done on one pass
    of the array
--packet_counter is the number of values in the data array and if all
    the packets are present
--it will start the median calculation by setting ns to start
--
--state idle will trigger on the enable signal and then set the ns
   signal to read_values
--the read_values state will store the input in the location ,in the
   data array,denoted by packet_counter
--state one will loop through all the values and swap values if the
   higher value is at the higher position
--when all the values have been sorted count = w_data*w_size-1, and
   the ns signal is three
--In the state three the median is set on the output, the done flag
   is set to '1' and ns is idle,
--and the median module is ready to read other values
--output signals are given default values in the begining of the
   process
```

```
--------------------------------------------------------
    comb_proc :process(cs,enable,data_in)
  variable temp        : natural range 0 to 255  := 0;
  variable index        : natural range 0 to 255  := 1;
  variable count       : natural range 0 to 255  := 0;
  variable packet_counter : natural range 0 to 255  := 0;


  begin
    done       <= '0';
    median       <=  0;

      ns <= idle;
      case cs is

        when reset =>
          median <= 0;
          packet_counter := 0;
          packet_counter:=0;
          count := 0;
          index := 0;
          ns <= idle;


        when idle =>
          if(enable = '1')then
            ns <= read_values;
          else
            ns <= cs;
          end if;


        when read_values =>
          data(packet_counter) <= data_in;
          packet_counter := packet_counter +1;
          if(packet_counter = data'length)then
            ns <= start;
          else
            ns <= idle;
          end if;

        when start =>
          packet_counter := 0;
          count := 0;
          ns <= one;
          index := 1;

        when one =>
          if(data(9 - index) > data(8 - index))then
```

```
            temp := data(9 - index);
            data(9 - index)<= data(8 - index);
            data(8 - index)<= temp;
            count:= 0;
          else
            count := count +1;
            temp := 0;

          end if;

          if(index = 8)then
            ns <= start;
            index := 0;
          else
            ns <= two;
          end if;

          index := index +1;

          if(count = 8)then
            ns <= three;
          else
            ns <= two;
          end if;

        when two =>
          ns <= one;

        when three =>
          ns <= idle;
          done <= '1';
          median <= data(4);

        when others =>
          ns <= idle;


      end case;

    end process;

  end architecture;
```

Image_core Module

```
--------------------------------------------------------------------------
--
-- Title       : image_core
-- Design      : defect_pixel_corr
-- Author      : Henrik Backe-Hansen
-- Company     : NTNU / Aptina Norway
--
--------------------------------------------------------------------------
--
-- File        : image_core.vhd
-- Generated   : Mon Feb  1 14:09:23 2010
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.20
--
--------------------------------------------------------------------------
-- Description : This image core is the top level unit of the
   defective pixel correction algorithm.It uses
--the median module in order to calculate the median of the values at
    its input.
--The image core is enabled by the mod_en signal and will then read
   in one pixel-value at the time untill all pixels have been read
--The data array will consist of the read pixel-values and a two
   pixel border with the value 127. This is to make it
--easier to read and correct edge and corner pixels.
--------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.size.all;

entity image_core is
   port(
    clk              : in  std_logic;
    mod_en             : in  std_logic;
    rst              :   in  std_logic;
    core_in            :   in  integer range 0 to bit_depth;
    di               : in  std_logic;      --data in
    do               : out std_logic;      --data out
    core_out           : out integer range 0 to bit_depth
       );
end image_core;

architecture image_core_arch of image_core is

------Signals to the median_mod-------
--These signals are the signals that communicates between the median
   module and the image core
```

```
signal enable :std_logic;
signal data_in  :integer range 0 to bit_depth;
signal done   :std_logic;
signal median :integer range 0 to bit_depth;
----------------------------------------


-------------type definitions----------
type pix_map is array(0 to size_x+3,0 to size_y+3)of bit;
        --pixel map
type img_array is array(0 to size_x+3,0 to size_y+3)of integer range
   0 to bit_depth;    --image array with 2 pixel border
type bye_array is array(0 to size_x-1,0 to size_y-1)of integer range
   0 to bit_depth;    --bayer image array data
type median_array is array(8 downto 0) of integer range 0 to
   bit_depth;           --array with the array to the median mod

----------------------------------------


-----------------FSM---------------------
type state_type is(idle,read_values,correct,read_out,padding127,
data_mapping,de_mapping,find_defects,reset);                        --FSM
    states
signal cs,ns:state_type;                                 --FSM signals
----------------------------------------


----------------------------------------
--Instansiation of the median module component and mapping signals to
    the median module signals
----------------------------------------
begin
median_mod:entity work.median(med_arch)
  port map(
  clk,
  rst,
  data_in,
  enable,
  done,
  median
  );

----------------------------------------
--process that is sensitive to the input values, clk, reset, and done
    signal. The process will read the pixels into the
--bayer array, then add a border and map it to the data array.

----------------------------------------
sync_proc :process(clk,rst,core_in,done)
variable bayer_array  :bye_array;
variable pixel_map    :pix_map;
variable data      :img_array;
variable med_array     :median_array;
variable state       :integer:= 1;
variable a          :integer :=0;
variable b          :integer :=0;
variable high       :integer range 0 to bit_depth:= 0;
variable low        :integer range 0 to bit_depth:=bit_depth;
variable timer        :integer range 0 to 2   := 0;
variable x_cor        :integer range 0 to size_x-1:= 0;
variable y_cor        :integer range 0 to size_y-1:= 0;
variable read_state    :integer range 0 to 3:= 1;
variable testing     :integer := th;
begin


  if(rst = '1')then
    cs <= reset;
  elsif(rising_edge(clk))then
    cs <= ns;

  case cs is

-------------------------------------------------------
--The reset state will set default values to all the variables used
-------------------------------------------------------
    when reset =>
      timer   := 0;
      a      := 0;
      b      := 0;
      med_array := (others => 0);
      data    := (others => (others => 0));
      bayer_array := (others => (others => 0));
      pixel_map :=(others =>(others => '0'));
      ns <= idle;

-------------------------------------------------------
--The idle state will wait for the mod_en signal to start its work
-------------------------------------------------------
when idle =>
  do<= '0';
    if(mod_en = '1')then
      core_out  <= 0;
      ns <= read_values;
    else
      ns <= cs;
    end if;



-------------------------------------------------------
```

```
--The values are read into the bayer_array in this state.The values
   come from the input integer_in.
----------------------------------------
  when read_values =>
      --read the image values and store them in the data array
        case read_state is
        when 0 =>
        if(x_cor = size_x-1)then
          x_cor := 0;
          read_state := 2;
        else
          ns <= read_values;
          x_cor := x_cor+1;
          read_state := 1;
        end if;

        when 1=>
          if(di = '1')then
            bayer_array(x_cor,y_cor):= core_in;
            if(y_cor = size_y-1)then
              read_state := 0;
              y_cor := 0;
            else
              y_cor := y_cor +1;
            end if;
          else
            read_state := 1;
          end if;

        when 2=>
        ns <= padding127;
        read_state:= 1;


        when others =>
         read_state := 1;
      end case;


----------------------------------------
--The padding127 state will fill the entire data array with the
   values of 127. This is in order to make a two pixel border
--when the bayer_array is mapped to the data array.The data array
   hence have the size 4 greater than the bayer_array
----------------------------------------
  when padding127 =>              --filling the entire 2D array with
     the value 127
          for a in size_x+3 downto 0 loop
          for b in size_y+3 downto 0 loop
          data(a,b):= 127;
        end loop;
      end loop;
      ns <= data_mapping;



-----------------------------------------------
--Data_mapping is where the bayer_array is mapped onto the data array
   .
--The result is when an array with the bayer_array with a two pixel
   border
--of the values 127(halv way between 0 and 255)
-----------------------------------------------
    when data_mapping =>
      for a in size_x-1 downto 0 loop        --mapping the data from
         the bayer_array to the data array
          for b in size_y-1 downto 0 loop
          data(a+2,b+2):= bayer_array(a,b);
        end loop;
      end loop;
      ns <=find_defects;



-----------------------------------------------
--This state will go through the enitire image,exept the borders,and
   place a '1' entry into the pixe_map at that possision
--if the value is classefied to be defective. For every pixel its
   eight pixels of the same color are evaluated in order to
--determine if the pixel is defective. If the middle pixel have less
   value than the smallest values - threshold or higher than
--the highest value + threshold, the pixel is classefied as defective
   , denoted by a '1' entry in the pixel_map at that position.
--The highest and lowest of the eight pixels have to be recalculated
   for every pixel.That is done with two for loops.
--When done going through the image tha next state will be correct.
-----------------------------------------------
when find_defects =>
      for j in size_y+1 downto 2 loop
        for i in size_x+1 downto 2 loop
            med_array(0)  := data(i-2,j-2);
          med_array(1)  := data(i,j-2);
          med_array(2)  := data(i+2,j-2);
          med_array(3)  := data(i-2,j+2);
          med_array(4)  := data(i,j+2);
          med_array(5)  := data(i+2,j+2);
          med_array(6)  := data(i-2,j);
          med_array(7)  := data(i+2,j);
          med_array(8)  := data(i,j);
          high:=  0;
          low:=  255;

          for count in 0 to 7 loop        --finding highest and
             lowest value
            if(med_array(count)>high)then
              high:= med_array(count);
            end if;
          end loop;

          for count in 0 to 7 loop        --finding highest and
             highest value
            if(med_array(count)<low)then
              low:= med_array(count);
            end if;
          end loop;

          if(data(i,j) > high+testing)then        --defective pixel
             with higher value
            pixel_map(i,j):= '1';      --adding the location to the
               pixel_map
          elsif(data(i,j) < low-testing)then       --defective pixel
             with lower value
            pixel_map(i,j):= '1';      --adding the location to the
               pixel_map
          end if;
        end loop;
      end loop;
        ns <= correct;


-------------------------------------
--The correct state consist of a state machine that is triggered by
   the state variable.
--The 0 and 1 state acts as a double for loop in order to go through
   the pixel_map to search for '1' entries.
--If there is a '1' entry, denoteing a defective pixel, the median
   array if filled.
--The value of the neigboring pixel is added to the median array as
   long as that position does not contain a '1' entry in
--the pixel_map.
--State 2-10 will set the enable flag, set the data_in value
   according to the values in the median array, then reset the enable
    flag
--When all the values in the median array have been sent the state
   will be 11, and will be there until the median value
--is not zero.
--At that point the median value will be placed in its proper
   position in the data array, and then set the state variable to 1;
```

```
--Being in state 12 indicates that the entire image have been
   corrected according to the '1' entries in the pixel_map
------------------------------------------
    when correct =>
      case state is

        when 0=>        --outer for loop
          if(a=size_x+2)then
            state:= 12;
            a:=0;
          else
            a := a+1;
            state:= 1;
            b:=0;
          end if;

        when 1 =>       --inner for loop
          if(b=size_y+3)then
            state:= 0;
          else
            if(pixel_map(a,b)= '1')then          --a+2,b+2
              if(pixel_map(a+2,b+2)='1')then
                med_array(0):= 127;
              else
                med_array(0)   := data(a+2,b+2);
              end if;

              if(pixel_map(a+2,b)='1')then       --a+2,b
                med_array(1):= 127;
              else
                med_array(1)   := data(a+2,b);
              end if;

              if(pixel_map(a+2,b-2)='1')then       --a+2,b-2
                med_array(2):= 127;
              else
                med_array(2)   := data(a+2,b-2);
              end if;

              if(pixel_map(a,b+2)='1')then       --a,b+2
                med_array(3):= 127;
              else
                med_array(3)   := data(a,b+2);
              end if;


                med_array(4)   := data(a,b); --a,b

              if(pixel_map(a,b-2)='1')then       --a,b-2
            med_array(5):= 127;
          else
            med_array(5)   := data(a,b-2);
          end if;

          if(pixel_map(a-2,b+2)='1')then      --a-2,b+2
            med_array(6):= 127;
          else
            med_array(6)   := data(a-2,b+2);
          end if;

          if(pixel_map(a-2,b)='1')then      --a-2,b
            med_array(7):= 127;
          else
            med_array(7)   := data(a-2,b);
          end if;

          if(pixel_map(a-2,b-2)='1')then      --a-2,b-2
            med_array(8):= 127;
          else
            med_array(8)   := data(a-2,b-2);
          end if;

          state := 2;
        else
          state:= 1;
        end if;
        b:= b+1;
      end if;



   when 2=>          --start sending data
   enable <= '1';
   data_in <= med_array(0);    --1st value
   if(timer = 2)then
     ns <= correct;
     state := 3;
     enable <= '0';
     timer := 0;
   else
     ns<= correct;
     state := 2;
     timer := timer +1;
   end if;

   when 3 =>
   enable <= '1';
```

```
data_in <= med_array(1);  --2nd value
if(timer = 2)then
  ns <= correct;
  state := 4;
  enable <= '0';
  timer := 0;
else
  ns<= correct;
  state := 3;
  timer := timer +1;
end if;

when 4 =>
enable <= '1';
data_in <= med_array(2);    --3rd value
if(timer = 2)then
  ns <= correct;
  state := 5;
  enable <= '0';
  timer := 0;
else
  ns<= correct;
  state := 4;
  timer := timer +1;
end if;

when 5 =>
enable <= '1';
data_in <= med_array(3);    --4th value
if(timer = 2)then
  ns <= correct;
  state := 6;
  enable <= '0';
  timer := 0;
else
  ns<= correct;
  state := 5;
  timer := timer +1;
end if;

when 6 =>
enable <= '1';
data_in <= med_array(4);    --5th value
if(timer = 2)then
  ns <= correct;
  state := 7;
  enable <= '0';
  timer := 0;
else
```

```
      ns<= correct;
      state := 6;
      timer := timer +1;
    end if;

    when 7 =>
    enable <= '1';
    data_in <= med_array(5);      --6th value
    if(timer = 2)then
      ns <= correct;
      state := 8;
      enable <= '0';
      timer := 0;
    else
      ns<= correct;
      state := 7;
      timer := timer +1;
    end if;

    when 8=>
    enable <= '1';
    data_in <= med_array(6);      --7th value
    if(timer = 2)then
      ns <= correct;
      state := 9;
      enable <= '0';
      timer := 0;
    else
      ns<= correct;
      state := 8;
      timer := timer +1;
    end if;

    when 9 =>
    enable <= '1';
    data_in <= med_array(7);      --8th value
    if(timer = 2)then
      ns <= correct;
      state := 10;
      enable <= '0';
      timer := 0;
    else
      ns<= correct;
      state := 9;
      timer := timer +1;
    end if;

    when 10 =>
    enable <= '1';
```

```
      data_in <= med_array(8);     --9th value
      if(timer = 2)then
        ns <= correct;
        state := 11;
        enable <= '0';
        timer := 0;
      else
        ns<= correct;
        state := 10;
        timer := timer +1;
      end if;

      when 11=>
      if(done = '1')then         --waiting for the median module to
         produce its output
        data(a,b-1):= median;    --replacing the defective value with
           the median of its neigboring pixels
        state := 1;
      else
        state := 11;
      end if;
        ns <= correct;


      when 12 =>
      ns <= de_mapping;

      when others =>
        state:= 1;
      end case;


-----------------------------------------------
--The two pixel border is removed in the de_mapping state.
--The new bayer_array has size of the original array, but with
   corrected values.
-----------------------------------------------
    when de_mapping =>
       for a in size_x-1 downto 0 loop
          for b in size_y-1 downto 0 loop
          bayer_array(a,b):=data(a+2,b+2);
         end loop;
       end loop;
       ns <= read_out;

-----------------------------------------------
--Read_out will read out the data array, one value at the time
-----------------------------------------------
    when read_out =>
```

```
         case read_state is
           when 0=>
           if(x_cor = size_x-1)then
             read_state:= 3;
             x_cor := 0;
           else
             x_cor :=x_cor+1;
             read_state:= 1;
           end if;


           when 1=>
           do <= '1';           --x_cor,y_cor
           core_out<= bayer_array(x_cor,y_cor);
           read_state:= 2;

           when 2=>

             do <= '0';
             if(y_cor = size_y-1)then
               y_cor := 0;
               read_state:= 0;
             else
               read_state := 1;
               y_cor := y_cor+1;
             end if;


           when 3=>
           ns<= idle;

           when others =>
            read_state := 1;
         end case;


--------------------------------------------
--Required state as to avioid cs signal with a wrong value(not a
    defined state)
--------------------------------------------
      when others =>
        ns <= idle;

      end case;
      end if;
    end process;
end image_core_arch;
```

Ctrl Module

```
----------------------------------------------------------
--
-- Title        : ctrl
-- Design       : defect_pixel_corr
-- Author       : Henrik Backe-Hansen
-- Company      : NTNU / Aptina Norway
--
----------------------------------------------------------
--
-- File         : ctrl.vhd
-- Generated    : Fri Mar  5 13:51:13 2010
-- From         : interface description file
-- By           : Itf2Vhdl ver. 1.20
--
----------------------------------------------------------
-- Description : Top level unit in the defective pixel correction
   algorithm.Reads
--data from RAM and inputs them to image_core, and reads the values
   from image_core
-- back to RAM
----------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.size.all;
use IEEE.numeric_std.all;

entity ctrl is
port(
clk      : in std_logic;
data_out   : out integer;
rst      :   in std_logic
);
end ctrl;

architecture ctrl_arch of ctrl is
----------------------------------------
--The following signals are used to map to the image_core
--signals to these signals in order to communicate with
--sub-modules
----------------------------------------
signal mod_en    :std_logic;
signal core_in     :integer range 0 to bit_depth;
signal di      :std_logic;
signal do      :std_logic;
signal core_out  :integer range 0 to bit_depth;
```

```
--State machine signals for the two state machines.
--The cs_r and ns_r denotes the read operation from ram while
--cs_w and ns_w denotes a write operation from ram
--------------------------------
type state_type is (idle ,one, two,start ,wait1 ,wait2);
signal cs_r,cs_w,ns_r,ns_w:state_type;

begin

--------------------------------
--Initialize the image_core module and mapping signals to the module
--------------------------------
  image_core_module:entity work.image_core(image_core_arch)
  port map(
  clk ,
  mod_en,
  rst ,
  core_in ,
  di ,
  do,
  core_out
  );


--------------------------------
--Synchrounes process that makes the state transistions on the rising
    edge of the clock
--------------------------------
  sync_proc:process(clk)
  begin
  if(rising_edge(clk))then
    if(rst = '1')then
      cs_r<= idle;
      cs_w <= idle;
    else
      cs_r <= ns_r;
      cs_w<= ns_w;
    end if;
  end if;
  end process;

--------------------------------
--This process will read all the vlaues of an image from ram into the
    image_core
--the ram module inputs and outputs only std_logic_vector so the
    values(ram_out)
--needs to be converted from std_logic_vector to integer before
    inputting
```

```
--them to the image core. The flags data in(di) also needs to be set
   in order for the iamge_core
--to know that its receiving data. The mod_en signal also needs to be
    set in order to enable the
--image_core.
--The state machine is necesarry in order to get the right timing on
   the input signals to the image_core
--Read_adr is a variable that needs to be converted to
   std_logic_vector in order to get correct
--communication between ram and image_core
------------------------------------------
  write_to_core_from_ram:process(cs_r, data_array)
  variable sent_count:natural:= 0;

  begin

    case cs_r is

      when idle =>
        sent_count := 0;
        ns_r <= start;
        mod_en <= '1';

      when start =>
      ns_r <= one;


      when one =>
        if(sent_count = ((size_x*size_y)))then
          ns_r <= start;
        else
          ns_r     <= two;
        end if;
          mod_en     <= '0';
          di       <= '0';

      when two =>
        di         <= '1';
        core_in      <= data_array(sent_count);
        sent_count     := sent_count +1;
        ns_r       <= one;

      when others =>
      ns_r <= idle;

    end case;
  end process;

------------------------------------------
```

```
--This process will write the output values from the image_core back
    to the ram module.
--There they will be stored in the ram_block array.
--Data out from the image_core is denoted by the data out(do) flag so
     cheching the flag
--will ensure that all data out values are stored to ram.
--The ram module only accepts std_logic_vector so the write_adr and
    value will be converted from
--integer to eight bit std_logic_vector.
--The process will start writing values at position 0 and end with
    position size_x * size_y -1
----------------------------
  read_from_core_to_ram:process(do,cs_w,mod_en,core_out)
  variable recv_count:natural range 0 to ((size_x*size_y)):= 0;
  begin
    case cs_w is
    when idle =>
    if(do = '1' and mod_en = '0')then
      data_out <= core_out;
      recv_count := recv_count +1;
    end if;

    when others =>
      ns_w <= idle;

    end case;

  end process;

end ctrl_arch;
```

Size Package

```
---------------------------------------------------------------------
--
-- Title       : size_package
-- Design      : defect_pixel_corr
-- Author      : Henrik Backe-Hansen
-- Company     : NTNU / Aptina Norway
--
---------------------------------------------------------------------
--
-- File        : size_package.vhd
-- Generated   : Fri Feb 26 15:09:28 2010
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.20
--
---------------------------------------------------------------------
--
-- Description :
--
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

package size is

   constant size_y      :natural := 256;       --size of the image in
      the y direction
   constant size_x      :natural := 256;       --size of the image in
      the x direction
   constant th          :natural := 50;        --threshold for detecting
      defective pixels
   constant bits        :natural :=8;          --number of bits in the
      pixelvalues (8=255)
   constant bit_depth     :natural := 2**bits;   --255 for 8 bits



   type ram_type is array(0 to 65535)of integer;
   signal data_array:ram_type;
   end size;
```

User_logic

```
------------------------------------------------------------------------------
-- user_logic.vhd - entity/architecture pair
------------------------------------------------------------------------------
--
-- ***************************************************************************
-- ** Copyright (c) 1995-2010 Xilinx, Inc.  All rights reserved.
                **
-- **
                                                                            
   **
-- ** Xilinx, Inc.
                                                                       **
-- ** XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
            **
-- ** AS A COURTESY TO YOU, SOLELY FOR USE IN DEVELOPING PROGRAMS AND
          **
-- ** SOLUTIONS FOR XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE,
          **
-- ** OR INFORMATION AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE,
          **
-- ** APPLICATION OR STANDARD, XILINX IS MAKING NO REPRESENTATION
               **
-- ** THAT THIS IMPLEMENTATION IS FREE FROM ANY CLAIMS OF
   INFRINGEMENT,       **
-- ** AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY
   REQUIRE         **
-- ** FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY
                  **
-- ** WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE
                   **
-- ** IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR
          **
-- ** REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF
          **
-- ** INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
          **
-- ** FOR A PARTICULAR PURPOSE.
                                                       **
-- **
                                                                            
   **
-- ***************************************************************************
--
------------------------------------------------------------------------------
-- Filename:          user_logic.vhd
-- Version:           1.00.a
-- Description:       User logic.
-- Date:              Fri May 28 23:21:09 2010 (by Create and Import
   Peripheral Wizard)
-- VHDL Standard:     VHDL'93
------------------------------------------------------------------
-- Naming Conventions:
--   active low signals:                    "*_n"
--   clock signals:                         "clk", "clk_div#", "clk_#
   x"
--   reset signals:                         "rst", "rst_n"
--   generics:                              "C_*"
--   user defined types:                    "*_TYPE"
--   state machine next state:              "*_ns"
--   state machine current state:           "*_cs"
--   combinatorial signals:                 "*_com"
--   pipelined or register delay signals:   "*_d#"
--   counter signals:                       "*cnt*"
--   clock enable signals:                  "*_ce"
--   internal version of output port:       "*_i"
--   device pins:                           "*_pin"
--   ports:                                 "- Names begin with
   Uppercase"
--   processes:                             "*_PROCESS"
--   component instantiations:              "<ENTITY_>I_<#|FUNC>"
-------------------------------------------------------------------


-- DO NOT EDIT BELOW THIS LINE ----------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use IEEE.NUMERIC_STD.all;

library proc_common_v3_00_a;
use proc_common_v3_00_a.proc_common_pkg.all;

-- DO NOT EDIT ABOVE THIS LINE ----------

--USER libraries added here

------------------------------------------------------------------
-- Entity section
------------------------------------------------------------------
-- Definition of Generics:
--   C_SLV_DWIDTH                 -- Slave interface data bus width
--   C_NUM_REG                    -- Number of software accessible
   registers
--   C_NUM_INTR                   -- Number of interrupt event
--
-- Definition of Ports:
```

```
--    Bus2IP_Clk                    -- Bus to IP clock
--    Bus2IP_Reset                  -- Bus to IP reset
--    Bus2IP_Data                   -- Bus to IP data bus
--    Bus2IP_BE                     -- Bus to IP byte enables
--    Bus2IP_RdCE                   -- Bus to IP read chip enable
--    Bus2IP_WrCE                   -- Bus to IP write chip enable
--    IP2Bus_Data                   -- IP to Bus data bus
--    IP2Bus_RdAck                  -- IP to Bus read transfer
   acknowledgement
--    IP2Bus_WrAck                  -- IP to Bus write transfer
   acknowledgement
--    IP2Bus_Error                  -- IP to Bus error response
--    IP2Bus_IntrEvent              -- IP to Bus interrupt event
------------------------------------------------------------


entity user_logic is
  generic
  (
    -- ADD USER GENERICS BELOW THIS LINE -----
    --USER generics added here
    -- ADD USER GENERICS ABOVE THIS LINE ----

    -- DO NOT EDIT BELOW THIS LINE ----------
    -- Bus protocol parameters, do not add to or delete
    C_SLV_DWIDTH                   : integer              := 32;
    C_NUM_REG                      : integer              := 6;
    C_NUM_INTR                     : integer              := 1
    -- DO NOT EDIT ABOVE THIS LINE ----------
  );
  port
  (
    -- ADD USER PORTS BELOW THIS LINE ---------
    --USER ports added here
    -- ADD USER PORTS ABOVE THIS LINE ------------------

    -- DO NOT EDIT BELOW THIS LINE ----------------------
    -- Bus protocol ports, do not add to or delete
    Bus2IP_Clk                     : in  std_logic;
    Bus2IP_Reset                   : in  std_logic;
    Bus2IP_Data                    : in  std_logic_vector(0 to
       C_SLV_DWIDTH-1);
    Bus2IP_BE                      : in  std_logic_vector(0 to
       C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE                    : in  std_logic_vector(0 to
       C_NUM_REG-1);
    Bus2IP_WrCE                    : in  std_logic_vector(0 to
       C_NUM_REG-1);
    IP2Bus_Data                    : out std_logic_vector(0 to
       C_SLV_DWIDTH-1);
    IP2Bus_RdAck                    : out std_logic;
    IP2Bus_WrAck                    : out std_logic;
    IP2Bus_Error                    : out std_logic;
    IP2Bus_IntrEvent                : out std_logic_vector(0 to
       C_NUM_INTR-1)
    -- DO NOT EDIT ABOVE THIS LINE -----------------------
  );

  attribute SIGIS : string;
  attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
  attribute SIGIS of Bus2IP_Reset  : signal is "RST";

end entity user_logic;

------------------------------------------------------------
-- Architecture section
------------------------------------------------------------

architecture IMP of user_logic is

  --USER signal declarations added here, as needed for user logic

  ------------------------------------------
  -- Signals for user logic slave model s/w accessible register
     example
  ------------------------------------------
  signal slv_reg0                       : std_logic_vector(0 to
     C_SLV_DWIDTH-1);--vals read by cpu
  signal slv_reg1                       : std_logic_vector(0 to
     C_SLV_DWIDTH-1);--reset write
  signal slv_reg2                       : std_logic_vector(0 to
     C_SLV_DWIDTH-1);--reset read
  signal slv_reg3                       : std_logic_vector(0 to
     C_SLV_DWIDTH-1);--di write
  signal slv_reg4                       : std_logic_vector(0 to
     C_SLV_DWIDTH-1);--di read
  signal slv_reg5                       : std_logic_vector(0 to
     C_SLV_DWIDTH-1);--value in
  signal slv_reg_write_sel              : std_logic_vector(0 to 5);
  signal slv_reg_read_sel               : std_logic_vector(0 to 5);
  signal slv_ip2bus_data                : std_logic_vector(0 to
     C_SLV_DWIDTH-1);
  signal slv_read_ack                   : std_logic;
  signal slv_write_ack                  : std_logic;

  ------------------------------------------
  -- Signals for user logic interrupt example
  ------------------------------------------
  signal intr_counter       : std_logic;
  signal reset_wr        :  std_logic;
  signal output        :  std_logic_vector(31 downto 0);
  signal clk           :std_logic;
  signal rst           :std_logic;



  component ctrl
    port(
    Bus2IP_Clk      :in std_logic;
    output       :out std_logic_vector(31 downto 0);
    Bus2IP_Reset   :in std_logic
    );
  end component;
    begin

  reset_wr        <= slv_reg0(0);


  --USER logic implementation added here
read_values:process(Bus2IP_Clk)is
variable i:integer:= 0;
begin
  if(rising_edge(Bus2IP_Clk))then
      if(Bus2IP_Reset = '1')then
        intr_counter <= '0';
      else
        intr_counter  <= '1';
        if(reset_wr = '1')then
          intr_counter <= '0';
        end if;
      end if;
      end if;

  end process;

  IP2Bus_IntrEvent(0) <= intr_counter;




  ------------------------------------------
  -- Example code to read/write user logic slave model s/w accessible
       registers
  --
  -- Note:
  -- The example code presented here is to show you one way of
      reading/writing
  -- software accessible registers implemented in the user logic
      slave model.
```

```
-- Each bit of the Bus2IP_WrCE/Bus2IP_RdCE signals is configured to
    correspond
-- to one software accessible register by the top level template.
   For example,
-- if you have four 32 bit software accessible registers in the
   user logic,
-- you are basically operating on the following memory mapped
   registers:
--
--    Bus2IP_WrCE/Bus2IP_RdCE   Memory Mapped Register
--                    "1000"    C_BASEADDR + 0x0
--                    "0100"    C_BASEADDR + 0x4
--                    "0010"    C_BASEADDR + 0x8
--                    "0001"    C_BASEADDR + 0xC
--
------------------------------------------------------------
slv_reg_write_sel <= Bus2IP_WrCE(0 to 5);
slv_reg_read_sel  <= Bus2IP_RdCE(0 to 5);
slv_write_ack     <= Bus2IP_WrCE(0) or Bus2IP_WrCE(1) or
   Bus2IP_WrCE(2) or Bus2IP_WrCE(3) or Bus2IP_WrCE(4) or
   Bus2IP_WrCE(5);
slv_read_ack      <= Bus2IP_RdCE(0) or Bus2IP_RdCE(1) or
   Bus2IP_RdCE(2) or Bus2IP_RdCE(3) or Bus2IP_RdCE(4) or
   Bus2IP_RdCE(5);

-- implement slave model software accessible register(s)
SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
begin

  if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
    if Bus2IP_Reset = '1' then
      slv_reg5 <= (others => '0');
    else
      case slv_reg_write_sel is
        when "100000" =>
          for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
            if ( Bus2IP_BE(byte_index) = '1' ) then
              slv_reg0(byte_index*8 to byte_index*8+7) <=
                 Bus2IP_Data(byte_index*8 to byte_index*8+7);
            end if;
          end loop;
        when "010000" =>
          for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
            if ( Bus2IP_BE(byte_index) = '1' ) then
              slv_reg1(byte_index*8 to byte_index*8+7) <=
                 Bus2IP_Data(byte_index*8 to byte_index*8+7);
            end if;
          end loop;
        when "001000" =>
```

```
          for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
            if ( Bus2IP_BE(byte_index) = '1' ) then
              slv_reg2(byte_index*8 to byte_index*8+7) <=
                 Bus2IP_Data(byte_index*8 to byte_index*8+7);
            end if;
          end loop;
        when "000100" =>
          for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
            if ( Bus2IP_BE(byte_index) = '1' ) then
              slv_reg3(byte_index*8 to byte_index*8+7) <=
                 Bus2IP_Data(byte_index*8 to byte_index*8+7);
            end if;
          end loop;
        when "000010" =>
          for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
            if ( Bus2IP_BE(byte_index) = '1' ) then
              slv_reg4(byte_index*8 to byte_index*8+7) <=
                 Bus2IP_Data(byte_index*8 to byte_index*8+7);
            end if;
          end loop;
        when "000001" =>
          for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
            if ( Bus2IP_BE(byte_index) = '1' ) then
              slv_reg5(byte_index*8 to byte_index*8+7) <=
                 Bus2IP_Data(byte_index*8 to byte_index*8+7);
            end if;
          end loop;
        when others => null;
      end case;
    end if;
  end if;

end process SLAVE_REG_WRITE_PROC;

-- implement slave model software accessible register(s) read mux
SLAVE_REG_READ_PROC : process( slv_reg_read_sel, slv_reg0, slv_reg1
   , slv_reg2, output, slv_reg4, slv_reg5 ) is
begin

  case slv_reg_read_sel is
    when "100000" => slv_ip2bus_data <= slv_reg0;
    when "010000" => slv_ip2bus_data <= output;
    when "001000" => slv_ip2bus_data <= slv_reg2;
    when "000100" => slv_ip2bus_data <= slv_reg3;
    when "000010" => slv_ip2bus_data <= slv_reg4;
    when "000001" => slv_ip2bus_data <= slv_reg5;
    when others => slv_ip2bus_data <= (others => '0');
  end case;
  end process SLAVE_REG_READ_PROC;



  ------------------------------------------
  -- Example code to drive IP to Bus signals
  ------------------------------------------
  IP2Bus_Data  <= slv_ip2bus_data;

  IP2Bus_WrAck <= slv_write_ack;
  IP2Bus_RdAck <= slv_read_ack;
  IP2Bus_Error <= '0';

end IMP;
```

My_custom_ip_register

```
------------------------------------------------------------------------------
-- my_custom_ip_register.vhd - entity/architecture pair
------------------------------------------------------------------------------
-- IMPORTANT:
-- DO NOT MODIFY THIS FILE EXCEPT IN THE DESIGNATED SECTIONS.
--
-- SEARCH FOR --USER TO DETERMINE WHERE CHANGES ARE ALLOWED.
--
-- TYPICALLY, THE ONLY ACCEPTABLE CHANGES INVOLVE ADDING NEW
-- PORTS AND GENERICS THAT GET PASSED THROUGH TO THE INSTANTIATION
-- OF THE USER_LOGIC ENTITY.
------------------------------------------------------------------------------
--
-- ***************************************************************************
-- ** Copyright (c) 1995-2010 Xilinx, Inc.  All rights reserved.
             **
-- **

   **
-- ** Xilinx, Inc.
                                                                         **
-- ** XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
          **
-- ** AS A COURTESY TO YOU, SOLELY FOR USE IN DEVELOPING PROGRAMS AND
         **
-- ** SOLUTIONS FOR XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE,
         **
-- ** OR INFORMATION AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE,
         **
-- ** APPLICATION OR STANDARD, XILINX IS MAKING NO REPRESENTATION
            **
-- ** THAT THIS IMPLEMENTATION IS FREE FROM ANY CLAIMS OF
   INFRINGEMENT,      **
-- ** AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY
   REQUIRE       **
-- ** FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY
               **
-- ** WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE
                **
-- ** IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR
         **
-- ** REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF
         **
-- ** INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
         **
-- ** FOR A PARTICULAR PURPOSE.
                                                            **
```

```
-- **
   **
-- ***********************************
--
------------------------------------------
-- Filename:          my_custom_ip_register.vhd
-- Version:           9.00.a
-- Description:       Top level design, instantiates library
   components and user logic.
-- Date:              Sat May 29 12:48:00 2010 (by Create and Import
   Peripheral Wizard)
-- VHDL Standard:     VHDL'93
------------------------------------------
-- Naming Conventions:
--   active low signals:                   "*_n"
--   clock signals:                        "clk", "clk_div#", "clk_#
   x"
--   reset signals:                        "rst", "rst_n"
--   generics:                             "C_*"
--   user defined types:                   "*_TYPE"
--   state machine next state:             "*_ns"
--   state machine current state:          "*_cs"
--   combinatorial signals:                "*_com"
--   pipelined or register delay signals:  "*_d#"
--   counter signals:                      "*cnt*"
--   clock enable signals:                 "*_ce"
--   internal version of output port:      "*_i"
--   device pins:                          "*_pin"
--   ports:                                "- Names begin with
   Uppercase"
--   processes:                            "*_PROCESS"
--   component instantiations:             "<ENTITY_>I_<#|FUNC>"
--------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library proc_common_v3_00_a;
use proc_common_v3_00_a.proc_common_pkg.all;
use proc_common_v3_00_a.ipif_pkg.all;

library interrupt_control_v2_01_a;
use interrupt_control_v2_01_a.interrupt_control;

library plbv46_slave_single_v1_01_a;
use plbv46_slave_single_v1_01_a.plbv46_slave_single;
```

```
library my_custom_ip_register_v9_00_a;
use my_custom_ip_register_v9_00_a.user_logic;

------------------------------------------------------------------------------
-- Entity section
------------------------------------------------------------------------------
-- Definition of Generics:
--   C_BASEADDR                   -- PLBv46 slave: base address
--   C_HIGHADDR                   -- PLBv46 slave: high address
--   C_SPLB_AWIDTH                -- PLBv46 slave: address bus width
--   C_SPLB_DWIDTH                -- PLBv46 slave: data bus width
--   C_SPLB_NUM_MASTERS           -- PLBv46 slave: Number of masters
--   C_SPLB_MID_WIDTH             -- PLBv46 slave: master ID bus
   width
--   C_SPLB_NATIVE_DWIDTH         -- PLBv46 slave: internal native
   data bus width
--   C_SPLB_P2P                   -- PLBv46 slave: point to point
   interconnect scheme
--   C_SPLB_SUPPORT_BURSTS        -- PLBv46 slave: support bursts
--   C_SPLB_SMALLEST_MASTER       -- PLBv46 slave: width of the
   smallest master
--   C_SPLB_CLK_PERIOD_PS         -- PLBv46 slave: bus clock in
   picoseconds
--   C_INCLUDE_DPHASE_TIMER       -- PLBv46 slave: Data Phase Timer
   configuration; 0 = exclude timer, 1 = include timer
--   C_FAMILY                     -- Xilinx FPGA family
--
-- Definition of Ports:
--   SPLB_Clk                     -- PLB main bus clock
--   SPLB_Rst                     -- PLB main bus reset
--   PLB_ABus                     -- PLB address bus
--   PLB_UABus                    -- PLB upper address bus
--   PLB_PAValid                  -- PLB primary address valid
   indicator
--   PLB_SAValid                  -- PLB secondary address valid
   indicator
--   PLB_rdPrim                   -- PLB secondary to primary read
   request indicator
--   PLB_wrPrim                   -- PLB secondary to primary write
   request indicator
--   PLB_masterID                 -- PLB current master identifier
--   PLB_abort                    -- PLB abort request indicator
--   PLB_busLock                  -- PLB bus lock
--   PLB_RNW                      -- PLB read/not write
--   PLB_BE                       -- PLB byte enables
--   PLB_MSize                    -- PLB master data bus size
--   PLB_size                     -- PLB transfer size
--   PLB_type                     -- PLB transfer type
--   PLB_lockErr                   -- PLB lock error indicator
--   PLB_wrDBus                    -- PLB write data bus
--   PLB_wrBurst                   -- PLB burst write transfer
   indicator
--   PLB_rdBurst                   -- PLB burst read transfer
   indicator
--   PLB_wrPendReq                 -- PLB write pending bus request
   indicator
--   PLB_rdPendReq                 -- PLB read pending bus request
   indicator
--   PLB_wrPendPri                 -- PLB write pending request
   priority
--   PLB_rdPendPri                 -- PLB read pending request
   priority
--   PLB_reqPri                    -- PLB current request priority
--   PLB_TAttribute                -- PLB transfer attribute
--   Sl_addrAck                    -- Slave address acknowledge
--   Sl_SSize                      -- Slave data bus size
--   Sl_wait                       -- Slave wait indicator
--   Sl_rearbitrate                -- Slave re-arbitrate bus indicator
--   Sl_wrDAck                     -- Slave write data acknowledge
--   Sl_wrComp                     -- Slave write transfer complete
   indicator
--   Sl_wrBTerm                    -- Slave terminate write burst
   transfer
--   Sl_rdDBus                     -- Slave read data bus
--   Sl_rdWdAddr                   -- Slave read word address
--   Sl_rdDAck                     -- Slave read data acknowledge
--   Sl_rdComp                     -- Slave read transfer complete
   indicator
--   Sl_rdBTerm                    -- Slave terminate read burst
   transfer
--   Sl_MBusy                      -- Slave busy indicator
--   Sl_MWrErr                     -- Slave write error indicator
--   Sl_MRdErr                     -- Slave read error indicator
--   Sl_MIRQ                       -- Slave interrupt indicator
--   IP2INTC_Irpt                  -- Interrupt output to processor
------------------------------------------------------------------

entity my_custom_ip_register is
  generic
  (
    -- ADD USER GENERICS BELOW THIS LINE ---------------
    --USER generics added here
    -- ADD USER GENERICS ABOVE THIS LINE ---------------

    -- DO NOT EDIT BELOW THIS LINE ---------------------
    -- Bus protocol parameters, do not add to or delete
    C_BASEADDR                    : std_logic_vector       := X"
        FFFFFFFF";
    C_HIGHADDR                    : std_logic_vector       := X"
        00000000";
    C_SPLB_AWIDTH                 : integer                := 32;
    C_SPLB_DWIDTH                 : integer                := 128;
    C_SPLB_NUM_MASTERS            : integer                := 8;
    C_SPLB_MID_WIDTH              : integer                := 3;
    C_SPLB_NATIVE_DWIDTH          : integer                := 32;
    C_SPLB_P2P                    : integer                := 0;
    C_SPLB_SUPPORT_BURSTS         : integer                := 0;
    C_SPLB_SMALLEST_MASTER        : integer                := 32;
    C_SPLB_CLK_PERIOD_PS          : integer                := 10000;
    C_INCLUDE_DPHASE_TIMER        : integer                := 1;
    C_FAMILY                      : string                 := "virtex5
        "
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
);
port
(
    -- ADD USER PORTS BELOW THIS LINE ------------------
    --USER ports added here
    -- ADD USER PORTS ABOVE THIS LINE ------------------

    -- DO NOT EDIT BELOW THIS LINE ---------------------
    -- Bus protocol ports, do not add to or delete
    SPLB_Clk                      : in   std_logic;
    SPLB_Rst                      : in   std_logic;
    PLB_ABus                      : in   std_logic_vector(0 to 31);
    PLB_UABus                     : in   std_logic_vector(0 to 31);
    PLB_PAValid                   : in   std_logic;
    PLB_SAValid                   : in   std_logic;
    PLB_rdPrim                    : in   std_logic;
    PLB_wrPrim                    : in   std_logic;
    PLB_masterID                  : in   std_logic_vector(0 to
        C_SPLB_MID_WIDTH-1);
    PLB_abort                     : in   std_logic;
    PLB_busLock                   : in   std_logic;
    PLB_RNW                       : in   std_logic;
    PLB_BE                        : in   std_logic_vector(0 to
        C_SPLB_DWIDTH/8-1);
    PLB_MSize                     : in   std_logic_vector(0 to 1);
    PLB_size                      : in   std_logic_vector(0 to 3);
    PLB_type                      : in   std_logic_vector(0 to 2);
    PLB_lockErr                   : in   std_logic;
    PLB_wrDBus                    : in   std_logic_vector(0 to
        C_SPLB_DWIDTH-1);
    PLB_wrBurst                   : in   std_logic;
    PLB_rdBurst                   : in   std_logic;
    PLB_wrPendReq                  : in   std_logic;
    PLB_rdPendReq                  : in   std_logic;
    PLB_wrPendPri                  : in   std_logic_vector(0 to 1);
    PLB_rdPendPri                  : in   std_logic_vector(0 to 1);
    PLB_reqPri                     : in   std_logic_vector(0 to 1);
    PLB_TAttribute                 : in   std_logic_vector(0 to 15);
    Sl_addrAck                     : out  std_logic;
    Sl_SSize                       : out  std_logic_vector(0 to 1);
    Sl_wait                        : out  std_logic;
    Sl_rearbitrate                 : out  std_logic;
    Sl_wrDAck                      : out  std_logic;
    Sl_wrComp                      : out  std_logic;
    Sl_wrBTerm                     : out  std_logic;
    Sl_rdDBus                      : out  std_logic_vector(0 to
       C_SPLB_DWIDTH-1);
    Sl_rdWdAddr                    : out  std_logic_vector(0 to 3);
    Sl_rdDAck                      : out  std_logic;
    Sl_rdComp                      : out  std_logic;
    Sl_rdBTerm                     : out  std_logic;
    Sl_MBusy                       : out  std_logic_vector(0 to
       C_SPLB_NUM_MASTERS-1);
    Sl_MWrErr                      : out  std_logic_vector(0 to
       C_SPLB_NUM_MASTERS-1);
    Sl_MRdErr                      : out  std_logic_vector(0 to
       C_SPLB_NUM_MASTERS-1);
    Sl_MIRQ                        : out  std_logic_vector(0 to
       C_SPLB_NUM_MASTERS-1);
    IP2INTC_Irpt                   : out  std_logic
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
  );

  attribute SIGIS : string;
  attribute SIGIS of SPLB_Clk      : signal is "CLK";
  attribute SIGIS of SPLB_Rst      : signal is "RST";
  attribute SIGIS of IP2INTC_Irpt  : signal is "INTR_LEVEL_HIGH";

end entity my_custom_ip_register;

------------------------------------------------------------------------------
-- Architecture section
------------------------------------------------------------------------------

architecture IMP of my_custom_ip_register is

  ------------------------------------------
  -- Array of base/high address pairs for each address range
  ------------------------------------------
  constant ZERO_ADDR_PAD                   : std_logic_vector(0 to 31)
     := (others => '0');
```

```
constant USER_SLV_BASEADDR              : std_logic_vector      :=
   C_BASEADDR or X"00000000";
constant USER_SLV_HIGHADDR              : std_logic_vector      :=
   C_BASEADDR or X"000000FF";
constant INTR_BASEADDR                  : std_logic_vector      :=
   C_BASEADDR or X"00000100";
constant INTR_HIGHADDR                  : std_logic_vector      :=
   C_BASEADDR or X"000001FF";

constant IPIF_ARD_ADDR_RANGE_ARRAY      : SLV64_ARRAY_TYPE      :=
  (
    ZERO_ADDR_PAD & USER_SLV_BASEADDR,  -- user logic slave space
        base address
    ZERO_ADDR_PAD & USER_SLV_HIGHADDR,  -- user logic slave space
        high address
    ZERO_ADDR_PAD & INTR_BASEADDR,      -- interrupt control space
        base address
    ZERO_ADDR_PAD & INTR_HIGHADDR       -- interrupt control space
        high address
  );

------------------------------------------------------------
-- Array of desired number of chip enables for each address range
------------------------------------------------------------
constant USER_SLV_NUM_REG               : integer               :=
    6;
constant USER_NUM_REG                   : integer               :=
   USER_SLV_NUM_REG;
constant INTR_NUM_CE                    : integer               :=
    16;

constant IPIF_ARD_NUM_CE_ARRAY          : INTEGER_ARRAY_TYPE    :=
  (
    0  => pad_power2(USER_SLV_NUM_REG), -- number of ce for user
        logic slave space
    1  => INTR_NUM_CE                   -- number of ce for
        interrupt control space
  );

------------------------------------------------------------
-- Ratio of bus clock to core clock (for use in dual clock systems)
-- 1 = ratio is 1:1
-- 2 = ratio is 2:1
------------------------------------------------------------
constant IPIF_BUS2CORE_CLK_RATIO        : integer               :=
    1;

------------------------------------------------------------
-- Width of the slave data bus (32 only)
```

```
------------------------------------------------------------------
constant USER_SLV_DWIDTH                : integer              :=
    C_SPLB_NATIVE_DWIDTH;

constant IPIF_SLV_DWIDTH                : integer              :=
    C_SPLB_NATIVE_DWIDTH;

------------------------------------------------------------------
-- Number of device level interrupts
------------------------------------------------------------------
constant INTR_NUM_IPIF_IRPT_SRC         : integer              :=
    4;

------------------------------------------------------------------
-- Capture mode for each IP interrupt (generated by user logic)
-- 1 = pass through (non-inverting)
-- 2 = pass through (inverting)
-- 3 = registered level (non-inverting)
-- 4 = registered level (inverting)
-- 5 = positive edge detect
-- 6 = negative edge detect
------------------------------------------------------------------
constant USER_NUM_INTR                  : integer              :=
    1;
constant USER_INTR_CAPTURE_MODE         : integer              :=
    1;

constant INTR_IP_INTR_MODE_ARRAY        : INTEGER_ARRAY_TYPE   :=
   (
     0   => USER_INTR_CAPTURE_MODE
   );

------------------------------------------------------------------
-- Device priority encoder feature inclusion/omission
-- true  = include priority encoder
-- false = omit priority encoder
------------------------------------------------------------------
constant INTR_INCLUDE_DEV_PENCODER      : boolean              :=
    true;

------------------------------------------------------------------
-- Device ISC feature inclusion/omission
-- true  = include device ISC
-- false = omit device ISC
------------------------------------------------------------------
constant INTR_INCLUDE_DEV_ISC           : boolean              :=
    true;

------------------------------------------------------------------
```

```
-- Index for CS/CE
------------------------------------------------------------------
constant USER_SLV_CS_INDEX               : integer               :=
    0;
constant USER_SLV_CE_INDEX               : integer               :=
    calc_start_ce_index(IPIF_ARD_NUM_CE_ARRAY, USER_SLV_CS_INDEX);
constant INTR_CS_INDEX                   : integer               :=
    1;
constant INTR_CE_INDEX                   : integer               :=
    calc_start_ce_index(IPIF_ARD_NUM_CE_ARRAY, INTR_CS_INDEX);

constant USER_CE_INDEX                   : integer               :=
    USER_SLV_CE_INDEX;

------------------------------------------------------------------
-- IP Interconnect (IPIC) signal declarations
------------------------------------------------------------------
signal ipif_Bus2IP_Clk                 : std_logic;
signal ipif_Bus2IP_Reset               : std_logic;
signal ipif_IP2Bus_Data                : std_logic_vector(0 to
    IPIF_SLV_DWIDTH-1);
signal ipif_IP2Bus_WrAck               : std_logic;
signal ipif_IP2Bus_RdAck               : std_logic;
signal ipif_IP2Bus_Error               : std_logic;
signal ipif_Bus2IP_Addr                : std_logic_vector(0 to
    C_SPLB_AWIDTH-1);
signal ipif_Bus2IP_Data                : std_logic_vector(0 to
    IPIF_SLV_DWIDTH-1);
signal ipif_Bus2IP_RNW                 : std_logic;
signal ipif_Bus2IP_BE                  : std_logic_vector(0 to
    IPIF_SLV_DWIDTH/8-1);
signal ipif_Bus2IP_CS                  : std_logic_vector(0 to ((
    IPIF_ARD_ADDR_RANGE_ARRAY'length)/2)-1);
signal ipif_Bus2IP_RdCE                : std_logic_vector(0 to
    calc_num_ce(IPIF_ARD_NUM_CE_ARRAY)-1);
signal ipif_Bus2IP_WrCE                : std_logic_vector(0 to
    calc_num_ce(IPIF_ARD_NUM_CE_ARRAY)-1);
signal intr_IPIF_Reg_Interrupts        : std_logic_vector(0 to 1);
signal intr_IPIF_Lvl_Interrupts        : std_logic_vector(0 to
    INTR_NUM_IPIF_IRPT_SRC-1);
signal intr_IP2Bus_Data                : std_logic_vector(0 to
    IPIF_SLV_DWIDTH-1);
signal intr_IP2Bus_WrAck               : std_logic;
signal intr_IP2Bus_RdAck               : std_logic;
signal intr_IP2Bus_Error               : std_logic;
signal user_Bus2IP_RdCE                : std_logic_vector(0 to
    USER_NUM_REG-1);
signal user_Bus2IP_WrCE                : std_logic_vector(0 to
    USER_NUM_REG-1);
```

```
  signal user_IP2Bus_Data                : std_logic_vector(0 to
     USER_SLV_DWIDTH-1);
  signal user_IP2Bus_RdAck               : std_logic;
  signal user_IP2Bus_WrAck               : std_logic;
  signal user_IP2Bus_Error               : std_logic;
  signal user_IP2Bus_IntrEvent           : std_logic_vector(0 to
     USER_NUM_INTR-1);

begin

  ------------------------------------------
  -- instantiate plbv46_slave_single
  ------------------------------------------
  PLBV46_SLAVE_SINGLE_I : entity plbv46_slave_single_v1_01_a.
      plbv46_slave_single
    generic map
    (
      C_ARD_ADDR_RANGE_ARRAY          => IPIF_ARD_ADDR_RANGE_ARRAY,
      C_ARD_NUM_CE_ARRAY              => IPIF_ARD_NUM_CE_ARRAY,
      C_SPLB_P2P                      => C_SPLB_P2P,
      C_BUS2CORE_CLK_RATIO            => IPIF_BUS2CORE_CLK_RATIO,
      C_SPLB_MID_WIDTH                => C_SPLB_MID_WIDTH,
      C_SPLB_NUM_MASTERS              => C_SPLB_NUM_MASTERS,
      C_SPLB_AWIDTH                   => C_SPLB_AWIDTH,
      C_SPLB_DWIDTH                   => C_SPLB_DWIDTH,
      C_SIPIF_DWIDTH                  => IPIF_SLV_DWIDTH,
      C_INCLUDE_DPHASE_TIMER          => C_INCLUDE_DPHASE_TIMER,
      C_FAMILY                        => C_FAMILY
    )
    port map
    (
      SPLB_Clk                        => SPLB_Clk,
      SPLB_Rst                        => SPLB_Rst,
      PLB_ABus                        => PLB_ABus,
      PLB_UABus                       => PLB_UABus,
      PLB_PAValid                     => PLB_PAValid,
      PLB_SAValid                     => PLB_SAValid,
      PLB_rdPrim                      => PLB_rdPrim,
      PLB_wrPrim                      => PLB_wrPrim,
      PLB_masterID                    => PLB_masterID,
      PLB_abort                       => PLB_abort,
      PLB_busLock                     => PLB_busLock,
      PLB_RNW                         => PLB_RNW,
      PLB_BE                          => PLB_BE,
      PLB_MSize                       => PLB_MSize,
      PLB_size                        => PLB_size,
      PLB_type                        => PLB_type,
      PLB_lockErr                     => PLB_lockErr,
      PLB_wrDBus                      => PLB_wrDBus,
      PLB_wrBurst                             => PLB_wrBurst,
      PLB_rdBurst                             => PLB_rdBurst,
      PLB_wrPendReq                           => PLB_wrPendReq,
      PLB_rdPendReq                           => PLB_rdPendReq,
      PLB_wrPendPri                           => PLB_wrPendPri,
      PLB_rdPendPri                           => PLB_rdPendPri,
      PLB_reqPri                              => PLB_reqPri,
      PLB_TAttribute                          => PLB_TAttribute,
      Sl_addrAck                              => Sl_addrAck,
      Sl_SSize                                => Sl_SSize,
      Sl_wait                                 => Sl_wait,
      Sl_rearbitrate                          => Sl_rearbitrate,
      Sl_wrDAck                               => Sl_wrDAck,
      Sl_wrComp                               => Sl_wrComp,
      Sl_wrBTerm                              => Sl_wrBTerm,
      Sl_rdDBus                               => Sl_rdDBus,
      Sl_rdWdAddr                             => Sl_rdWdAddr,
      Sl_rdDAck                               => Sl_rdDAck,
      Sl_rdComp                               => Sl_rdComp,
      Sl_rdBTerm                              => Sl_rdBTerm,
      Sl_MBusy                                => Sl_MBusy,
      Sl_MWrErr                               => Sl_MWrErr,
      Sl_MRdErr                               => Sl_MRdErr,
      Sl_MIRQ                                 => Sl_MIRQ,
      Bus2IP_Clk                              => ipif_Bus2IP_Clk,
      Bus2IP_Reset                            => ipif_Bus2IP_Reset,
      IP2Bus_Data                             => ipif_IP2Bus_Data,
      IP2Bus_WrAck                            => ipif_IP2Bus_WrAck,
      IP2Bus_RdAck                            => ipif_IP2Bus_RdAck,
      IP2Bus_Error                            => ipif_IP2Bus_Error,
      Bus2IP_Addr                             => ipif_Bus2IP_Addr,
      Bus2IP_Data                             => ipif_Bus2IP_Data,
      Bus2IP_RNW                              => ipif_Bus2IP_RNW,
      Bus2IP_BE                               => ipif_Bus2IP_BE,
      Bus2IP_CS                               => ipif_Bus2IP_CS,
      Bus2IP_RdCE                             => ipif_Bus2IP_RdCE,
      Bus2IP_WrCE                             => ipif_Bus2IP_WrCE
    );

------------------------------------------
-- instantiate interrupt_control
------------------------------------------
INTERRUPT_CONTROL_I : entity interrupt_control_v2_01_a.
    interrupt_control
  generic map
  (
      C_NUM_CE                                => INTR_NUM_CE,
      C_NUM_IPIF_IRPT_SRC                     => INTR_NUM_IPIF_IRPT_SRC,
      C_IP_INTR_MODE_ARRAY                    => INTR_IP_INTR_MODE_ARRAY,
      C_INCLUDE_DEV_PENCODER              => INTR_INCLUDE_DEV_PENCODER,
      C_INCLUDE_DEV_ISC                   => INTR_INCLUDE_DEV_ISC,
      C_IPIF_DWIDTH                       => IPIF_SLV_DWIDTH
    )
    port map
    (
      Bus2IP_Clk                          => ipif_Bus2IP_Clk,
      Bus2IP_Reset                        => ipif_Bus2IP_Reset,
      Bus2IP_Data                         => ipif_Bus2IP_Data,
      Bus2IP_BE                           => ipif_Bus2IP_BE,
      Interrupt_RdCE                      => ipif_Bus2IP_RdCE(
         INTR_CE_INDEX to INTR_CE_INDEX+INTR_NUM_CE-1),
      Interrupt_WrCE                      => ipif_Bus2IP_WrCE(
         INTR_CE_INDEX to INTR_CE_INDEX+INTR_NUM_CE-1),
      IPIF_Reg_Interrupts                 => intr_IPIF_Reg_Interrupts,
      IPIF_Lvl_Interrupts                 => intr_IPIF_Lvl_Interrupts,
      IP2Bus_IntrEvent                    => user_IP2Bus_IntrEvent,
      Intr2Bus_DevIntr                    => IP2INTC_Irpt,
      Intr2Bus_DBus                       => intr_IP2Bus_Data,
      Intr2Bus_WrAck                      => intr_IP2Bus_WrAck,
      Intr2Bus_RdAck                      => intr_IP2Bus_RdAck,
      Intr2Bus_Error                      => intr_IP2Bus_Error,
      Intr2Bus_Retry                      => open,
      Intr2Bus_ToutSup                    => open
    );

  -- feed registered and level-pass-through interrupts into Device
     ISC if exists, otherwise ignored
  intr_IPIF_Reg_Interrupts(0) <= '0';
  intr_IPIF_Reg_Interrupts(1) <= '0';
  intr_IPIF_Lvl_Interrupts(0) <= '0';
  intr_IPIF_Lvl_Interrupts(1) <= '0';
  intr_IPIF_Lvl_Interrupts(2) <= '0';
  intr_IPIF_Lvl_Interrupts(3) <= '0';

  ------------------------------------------
  -- instantiate User Logic
  ------------------------------------------
  USER_LOGIC_I : entity my_custom_ip_register_v9_00_a.user_logic
    generic map
    (
      -- MAP USER GENERICS BELOW THIS LINE ---------------
      --USER generics mapped here
      -- MAP USER GENERICS ABOVE THIS LINE ---------------

      C_SLV_DWIDTH                        => USER_SLV_DWIDTH,
      C_NUM_REG                           => USER_NUM_REG,
      C_NUM_INTR                          => USER_NUM_INTR
    )
    port map
    (
      -- MAP USER PORTS BELOW THIS LINE ------------------
      --USER ports mapped here
      -- MAP USER PORTS ABOVE THIS LINE ------------------

      Bus2IP_Clk                    => ipif_Bus2IP_Clk,
      Bus2IP_Reset                  => ipif_Bus2IP_Reset,
      Bus2IP_Data                   => ipif_Bus2IP_Data,
      Bus2IP_BE                     => ipif_Bus2IP_BE,
      Bus2IP_RdCE                   => user_Bus2IP_RdCE,
      Bus2IP_WrCE                   => user_Bus2IP_WrCE,
      IP2Bus_Data                   => user_IP2Bus_Data,
      IP2Bus_RdAck                  => user_IP2Bus_RdAck,
      IP2Bus_WrAck                  => user_IP2Bus_WrAck,
      IP2Bus_Error                  => user_IP2Bus_Error,
      IP2Bus_IntrEvent              => user_IP2Bus_IntrEvent
    );

  ------------------------------------------
  -- connect internal signals
  ------------------------------------------
  IP2BUS_DATA_MUX_PROC : process( ipif_Bus2IP_CS, user_IP2Bus_Data,
      intr_IP2Bus_Data ) is
  begin

    case ipif_Bus2IP_CS is
      when "10" => ipif_IP2Bus_Data <= user_IP2Bus_Data;
      when "01" => ipif_IP2Bus_Data <= intr_IP2Bus_Data;
      when others => ipif_IP2Bus_Data <= (others => '0');
    end case;

  end process IP2BUS_DATA_MUX_PROC;

  ipif_IP2Bus_WrAck <= user_IP2Bus_WrAck or intr_IP2Bus_WrAck;
  ipif_IP2Bus_RdAck <= user_IP2Bus_RdAck or intr_IP2Bus_RdAck;
  ipif_IP2Bus_Error <= user_IP2Bus_Error or intr_IP2Bus_Error;

  user_Bus2IP_RdCE <= ipif_Bus2IP_RdCE(USER_CE_INDEX to USER_CE_INDEX
      +USER_NUM_REG-1);
  user_Bus2IP_WrCE <= ipif_Bus2IP_WrCE(USER_CE_INDEX to USER_CE_INDEX
      +USER_NUM_REG-1);

end IMP;
```

.1.2 VHDL Test Benches

Median

```
--------------------------------------------------
--
-- Title        : median_tb
-- Design       : defect_pixel_corr
-- Author       : Henrik Backe-Hansen
-- Company      : NTNU / Aptina Norway
--
--------------------------------------------------
--
-- File         : median_mod_tb.vhd
-- Generated    : Tue Jan 26 23:24:31 2010
-- From         : interface description file
-- By           : Itf2Vhdl ver. 1.20
--
--------------------------------------------------
-- Description :
--This testbench the input will be entered one-by-by and compared
   with the data in a certain position in the data_golden array
--The display will show ERROR if the input value does not match the
   data_golden value for that position.
--A FAILURE message will appear if the median value is different from
    the median calculated for the data_golden array (123)
--a FAILURE message will be shown saying "simulation completed
   correctly without error" to indicate that the simulation has
   completed
--------------------------------------------------


library IEEE;
use IEEE.std_logic_1164.all;
use work.size.all;

entity median_testbench is
end entity;

architecture struct of median_testbench is
--------------------------------------------------
--Component declaration
--------------------------------------------------
component median_mod
  port(                        --component declaration
  clk       :in    std_logic;
  rst       :in    std_logic;
  data_in     :in    integer;
  enable      :in    std_logic;
  done      :out   std_logic;
  median      :out   integer
  );
  end component;

--------------------------------------------
--Signal declaration
--------------------------------------------
  signal clk     : std_logic :=   '0';
  signal rst     : std_logic :=   '0';
  signal data_in   : integer    :=    0 ;
  signal enable : std_logic :=   '0';
  signal done    : std_logic :=   '0';
  signal median : integer    :=    0 ;

  type data_format is array (8 downto 0) of integer range 0 to
     bit_depth;   --input data array size setup
  signal data_golden:data_format:=
     (127,105,141,127,127,127,127,113,115);

--------------------------------------------
--Device under test(DUT) instansiation
--------------------------------------------
begin
  DUT:median_mod
  port map(
  clk       =>   clk,
  rst       =>   rst,
  data_in     =>   data_in,
  enable      =>   enable,
  done      =>   done,
  median      =>   median
  );

--------------------------------------------
--process to generate the clock for 100MHz
--------------------------------------------
  clk_gen  :process
  begin
  wait for 5ns;
  clk <= not(clk);
  end process;

--------------------------------------------
--Stimuli
--------------------------------------------
  test_bench  :  process
  variable test:integer:= 0;
  variable test2:integer:= 0;
  begin
```

```
    rst <= '1';
    wait for 20ns;
    rst <= '0';
    wait for 20ns;

-----------------------------------------
--latching in input values
-----------------------------------------
    enable <= '1';                          --data(0) <= 115
    data_in <= 115;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_golden(0) = data_in
        report"data_golden(0) differs from input 0"
          severity error;



    enable <= '1';                          --data(1) <= 123
    data_in <= 113;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(1)
        report"data_golden(1) differs from input 1"
            severity error;


    enable <= '1';                          --data(2) <= 105
    data_in <= 127;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(2)
        report"data_golden(2) differs from input 2"
            severity error;

    enable <= '1';                          --data(3) <= 129
    data_in <= 127;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(3)
        report"data_golden(3) differs from input 3"
          severity error;


    enable <= '1';                          --data(4) <= 249
    data_in <= 127;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(4)
        report"data_golden(4) differs from input 4"
            severity error;


    enable <= '1';                      --data(5) <= 135
    data_in <= 127;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(5)
        report"data_golden(5) differs from input 5"
            severity error;


    enable <= '1';                      --data(6) <= 102
    data_in <= 141;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(6)
        report"data_golden(6) differs from input 6"
            severity error;


    enable <= '1';                      --data(7) <= 136
    data_in <= 105;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(7)
        report"data_golden(7) differs from input 7"
            severity error;

    enable <= '1';                      --data(8) <= 117
    data_in <= 127;
    wait for 20ns;
    enable <= '0';
    wait for 20ns;
      assert data_in = data_golden(8)
         report"data_golden(8) differs from input 8"
            severity error;

-----------------------------------------------
--Comparing the output with the golden output
```

```
--------------------------------------------------------------
    wait on done;
      assert median = 127
        report"wrong median value calculated"
            severity failure;


    wait for 50ns;
    assert true;
      report "simulation completed correctly without error"
        severity failure;



  wait;
  end process;


end architecture;
```

Image_core

```
-------------------------------------------------------------------------------
--
-- Title       : image_core_tb
-- Design      : defect_pixel_corr
-- Author      : Henrik Backe-Hansen
-- Company     : NTNU / Aptina Norway
--
-------------------------------------------------------------------------------
--
-- File        : image_core_tb.vhd
-- Generated   : Mon Feb 22 17:01:01 2010
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.20
--
-------------------------------------------------------------------------------
--Testbench to test the image_core.vhd file. The stimuli is fed to
   the image_core file
--value by value, row by row until all values are read.
--
-------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.size.all;
use IEEE.numeric_std.all;
use std.textio.all;
use IEEE.std_logic_unsigned.all;

entity image_core_tb is
end image_core_tb;

architecture arch of image_core_tb is

type in_array is array(0 to size_x-1)of INTEGER;
type out_array is array(0 to size_y)of INTEGER;

component image_core
  port(                          --component declaration
  clk       :in    std_logic;
  mod_en     :in    std_logic;
  rst       :in    std_logic;
  core_in   :in    integer;
  di        :in    std_logic;
  do        :out   std_logic;
  core_out   :out   integer
  );
  end component;
```

```
signal clk      :std_logic  :=  '0';
signal mod_en   :std_logic  :=  '0';
signal rst      :std_logic  :=  '0';
signal core_in  :integer   :=   0 ;
signal core_out :integer   :=   0 ;
signal di      :std_logic  :=  '0';
signal do      :std_logic  :=  '0';

begin

DUT:image_core        --device under test signal mapping
port map(
clk      =>  clk,
mod_en     =>  mod_en,
rst      =>  rst,
core_in    =>  core_in,
di       =>  di,
do       =>  do,
core_out   =>  core_out
);

clk_gen:process
begin
  wait for 5ns;
  clk <= not(clk);
end process;


read_values:process
begin

      mod_en  <= '1';
      wait for 20ns;
      mod_en <= '0';
      wait for 20ns;

    for i in 0 to ((size_y*size_x)-1) loop

      di <= '1';                         --setting the data in (di)flag
      core_in <= data_array(i);    --inputing the values to
         integer_in
      wait for 10ns;
      di<= '0';
      wait for 10ns;
    end loop;

  wait;
end process;
```

```
  write_file:process
  FILE file_out    :TEXT open WRITE_MODE IS "src\data_out_th_50.pgm";
  variable line_out  :LINE;
  variable output:out_array;
  variable i   :integer:=0;

  begin
    wait for 10ns;
    if(do='1')then
      output(i):=core_out;
      write(line_out,output(i));
      write(line_out, string'(" "));

      i:= i+1;
      if(i = size_y)then
        i:= 0;
        writeline(file_out,line_out);

      else

      end if;

    end if;

  end process;

end arch;
```

Ctrl

```
-------------------------------------------------
--
-- Title        : ctrl_tb
-- Design       : defect_pixel_corr
-- Author       : Henrik Backe-Hansen
-- Company      : NTNU / Aptina Norway
--
-------------------------------------------------
--
-- File         : ctrl_tb.vhd
-- Generated    : Mon Mar 15 21:21:11 2010
-- From         : interface description file
-- By           : Itf2Vhdl ver. 1.20
--
-------------------------------------------------
-- Description : Testbench for the ctrl module
--
-------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.all;
use std.textio.all;
use work.size.all;

entity ctrl_tb is
end ctrl_tb;

architecture ctrl_tb_arch of ctrl_tb is

type out_array is array(0 to size_y)of INTEGER;

component ctrl
  port(
  clk    :in std_logic;
  rst    :in std_logic;
  data_out:out integer
  );
end component;
signal clk        : std_logic := '0';
signal rst        : std_logic := '0';
signal data_out    : integer    := 0;

begin

  DUT:ctrl
  port map(
    clk      => clk,
    rst      =>  rst,
    data_out=> data_out
    );

clk_gen:process
begin
  wait for 5ns;
  clk <= not(clk);
end process;

write_to_file:process
FILE file_out    :TEXT open WRITE_MODE is "src\out_ctrl.pgm";
variable line_out :LINE;
variable output:out_array;
variable i:integer:=0;

begin
  wait for 100ns;

    if(data_out /= -2147483648)then
    output(i)    :=data_out;
    textio.write(line_out,output(i));
    textio.write(line_out,string'(" "));

    i := i +1;
      if(i = size_y)then
        i := 0;
        writeline(file_out,line_out);
      end if;
    end if;


end process;
end ctrl_tb_arch;
```

.2 Microblaze files

.2.1 mss

```
 PARAMETER VERSION = 2.2.0


BEGIN OS
 PARAMETER OS_NAME = standalone
 PARAMETER OS_VER = 3.00.a
 PARAMETER PROC_INSTANCE = microblaze_0
 PARAMETER stdin = RS232_Uart
 PARAMETER stdout = RS232_Uart
END


BEGIN PROCESSOR
 PARAMETER DRIVER_NAME = cpu
 PARAMETER DRIVER_VER = 1.12.b
 PARAMETER HW_INSTANCE = microblaze_0
 PARAMETER xmdstub_peripheral = RS232_Uart
END


BEGIN DRIVER
 PARAMETER DRIVER_NAME = bram
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = dlmb_cntlr
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = bram
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = ilmb_cntlr
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = generic
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = lmb_bram
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = uartlite
 PARAMETER DRIVER_VER = 1.14.a
 PARAMETER HW_INSTANCE = RS232_Uart
END
```

```
BEGIN DRIVER
 PARAMETER DRIVER_NAME = sysace
 PARAMETER DRIVER_VER = 1.12.a
 PARAMETER HW_INSTANCE = SysACE_CompactFlash
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = generic
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = clock_generator_0
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = uartlite
 PARAMETER DRIVER_VER = 1.14.a
 PARAMETER HW_INSTANCE = mdm_0
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = generic
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = proc_sys_reset_0
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = intc
 PARAMETER DRIVER_VER = 1.11.a
 PARAMETER HW_INSTANCE = xps_intc_0
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = generic
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = mb_plb
END

BEGIN DRIVER
 PARAMETER DRIVER_NAME = my_custom_ip_register
 PARAMETER DRIVER_VER = 9.00.a
 PARAMETER HW_INSTANCE = my_custom_ip_register_0
END


BEGIN LIBRARY
 PARAMETER LIBRARY_NAME = xilfatfs
 PARAMETER LIBRARY_VER = 1.00.a
 PARAMETER PROC_INSTANCE = microblaze_0
 PARAMETER CONFIG_WRITE = true
 PARAMETER CONFIG_DIR_SUPPORT = true
 PARAMETER CONFIG_FAT12 = true
END
```

.2.2 mhs

```
#
  ##########################################################################################################
# Created by Base System Builder Wizard for Xilinx EDK 11.5 Build
  EDK_LS5.70
# Sat Apr 17 13:59:14 2010
# Target Board:  Xilinx Virtex 5 ML501 Evaluation Platform Rev 1
# Family:     virtex5
# Device:     xc5vlx50
# Package:    ff676
# Speed Grade:  -1
# Processor number: 1
# Processor 1: microblaze_0
# System clock frequency: 125.0
# Debug Interface: On-Chip HW Debug Module
#
  ##########################################################################################################

 PARAMETER VERSION = 2.1.0


 PORT fpga_0_RS232_Uart_RX_pin = fpga_0_RS232_Uart_RX_pin, DIR = I
 PORT fpga_0_RS232_Uart_TX_pin = fpga_0_RS232_Uart_TX_pin, DIR = O
 PORT fpga_0_SysACE_CompactFlash_SysACE_MPA_pin =
    fpga_0_SysACE_CompactFlash_SysACE_MPA_pin, DIR = O, VEC = [6:0]
 PORT fpga_0_SysACE_CompactFlash_SysACE_CLK_pin =
    fpga_0_SysACE_CompactFlash_SysACE_CLK_pin, DIR = I
 PORT fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin =
    fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin, DIR = I
 PORT fpga_0_SysACE_CompactFlash_SysACE_CEN_pin =
    fpga_0_SysACE_CompactFlash_SysACE_CEN_pin, DIR = O
 PORT fpga_0_SysACE_CompactFlash_SysACE_OEN_pin =
    fpga_0_SysACE_CompactFlash_SysACE_OEN_pin, DIR = O
 PORT fpga_0_SysACE_CompactFlash_SysACE_WEN_pin =
    fpga_0_SysACE_CompactFlash_SysACE_WEN_pin, DIR = O
 PORT fpga_0_SysACE_CompactFlash_SysACE_MPD_pin =
    fpga_0_SysACE_CompactFlash_SysACE_MPD_pin, DIR = IO, VEC = [15:0]
 PORT fpga_0_clk_1_sys_clk_pin = dcm_clk_s, DIR = I, SIGIS = CLK,
    CLK_FREQ = 100000000
 PORT fpga_0_rst_1_sys_rst_pin = sys_rst_s, DIR = I, SIGIS = RST,
    RST_POLARITY = 0


BEGIN xps_intc
 PARAMETER INSTANCE = xps_intc_0
 PARAMETER HW_VER = 2.01.a
 PARAMETER C_BASEADDR = 0x81800000
 PARAMETER C_HIGHADDR = 0x8180ffff
 BUS_INTERFACE SPLB = mb_plb
 PORT Irq = microblaze_0_INTERRUPT
 PORT Intr = my_custom_ip_register_0_IP2INTC_Irpt
END

BEGIN proc_sys_reset
 PARAMETER INSTANCE = proc_sys_reset_0
 PARAMETER C_EXT_RESET_HIGH = 0
 PARAMETER HW_VER = 2.00.a
 PORT Slowest_sync_clk = clk_125_0000MHz
 PORT Ext_Reset_In = sys_rst_s
 PORT MB_Debug_Sys_Rst = Debug_SYS_Rst
 PORT Dcm_locked = Dcm_all_locked
 PORT MB_Reset = mb_reset
 PORT Bus_Struct_Reset = sys_bus_reset
END

BEGIN microblaze
 PARAMETER INSTANCE = microblaze_0
 PARAMETER C_INTERCONNECT = 1
 PARAMETER C_USE_FPU = 1
 PARAMETER C_DEBUG_ENABLED = 1
 PARAMETER HW_VER = 7.30.a
 BUS_INTERFACE DLMB = dlmb
 BUS_INTERFACE ILMB = ilmb
 BUS_INTERFACE DPLB = mb_plb
 BUS_INTERFACE IPLB = mb_plb
 BUS_INTERFACE DEBUG = microblaze_0_mdm_bus
 PORT MB_RESET = mb_reset
 PORT INTERRUPT = microblaze_0_INTERRUPT
END

BEGIN mdm
 PARAMETER INSTANCE = mdm_0
 PARAMETER C_MB_DBG_PORTS = 1
 PARAMETER C_USE_UART = 1
 PARAMETER C_UART_WIDTH = 8
 PARAMETER HW_VER = 1.00.g
 PARAMETER C_BASEADDR = 0x84400000
 PARAMETER C_HIGHADDR = 0x8440ffff
 BUS_INTERFACE SPLB = mb_plb
 BUS_INTERFACE MBDEBUG_0 = microblaze_0_mdm_bus
 PORT Debug_SYS_Rst = Debug_SYS_Rst
END

BEGIN plb_v46
 PARAMETER INSTANCE = mb_plb
 PARAMETER HW_VER = 1.04.a
 PORT PLB_Clk = clk_125_0000MHz
 PORT SYS_Rst = sys_bus_reset
END

BEGIN bram_block
 PARAMETER INSTANCE = lmb_bram
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = ilmb_port
 BUS_INTERFACE PORTB = dlmb_port
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = ilmb_cntlr
 PARAMETER HW_VER = 2.10.b
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x0001ffff
 BUS_INTERFACE SLMB = ilmb
 BUS_INTERFACE BRAM_PORT = ilmb_port
END

BEGIN lmb_v10
 PARAMETER INSTANCE = ilmb
 PARAMETER HW_VER = 1.00.a
 PORT LMB_Clk = clk_125_0000MHz
 PORT SYS_Rst = sys_bus_reset
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = dlmb_cntlr
 PARAMETER HW_VER = 2.10.b
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x0001ffff
 BUS_INTERFACE SLMB = dlmb
 BUS_INTERFACE BRAM_PORT = dlmb_port
END

BEGIN lmb_v10
 PARAMETER INSTANCE = dlmb
 PARAMETER HW_VER = 1.00.a
 PORT LMB_Clk = clk_125_0000MHz
 PORT SYS_Rst = sys_bus_reset
END

BEGIN clock_generator
 PARAMETER INSTANCE = clock_generator_0
 PARAMETER C_EXT_RESET_HIGH = 0
 PARAMETER C_CLKIN_FREQ = 100000000
 PARAMETER C_CLKOUT0_FREQ = 125000000
 PARAMETER C_CLKOUT0_PHASE = 0
 PARAMETER C_CLKOUT0_GROUP = NONE
 PARAMETER C_CLKOUT0_BUF = TRUE
 PARAMETER HW_VER = 3.02.a
 PORT CLKIN = dcm_clk_s
 PORT CLKOUT0 = clk_125_0000MHz
 PORT RST = sys_rst_s
 PORT LOCKED = Dcm_all_locked
END

BEGIN xps_sysace
 PARAMETER INSTANCE = SysACE_CompactFlash
 PARAMETER C_MEM_WIDTH = 16
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BASEADDR = 0x83600000
 PARAMETER C_HIGHADDR = 0x8360ffff
 BUS_INTERFACE SPLB = mb_plb
 PORT SysACE_MPA = fpga_0_SysACE_CompactFlash_SysACE_MPA_pin
 PORT SysACE_CLK = fpga_0_SysACE_CompactFlash_SysACE_CLK_pin
 PORT SysACE_MPIRQ = fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin
 PORT SysACE_CEN = fpga_0_SysACE_CompactFlash_SysACE_CEN_pin
 PORT SysACE_OEN = fpga_0_SysACE_CompactFlash_SysACE_OEN_pin
 PORT SysACE_WEN = fpga_0_SysACE_CompactFlash_SysACE_WEN_pin
 PORT SysACE_MPD = fpga_0_SysACE_CompactFlash_SysACE_MPD_pin
END

BEGIN xps_uartlite
 PARAMETER INSTANCE = RS232_Uart
 PARAMETER C_BAUDRATE = 9600
 PARAMETER C_DATA_BITS = 8
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_ODD_PARITY = 0
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BASEADDR = 0x84000000
 PARAMETER C_HIGHADDR = 0x8400ffff
 BUS_INTERFACE SPLB = mb_plb
 PORT RX = fpga_0_RS232_Uart_RX_pin
 PORT TX = fpga_0_RS232_Uart_TX_pin
END

BEGIN my_custom_ip_register
 PARAMETER INSTANCE = my_custom_ip_register_0
 PARAMETER HW_VER = 9.00.a
 PARAMETER C_BASEADDR = 0xccc00000
 PARAMETER C_HIGHADDR = 0xccc0ffff
 BUS_INTERFACE SPLB = mb_plb
 PORT IP2INTC_Irpt = my_custom_ip_register_0_IP2INTC_Irpt
END
```

.2.3 ucf

```
#  Virtex 5 ML501 Evaluation Platform
Net fpga_0_RS232_Uart_RX_pin LOC = AC7  |  IOSTANDARD=LVCMOS33;
Net fpga_0_RS232_Uart_TX_pin LOC = AD14  |  IOSTANDARD=LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<0> LOC=N6  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<1> LOC=E5  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<2> LOC=F5  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<3> LOC=F4  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<4> LOC=J5  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<5> LOC=E7  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<6> LOC=G7  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_CLK_pin LOC=AB12  |  IOSTANDARD
     = LVCMOS33  |  PERIOD = 30000 ps;
Net fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin LOC=G6  |  IOSTANDARD
     = LVCMOS33  |  TIG;
Net fpga_0_SysACE_CompactFlash_SysACE_CEN_pin LOC=F7  |  IOSTANDARD =
     LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_OEN_pin LOC=E6  |  IOSTANDARD =
     LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_WEN_pin LOC=M5  |  IOSTANDARD =
     LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<0> LOC=M6  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<1> LOC=K5  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<2> LOC=L3  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<3> LOC=L4  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<4> LOC=L7  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<5> LOC=L5  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<6> LOC=H6  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<7> LOC=G5  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<8> LOC=M7  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<9> LOC=H7  |
    IOSTANDARD = LVCMOS33;
```

```
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<10> LOC=J6  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<11> LOC=G4  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<12> LOC=K7  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<13> LOC=J4  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<14> LOC=H4  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<15> LOC=K6  |
    IOSTANDARD = LVCMOS33;
Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;
Net fpga_0_clk_1_sys_clk_pin LOC = AD8  |  IOSTANDARD=LVCMOS33;
Net fpga_0_rst_1_sys_rst_pin TIG;
Net fpga_0_rst_1_sys_rst_pin LOC = T23  |  IOSTANDARD=LVCMOS33  |
    PULLUP;
```

.2.4 xparameters.h

```
/*******************************************************************
*
* CAUTION: This file is automatically generated by libgen.
* Version: Xilinx EDK 12.1 EDK_MS1.53d
* DO NOT EDIT.
*
* Copyright (c) 1995-2010 Xilinx, Inc.  All rights reserved.

*
* Description: Driver parameters
*
*******************************************************************/

#define STDIN_BASEADDRESS 0x84000000
#define STDOUT_BASEADDRESS 0x84000000

/******************************************************************/

/* Definitions for driver UARTLITE */
#define XPAR_XUARTLITE_NUM_INSTANCES 2

/* Definitions for peripheral MDM_0 */
#define XPAR_MDM_0_BASEADDR 0x84400000
#define XPAR_MDM_0_HIGHADDR 0x8440FFFF
#define XPAR_MDM_0_DEVICE_ID 0
#define XPAR_MDM_0_BAUDRATE 0
#define XPAR_MDM_0_USE_PARITY 0
#define XPAR_MDM_0_ODD_PARITY 0
#define XPAR_MDM_0_DATA_BITS 0


/* Definitions for peripheral RS232_UART */
#define XPAR_RS232_UART_BASEADDR 0x84000000
#define XPAR_RS232_UART_HIGHADDR 0x8400FFFF
#define XPAR_RS232_UART_DEVICE_ID 1
#define XPAR_RS232_UART_BAUDRATE 9600
#define XPAR_RS232_UART_USE_PARITY 0
#define XPAR_RS232_UART_ODD_PARITY 0
#define XPAR_RS232_UART_DATA_BITS 8


/******************************************************************/


/* Canonical definitions for peripheral MDM_0 */
#define XPAR_UARTLITE_0_DEVICE_ID XPAR_MDM_0_DEVICE_ID
```

```
#define XPAR_UARTLITE_0_BASEADDR 0x84400000
#define XPAR_UARTLITE_0_HIGHADDR 0x8440FFFF
#define XPAR_UARTLITE_0_BAUDRATE 0
#define XPAR_UARTLITE_0_USE_PARITY 0
#define XPAR_UARTLITE_0_ODD_PARITY 0
#define XPAR_UARTLITE_0_DATA_BITS 0
#define XPAR_UARTLITE_0_SIO_CHAN -1


/* Canonical definitions for peripheral RS232_UART */
#define XPAR_UARTLITE_1_DEVICE_ID XPAR_RS232_UART_DEVICE_ID
#define XPAR_UARTLITE_1_BASEADDR 0x84000000
#define XPAR_UARTLITE_1_HIGHADDR 0x8400FFFF
#define XPAR_UARTLITE_1_BAUDRATE 9600
#define XPAR_UARTLITE_1_USE_PARITY 0
#define XPAR_UARTLITE_1_ODD_PARITY 0
#define XPAR_UARTLITE_1_DATA_BITS 8
#define XPAR_UARTLITE_1_SIO_CHAN -1


/******************************************************************/

#define XPAR_XSYSACE_MEM_WIDTH 16
/* Definitions for driver SYSACE */
#define XPAR_XSYSACE_NUM_INSTANCES 1

/* Definitions for peripheral SYSACE_COMPACTFLASH */
#define XPAR_SYSACE_COMPACTFLASH_DEVICE_ID 0
#define XPAR_SYSACE_COMPACTFLASH_BASEADDR 0x83600000
#define XPAR_SYSACE_COMPACTFLASH_HIGHADDR 0x8360FFFF
#define XPAR_SYSACE_COMPACTFLASH_MEM_WIDTH 16


/******************************************************************/


/* Canonical definitions for peripheral SYSACE_COMPACTFLASH */
#define XPAR_SYSACE_0_DEVICE_ID XPAR_SYSACE_COMPACTFLASH_DEVICE_ID
#define XPAR_SYSACE_0_BASEADDR 0x83600000
#define XPAR_SYSACE_0_HIGHADDR 0x8360FFFF
#define XPAR_SYSACE_0_MEM_WIDTH 16


/******************************************************************/


/* Definitions for peripheral DLMB_CNTLR */
#define XPAR_DLMB_CNTLR_BASEADDR 0x00000000
#define XPAR_DLMB_CNTLR_HIGHADDR 0x0001FFFF
```

```
/* Definitions for peripheral ILMB_CNTLR */
#define XPAR_ILMB_CNTLR_BASEADDR 0x00000000
#define XPAR_ILMB_CNTLR_HIGHADDR 0x0001FFFF


/******************************************************************/

/* Definitions for driver MY_CUSTOM_IP_REGISTER */
#define XPAR_MY_CUSTOM_IP_REGISTER_NUM_INSTANCES 1

/* Definitions for peripheral MY_CUSTOM_IP_REGISTER_0 */
#define XPAR_MY_CUSTOM_IP_REGISTER_0_DEVICE_ID 0
#define XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR 0xCCC00000
#define XPAR_MY_CUSTOM_IP_REGISTER_0_HIGHADDR 0xCCC0FFFF


/******************************************************************/

#define XPAR_INTC_MAX_NUM_INTR_INPUTS 1
#define XPAR_XINTC_HAS_IPR 1
#define XPAR_XINTC_USE_DCR 0
/* Definitions for driver INTC */
#define XPAR_XINTC_NUM_INSTANCES 1

/* Definitions for peripheral XPS_INTC_0 */
#define XPAR_XPS_INTC_0_DEVICE_ID 0
#define XPAR_XPS_INTC_0_BASEADDR 0x81800000
#define XPAR_XPS_INTC_0_HIGHADDR 0x8180FFFF
#define XPAR_XPS_INTC_0_KIND_OF_INTR 0xFFFFFFFE


/******************************************************************/

#define XPAR_INTC_SINGLE_BASEADDR 0x81800000
#define XPAR_INTC_SINGLE_HIGHADDR 0x8180FFFF
#define XPAR_INTC_SINGLE_DEVICE_ID XPAR_XPS_INTC_0_DEVICE_ID
#define XPAR_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_MASK 0X000001
#define XPAR_XPS_INTC_0_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_INTR 0

/******************************************************************/

/* Canonical definitions for peripheral XPS_INTC_0 */
#define XPAR_INTC_0_DEVICE_ID XPAR_XPS_INTC_0_DEVICE_ID
#define XPAR_INTC_0_BASEADDR 0x81800000
#define XPAR_INTC_0_HIGHADDR 0x8180FFFF
#define XPAR_INTC_0_KIND_OF_INTR 0xFFFFFFFE
```

```
#define XPAR_INTC_0_MY_CUSTOM_IP_REGISTER_0_VEC_ID
   XPAR_XPS_INTC_0_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_INTR

/******************************************************************/

/* Definitions for bus frequencies */
#define XPAR_CPU_DPLB_FREQ_HZ 125000000
#define XPAR_CPU_IPLB_FREQ_HZ 125000000
/******************************************************************/

/* Canonical definitions for bus frequencies */
#define XPAR_PROC_BUS_0_FREQ_HZ 125000000
/******************************************************************/

#define XPAR_CPU_CORE_CLOCK_FREQ_HZ 125000000
#define XPAR_MICROBLAZE_CORE_CLOCK_FREQ_HZ 125000000

/******************************************************************/


/* Definitions for peripheral MICROBLAZE_0 */
#define XPAR_MICROBLAZE_0_SCO 0
#define XPAR_MICROBLAZE_0_FREQ 125000000
#define XPAR_MICROBLAZE_0_DATA_SIZE 32
#define XPAR_MICROBLAZE_0_DYNAMIC_BUS_SIZING 1
#define XPAR_MICROBLAZE_0_AREA_OPTIMIZED 0
#define XPAR_MICROBLAZE_0_OPTIMIZATION 0
#define XPAR_MICROBLAZE_0_INTERCONNECT 1
#define XPAR_MICROBLAZE_0_DPLB_DWIDTH 32
#define XPAR_MICROBLAZE_0_DPLB_NATIVE_DWIDTH 32
#define XPAR_MICROBLAZE_0_DPLB_BURST_EN 0
#define XPAR_MICROBLAZE_0_DPLB_P2P 0
#define XPAR_MICROBLAZE_0_IPLB_DWIDTH 32
#define XPAR_MICROBLAZE_0_IPLB_NATIVE_DWIDTH 32
#define XPAR_MICROBLAZE_0_IPLB_BURST_EN 0
#define XPAR_MICROBLAZE_0_IPLB_P2P 0
#define XPAR_MICROBLAZE_0_D_PLB 1
#define XPAR_MICROBLAZE_0_D_LMB 1
#define XPAR_MICROBLAZE_0_I_PLB 1
#define XPAR_MICROBLAZE_0_I_LMB 1
#define XPAR_MICROBLAZE_0_USE_MSR_INSTR 1
#define XPAR_MICROBLAZE_0_USE_PCMP_INSTR 1
#define XPAR_MICROBLAZE_0_USE_BARREL 0
#define XPAR_MICROBLAZE_0_USE_DIV 0
#define XPAR_MICROBLAZE_0_USE_HW_MUL 1
#define XPAR_MICROBLAZE_0_USE_FPU 1
#define XPAR_MICROBLAZE_0_UNALIGNED_EXCEPTIONS 0
#define XPAR_MICROBLAZE_0_ILL_OPCODE_EXCEPTION 0
#define XPAR_MICROBLAZE_0_IPLB_BUS_EXCEPTION 0
```

```
#define XPAR_MICROBLAZE_0_DPLB_BUS_EXCEPTION 0
#define XPAR_MICROBLAZE_0_DIV_ZERO_EXCEPTION 0
#define XPAR_MICROBLAZE_0_FPU_EXCEPTION 0
#define XPAR_MICROBLAZE_0_FSL_EXCEPTION 0
#define XPAR_MICROBLAZE_0_PVR 0
#define XPAR_MICROBLAZE_0_PVR_USER1 0x00
#define XPAR_MICROBLAZE_0_PVR_USER2 0x00000000
#define XPAR_MICROBLAZE_0_DEBUG_ENABLED 1
#define XPAR_MICROBLAZE_0_NUMBER_OF_PC_BRK 1
#define XPAR_MICROBLAZE_0_NUMBER_OF_RD_ADDR_BRK 0
#define XPAR_MICROBLAZE_0_NUMBER_OF_WR_ADDR_BRK 0
#define XPAR_MICROBLAZE_0_INTERRUPT_IS_EDGE 0
#define XPAR_MICROBLAZE_0_EDGE_IS_POSITIVE 1
#define XPAR_MICROBLAZE_0_RESET_MSR 0x00000000
#define XPAR_MICROBLAZE_0_OPCODE_0X0_ILLEGAL 0
#define XPAR_MICROBLAZE_0_FSL_LINKS 0
#define XPAR_MICROBLAZE_0_FSL_DATA_SIZE 32
#define XPAR_MICROBLAZE_0_USE_EXTENDED_FSL_INSTR 0
#define XPAR_MICROBLAZE_0_ICACHE_BASEADDR 0x00000000
#define XPAR_MICROBLAZE_0_ICACHE_HIGHADDR 0x3FFFFFFF
#define XPAR_MICROBLAZE_0_USE_ICACHE 0
#define XPAR_MICROBLAZE_0_ALLOW_ICACHE_WR 1
#define XPAR_MICROBLAZE_0_ADDR_TAG_BITS 0
#define XPAR_MICROBLAZE_0_CACHE_BYTE_SIZE 8192
#define XPAR_MICROBLAZE_0_ICACHE_USE_FSL 1
#define XPAR_MICROBLAZE_0_ICACHE_LINE_LEN 4
#define XPAR_MICROBLAZE_0_ICACHE_ALWAYS_USED 0
#define XPAR_MICROBLAZE_0_ICACHE_INTERFACE 0
#define XPAR_MICROBLAZE_0_ICACHE_VICTIMS 0
#define XPAR_MICROBLAZE_0_ICACHE_STREAMS 0
#define XPAR_MICROBLAZE_0_DCACHE_BASEADDR 0x00000000
#define XPAR_MICROBLAZE_0_DCACHE_HIGHADDR 0x3FFFFFFF
#define XPAR_MICROBLAZE_0_USE_DCACHE 0
#define XPAR_MICROBLAZE_0_ALLOW_DCACHE_WR 1
#define XPAR_MICROBLAZE_0_DCACHE_ADDR_TAG 0
#define XPAR_MICROBLAZE_0_DCACHE_BYTE_SIZE 8192
#define XPAR_MICROBLAZE_0_DCACHE_USE_FSL 1
#define XPAR_MICROBLAZE_0_DCACHE_LINE_LEN 4
#define XPAR_MICROBLAZE_0_DCACHE_ALWAYS_USED 0
#define XPAR_MICROBLAZE_0_DCACHE_INTERFACE 0
#define XPAR_MICROBLAZE_0_DCACHE_USE_WRITEBACK 0
#define XPAR_MICROBLAZE_0_DCACHE_VICTIMS 0
#define XPAR_MICROBLAZE_0_USE_MMU 0
#define XPAR_MICROBLAZE_0_MMU_DTLB_SIZE 4
#define XPAR_MICROBLAZE_0_MMU_ITLB_SIZE 2
#define XPAR_MICROBLAZE_0_MMU_TLB_ACCESS 3
#define XPAR_MICROBLAZE_0_MMU_ZONES 16
#define XPAR_MICROBLAZE_0_USE_INTERRUPT 1
#define XPAR_MICROBLAZE_0_USE_EXT_BRK 1
```

```
#define XPAR_MICROBLAZE_0_USE_EXT_NM_BRK 1
#define XPAR_MICROBLAZE_0_USE_BRANCH_TARGET_CACHE 0
#define XPAR_MICROBLAZE_0_BRANCH_TARGET_CACHE_SIZE 0

/******************************************************************/

#define XPAR_CPU_ID 0
#define XPAR_MICROBLAZE_ID 0
#define XPAR_MICROBLAZE_SCO 0
#define XPAR_MICROBLAZE_FREQ 125000000
#define XPAR_MICROBLAZE_DATA_SIZE 32
#define XPAR_MICROBLAZE_DYNAMIC_BUS_SIZING 1
#define XPAR_MICROBLAZE_AREA_OPTIMIZED 0
#define XPAR_MICROBLAZE_OPTIMIZATION 0
#define XPAR_MICROBLAZE_INTERCONNECT 1
#define XPAR_MICROBLAZE_DPLB_DWIDTH 32
#define XPAR_MICROBLAZE_DPLB_NATIVE_DWIDTH 32
#define XPAR_MICROBLAZE_DPLB_BURST_EN 0
#define XPAR_MICROBLAZE_DPLB_P2P 0
#define XPAR_MICROBLAZE_IPLB_DWIDTH 32
#define XPAR_MICROBLAZE_IPLB_NATIVE_DWIDTH 32
#define XPAR_MICROBLAZE_IPLB_BURST_EN 0
#define XPAR_MICROBLAZE_IPLB_P2P 0
#define XPAR_MICROBLAZE_D_PLB 1
#define XPAR_MICROBLAZE_D_LMB 1
#define XPAR_MICROBLAZE_I_PLB 1
#define XPAR_MICROBLAZE_I_LMB 1
#define XPAR_MICROBLAZE_USE_MSR_INSTR 1
#define XPAR_MICROBLAZE_USE_PCMP_INSTR 1
#define XPAR_MICROBLAZE_USE_BARREL 0
#define XPAR_MICROBLAZE_USE_DIV 0
#define XPAR_MICROBLAZE_USE_HW_MUL 1
#define XPAR_MICROBLAZE_USE_FPU 1
#define XPAR_MICROBLAZE_UNALIGNED_EXCEPTIONS 0
#define XPAR_MICROBLAZE_ILL_OPCODE_EXCEPTION 0
#define XPAR_MICROBLAZE_IPLB_BUS_EXCEPTION 0
#define XPAR_MICROBLAZE_DPLB_BUS_EXCEPTION 0
#define XPAR_MICROBLAZE_DIV_ZERO_EXCEPTION 0
#define XPAR_MICROBLAZE_FPU_EXCEPTION 0
#define XPAR_MICROBLAZE_FSL_EXCEPTION 0
#define XPAR_MICROBLAZE_PVR 0
#define XPAR_MICROBLAZE_PVR_USER1 0x00
#define XPAR_MICROBLAZE_PVR_USER2 0x00000000
#define XPAR_MICROBLAZE_DEBUG_ENABLED 1
#define XPAR_MICROBLAZE_NUMBER_OF_PC_BRK 1
#define XPAR_MICROBLAZE_NUMBER_OF_RD_ADDR_BRK 0
#define XPAR_MICROBLAZE_NUMBER_OF_WR_ADDR_BRK 0
#define XPAR_MICROBLAZE_INTERRUPT_IS_EDGE 0
#define XPAR_MICROBLAZE_EDGE_IS_POSITIVE 1
```

```
#define XPAR_MICROBLAZE_RESET_MSR 0x00000000
#define XPAR_MICROBLAZE_OPCODE_0X0_ILLEGAL 0
#define XPAR_MICROBLAZE_FSL_LINKS 0
#define XPAR_MICROBLAZE_FSL_DATA_SIZE 32
#define XPAR_MICROBLAZE_USE_EXTENDED_FSL_INSTR 0
#define XPAR_MICROBLAZE_ICACHE_BASEADDR 0x00000000
#define XPAR_MICROBLAZE_ICACHE_HIGHADDR 0x3FFFFFFF
#define XPAR_MICROBLAZE_USE_ICACHE 0
#define XPAR_MICROBLAZE_ALLOW_ICACHE_WR 1
#define XPAR_MICROBLAZE_ADDR_TAG_BITS 0
#define XPAR_MICROBLAZE_CACHE_BYTE_SIZE 8192
#define XPAR_MICROBLAZE_ICACHE_USE_FSL 1
#define XPAR_MICROBLAZE_ICACHE_LINE_LEN 4
#define XPAR_MICROBLAZE_ICACHE_ALWAYS_USED 0
#define XPAR_MICROBLAZE_ICACHE_INTERFACE 0
#define XPAR_MICROBLAZE_ICACHE_VICTIMS 0
#define XPAR_MICROBLAZE_ICACHE_STREAMS 0
#define XPAR_MICROBLAZE_DCACHE_BASEADDR 0x00000000
#define XPAR_MICROBLAZE_DCACHE_HIGHADDR 0x3FFFFFFF
#define XPAR_MICROBLAZE_USE_DCACHE 0
#define XPAR_MICROBLAZE_ALLOW_DCACHE_WR 1
#define XPAR_MICROBLAZE_DCACHE_ADDR_TAG 0
#define XPAR_MICROBLAZE_DCACHE_BYTE_SIZE 8192
#define XPAR_MICROBLAZE_DCACHE_USE_FSL 1
#define XPAR_MICROBLAZE_DCACHE_LINE_LEN 4
#define XPAR_MICROBLAZE_DCACHE_ALWAYS_USED 0
#define XPAR_MICROBLAZE_DCACHE_INTERFACE 0
#define XPAR_MICROBLAZE_DCACHE_USE_WRITEBACK 0
#define XPAR_MICROBLAZE_DCACHE_VICTIMS 0
#define XPAR_MICROBLAZE_USE_MMU 0
#define XPAR_MICROBLAZE_MMU_DTLB_SIZE 4
#define XPAR_MICROBLAZE_MMU_ITLB_SIZE 2
#define XPAR_MICROBLAZE_MMU_TLB_ACCESS 3
#define XPAR_MICROBLAZE_MMU_ZONES 16
#define XPAR_MICROBLAZE_USE_INTERRUPT 1
#define XPAR_MICROBLAZE_USE_EXT_BRK 1
#define XPAR_MICROBLAZE_USE_EXT_NM_BRK 1
#define XPAR_MICROBLAZE_USE_BRANCH_TARGET_CACHE 0
#define XPAR_MICROBLAZE_BRANCH_TARGET_CACHE_SIZE 0
#define XPAR_MICROBLAZE_HW_VER "7.30.a"

/******************************************************************/

#define XILFATFS_MAXFILES 5
#define XILFATFS_BUFCACHE_SIZE 10240
```

.3 Microblaze Synthesize

```
Release 12.1 - xst M.53d (nt)
Copyright (c) 1995-2010 Xilinx, Inc.  All rights reserved.
-->
TABLE OF CONTENTS
  1) Synthesis Options Summary
  2) HDL Compilation
  3) Design Hierarchy Analysis
  4) HDL Analysis
  5) HDL Synthesis
     5.1) HDL Synthesis Report
  6) Advanced HDL Synthesis
     6.1) Advanced HDL Synthesis Report
  7) Low Level Synthesis
  8) Partition Report
  9) Final Report
  9.1) Device utilization summary
  9.2) Partition Resource Summary
  9.3) TIMING REPORT


=========================================================================

*                      Synthesis Options Summary
                          *
=========================================================================

---- Source Parameters
Input Format                       : MIXED
Input File Name                    : "system_xst.prj"
Verilog Include Directory          : {"C:\Users\Magician\Desktop\
   bsb_project\pcores\" "C:\Xilinx\12.1\ISE_DS\EDK\hw\
   XilinxProcessorIPLib\pcores\" }

---- Target Parameters
Target Device                      : xc5vlx50ff676-1
Output File Name                   : "../implementation/system.ngc"

---- Source Options
Top Module Name                    : system

---- Target Options
Add IO Buffers                     : YES
Global Maximum Fanout              : 10000

---- General Options
Optimization Goal                  : speed
Netlist Hierarchy                  : as_optimized
```

```
Optimization Effort                : 1
Hierarchy Separator                : /

---- Other Options
Cores Search Directories           : {../implementation}

=========================================================================


WARNING:UtilitiesC - The message filter file "../__xps/ise/filter.
    filter" specified with the -filter switch can't be found.
WARNING:UtilitiesC - The message filter file "../__xps/ise/filter.
    filter" specified with the -filter switch can't be found.

=========================================================================

*                          HDL Compilation
                                *
=========================================================================

Compiling vhdl file "C:/Users/Magician/Desktop/bsb_project/hdl/system
    .vhd" in Library work.
Entity <system> compiled.
Entity <system> (Architecture <STRUCTURE>) compiled.

=========================================================================

*                    Design Hierarchy Analysis
                             *
=========================================================================

Analyzing hierarchy for entity <system> in library <work> (
    architecture <STRUCTURE>).



=========================================================================

*                           HDL Analysis
                                 *
=========================================================================

Analyzing Entity <system> in library <work> (Architecture <STRUCTURE
    >).
     Set property "BUFFER_TYPE = BUFGP" for signal <
        fpga_0_SysACE_CompactFlash_SysACE_CLK_pin> in unit <system>.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'RstcPPCresetcore_0' of
    component 'proc_sys_reset_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'RstcPPCresetchip_0' of
    component 'proc_sys_reset_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'RstcPPCresetsys_0' of
    component 'proc_sys_reset_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'RstcPPCresetcore_1' of
    component 'proc_sys_reset_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'RstcPPCresetchip_1' of
    component 'proc_sys_reset_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'RstcPPCresetsys_1' of
    component 'proc_sys_reset_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1191: Unconnected output port 'Peripheral_Reset' of
    component 'proc_sys_reset_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'MB_Halted' of component '
    microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Instruction' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Valid_Instr' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_PC' of component '
    microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Reg_Write' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Reg_Addr' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_MSR_Reg' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_PID_Reg' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_New_Reg_Value' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Exception_Taken' of
     component 'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Exception_Kind' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Jump_Taken' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Delay_Slot' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Data_Address' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Data_Access' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Data_Read' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Data_Write' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Data_Write_Value'
    of component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Data_Byte_Enable'
    of component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_DCache_Req' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_DCache_Hit' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_DCache_Rdy' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_DCache_Read' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_ICache_Req' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_ICache_Hit' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_ICache_Rdy' of
    component 'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_OF_PipeRun' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_EX_PipeRun' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_MEM_PipeRun' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_MB_Halted' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'Trace_Jump_Hit' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL0_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL0_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL0_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL0_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL0_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL0_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL1_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL1_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL1_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL1_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL1_M_DATA' of component
     'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL1_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL2_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL2_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL2_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL2_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL2_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL2_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL3_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL3_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL3_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL3_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL3_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL3_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL4_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL4_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL4_M_CLK' of component
    'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL4_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL4_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL4_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL5_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL5_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL5_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL5_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL5_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL5_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL6_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL6_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL6_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL6_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL6_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL6_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL7_S_CLK' of component
    'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL7_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL7_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL7_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL7_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL7_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL8_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL8_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL8_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL8_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL8_M_DATA' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL8_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL9_S_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL9_S_READ' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL9_M_CLK' of component
    'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL9_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL9_M_DATA' of component
     'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL9_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL10_S_CLK' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL10_S_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL10_M_CLK' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL10_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL10_M_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL10_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL11_S_CLK' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL11_S_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL11_M_CLK' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL11_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL11_M_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL11_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL12_S_CLK' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL12_S_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL12_M_CLK' of component
     'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL12_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL12_M_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL12_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL13_S_CLK' of component
      'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL13_S_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL13_M_CLK' of component
      'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL13_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL13_M_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL13_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL14_S_CLK' of component
      'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL14_S_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL14_M_CLK' of component
      'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL14_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL14_M_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL14_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL15_S_CLK' of component
      'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL15_S_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL15_M_CLK' of component
     'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL15_M_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL15_M_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'FSL15_M_CONTROL' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'ICACHE_FSL_IN_CLK' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'ICACHE_FSL_IN_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'ICACHE_FSL_OUT_CLK' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'ICACHE_FSL_OUT_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'ICACHE_FSL_OUT_DATA' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'ICACHE_FSL_OUT_CONTROL'
    of component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'DCACHE_FSL_IN_CLK' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'DCACHE_FSL_IN_READ' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'DCACHE_FSL_OUT_CLK' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'DCACHE_FSL_OUT_WRITE' of
    component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'DCACHE_FSL_OUT_DATA' of
    component 'microblaze_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1215: Unconnected output port 'DCACHE_FSL_OUT_CONTROL'
    of component 'microblaze_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Interrupt' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'MDM_DBus' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'MDM_errAck' of component
    'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'MDM_retry' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'MDM_toutSup' of component
     'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'MDM_xferAck' of component
     'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_1' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_1' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_1' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_1' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_1' of component
     'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_1' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_1' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_2' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_2' of component '
    mdm_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_2' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_2' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_2' of component
     'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_2' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_2' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_3' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_3' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_3' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_3' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_3' of component
     'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_3' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_3' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_4' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_4' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_4' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_4' of
    component 'mdm_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_4' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_4' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_4' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_5' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_5' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_5' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_5' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_5' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_5' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_5' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_6' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_6' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_6' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_6' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_6' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_6' of
    component 'mdm_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_6' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Clk_7' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_TDI_7' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Reg_En_7' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Capture_7' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Shift_7' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Update_7' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Dbg_Rst_7' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_tdi' of component '
    mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_reset' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_shift' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_update' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_capture' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_sel1' of component
    'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'bscan_drck1' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'FSL0_S_CLK' of component
    'mdm_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'FSL0_S_READ' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'FSL0_M_CLK' of component
    'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'FSL0_M_WRITE' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'FSL0_M_DATA' of component
      'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'FSL0_M_CONTROL' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_DRCK' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_RESET' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_SEL' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_CAPTURE' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_SHIFT' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_UPDATE' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1520: Unconnected output port 'Ext_JTAG_TDI' of
    component 'mdm_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_Rst' of component '
    mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'MPLB_Rst' of component '
    mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_dcrAck' of component
    'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_dcrDBus' of component
      'mb_plb_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SaddrAck' of
    component 'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SMRdErr' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SMWrErr' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SMBusy' of component
    'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SrdBTerm' of
    component 'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SrdComp' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SrdDAck' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SrdDBus' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SrdWdAddr' of
    component 'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_Srearbitrate' of
    component 'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_Sssize' of component
    'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_Swait' of component '
    mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SwrBTerm' of
    component 'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SwrComp' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'PLB_SwrDAck' of component
     'mb_plb_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1673: Unconnected output port 'Bus_Error_Det' of
    component 'mb_plb_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT1' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT2' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT3' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT4' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT5' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT6' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT7' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT8' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT9' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT10' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT11' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT12' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT13' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT14' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKOUT15' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'CLKFBOUT' of component '
    clock_generator_0_wrapper'.
```

```
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1880: Unconnected output port 'PSDONE' of component '
    clock_generator_0_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1909: Unconnected output port 'SysACE_IRQ' of component
    'sysace_compactflash_wrapper'.
WARNING:Xst:753 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 1965: Unconnected output port 'Interrupt' of component '
    rs232_uart_wrapper'.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2061: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2069: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2077: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2085: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2093: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2101: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2109: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2117: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2125: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2133: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2141: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2149: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2157: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2165: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2173: Instantiating black box module <IOBUF>.
WARNING:Xst:2211 - "C:/Users/Magician/Desktop/bsb_project/hdl/system.
    vhd" line 2181: Instantiating black box module <IOBUF>.
 Entity <system> analyzed. Unit <system> generated.


=========================================================================
*                            HDL Synthesis
                                  *
```

```
=========================================================================



Performing bidirectional port resolution...

Synthesizing Unit <system>.
    Related source file is "C:/Users/Magician/Desktop/bsb_project/hdl
        /system.vhd".
Unit <system> synthesized.



=========================================================================

HDL Synthesis Report

Found no macro
=========================================================================


=========================================================================

*                           Advanced HDL Synthesis
                                  *
=========================================================================



Reading core <../implementation/xps_intc_0_wrapper.ngc>.
Reading core <../implementation/proc_sys_reset_0_wrapper.ngc>.
Reading core <../implementation/microblaze_0_wrapper.ngc>.
Reading core <../implementation/mdm_0_wrapper.ngc>.
Reading core <../implementation/mb_plb_wrapper.ngc>.
Reading core <../implementation/lmb_bram_wrapper.ngc>.
Reading core <../implementation/ilmb_cntlr_wrapper.ngc>.
Reading core <../implementation/ilmb_wrapper.ngc>.
Reading core <../implementation/dlmb_cntlr_wrapper.ngc>.
Reading core <../implementation/dlmb_wrapper.ngc>.
Reading core <../implementation/clock_generator_0_wrapper.ngc>.
Reading core <../implementation/sysace_compactflash_wrapper.ngc>.
Reading core <../implementation/rs232_uart_wrapper.ngc>.
Reading core <../implementation/my_custom_ip_register_0_wrapper.ngc>.
Loading core <xps_intc_0_wrapper> for timing and area information for
    instance <xps_intc_0>.
Loading core <proc_sys_reset_0_wrapper> for timing and area
   information for instance <proc_sys_reset_0>.
Loading core <microblaze_0_wrapper> for timing and area information
   for instance <microblaze_0>.
Loading core <mdm_0_wrapper> for timing and area information for
   instance <mdm_0>.
```

```
Loading core <mb_plb_wrapper> for timing and area information for
   instance <mb_plb>.
Loading core <lmb_bram_wrapper> for timing and area information for
   instance <lmb_bram>.
Loading core <ilmb_cntlr_wrapper> for timing and area information for
    instance <ilmb_cntlr>.
Loading core <ilmb_wrapper> for timing and area information for
   instance <ilmb>.
Loading core <dlmb_cntlr_wrapper> for timing and area information for
    instance <dlmb_cntlr>.
Loading core <dlmb_wrapper> for timing and area information for
   instance <dlmb>.
Loading core <clock_generator_0_wrapper> for timing and area
   information for instance <clock_generator_0>.
Loading core <sysace_compactflash_wrapper> for timing and area
   information for instance <SysACE_CompactFlash>.
Loading core <rs232_uart_wrapper> for timing and area information for
    instance <RS232_Uart>.
Loading core <my_custom_ip_register_0_wrapper> for timing and area
   information for instance <my_custom_ip_register_0>.

==========================================================================

Advanced HDL Synthesis Report

Found no macro
==========================================================================


==========================================================================

*                          Low Level Synthesis
                               *
==========================================================================


Optimizing unit <system> ...

Mapping all equations...
Building and optimizing final netlist ...
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_MPLB_RST[1].I_MPLB_RST> in
   Unit <mb_plb> is equivalent to the following 7 FFs/Latches : <
   mb_plb/GEN_MPLB_RST[0].I_MPLB_RST> <mb_plb/GEN_SPLB_RST[4].
   I_SPLB_RST> <mb_plb/GEN_SPLB_RST[3].I_SPLB_RST> <mb_plb/
   GEN_SPLB_RST[2].I_SPLB_RST> <mb_plb/GEN_SPLB_RST[1].I_SPLB_RST> <
   mb_plb/GEN_SPLB_RST[0].I_SPLB_RST> <mb_plb/I_PLB_RST>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
   I_ARBCONTROL_SM/arbSecRdInProgReg_i> in Unit <mb_plb> is
   equivalent to the following 2 FFs/Latches : <mb_plb/GEN_SHARED.
```

```
    I_PLB_ARBITER_LOGIC/I_ARBCONTROL_SM/arbSecRdInProgReg_i_1> <mb_plb
    /GEN_SHARED.I_PLB_ARBITER_LOGIC/I_ARBCONTROL_SM/
    arbSecRdInProgReg_i_2>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_1> in Unit <mb_plb> is equivalent to
     the following FF/Latch : <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_1_1>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_0> in Unit <mb_plb> is equivalent to
     the following FF/Latch : <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_0_1>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARBCONTROL_SM/arbctrl_sm_cs_FSM_FFd3> in Unit <mb_plb> is
    equivalent to the following FF/Latch : <mb_plb/GEN_SHARED.
    I_PLB_ARBITER_LOGIC/I_ARBCONTROL_SM/arbctrl_sm_cs_FSM_FFd3_1>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARBCONTROL_SM/arbSecRdInProgReg_i> in Unit <mb_plb> is
    equivalent to the following 2 FFs/Latches : <mb_plb/GEN_SHARED.
    I_PLB_ARBITER_LOGIC/I_ARBCONTROL_SM/arbSecRdInProgReg_i_1> <mb_plb
    /GEN_SHARED.I_PLB_ARBITER_LOGIC/I_ARBCONTROL_SM/
    arbSecRdInProgReg_i_2>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_0> in Unit <mb_plb> is equivalent to
     the following FF/Latch : <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_0_1>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_1> in Unit <mb_plb> is equivalent to
     the following FF/Latch : <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARB_ENCODER/arbAddrSelReg_i_1_1>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_SHARED.I_PLB_ARBITER_LOGIC/
    I_ARBCONTROL_SM/arbctrl_sm_cs_FSM_FFd3> in Unit <mb_plb> is
    equivalent to the following FF/Latch : <mb_plb/GEN_SHARED.
    I_PLB_ARBITER_LOGIC/I_ARBCONTROL_SM/arbctrl_sm_cs_FSM_FFd3_1>
INFO:Xst:2260 - The FF/Latch <mb_plb/GEN_MPLB_RST[1].I_MPLB_RST> in
    Unit <mb_plb> is equivalent to the following 7 FFs/Latches : <
    mb_plb/GEN_MPLB_RST[0].I_MPLB_RST> <mb_plb/GEN_SPLB_RST[4].
    I_SPLB_RST> <mb_plb/GEN_SPLB_RST[3].I_SPLB_RST> <mb_plb/
    GEN_SPLB_RST[2].I_SPLB_RST> <mb_plb/GEN_SPLB_RST[1].I_SPLB_RST> <
    mb_plb/GEN_SPLB_RST[0].I_SPLB_RST> <mb_plb/I_PLB_RST>

Final Macro Processing ...

=========================================================================

Final Register Report

Found no macro
=========================================================================
=========================================================================
*                        Partition Report
                           *
=========================================================================


Partition Implementation Status
-------------------------------

  No Partitions were found in this design.

-------------------------------


=========================================================================
*                          Final Report
                              *
=========================================================================

Final Results
Top Level Output File Name          : ../implementation/system.ngc
Output Format                       : ngc
Optimization Goal                   : speed
Keep Hierarchy                      : no

Design Statistics
# IOs                               : 32

Cell Usage :
# BELS                              : 3937
#       GND                         : 16
#       INV                         : 54
#       LUT1                        : 128
#       LUT2                        : 251
#       LUT3                        : 519
#       LUT4                        : 522
#       LUT5                        : 506
#       LUT6                        : 948
#       LUT6_2                      : 31
#       MULT_AND                    : 12
#       MUXCY                       : 266
#       MUXCY_L                     : 196
#       MUXF5                       : 39
#       MUXF7                       : 123
#       MUXF7_L                     : 4
#       MUXF8_L                     : 2
#       VCC                         : 12
#        XORCY                          : 308
# FlipFlops/Latches                     : 3042
#        FD                             : 240
#        FDC                            : 63
#        FDC_1                          : 5
#        FDCE                           : 47
#        FDE                            : 534
#        FDE_1                          : 5
#        FDP                            : 25
#        FDR                            : 795
#        FDRE                           : 1184
#        FDRE_1                         : 1
#        FDRS                           : 20
#        FDRSE                          : 6
#        FDS                            : 59
#        FDSE                           : 58
# RAMS                                  : 55
#        RAM32M                         : 21
#        RAM32X1D                       : 2
#        RAMB36_EXP                     : 32
# Shift Registers                       : 90
#        SRL16                          : 3
#        SRL16E                         : 32
#        SRLC16E                        : 55
# Clock Buffers                         : 6
#        BUFG                           : 5
#        BUFGP                          : 1
# IO Buffers                            : 31
#        IBUF                           : 4
#        IOBUF                          : 16
#        OBUF                           : 11
# DSPs                                  : 5
#        DSP48E                         : 5
# Others                                : 2
#        BSCAN_VIRTEX5                  : 1
#        PLL_ADV                        : 1
=========================================================================


Device utilization summary:
---------------------------


Selected Device : 5vlx50ff676-1


Slice Logic Utilization:
 Number of Slice Registers:              2984  out of  28800    10%
 Number of Slice LUTs:                   3137  out of  28800    10%
    Number used as Logic:                2959  out of  28800    10%
    Number used as Memory:            178  out of   7680     2%
       Number used as RAM:             88
       Number used as SRL:             90

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  4892
   Number with an unused Flip Flop:   1908  out of   4892    39%
   Number with an unused LUT:         1755  out of   4892    35%
   Number of fully used LUT-FF pairs: 1229  out of   4892    25%
   Number of unique control sets:      295

IO Utilization:
 Number of IOs:                         32
 Number of bonded IOBs:                 32  out of    440     7%
    IOB Flip Flops/Latches:             58

Specific Feature Utilization:
 Number of Block RAM/FIFO:              32  out of     48    66%
    Number using Block RAM only:        32
 Number of BUFG/BUFGCTRLs:               6  out of     32    18%
 Number of DSP48Es:                      5  out of     48    10%
 Number of PLL_ADVs:                     1  out of      6    16%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------



=========================================================================

TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE
         REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------------------------------+------------------------+
Clock Signal                                               | Clock buffer(FF
   name)   | Load  |
-----------------------------------------------------------+------------------------+
clock_generator_0/clock_generator_0/PLL0_CLK_OUT<0>| BUFG
                            |  2858   |
mdm_0/mdm_0/drck_i                                        | BUFG
                            |  204    |
mdm_0/bscan_update1                                       | BUFG
                            |  35     |
fpga_0_SysACE_CompactFlash_SysACE_CLK_pin                 | BUFGP
                          |  95     |
-----------------------------------------------------------+------------------------------+--


Asynchronous Control Signals Information:
----------------------------------------
-----------------------------------------------------------------------------------------------

Control Signal

    | Buffer(FF name)

    | Load   |
-----------------------------------------------------------------------------------------------

lmb_bram/lmb_bram/net_gnd0(lmb_bram/lmb_bram/XST_GND:G)

    | NONE(lmb_bram/lmb_bram/ramb36_0)
                                                                                           |
    128     |
mb_plb/SPLB_Rst<2>(mb_plb/mb_plb/GEN_SPLB_RST[2].I_SPLB_RST:Q)

    | NONE(SysACE_CompactFlash/SysACE_CompactFlash/I_SYSACE_CONTROLLER
    /MEM_STATE_MACHINE_I/Done)          |  89     |
mdm_0/mdm_0/MDM_Core_I1/Config_Reg_Acst_inv(mdm_0/mdm_0/MDM_Core_I1/
    Config_Reg_Acst_inv1_INV_0:O)

    | NONE(mdm_0/mdm_0/MDM_Core_I1/Config_Reg_0)
                                                                              |  23     |
mdm_0/mdm_0/MDM_Core_I1/SEL_inv(mdm_0/mdm_0/MDM_Core_I1/
    SEL_inv1_INV_0:O)

    | NONE(mdm_0/mdm_0/MDM_Core_I1/PORT_Selector_0)
                                                                          |  12     |
SysACE_CompactFlash/SysACE_CompactFlash/I_SYSACE_CONTROLLER/
    SYNC_2_CLOCKS_I/done3(SysACE_CompactFlash/SysACE_CompactFlash/
    I_SYSACE_CONTROLLER/SYNC_2_CLOCKS_I/DONE_REG3:Q)| NONE(
    SysACE_CompactFlash/SysACE_CompactFlash/I_SYSACE_CONTROLLER/
    SYNC_2_CLOCKS_I/RDCE_REG1)         |  6      |
microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    Command_Reg_Rst(microblaze_0/microblaze_0/Performance.
    Use_Debug_Logic.Debug_I1/Command_Reg_Rst:Q)              | NONE(
    microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    command_reg_0)                    | 2       |
mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/data_cmd_inv(
    mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
    data_cmd_inv1_INV_0:O)                                | NONE(mdm_0/mdm_0/
    MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/execute)
                                     | 1       |
mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/local_sel_n3(
    mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
    Insert_Delays[4].LUT_Delay:O)                         | NONE(mdm_0/mdm_0/
    MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/FDC_I)
                                      | 1       |
mdm_0/mdm_0/MDM_Core_I1/reset_RX_FIFO(mdm_0/mdm_0/MDM_Core_I1/
    reset_RX_FIFO:Q)

    | NONE(mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
    Have_UARTs.RX_FIFO_I/data_Exists_I)| 1       |
mdm_0/mdm_0/MDM_Core_I1/reset_TX_FIFO(mdm_0/mdm_0/MDM_Core_I1/
    reset_TX_FIFO:Q)

    | NONE(mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
    Have_UARTs.TX_FIFO_I/data_Exists_I)| 1       |
microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    continue_from_brk(microblaze_0/microblaze_0/Performance.
    Use_Debug_Logic.Debug_I1/continue_from_brk:Q)        | NONE(
    microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    continue_from_brk_TClk)           | 1       |
microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    force_stop_CMD(microblaze_0/microblaze_0/Performance.
    Use_Debug_Logic.Debug_I1/force_stop_CMD:Q)              | NONE(
    microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    force_stop_TClk)                  | 1       |
microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    normal_stop_cmd(microblaze_0/microblaze_0/Performance.
    Use_Debug_Logic.Debug_I1/normal_stop_cmd:Q)           | NONE(
    microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    normal_stop_TClk)                 | 1       |
microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    start_single_step(microblaze_0/microblaze_0/Performance.
    Use_Debug_Logic.Debug_I1/start_single_step:Q)        | NONE(
    microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
    single_Step_TClk)                 | 1       |
-----------------------------------------------------------------------------------------


Timing Summary:
---------------
Speed Grade: -1
```

```
   Minimum period: 6.198ns (Maximum Frequency: 161.342MHz)
   Minimum input arrival time before clock: 2.750ns
   Maximum output required time after clock: 7.206ns
   Maximum combinational path delay: 1.397ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock '
   clock_generator_0/clock_generator_0/PLL0_CLK_OUT<0>'
  Clock period: 5.997ns (frequency: 166.750MHz)
  Total number of paths / destination ports: 763463 / 9782
-------------------------------------------------------------------------
Delay:               5.997ns (Levels of Logic = 6)
  Source:            microblaze_0/microblaze_0/Performance.
     Data_Flow_I/FPU_I/mem_MantA_2_31 (FF)
  Destination:       microblaze_0/microblaze_0/Performance.
     Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/mem_MantB_3_34 (FF)
  Source Clock:      clock_generator_0/clock_generator_0/PLL0_CLK_OUT
     <0> rising
  Destination Clock: clock_generator_0/clock_generator_0/PLL0_CLK_OUT
     <0> rising

  Data Path: microblaze_0/microblaze_0/Performance.Data_Flow_I/FPU_I/
     mem_MantA_2_31 to microblaze_0/microblaze_0/Performance.
     Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/mem_MantB_3_34
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRE:C->Q             3   0.471   1.080  microblaze_0/
        Performance.Data_Flow_I/FPU_I/mem_MantA_2_31 (microblaze_0/
        Performance.Data_Flow_I/FPU_I/mem_MantA_2<31>)
     LUT6:I0->O            4   0.094   0.726  microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_rot_mux1<31>1 (microblaze_0/Performance.Data_Flow_I/FPU_I
        /Use_FPU.FPU_ADDSUB_I/mem_rot_mux1<31>)
     LUT5:I2->O            6   0.094   0.816  microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_rot_mux2<11>1 (microblaze_0/Performance.Data_Flow_I/FPU_I
        /Use_FPU.FPU_ADDSUB_I/mem_rot_mux2<11>)
     LUT6:I2->O            2   0.094   0.794  microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_rot_mux3<27>1 (microblaze_0/Performance.Data_Flow_I/FPU_I
        /Use_FPU.FPU_ADDSUB_I/mem_rot_mux3<27>)
    LUT6:I2->O              1    0.094    0.000   microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_mant_sticky_3_cmb113_F (N535)
    MUXF7:I0->O             1    0.251    0.480   microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_mant_sticky_3_cmb113 (microblaze_0/Performance.
        Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_mant_sticky_3_cmb113)
    LUT5:I4->O              1    0.094    0.336   microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_mant_sticky_3_cmb164 (microblaze_0/Performance.
        Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_mant_sticky_3_cmb164)
    FDRS:S                       0.573            microblaze_0/
        Performance.Data_Flow_I/FPU_I/Use_FPU.FPU_ADDSUB_I/
        mem_MantB_3_34
   ----------------------------------------
   Total                         5.997ns (1.765ns logic, 4.232ns route)
                                         (29.4% logic, 70.6% route)

=========================================================================
Timing constraint: Default period analysis for Clock 'mdm_0/mdm_0/
   drck_i'
  Clock period: 6.198ns (frequency: 161.342MHz)
  Total number of paths / destination ports: 302 / 250
-------------------------------------------------------------------------
Delay:               3.099ns (Levels of Logic = 4)
  Source:            mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.
     JTAG_CONTROL_I/SYNC_FDRE (FF)
  Destination:       microblaze_0/microblaze_0/Performance.
     Use_Debug_Logic.Debug_I1/shift_Count_0 (FF)
  Source Clock:      mdm_0/mdm_0/drck_i falling
  Destination Clock: mdm_0/mdm_0/drck_i rising

  Data Path: mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
     SYNC_FDRE to microblaze_0/microblaze_0/Performance.
     Use_Debug_Logic.Debug_I1/shift_Count_0
                                  Gate     Net
   Cell:in->out        fanout    Delay    Delay   Logical Name (Net Name)
   ----------------------------------------   ------------
    FDRE_1:C->Q             1    0.467    0.973   PLB_Interconnect.
        JTAG_CONTROL_I/SYNC_FDRE (PLB_Interconnect.JTAG_CONTROL_I/
        sync)
    LUT6:I1->O              9    0.094    0.380   PLB_Interconnect.
        JTAG_CONTROL_I/shifting_Data (Dbg_Shift_7)
    end scope: 'mdm_0/MDM_Core_I1'
    end scope: 'mdm_0'
    begin scope: 'microblaze_0'
    INV:I->O              8   0.238   0.374  microblaze_0/
       Performance.Use_Debug_Logic.Debug_I1/Shift_inv1_INV_0 (
       microblaze_0/Performance.Use_Debug_Logic.Debug_I1/Shift_inv)
    FDR:R                     0.573          microblaze_0/
       Performance.Use_Debug_Logic.Debug_I1/shift_Count_0
   ----------------------------------------
   Total                      3.099ns (1.372ns logic, 1.727ns route)
                                      (44.3% logic, 55.7% route)

===========================================================================
Timing constraint: Default period analysis for Clock 'mdm_0/
   bscan_update1'
  Clock period: 5.746ns (frequency: 174.034MHz)
  Total number of paths / destination ports: 209 / 40
---------------------------------------------------------------------------
Delay:               2.873ns (Levels of Logic = 4)
  Source:            mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.
     JTAG_CONTROL_I/FDC_I (FF)
  Destination:       microblaze_0/microblaze_0/Performance.
     Use_Debug_Logic.Debug_I1/force_stop_TClk (FF)
  Source Clock:      mdm_0/bscan_update1 falling
  Destination Clock: mdm_0/bscan_update1 rising

  Data Path: mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
     FDC_I to microblaze_0/microblaze_0/Performance.Use_Debug_Logic.
     Debug_I1/force_stop_TClk
                                Gate     Net
   Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
    FDC_1:C->Q           10   0.467   0.625  PLB_Interconnect.
       JTAG_CONTROL_I/FDC_I (PLB_Interconnect.JTAG_CONTROL_I/
       data_cmd)
    LUT2:I0->O            6   0.094   1.000  PLB_Interconnect.
       JTAG_CONTROL_I/Dbg_Reg_En_I<4>1 (Dbg_Reg_En_0<4>)
    end scope: 'mdm_0/MDM_Core_I1'
    end scope: 'mdm_0'
    begin scope: 'microblaze_0'
    LUT5:I0->O            9   0.094   0.380  microblaze_0/
       Performance.Use_Debug_Logic.Debug_I1/
       Control_Reg_En_cmp_eq00001 (microblaze_0/Performance.
       Use_Debug_Logic.Debug_I1/Control_Reg_En)
    FDCE:CE                   0.213          microblaze_0/
       Performance.Use_Debug_Logic.Debug_I1/force_stop_TClk
   ----------------------------------------
   Total                      2.873ns (0.868ns logic, 2.005ns route)
                                      (30.2% logic, 69.8% route)
```

```
================================================================================
Timing constraint: Default period analysis for Clock '
   fpga_0_SysACE_CompactFlash_SysACE_CLK_pin'
  Clock period: 2.224ns (frequency: 449.640MHz)
  Total number of paths / destination ports: 227 / 70
--------------------------------------------------------------------------------
Delay:                  2.224ns (Levels of Logic = 2)
  Source:               SysACE_CompactFlash/SysACE_CompactFlash/
     I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/current_State_FSM_FFd4 (
     FF)
  Destination:          SysACE_CompactFlash/SysACE_CompactFlash/
     I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/SYSACE_WEN_REG (FF)
  Source Clock:         fpga_0_SysACE_CompactFlash_SysACE_CLK_pin rising
  Destination Clock: fpga_0_SysACE_CompactFlash_SysACE_CLK_pin rising

  Data Path: SysACE_CompactFlash/SysACE_CompactFlash/
     I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/current_State_FSM_FFd4
     to SysACE_CompactFlash/SysACE_CompactFlash/I_SYSACE_CONTROLLER/
     MEM_STATE_MACHINE_I/SYSACE_WEN_REG
                                Gate     Net
    Cell:in->out      fanout    Delay    Delay   Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q              3    0.471    1.080  SysACE_CompactFlash/
        I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/
        current_State_FSM_FFd4 (SysACE_CompactFlash/
        I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/
        current_State_FSM_FFd4)
     LUT6:I0->O            2    0.094    0.485  SysACE_CompactFlash/
        I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/sysace_wen_cmb21 (
        SysACE_CompactFlash/I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/
        N3)
     LUT3:I2->O            1    0.094    0.000  SysACE_CompactFlash/
        I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/sysace_wen_cmb1 (
        SysACE_CompactFlash/I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/
        sysace_wen_cmb)
     FDP:D                     -0.018          SysACE_CompactFlash/
        I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/SYSACE_WEN_REG
    ----------------------------------------
    Total                       2.224ns (0.659ns logic, 1.565ns route)
                                         (29.6% logic, 70.4% route)

================================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock '
   clock_generator_0/clock_generator_0/PLL0_CLK_OUT<0>'
  Total number of paths / destination ports: 3 / 3
```

```
--------------------------------------------------------------------------------

Offset:                 1.733ns (Levels of Logic = 3)
  Source:               fpga_0_rst_1_sys_rst_pin (PAD)
  Destination:          proc_sys_reset_0/proc_sys_reset_0/EXT_LPF/exr_d1
      (FF)
  Destination Clock: clock_generator_0/clock_generator_0/PLL0_CLK_OUT
     <0> rising

  Data Path: fpga_0_rst_1_sys_rst_pin to proc_sys_reset_0/
     proc_sys_reset_0/EXT_LPF/exr_d1
                                    Gate     Net
    Cell:in->out       fanout    Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O               2   0.818   0.341
        fpga_0_rst_1_sys_rst_pin_IBUF (fpga_0_rst_1_sys_rst_pin_IBUF)
     begin scope: 'proc_sys_reset_0'
     INV:I->O                1   0.238   0.336  proc_sys_reset_0/
        EXT_LPF/exr_d1_or000011_INV_0 (proc_sys_reset_0/EXT_LPF/
        exr_d1_or00001)
     FDS:D                      -0.018          proc_sys_reset_0/
        EXT_LPF/exr_d1
    ----------------------------------------
    Total                        1.733ns (1.056ns logic, 0.677ns route)
                                         (60.9% logic, 39.1% route)

================================================================================

Timing constraint: Default OFFSET IN BEFORE for Clock 'mdm_0/mdm_0/
   drck_i'
  Total number of paths / destination ports: 129 / 96
--------------------------------------------------------------------------------

Offset:                 2.750ns (Levels of Logic = 4)
  Source:               mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:SHIFT (
     PAD)
  Destination:          microblaze_0/microblaze_0/Performance.
     Use_Debug_Logic.Debug_I1/shift_Count_0 (FF)
  Destination Clock: mdm_0/mdm_0/drck_i rising

  Data Path: mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:SHIFT to
     microblaze_0/microblaze_0/Performance.Use_Debug_Logic.Debug_I1/
     shift_Count_0
                                    Gate     Net
    Cell:in->out       fanout    Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
    BSCAN_VIRTEX5:SHIFT      5   0.000   0.000  mdm_0/Use_Virtex5.
       BSCAN_VIRTEX5_I (bscan_shift)
     begin scope: 'mdm_0/MDM_Core_I1'
    LUT6:I0->O              9   0.094    0.380  PLB_Interconnect.
       JTAG_CONTROL_I/shifting_Data (Dbg_Shift_7)
    end scope: 'mdm_0/MDM_Core_I1'
    end scope: 'mdm_0'
    begin scope: 'microblaze_0'
    INV:I->O                8   0.238    0.374  microblaze_0/
       Performance.Use_Debug_Logic.Debug_I1/Shift_inv1_INV_0 (
       microblaze_0/Performance.Use_Debug_Logic.Debug_I1/Shift_inv)
    FDR:R                       0.573           microblaze_0/
       Performance.Use_Debug_Logic.Debug_I1/shift_Count_0
   ----------------------------------------
    Total                       2.750ns (1.996ns logic, 0.754ns route)
                                        (72.6% logic, 27.4% route)

=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'mdm_0/
   bscan_update1'
  Total number of paths / destination ports: 14 / 14
-------------------------------------------------------------------------
Offset:              1.670ns (Levels of Logic = 2)
  Source:            mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:SEL (PAD
     )
  Destination:       mdm_0/mdm_0/MDM_Core_I1/PORT_Selector_1_0 (FF)
  Destination Clock: mdm_0/bscan_update1 rising

  Data Path: mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:SEL to mdm_0/
     mdm_0/MDM_Core_I1/PORT_Selector_1_0
                                Gate     Net
    Cell:in->out      fanout    Delay   Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
   BSCAN_VIRTEX5:SEL        8   0.000    0.000  mdm_0/Use_Virtex5.
      BSCAN_VIRTEX5_I (mdm_0/sel)
    begin scope: 'mdm_0/MDM_Core_I1'
    LUT5:I0->O              4   0.094    0.352  MDM_SEL1 (MDM_SEL)
    FDCE:CE                     0.213           PORT_Selector_1_0
   ----------------------------------------
    Total                       1.670ns (1.318ns logic, 0.352ns route)
                                        (78.9% logic, 21.1% route)

=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock '
   fpga_0_SysACE_CompactFlash_SysACE_CLK_pin'
  Total number of paths / destination ports: 16 / 16
-------------------------------------------------------------------------
Offset:              1.154ns (Levels of Logic = 2)
```

```
  Source:              fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<0> (
     PAD)
  Destination:         SysACE_CompactFlash/SysACE_CompactFlash/
     I_SYSACE_CONTROLLER/SYNC_2_CLOCKS_I/MEM_DQ_I_GEN[7].MEM_DQ_I_1 (
     FF)
  Destination Clock: fpga_0_SysACE_CompactFlash_SysACE_CLK_pin rising

  Data Path: fpga_0_SysACE_CompactFlash_SysACE_MPD_pin<0> to
     SysACE_CompactFlash/SysACE_CompactFlash/I_SYSACE_CONTROLLER/
     SYNC_2_CLOCKS_I/MEM_DQ_I_GEN[7].MEM_DQ_I_1
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IOBUF:IO->O            1   0.818   0.336  iobuf_15 (
         fpga_0_SysACE_CompactFlash_SysACE_MPD_pin_I<0>)
      begin scope: 'SysACE_CompactFlash'
     FDCE:D                    -0.018          SysACE_CompactFlash/
         I_SYSACE_CONTROLLER/SYNC_2_CLOCKS_I/MEM_DQ_I_GEN[7].
         MEM_DQ_I_1
    ----------------------------------------
    Total                      1.154ns (0.818ns logic, 0.336ns route)
                                       (70.9% logic, 29.1% route)

=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock '
    fpga_0_SysACE_CompactFlash_SysACE_CLK_pin'
  Total number of paths / destination ports: 42 / 26
-------------------------------------------------------------------------
Offset:              3.259ns (Levels of Logic = 2)
  Source:            SysACE_CompactFlash/SysACE_CompactFlash/
     I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/SYSACE_CEN_REG (FF)
  Destination:       fpga_0_SysACE_CompactFlash_SysACE_CEN_pin (PAD)
  Source Clock:      fpga_0_SysACE_CompactFlash_SysACE_CLK_pin rising

  Data Path: SysACE_CompactFlash/SysACE_CompactFlash/
     I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/SYSACE_CEN_REG to
     fpga_0_SysACE_CompactFlash_SysACE_CEN_pin
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDP:C->Q               1   0.471   0.336  SysACE_CompactFlash/
         I_SYSACE_CONTROLLER/MEM_STATE_MACHINE_I/SYSACE_CEN_REG (
         SysACE_CEN)
      end scope: 'SysACE_CompactFlash'
     OBUF:I->O                  2.452
         fpga_0_SysACE_CompactFlash_SysACE_CEN_pin_OBUF (
         fpga_0_SysACE_CompactFlash_SysACE_CEN_pin)
  ---------------------------------------------------------
   Total                       3.259ns (2.923ns logic, 0.336ns route)
                                       (89.7% logic, 10.3% route)

================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock '
    clock_generator_0/clock_generator_0/PLL0_CLK_OUT<0>'
  Total number of paths / destination ports: 1 / 1
--------------------------------------------------------------------------------
Offset:                 3.259ns (Levels of Logic = 2)
  Source:               RS232_Uart/RS232_Uart/UARTLITE_CORE_I/
      UARTLITE_TX_I/TX (FF)
  Destination:          fpga_0_RS232_Uart_TX_pin (PAD)
  Source Clock:         clock_generator_0/clock_generator_0/PLL0_CLK_OUT
      <0> rising

  Data Path: RS232_Uart/RS232_Uart/UARTLITE_CORE_I/UARTLITE_TX_I/TX
      to fpga_0_RS232_Uart_TX_pin
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ---------------------------------------------  ------------
     FDS:C->Q              1   0.471   0.336  RS232_Uart/
        UARTLITE_CORE_I/UARTLITE_TX_I/TX (TX)
     end scope: 'RS232_Uart'
     OBUF:I->O                 2.452
        fpga_0_RS232_Uart_TX_pin_OBUF (fpga_0_RS232_Uart_TX_pin)
  ---------------------------------------------
   Total                       3.259ns (2.923ns logic, 0.336ns route)
                                       (89.7% logic, 10.3% route)

================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'mdm_0/mdm_0/
    drck_i'
  Total number of paths / destination ports: 119 / 1
--------------------------------------------------------------------------------
Offset:                 7.206ns (Levels of Logic = 8)
  Source:               microblaze_0/microblaze_0/Performance.
      Use_Debug_Logic.Debug_I1/Use_SRL16.The_Cache_Addresses[5].
      SRL16E_Cache_I (FF)
  Destination:          mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:TDO (PAD
      )
  Source Clock:         mdm_0/mdm_0/drck_i rising

  Data Path: microblaze_0/microblaze_0/Performance.Use_Debug_Logic.
      Debug_I1/Use_SRL16.The_Cache_Addresses[5].SRL16E_Cache_I to
    mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:TDO
                                    Gate     Net
    Cell:in->out       fanout    Delay    Delay   Logical Name (Net Name)
   ----------------------------------------   ------------
     SRL16E:CLK->Q           1    1.889    0.973   microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/Use_SRL16.
        The_Cache_Addresses[5].SRL16E_Cache_I (microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/tdo_config_word1<6>)
     LUT6:I1->O              1    0.094    0.973   microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/TDO42 (microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/TDO42)
     LUT6:I1->O              1    0.094    0.000   microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/TDO387_SW01 (
        microblaze_0/Performance.Use_Debug_Logic.Debug_I1/TDO387_SW0)
     MUXF7:I1->O             1    0.254    0.789   microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/TDO387_SW0_f7 (N463)
     LUT6:I2->O              1    0.094    1.069   microblaze_0/
        Performance.Use_Debug_Logic.Debug_I1/TDO387 (DBG_TDO)
     end scope: 'microblaze_0'
     begin scope: 'mdm_0'
     begin scope: 'mdm_0/MDM_Core_I1'
     LUT6:I0->O              1    0.094    0.789   TDO_i79 (TDO_i79)
     LUT6:I2->O              0    0.094    0.000   TDO_i225 (TDO)
     end scope: 'mdm_0/MDM_Core_I1'
   BSCAN_VIRTEX5:TDO               0.000            mdm_0/Use_Virtex5.
       BSCAN_VIRTEX5_I
   ----------------------------------------
    Total                          7.206ns (2.613ns logic, 4.593ns route)
                                           (36.3% logic, 63.7% route)

=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'mdm_0/
    bscan_update1'
  Total number of paths / destination ports: 36 / 1
-------------------------------------------------------------------------
Offset:                4.996ns (Levels of Logic = 9)
  Source:              mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.
     JTAG_CONTROL_I/FDC_I (FF)
  Destination:         mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:TDO (PAD
     )
  Source Clock:        mdm_0/bscan_update1 falling

  Data Path: mdm_0/mdm_0/MDM_Core_I1/PLB_Interconnect.JTAG_CONTROL_I/
     FDC_I to mdm_0/mdm_0/Use_Virtex5.BSCAN_VIRTEX5_I:TDO
                                    Gate     Net
    Cell:in->out       fanout    Delay    Delay   Logical Name (Net Name)
   ----------------------------------------   ------------
```

```
      FDC_1:C->Q              10    0.467    0.625  PLB_Interconnect.
         JTAG_CONTROL_I/FDC_I (PLB_Interconnect.JTAG_CONTROL_I/
         data_cmd)
      LUT2:I0->O               6    0.094    1.096  PLB_Interconnect.
         JTAG_CONTROL_I/Dbg_Reg_En_I<4>1 (Dbg_Reg_En_0<4>)
      end scope: 'mdm_0/MDM_Core_I1'
      end scope: 'mdm_0'
      begin scope: 'microblaze_0'
      LUT6:I0->O               1    0.094    0.480  microblaze_0/
         Performance.Use_Debug_Logic.Debug_I1/TDO351 (microblaze_0/
         Performance.Use_Debug_Logic.Debug_I1/TDO351)
      LUT6:I5->O               1    0.094    1.069  microblaze_0/
         Performance.Use_Debug_Logic.Debug_I1/TDO387 (DBG_TDO)
      end scope: 'microblaze_0'
      begin scope: 'mdm_0'
      begin scope: 'mdm_0/MDM_Core_I1'
      LUT6:I0->O               1    0.094    0.789  TDO_i79 (TDO_i79)
      LUT6:I2->O               0    0.094    0.000  TDO_i225 (TDO)
      end scope: 'mdm_0/MDM_Core_I1'
     BSCAN_VIRTEX5:TDO              0.000           mdm_0/Use_Virtex5.
        BSCAN_VIRTEX5_I
     ----------------------------------------
     Total                          4.996ns (0.937ns logic, 4.059ns route)
                                            (18.8% logic, 81.2% route)

=========================================================================

Timing constraint: Default path analysis
  Total number of paths / destination ports: 2 / 2
-------------------------------------------------------------------------

Delay:               1.397ns (Levels of Logic = 3)
  Source:            fpga_0_rst_1_sys_rst_pin (PAD)
  Destination:       clock_generator_0/clock_generator_0/Using_PLL0.
     PLL0_INST/PLL_INST/Using_PLL_ADV.PLL_ADV_inst:RST (PAD)

  Data Path: fpga_0_rst_1_sys_rst_pin to clock_generator_0/
     clock_generator_0/Using_PLL0.PLL0_INST/PLL_INST/Using_PLL_ADV.
     PLL_ADV_inst:RST
                                   Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
      IBUF:I->O                2    0.818    0.341
         fpga_0_rst_1_sys_rst_pin_IBUF (fpga_0_rst_1_sys_rst_pin_IBUF)
      begin scope: 'clock_generator_0'
      INV:I->O                 0    0.238    0.000  clock_generator_0/
         Using_PLL0.PLL0_INST/PLL_INST/rsti1_INV_0 (clock_generator_0/
         Using_PLL0.PLL0_INST/PLL_INST/rsti)
    PLL_ADV:RST                    0.000            clock_generator_0/
       Using_PLL0.PLL0_INST/PLL_INST/Using_PLL_ADV.PLL_ADV_inst
    ----------------------------------------
    Total                          1.397ns (1.056ns logic, 0.341ns route)
                                           (75.6% logic, 24.4% route)

=========================================================================



Total REAL time to Xst completion: 25.00 secs
Total CPU time to Xst completion: 25.75 secs

-->

Total memory usage is 257840 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :  278 (   0 filtered)
Number of infos    :   10 (   0 filtered)
```

.4 Matlab Scripts

.4.1 Read Image

```
a = imread('canyon.gif');
[y,x]= size(a);
size_x = x;
size_y = y;

b = imnoise(a,'salt & pepper',0.02);

string = '(';
for j = 1:size_x
    j
    for i = 1:size_y
         if(j==size_x && i==size_y)
           string = [string,num2str(b(i,j))];
        else
          string = [string,num2str(b(i,j)),','];
        end;

     end;
end;

string = [string,')'];
file_in = fopen('image_in_0.02.pgm','w');
fprintf(file_in,string);
fclose(file_in);
size_x
size_y
```

.4.2 Write Image

```
a = textread('one.pgm');
b = a';
[y,x]= size(b);
img = zeros(x,y-(y-x));

for i = 1:y-1
    for j = 1:x
      img(i,j) = b(i,j);
    end;
end;

size(img)
imwrite(img,'canyon_noise_0.01.pgm','encoding','ascii');
```

.4.3 Measuring Values

```
img = imread('field.gif');
noisy = imread('field_output_noise_0.001.png');
img = im2double(img);
noisy = im2double(noisy);

[x,y]= size(img);

max_val = max(max(img));
[x,y] = size(img);
mse = 0;
mae = 0;
        for i=2:x-1
              for j=2:y-1
                mse = mse + (img(i,j)-noisy(i,j))^2;
                mae = mae + abs(img(i,j)-noisy(i,j));
              end
        end

    mse = mse/(2*x*y);
    mae = mae/(2*x*y);


    psnr = 10*log10((max_val^2)/mse);

     mse = mse*256;
    mae = mae*256;

mse
mae
psnr
```

.4.4 Median algorithm

```
%image is the image to be noised then denoised
%m is the vindowsize
%th is the threshold in added value, a value of 0,2 will classefy any
    pixel
%of +-0,2 as defect
%n is the noise percentage
function image_median(image,m,th,n)
tic;                                        %start time
I = imread(image);                          %reading image file
[x,y,z]= size(I);                           %calculating the size of
   the input image
pixel_map = zeros(x,y,z);                   %a matrix which stores
   the positions of the pixels that have been corrected
img = im2double(I);                         %converting from unit8 to
    double: scale from 0..1 in stead of 0--256
img_noisy = imnoise(img,'salt & pepper',n); %adding noise to the
   input image
noise= img_noisy;                           %duplicating the noist
   image in order to dispay both of them
% imwrite(noise,'img_two_noisy_salt.jpg');%writing corrected image to
    file
                                            %The corners, edges and
                                                mid image will have to
                                                 be done separately
                                                due to
                                            %how matlab indexes and m
                                                *n*3 matrix. It is
                                                impossible to access a
                                                 position
                                            %outside the matrix.The
                                                image matrix will have
                                                 corners(1,1),(1,y),(x
                                                ,1) and
                                            %(x,y)
                                            %correcting the corners
                                                of all the layers in
                                                the image by replacing
                                                 a noisy
                                            %pixel with the median of
                                                 the three neigboring
                                                pixels.The z variable
                                                will
                                            %be 3 for an RGB image.
                                            %left upper corner(1,1)
                                                of all the layers(3
                                                layers for an RGB
                                                image)
```

```
for k=1:z
    c1=[img_noisy(2,1,k),img_noisy(2,2,k),img_noisy(1,2,k)];
    if(img_noisy(1,1,k)>th+max(c1)||img_noisy(1,1,k)<th+min(c1))
       pixel_map(1,1,k)=1;
       img_noisy(1,1,k)=median(c1);        %left upper corner(1,y)
           of all layers
    end
end

for k=1:z
    c2=[img_noisy(1,y-1,k),img_noisy(2,y,k),img_noisy(2,y-1,k)];
    if(img_noisy(1,y,k)>th+max(c2)||img_noisy(1,y,k)<th+min(c2))
       pixel_map(1,y,k)=1;
       img_noisy(1,y,k)=median(c2);        %right upper corner(1,y)
            of all layers
    end
end

for k=1:z
    c3= [img_noisy(x,2,k),img_noisy(x-1,1,k),img_noisy(x-1,2,k)];
    if(img_noisy(x,1,k)>th+max(c3)||img_noisy(x,1,k)<th+min(c3))
       pixel_map(x,1,k)=1;
       img_noisy(x,1,k)=median(c3);       %left lower corner(x,1)
           of all layers
     end
end

for k=1:z
   c4=[img_noisy(x,y,k),img_noisy(x-1,y-1,k),img_noisy(x-1,y,k)];
   if(img_noisy(x,y,k)>th+max(c4)||img_noisy(x,y,k)<th+min(c4))
      pixel_map(x,y,k)=1;
      img_noisy(x,y,k)=median(c4);         %right lower corner(x,y)
           of all layers
    end
end

                                                  %correcting the edges of
                                                     the image is done by
                                                     storing the neigboring
                                                  %pixel values and
                                                     assigning the median
                                                     of those values to the
                                                      pisition of
                                                  %the defect pixel.The
                                                     windowsize is m, and
                                                     correcting from ((m-((
                                                    m+1)/2)))
                                                  %to ((m-((m-1)/2))) will
                                                     correct the edges that
```

```
                                        the mid image
                                      correction
                                    %function can not reach
                                      with its for loop.

for k=1:z                            %upper horizontal line
    for i=2:((m-((m+1)/2)))
        for j=2:((m-((m-1)/2)))
            e1=[img_noisy(i,j-1,k),img_noisy(i,j+1,k),
                img_noisy(i+1,j-1,k),img_noisy(i+1,j,k),
                img_noisy(i+1,j+1,k)];
                 if(img_noisy(i,j,k)>th+max(e1)||img_noisy
                     (i,j,k)<th+min(e1))
                        pixel_map(i,j,k)=1;
                      img_noisy(i,j,k)=median(e1);

                 end
        end
    end
end


 for k=1:z                          %lower horizontal line
    for i=2:(x-(m-((m+1)/2)))
        for j=y:(y-(m-((m-1)/2)))
            e2=[img_noisy(i,j-1,k),img_noisy(i,j+1,k),
                img_noisy(i+1,j-1,k),img_noisy(i+1,j,k),
                img_noisy(i+1,j+1,k)];
            if(img_noisy(i,j,k)>th+max(e2)||img_noisy(i,j
                ,k)<th+min(e2))
              pixel_map(i,j,k)=1;
                 img_noisy(i,j,k)=median(e2);
            else
            end
        end
    end
 end


 for k=1:z                           %left vertical line
    for i=2:((m-((m+1)/2)))
        for j=2:((m-((m+1)/2)))
            e3=[img_noisy(i-1,j,k),img_noisy(i+1,j,k),
                img_noisy(i+1,j+1,k),img_noisy(i,j+1,k),
                img_noisy(i-1,j+1,k)];
            if(img_noisy(i,j,k)>th+max(e3)||img_noisy(i,j
                ,k)<th+min(e3))
                 pixel_map(i,j,k)=1;
                 img_noisy(i,j,k)=median(e3);
```

```
                    end
                end
            end
         end


        for k=1:z                               %right vertical line
            for i=y:(y-(m-((m+1)/2)))
                for j=2:(x-(m-((m+1)/2)))
                    e4=[img_noisy(i,j-1,k),img_noisy(i,j+1,k),
                        img_noisy(i+1,j-1,k),img_noisy(i+1,j,k),
                        img_noisy(i+1,j+1,k)];
                     if(img_noisy(i,j,k)>th+max(e4)||img_noisy(i,j
                         ,k)<th+min(e4))
                    pixel_map(i,j,k)=1;
                          img_noisy(i,j,k)=median(e4);

                    end
                end
            end
        end

                                                %Correcting mid image
                                                    from (m-(m-1)/2):x-(m
                                                    -(m+1)/2) to
                                                %(m-(m-1)/2):y-(m-(m+1)/2
                                                     will ensure that the
                                                    enitre image has been
                                                %corrected when first
                                                    doing the coreners,
                                                    then the edges and at
                                                    last the
                                                %mid image.
      for k=1:z
                for i=(m-(m-1)/2):x-(m-(m+1)/2)
                      for j=(m-(m-1)/2):y-(m-(m+1)/2)
                             val = 1;                      %counter for
                                the values array
                             count = 1;
                             values = zeros(1,m^2);     %array to
                                store the neigboring pixel values
                             array = zeros(1,m^1-1);
                             for a=-(m-((m+1)/2)):(m-((m+1)/2))
                                 for b=-(m-(m+1)/2):(m-((m+1)/2))
                                     if(a==0 && b==0)
                                     else
                                     array(count)=img_noisy(i+a,j+b,
                                        k);
```

```
                                                  end
                                                  values(val)=img_noisy(i+a,j+b,k
                                                      );
                                                  val = val+1;
                                              end
                                       end
                                          if(img_noisy(i,j,k)>th+max(array)||
                                              img_noisy(i,j,k)<th+min(array))
                                         img_noisy(i,j,k)=median(values);     %
                                              replacing the defect pixel with
                                              the median of the values array
                                         pixel_map(i,j,k)=1;                  %
                                              storing the position of the defect
                                               pixel
                                         end
                              end
                     end
            end

     toc;                                            %stop time
imwrite(img_noisy,'median_res.jpg'); % write the
                                                 %restored image to
                                                     predetermined filename
                                                 %plotting the original,
                                                     noisy and
                                                     reconstructed image
str=sprintf('Median filtered image with windowsize %d', m);
subplot(2,2,1:2), imshow(I)
title('Original image');
subplot(2,2,3), imshow(noise)
title('Noisy image');
subplot(2,2,4), imshow(img_noisy)
title(str);

                                                 %When the image has been
                                                     corrected it is time
                                                     to measure the mse,
                                                     mae and
                                                 %psnr.This is done by
                                                     looping through the
                                                     entire image in order
                                                     to
                                                 %calculate these values.
J = img_noisy;
max_val = max(max(max(img)));
[x,y,z] = size(I);
mse = 0;
mae = 0;
     for k=1:z
```

```
            for  i=2:x-1
                   for  j=2:y-1
                     mse = mse + (img(i,j,k)-J(i,j,k))^2;
                     mae = mae + abs(img(i,j,k)-J(i,j,k));
                   end
            end
     end
     mse = mse/(3*x*y);
     mae = mae/(3*x*y);
     psnr = 10*log10((max_val^2)/mse);
%defects = sum(sum(sum(pixel_map)));

mse = 256*mse;
mae = 256*mae;
              %displaying the measured values.
mse
mae
psnr
end
```

.5 C Code

.5.1 Test Compact flash

```
#include "sysace_stdio.h"
#include "xparameters.h"
#include "xutil.h"
#include "xio.h"
#include "stdio.h"

int main(void){

unsigned char buffer[3];

char*file_out = "output.txt"; //input and output files name, type the
char*file_in = "input.txt";    //directory
int numread;
char readBuffer[2];          //the variable that stored charackters read
    from the cf card

SYSACE_FILE * outfile;       //input file handler
SYSACE_FILE * infile;      //output file handler

outfile = sysace_fopen(file_out,"w");          //open the putfile in
   write mode
infile = sysace_fopen(file_in,"r");            //open the input file in
   read mode

numread = sysace_fread(readBuffer,1,19,infile); //reading 19
   characters from the input file into the readBuffer

sysace_fwrite(readBuffer,1,19,outfile);       //writing the characters
   from the readBuffer into the file on the cf card



sysace_fclose(outfile);    //closing the output file
sysace_fclose(infile);     //closing the input file

return 0;
}
```

.5.2 Test interrupt

```
#include "sysace_stdio.h"
#include "xparameters.h"
#include "xutil.h"
#include "xio.h"
#include "stdio.h"
#include "my_custom_ip_register.h"
#include "xintc.h"
#include "mb_interface.h"

#define SLV_REG0     0x00000000
#define SLV_REG1     0x00000004
#define SLV_REG2     0x00000008
#define SLV_REG3     0x0000000C
#define SLV_REG4     0x00000010
#define BaseAddress  0xCCC00000
#define read        0x00000001
#define reset       0x00000000


void intr_handler(){
  char*file_out = "outfile.txt";    //declaring the output file name,
      type and directory
  char readValue;            //variable to store data
  int numwrite;
  SYSACE_FILE * outfile;        //output file handler
  unsigned char buffer[4]= "test";  //data to be written to the file


  status = MY_CUSTOM_IP_REGISTER_mReadReg(
     XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
     MY_CUSTOM_IP_REGISTER_INTR_IPISR_OFFSET);   //read the status of
      the interrupt service register
  MY_CUSTOM_IP_REGISTER_mWriteReg(
     XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
     MY_CUSTOM_IP_REGISTER_INTR_IPISR_OFFSET, 0);        //Acknowledge
      interrupt on peripheral
  XIntc_mAckIntr(XPAR_XPS_INTC_0_BASEADDR,
     XPAR_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_MASK);
       //Acknowledge interrupt on interrupt controller


  outfile = sysace_fopen(file_out,"w");          //open file in
     write mode
  numwrite = sysace_fwrite(buffer,1,sizeof(buffer),outfile);  //write
      the data in buffer to the file
  sysace_fclose(outfile);                  //close file
}
```

```
void init(){
    // Registering an Interrupt handler
    XIntc_RegisterHandler(XPAR_XPS_INTC_0_BASEADDR,
        XPAR_XPS_INTC_0_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_INTR,
     intr_handler, (void *)XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR);

   // enabling interrupts on the interrupt controller, my_custom_ip
       and microblaze
    XIntc_mMasterEnable(XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR);
    XIntc_mEnableIntr(XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
     XPAR_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_MASK);
    MY_CUSTOM_IP_REGISTER_EnableInterrupt((void *)
        XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR);
    microblaze_enable_interrupts();

}


int main(void){
init();         //run the function to set up the interrupts
intr_handler();    //the function run for every interrupt

   while(1){

   }

return 0;
}
```

.5.3 Test interrupt with read and write

```
#include "sysace_stdio.h"
#include "xparameters.h"
#include "xutil.h"
#include "xio.h"
#include "stdio.h"
#include "my_custom_ip_register.h"
#include "xintc.h"
#include "mb_interface.h"

void intr_handler(){

SYSACE_FILE * outfile;
char* file_out = "outfile.txt"; //output file name, type and
    directory
Xuint32 readValue;          //variable to store data read from the
    register
Xuint32 status;            //status of the interrupt register
char  test[3];            //variable used to store char representation
    of readValue
char  data[30];           //variable that is written to the cf card

    status = MY_CUSTOM_IP_REGISTER_mReadReg(
        XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
        MY_CUSTOM_IP_REGISTER_INTR_IPISR_OFFSET);   //read the status
        of the interrupt service register
    MY_CUSTOM_IP_REGISTER_mWriteReg(
        XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
        MY_CUSTOM_IP_REGISTER_INTR_IPISR_OFFSET, 0);        //
        Acknowledge interrupt on peripheral
    XIntc_mAckIntr(XPAR_XPS_INTC_0_BASEADDR,
        XPAR_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_MASK);
            //Acknowledge interrupt on interrupt controller

    outfile = sysace_fopen(file_out,"w");
                        //open file in write mode
     readValue = MY_CUSTOM_IP_REGISTER_mReadReg(
        XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
        MY_CUSTOM_IP_REGISTER_SLV_REG3_OFFSET);    //read value from
        register 3
     test[0] = readValue;
                    //stores value in the test variable
    strncat(data,test,2);                                  //stores
        the test value into a char array
    sysace_fwrite((char *)data,1,sizeof(data),outfile);
            //writes the value of data to the cf card
    sysace_fclose(outfile);                                  //close
        the file
     }

int init(i){
   // Registering an Interrupt handler
   XIntc_RegisterHandler(XPAR_XPS_INTC_0_BASEADDR,
      XPAR_XPS_INTC_0_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_INTR,
    intr_handler, (void *)XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR);

   // enabling interrupts on the interrupt controller, my_custom_ip
      and microblaze
   XIntc_mMasterEnable(XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR);
   XIntc_mEnableIntr(XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR,
    XPAR_MY_CUSTOM_IP_REGISTER_0_IP2INTC_IRPT_MASK);
   MY_CUSTOM_IP_REGISTER_EnableInterrupt((void *)
      XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR);
   microblaze_enable_interrupts();
  i=0;
  return i;
}


int main(void){
MY_CUSTOM_IP_REGISTER_mWriteReg(XPAR_MY_CUSTOM_IP_REGISTER_0_BASEADDR
   ,MY_CUSTOM_IP_REGISTER_SLV_REG3_OFFSET,0xA2); //writing a test
   value to register 3

init();   //function to set up the interrupts


intr_handler();   //function that is invoked when there is an
   interrupt.


  while(1){
  }

return 0;
}
```

.6 Images

.6.1 Original Images

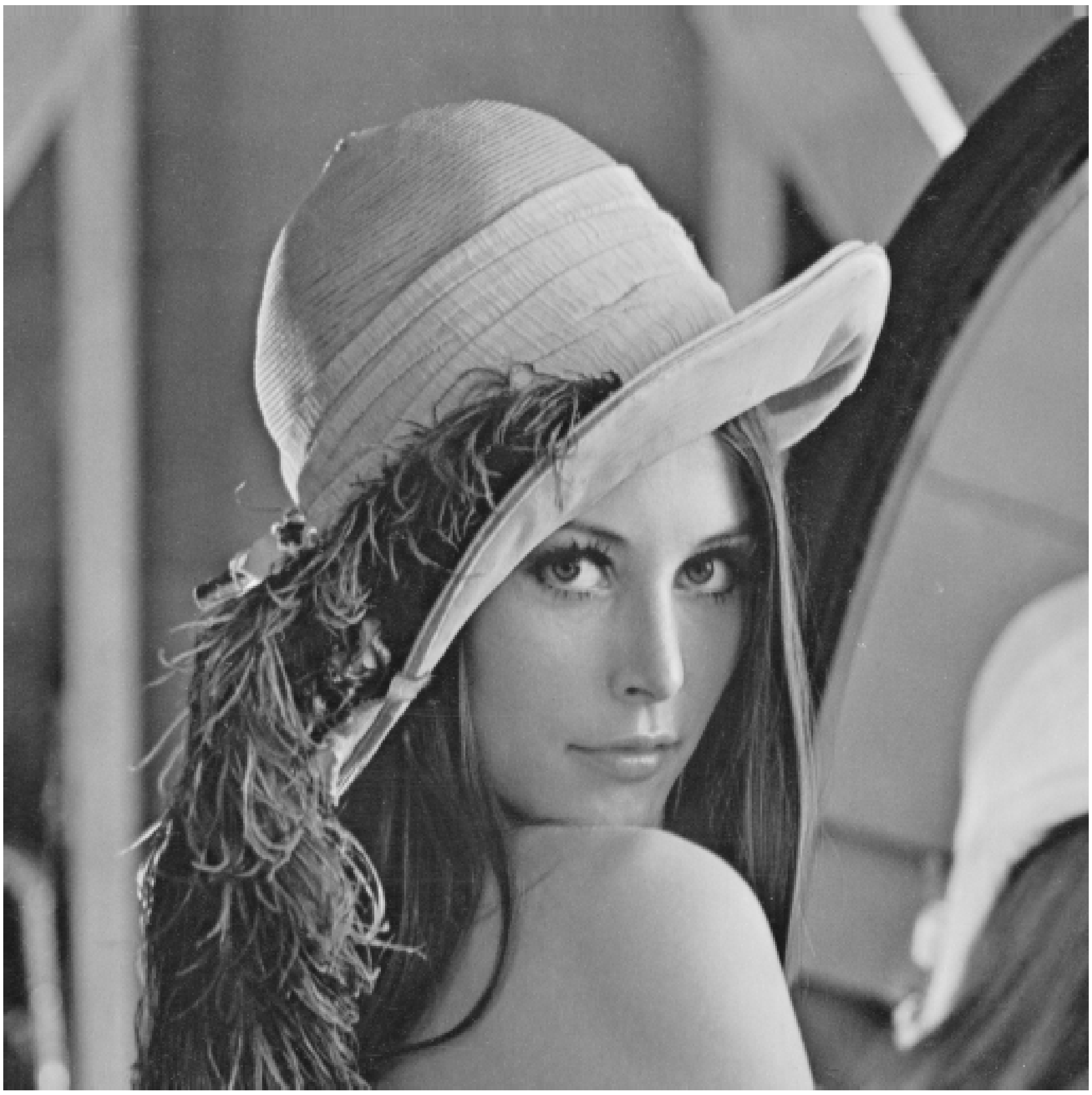


Figure 1: Original noise free Lena image

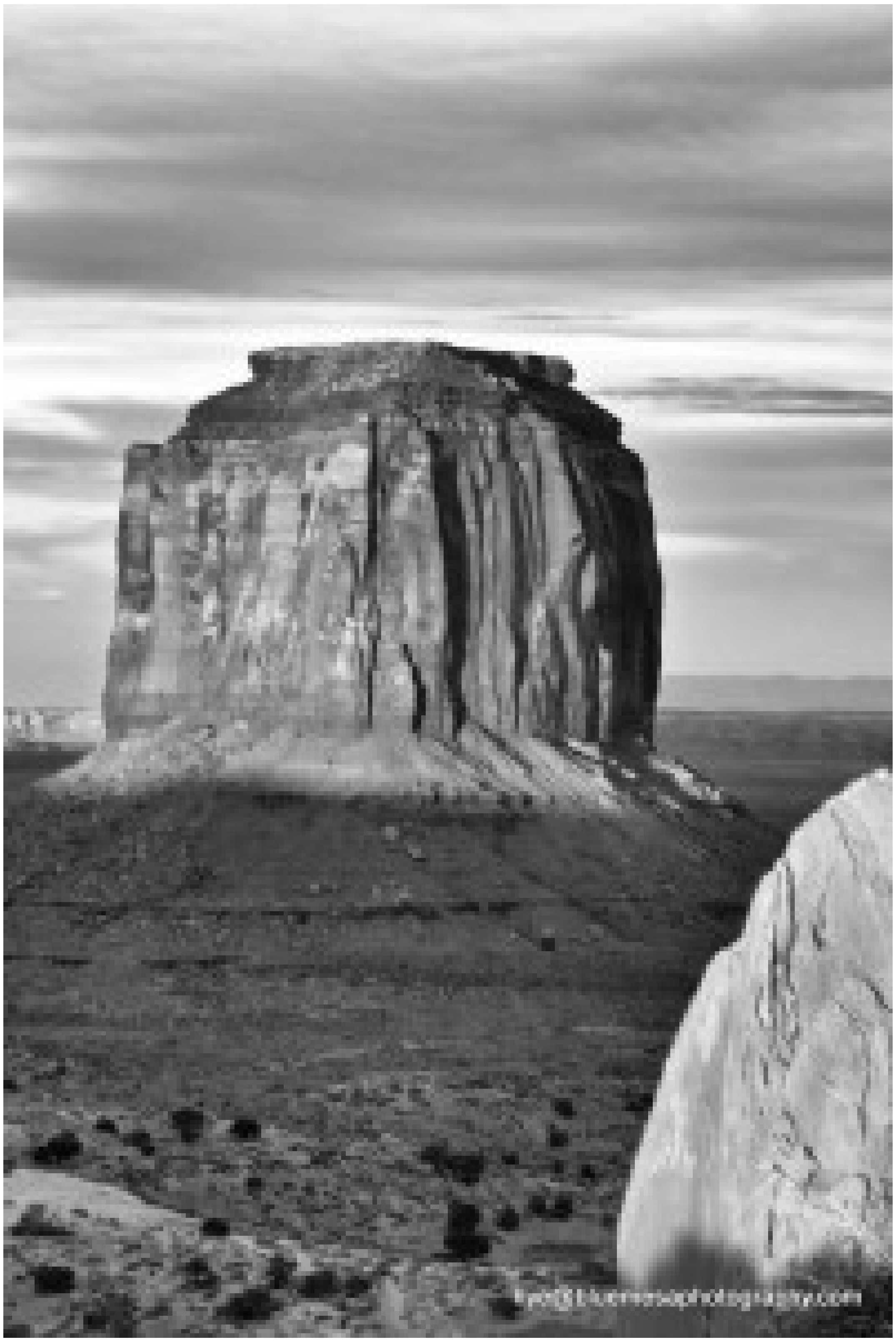


Figure 2: Original noise free Canyon image

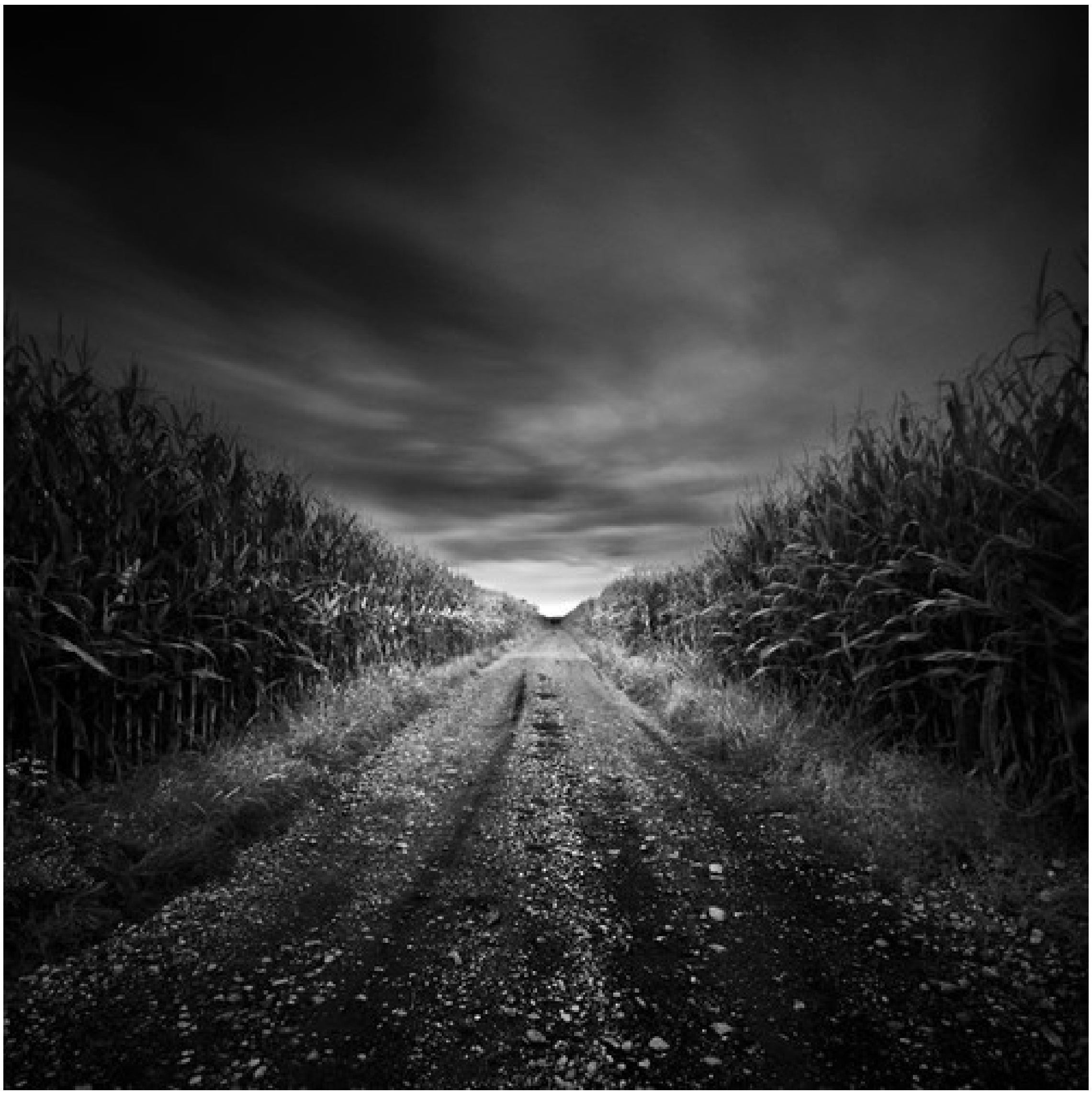


Figure 3: Original noise free Field image

.6.2 Lena Images from different noise levels

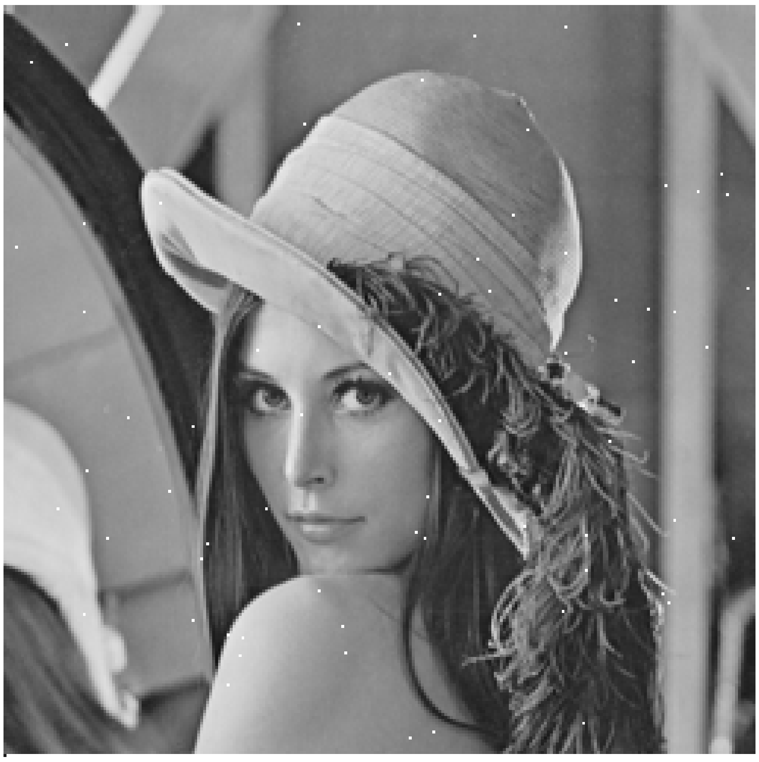


Figure 4: Lena image with noise level 0.001

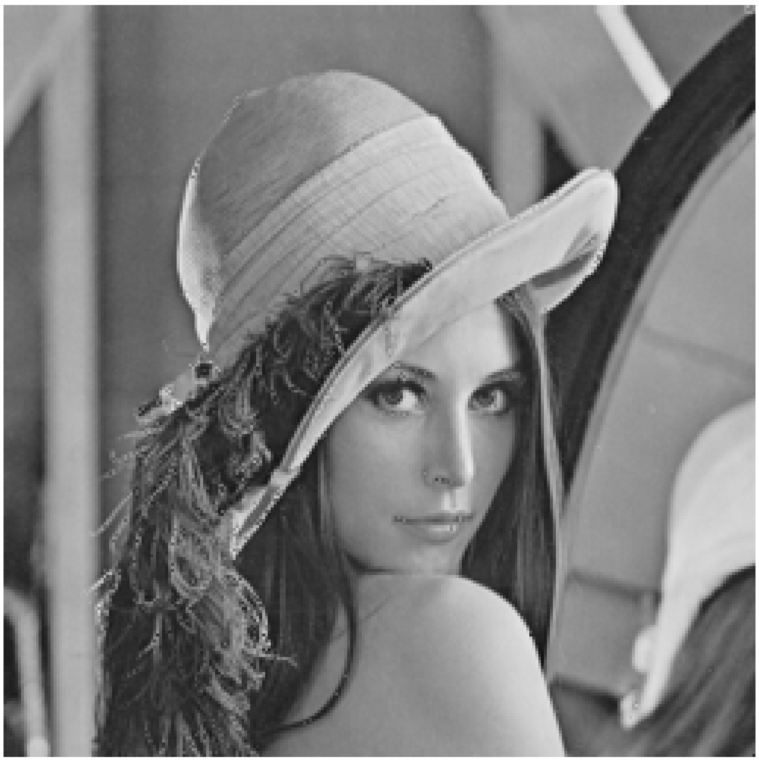


Figure 5: Restored Lena image from noise level 0.001

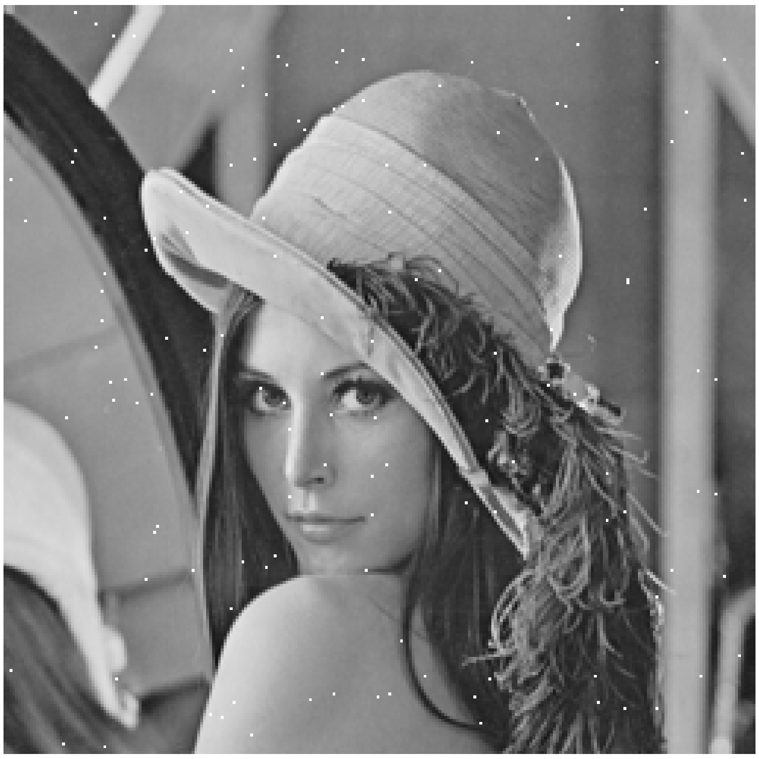


Figure 6: Lena image with noise level 0.002

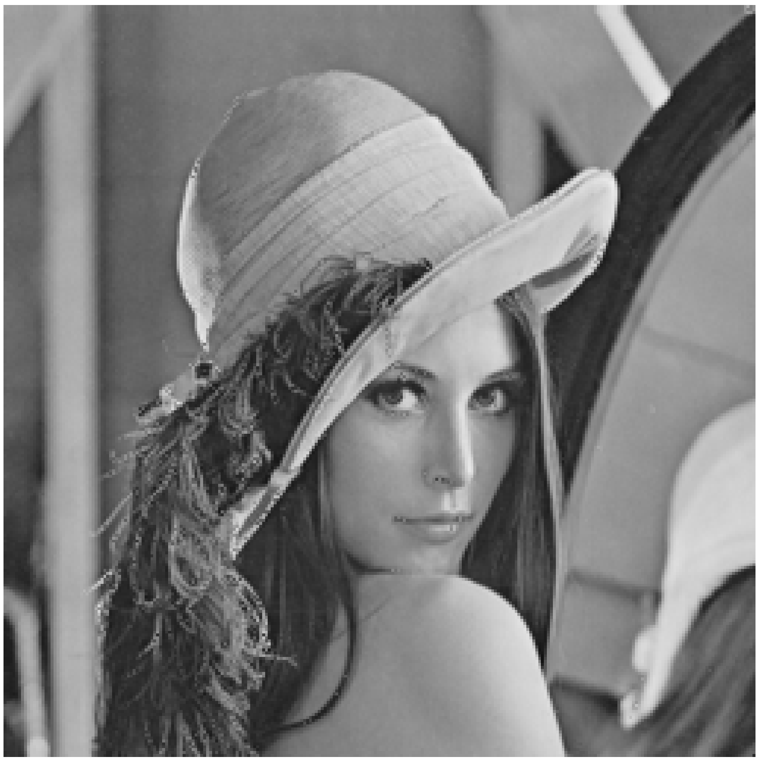


Figure 7: Restored Lena image from noise level 0.002

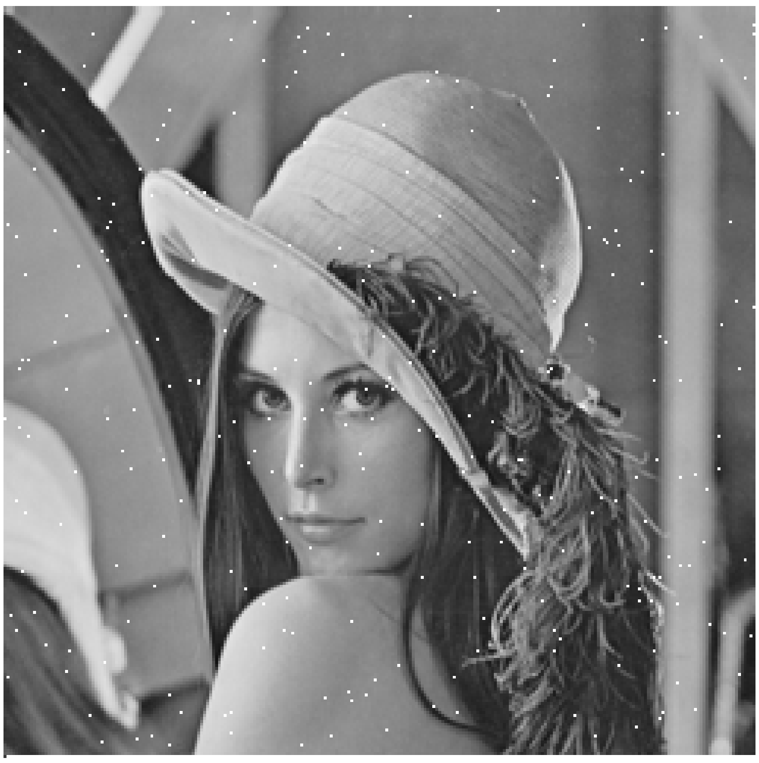


Figure 8: Lena image with noise level 0.004

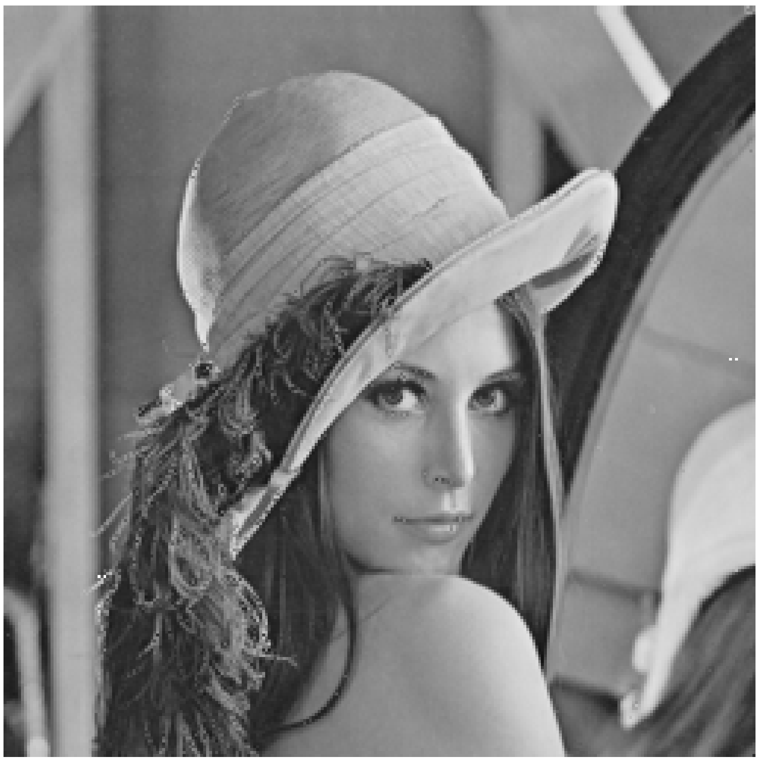


Figure 9: Restored Lena image from noise level 0.004

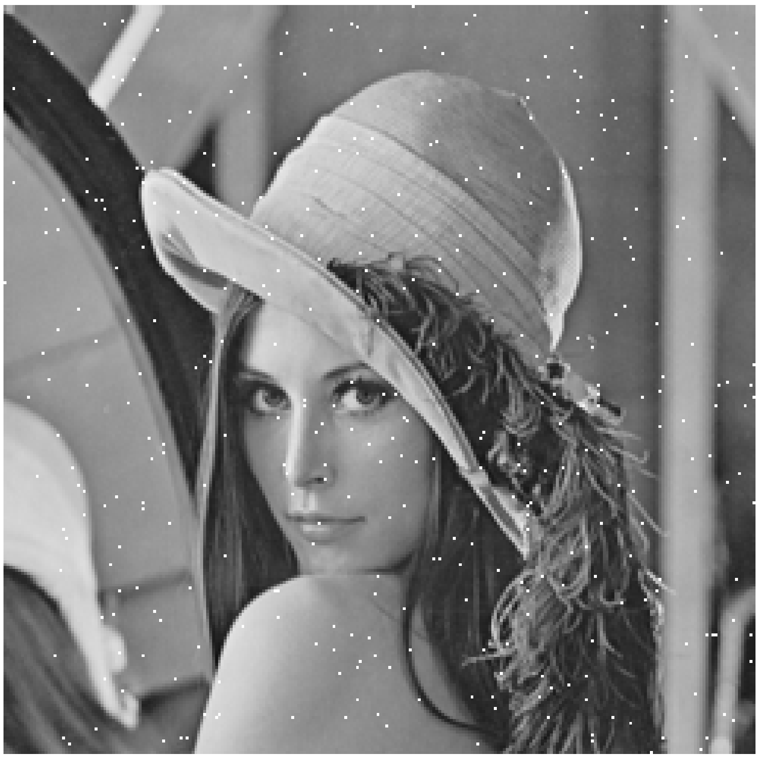


Figure 10: Lena image with noise level 0.006

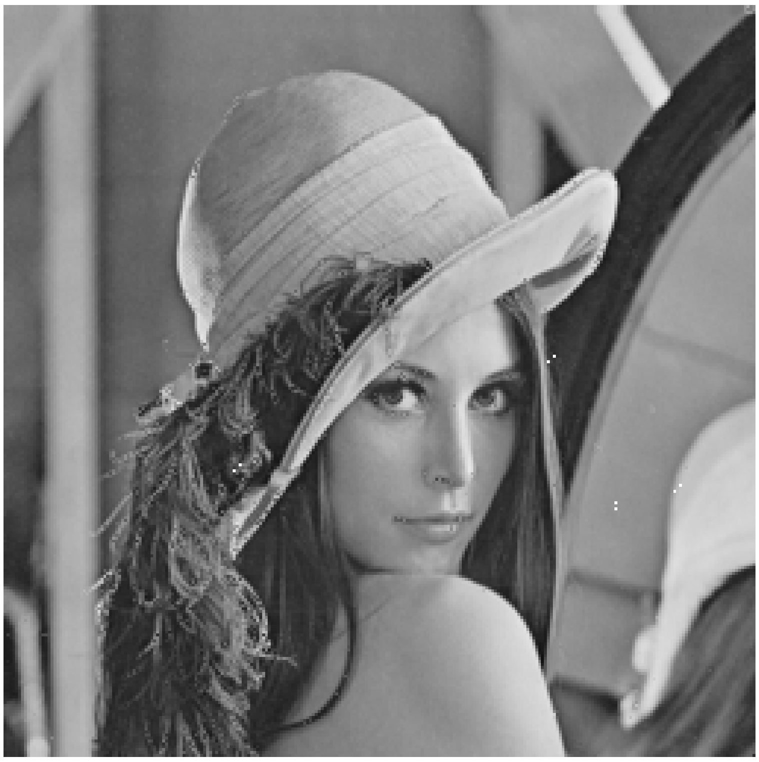


Figure 11: Restored image from noise level 0.006

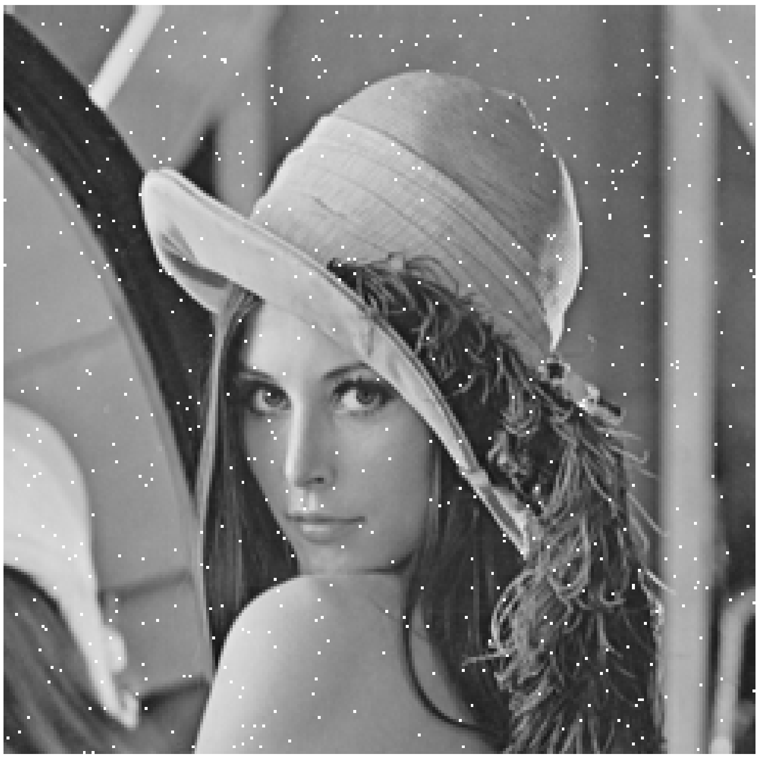


Figure 12: Lena image with noise level 0.008

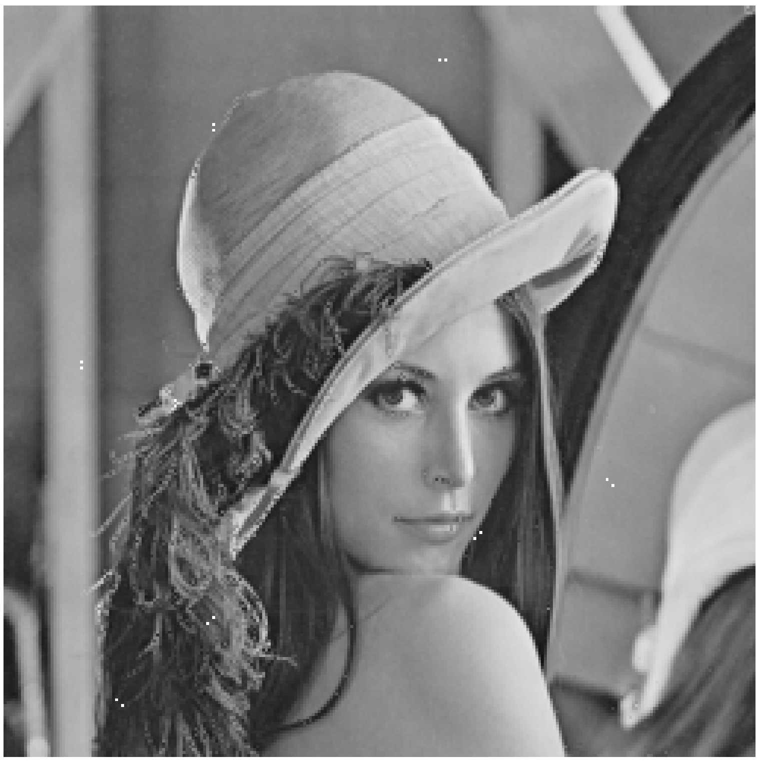


Figure 13: Restored Lena image from noise level 0.008

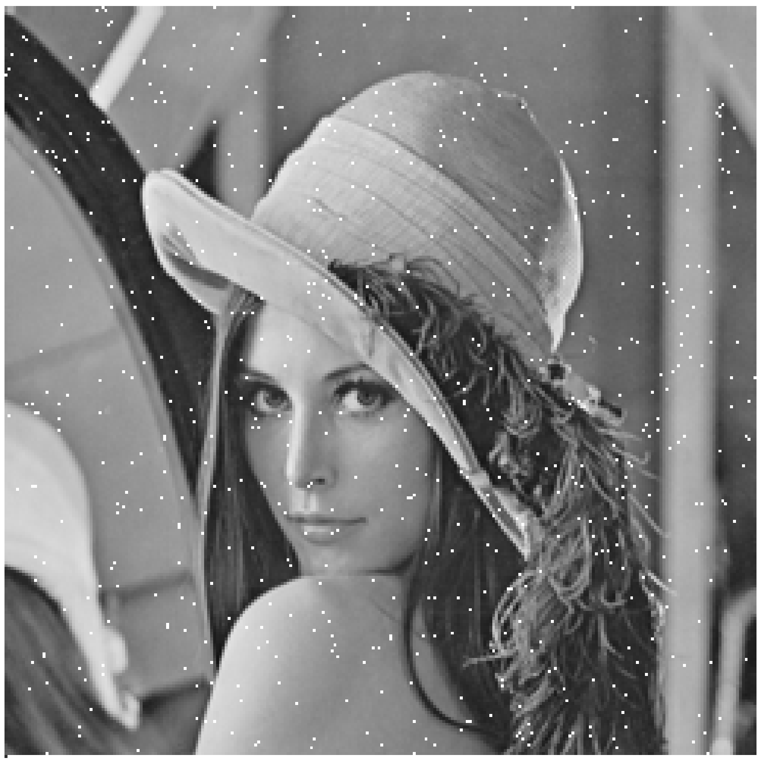


Figure 14: Lena image with noise level 0.01

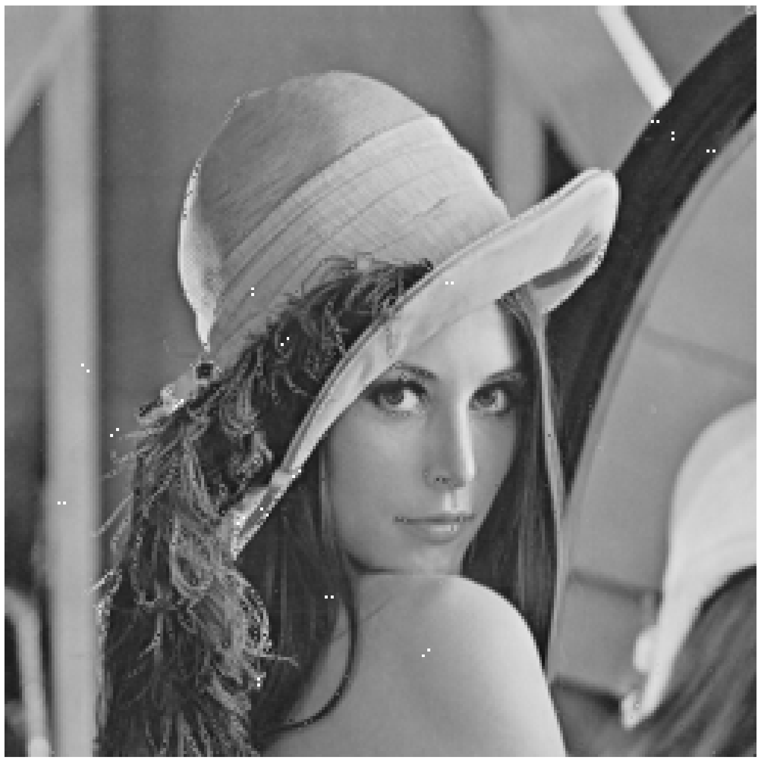


Figure 15: Restored Lena image from noise level 0.01

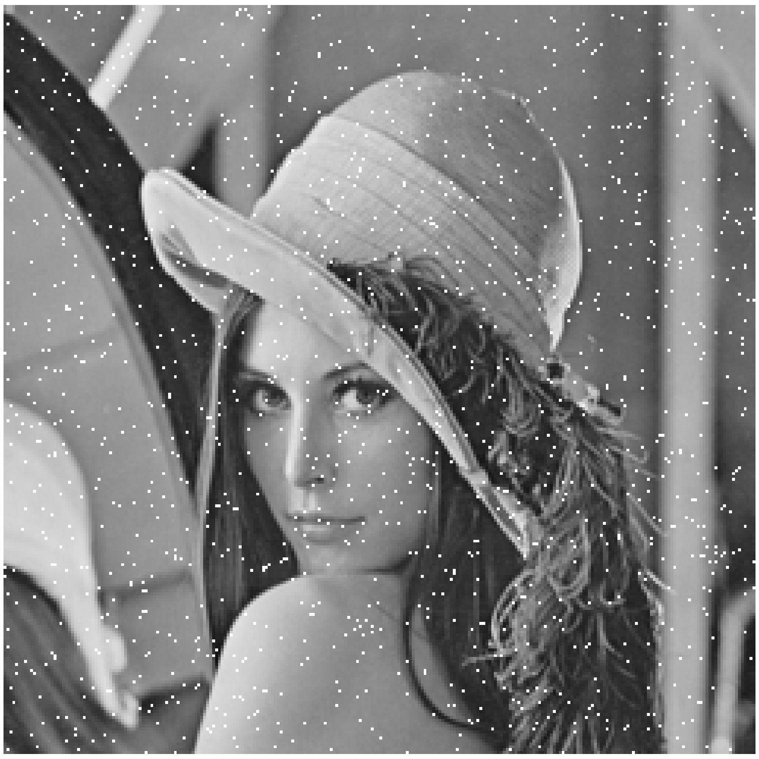


Figure 16: Lena image with noise level 0.02

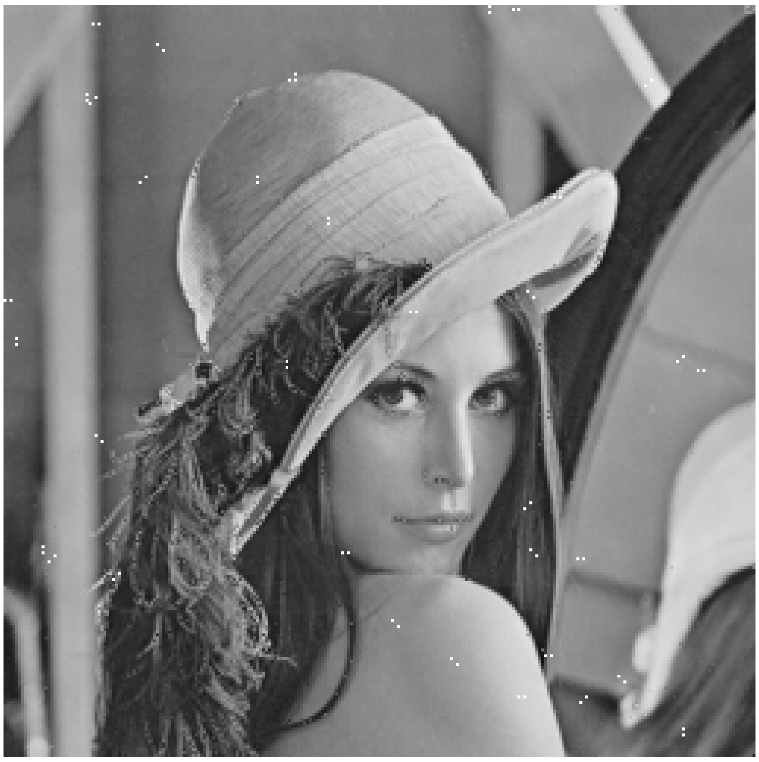


Figure 17: Restored Lena image from noise level 0.02

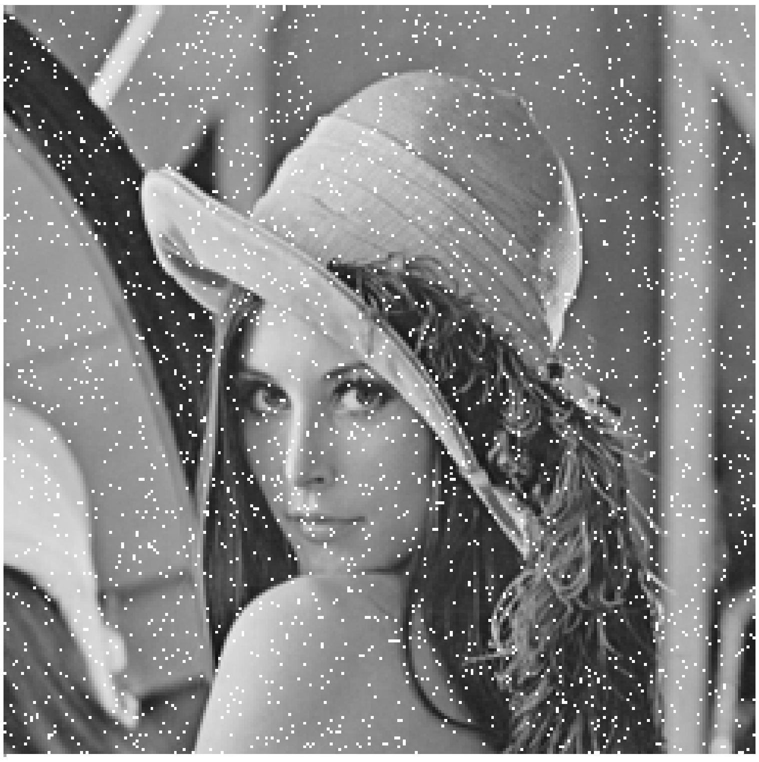


Figure 18: Lena image with noise level 0.04

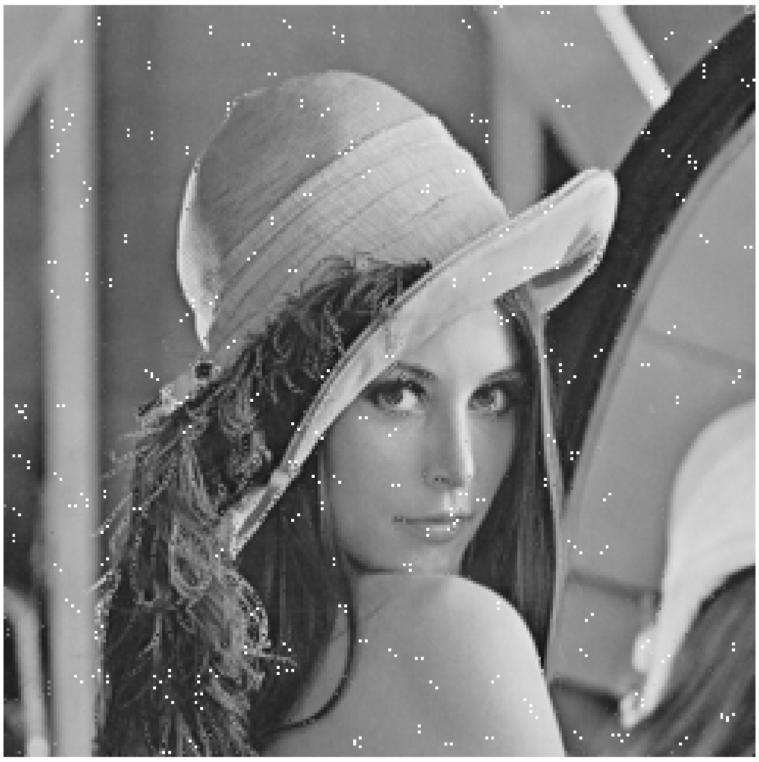


Figure 19: Restored Lena image from noise level 0.04

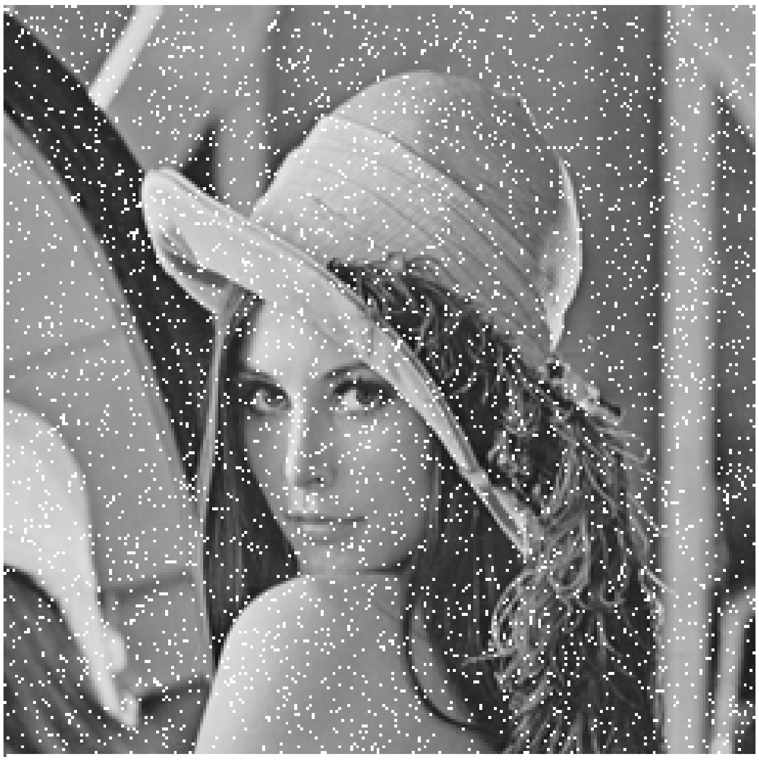


Figure 20: Lena image with noise level 0.06

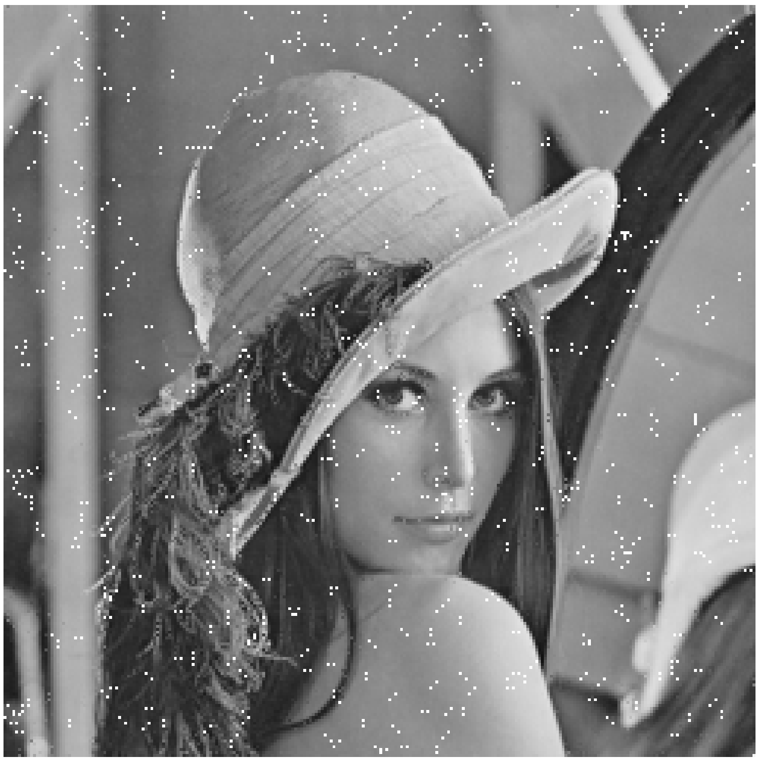


Figure 21: Restored Lena image from noise level 0.06

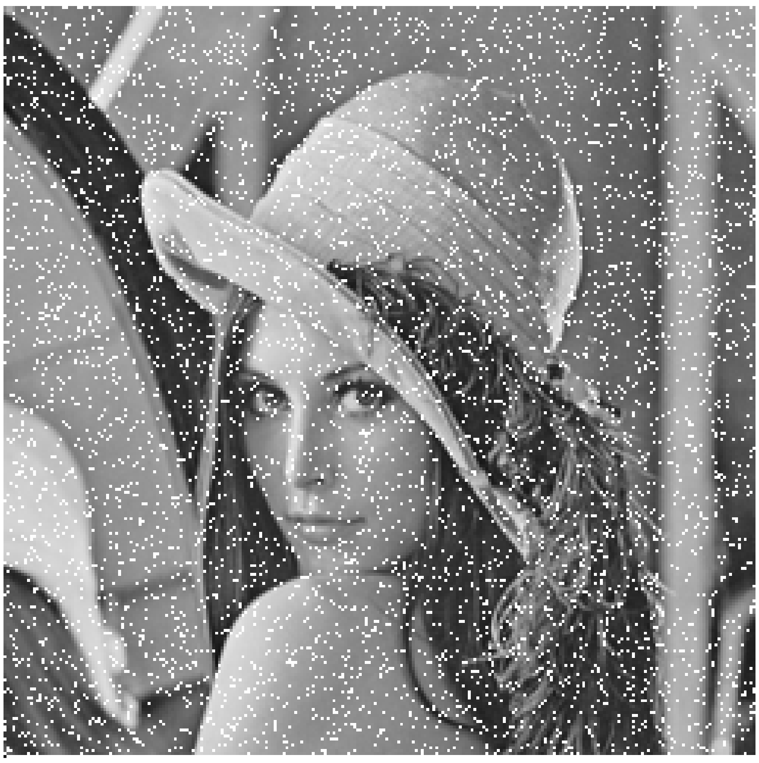


Figure 22: Lena image with noise level 0.08

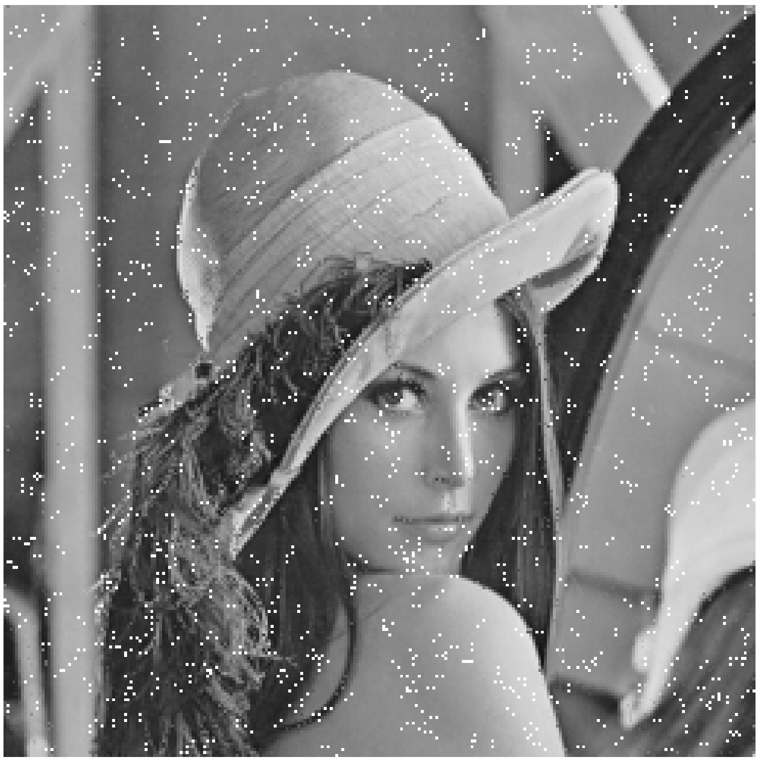


Figure 23: Restored Lena image from noise level 0.08

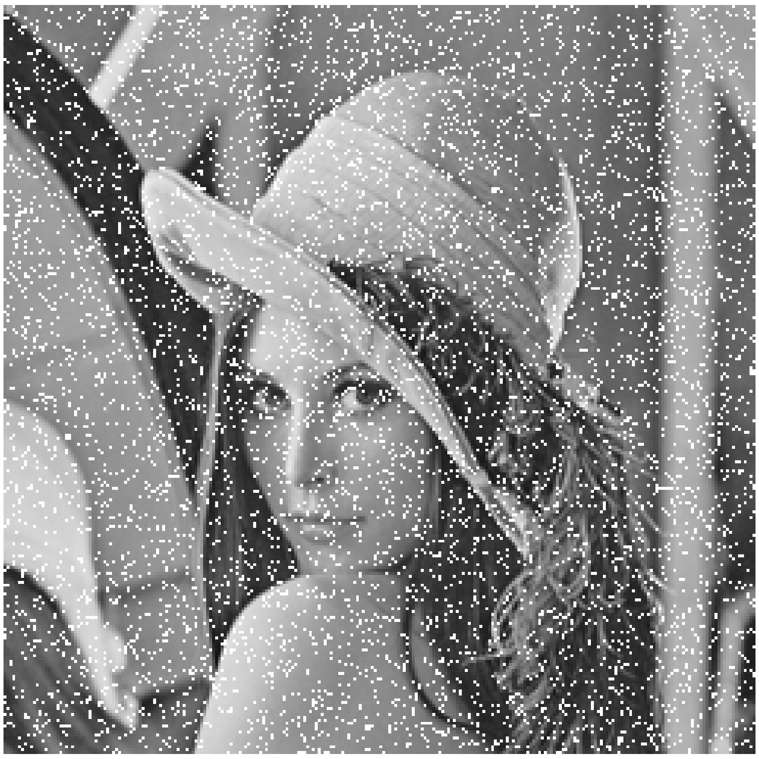


Figure 24: Lena image with noise level 0.1

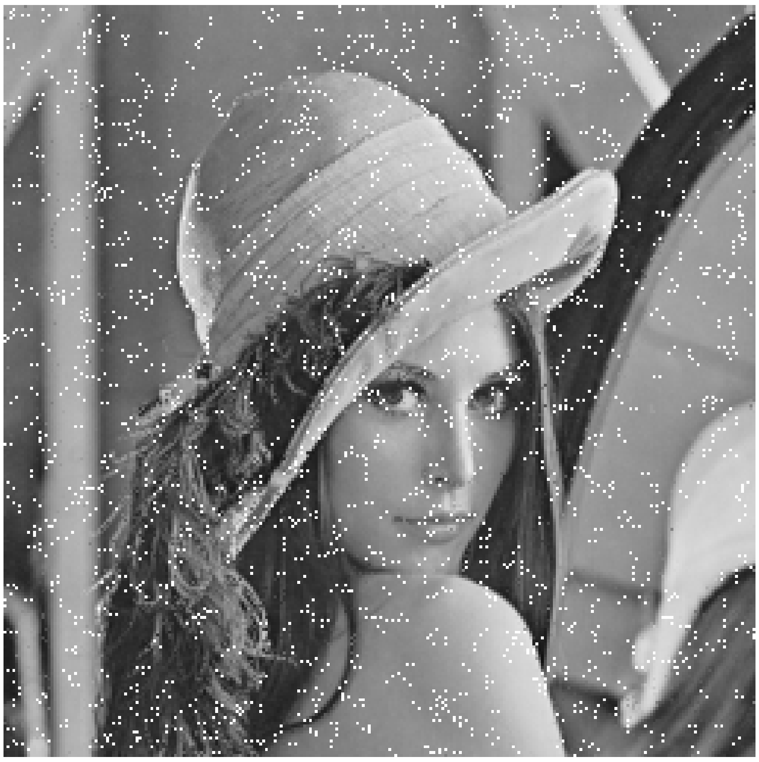


Figure 25: Restored Lena image from noise level 0.1

.6.3 Lena Images from different threshold levels

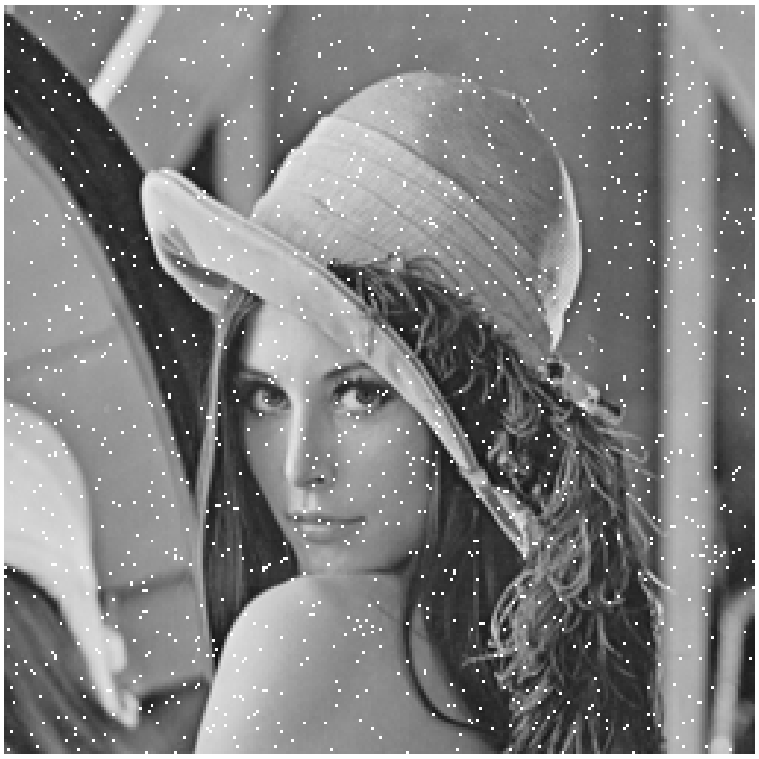


Figure 26: Lena image with noise level 0.02

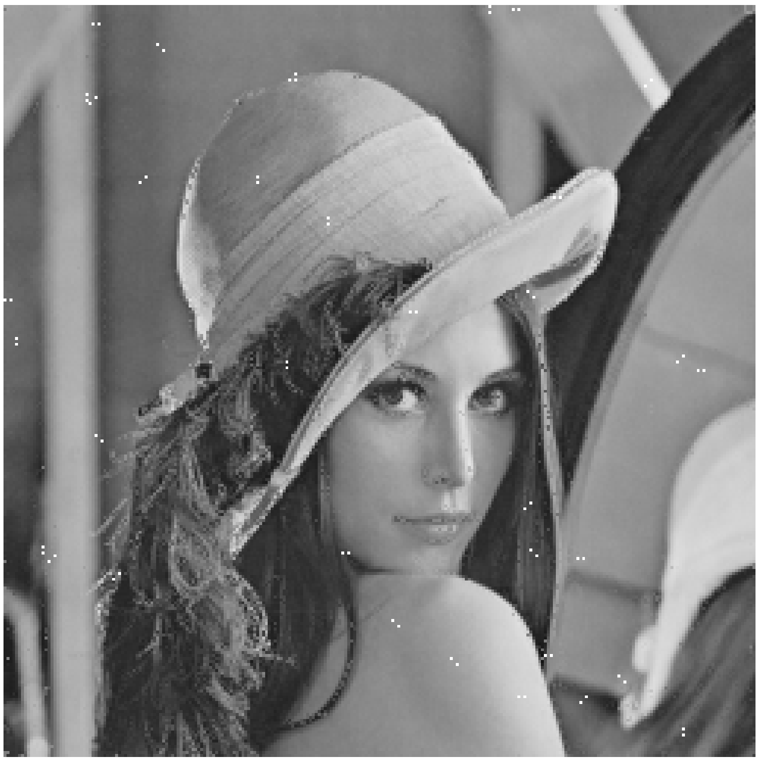


Figure 27: Restored Lena image from threshold level 1

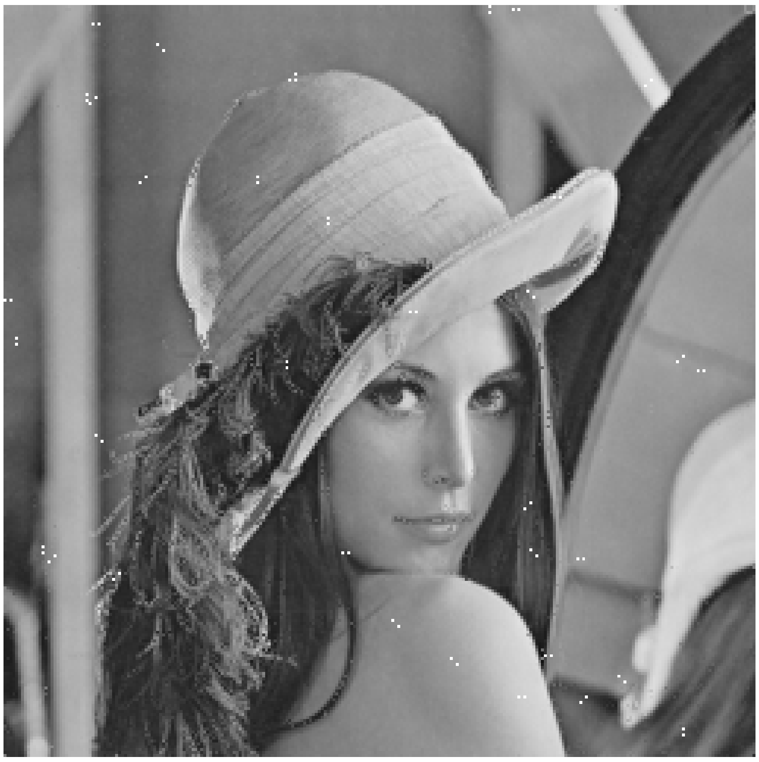


Figure 28: Restored Lena image from threshold level 4

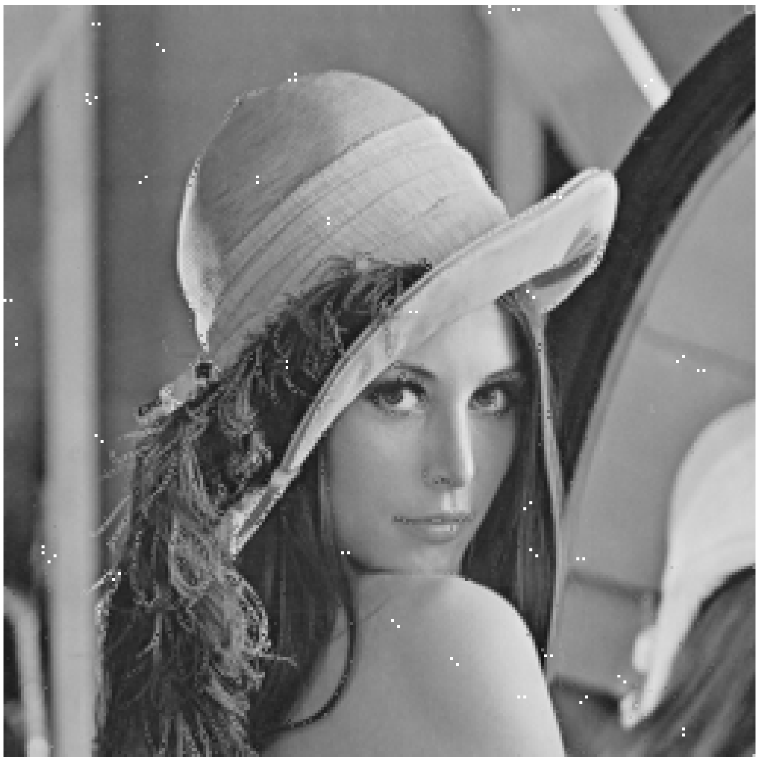


Figure 29: Restored Lena image from threshold level 8

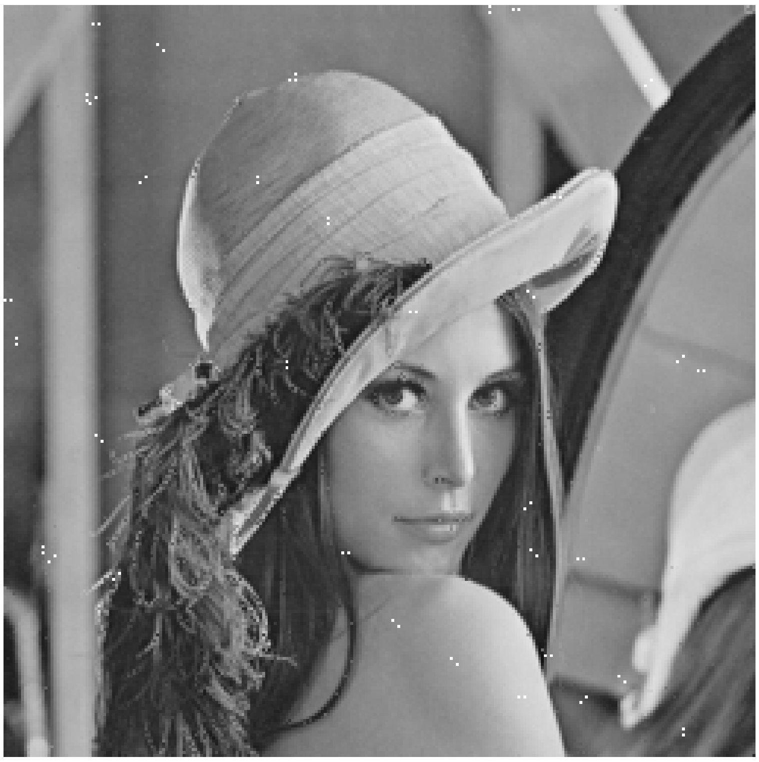


Figure 30: Restored Lena image from threshold level 12

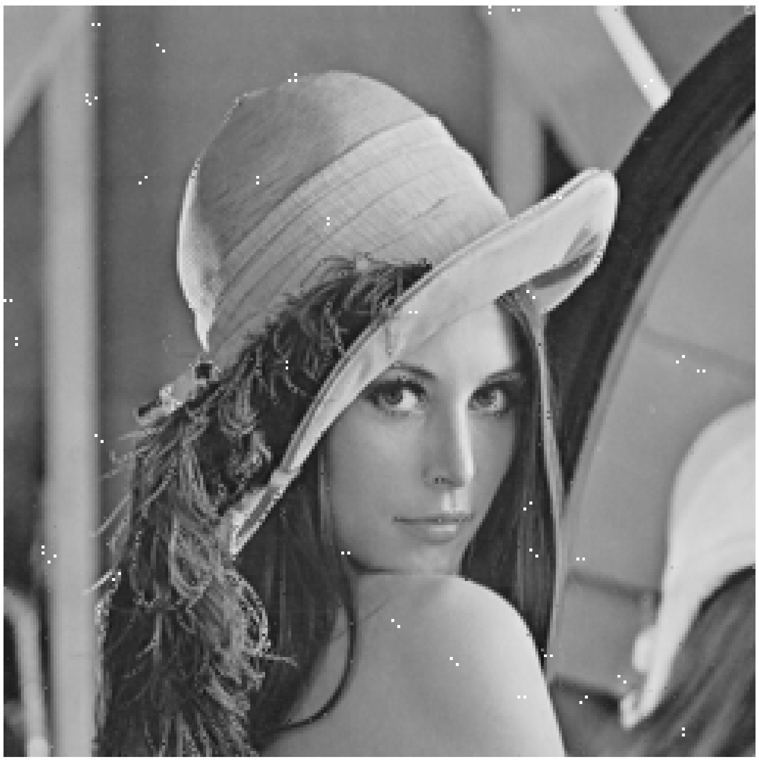


Figure 31: Restored image from threshold level 16

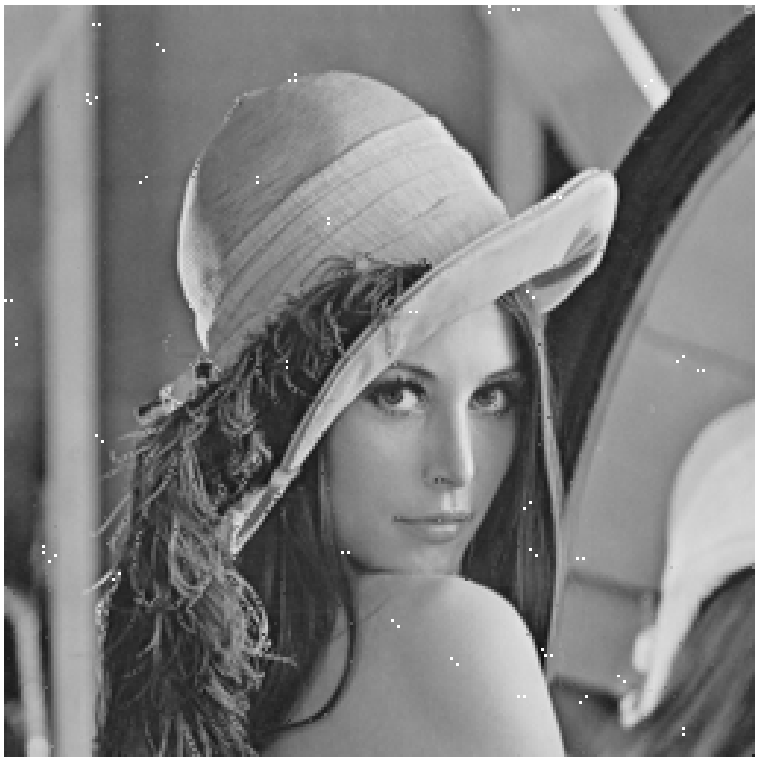


Figure 32: Restored Lena image from threshold level 20

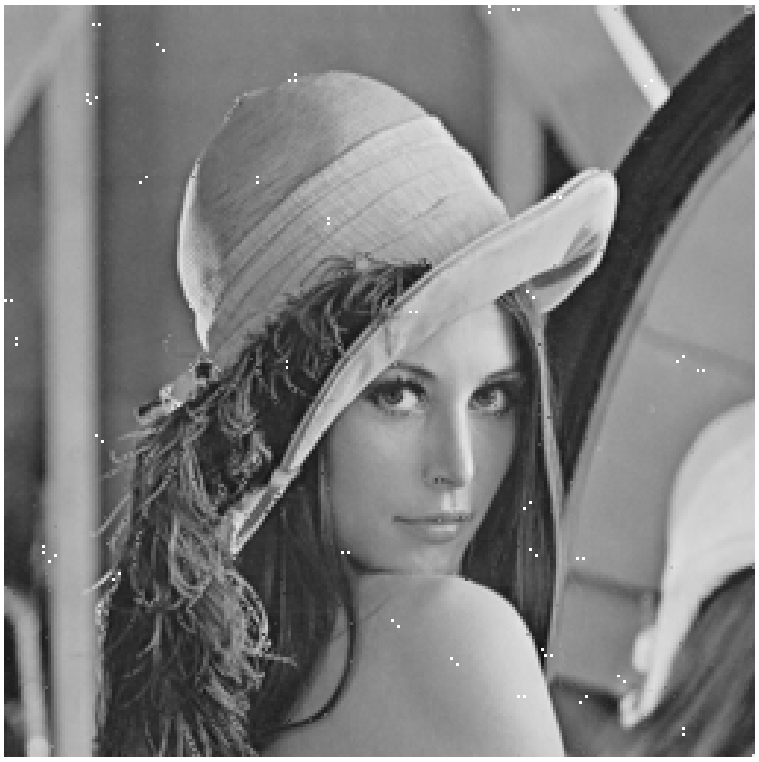


Figure 33: Restored Lena image from threshold level 22

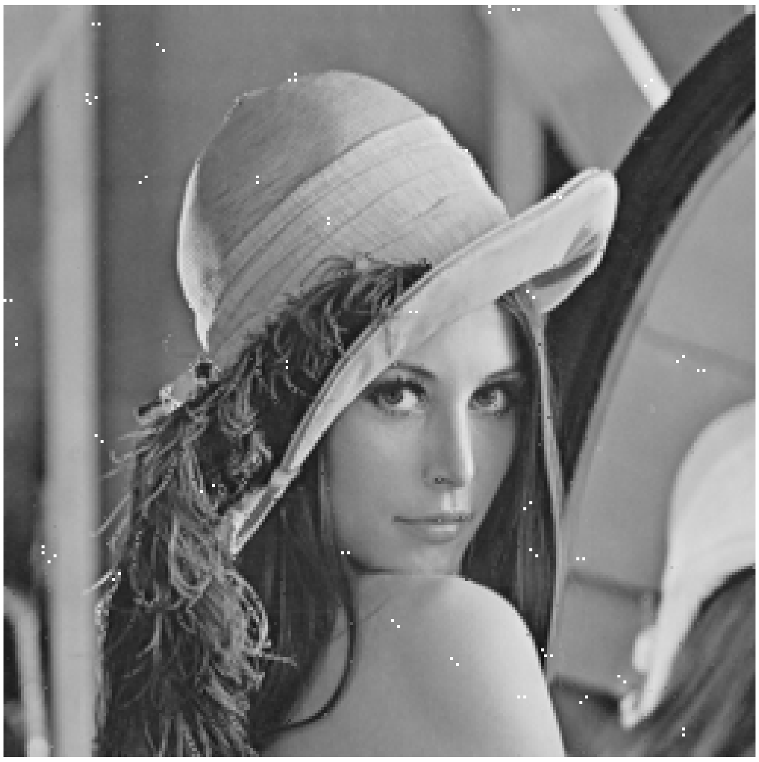


Figure 34: Restored Lena image from threshold level 26

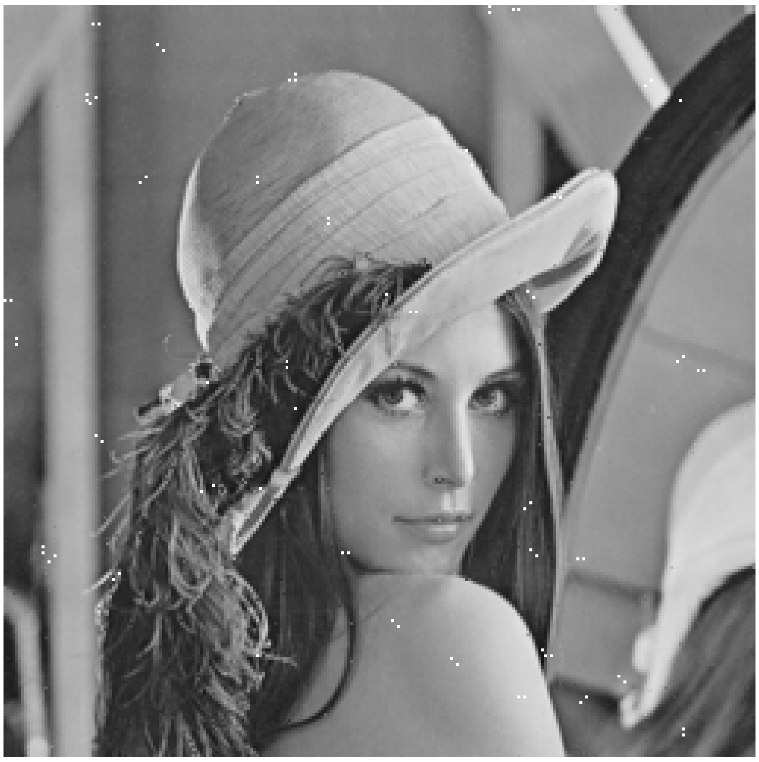


Figure 35: Restored Lena image from threshold level 30

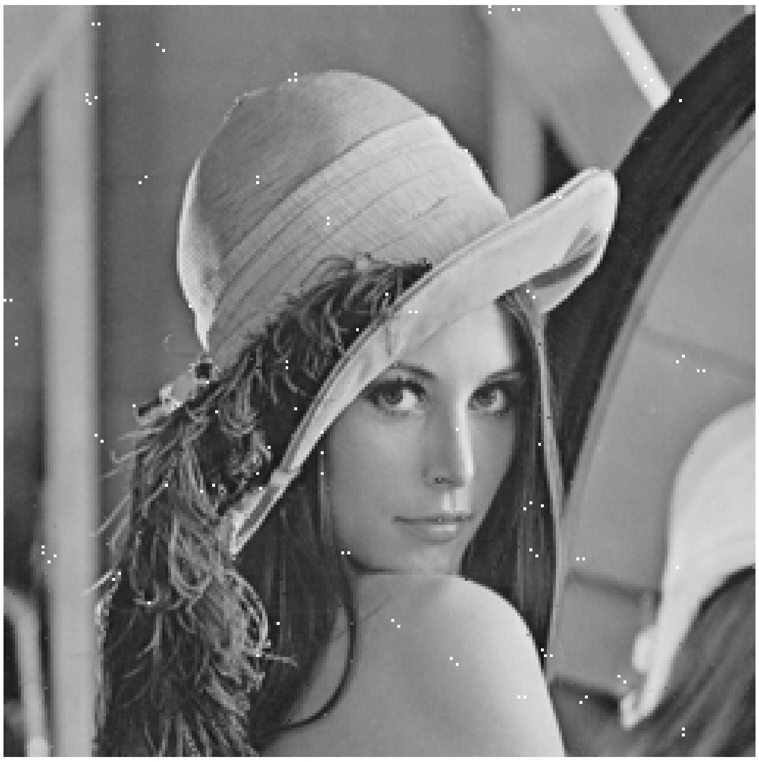


Figure 36: Restored Lena image from threshold level 34

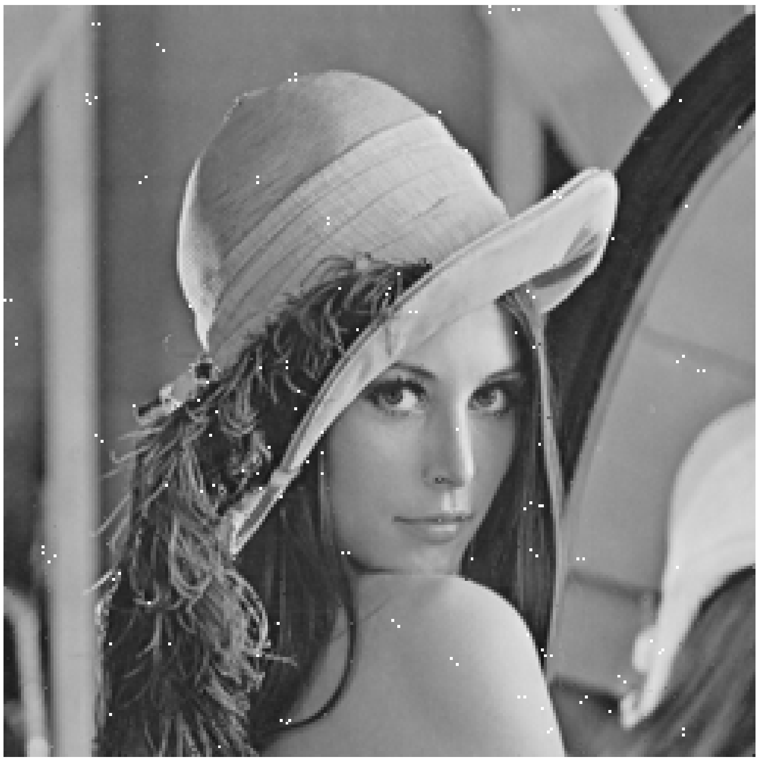


Figure 37: Restored Lena image from threshold level 38

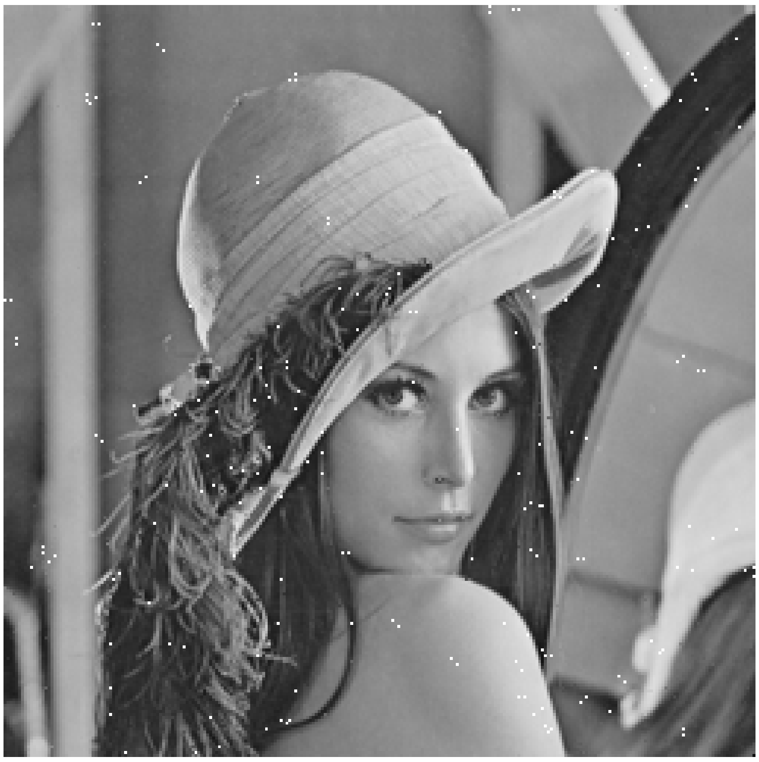


Figure 38: Restored Lena image from threshold level 42

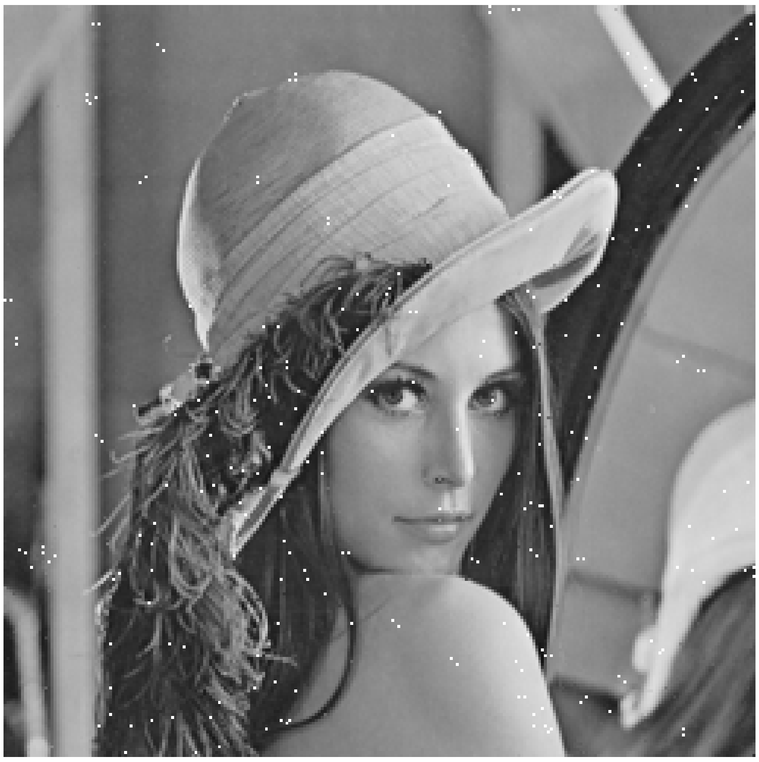


Figure 39: Restored Lena image from threshold level 46

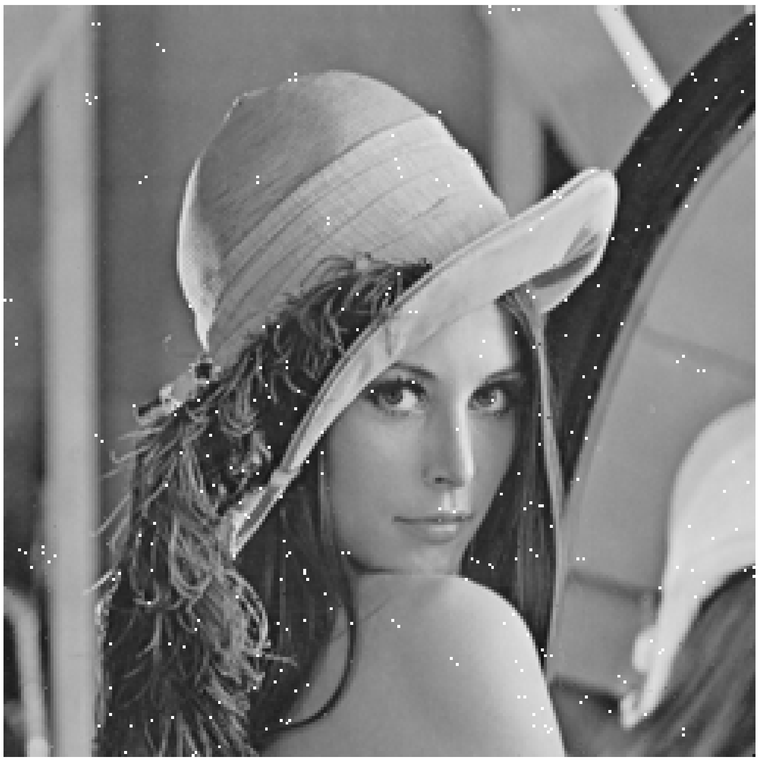


Figure 40: Restored Lena image from threshold level 50

.6.4 Field Images from different noise levels

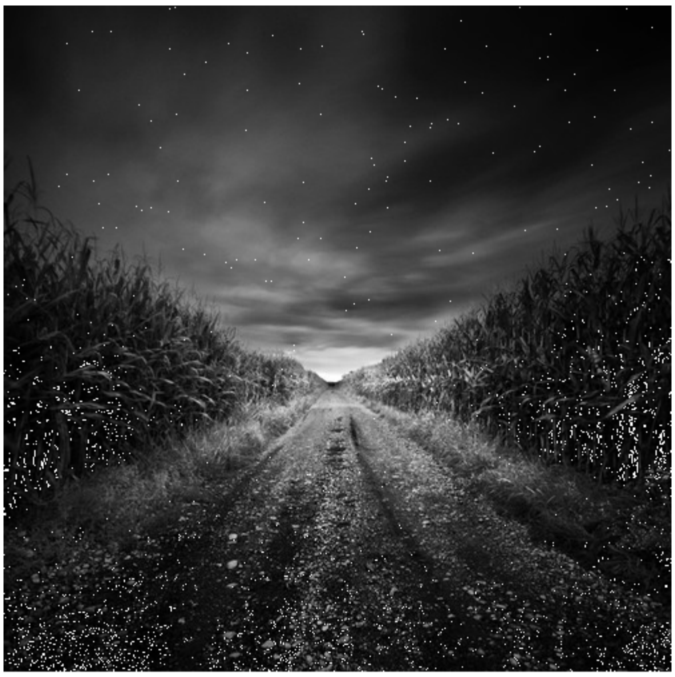


Figure 41: Field image with noise level 0.001

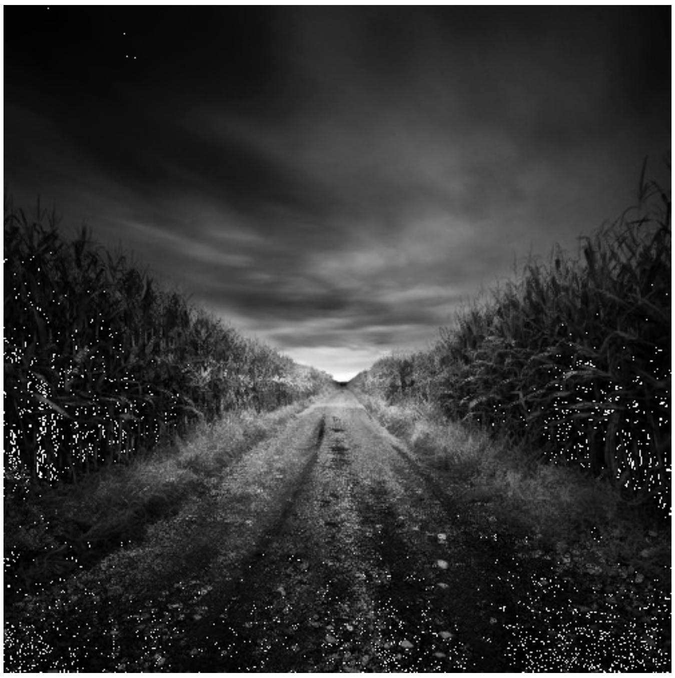


Figure 42: Restored Field image from noise level 0.001

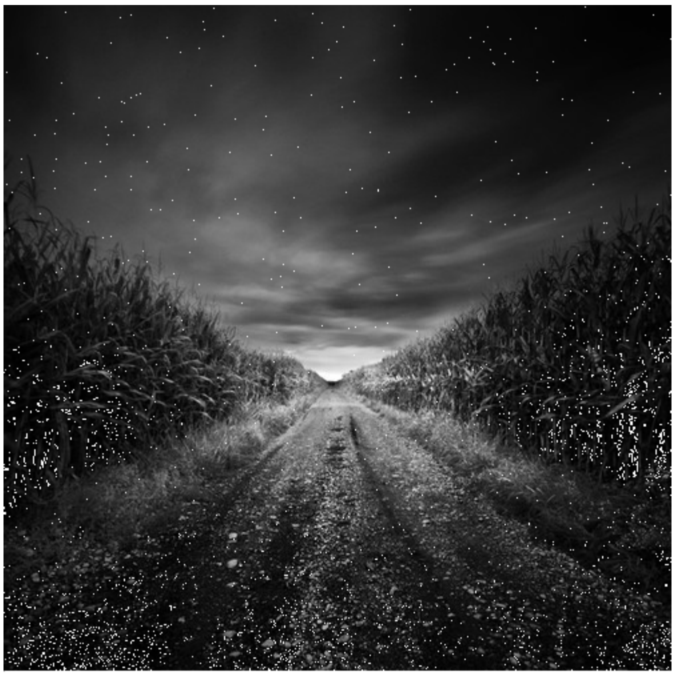


Figure 43: Field image with noise level 0.002

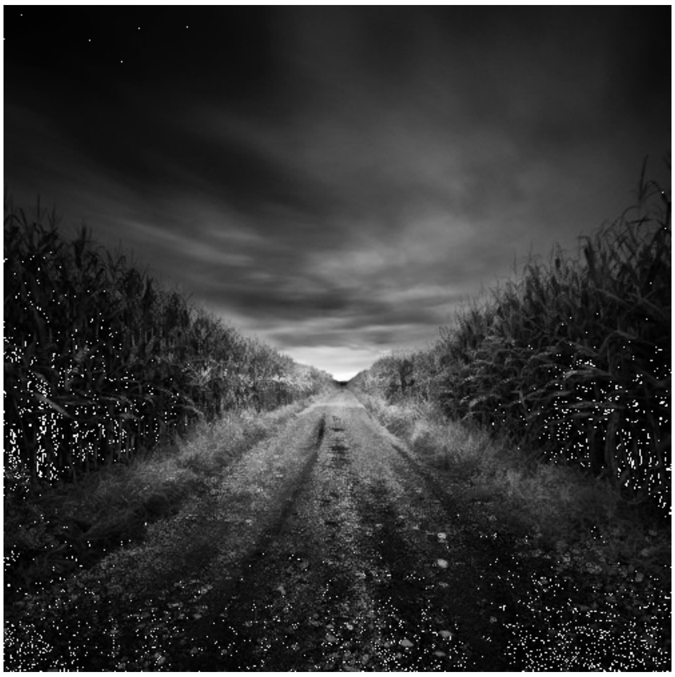


Figure 44: Restored Field image from noise level 0.002

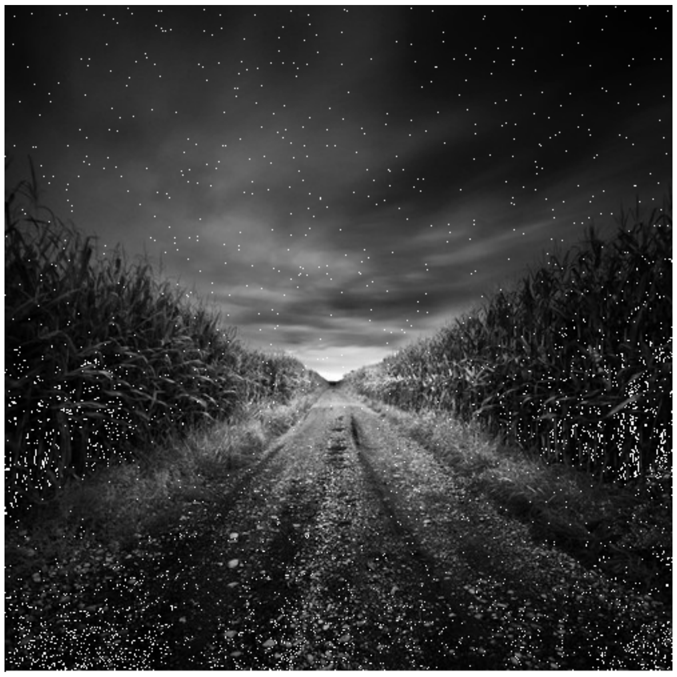


Figure 45: Field image with noise level 0.004

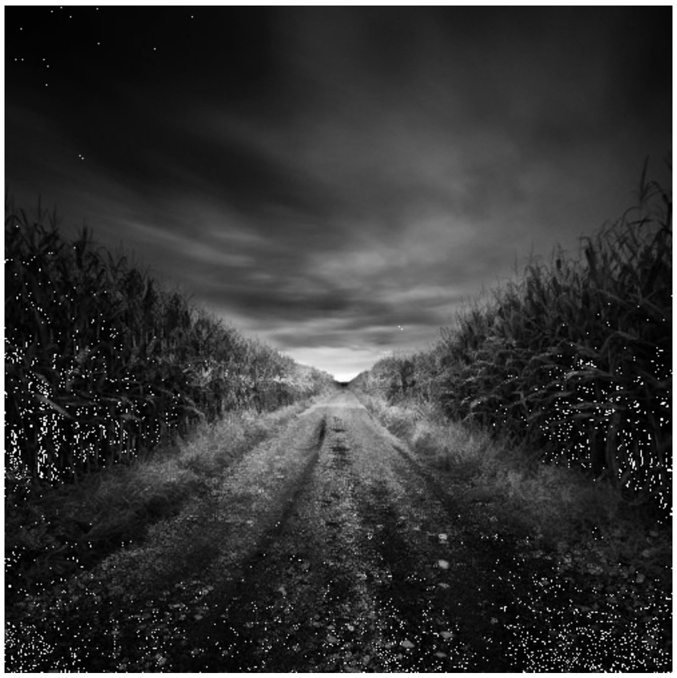


Figure 46: Restored Field image from noise level 0.004

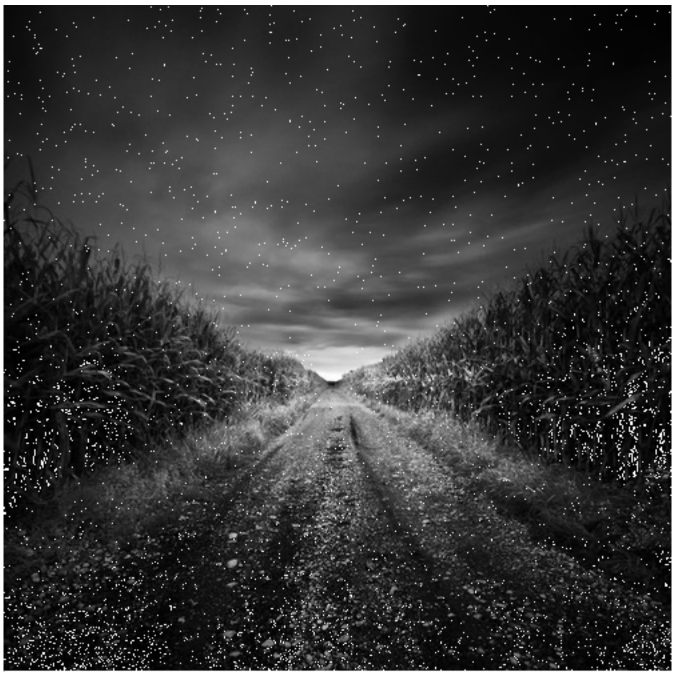


Figure 47: Field image with noise level 0.006

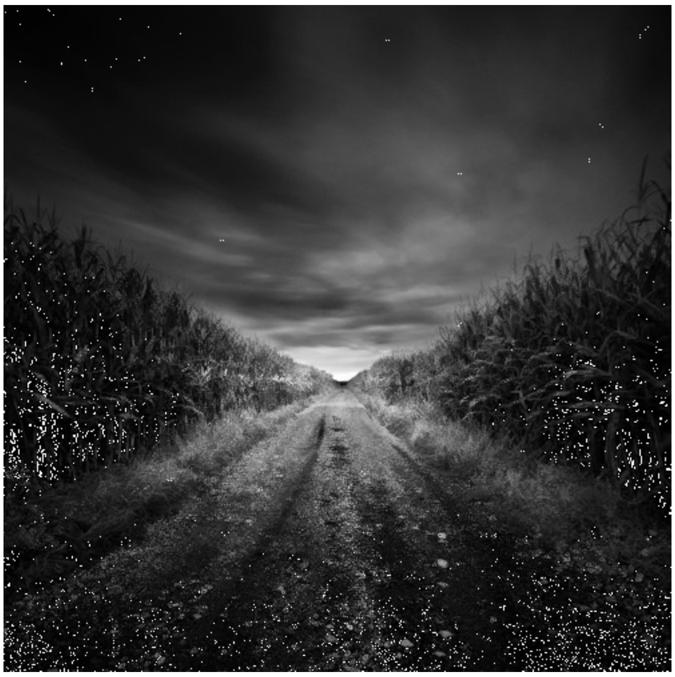


Figure 48: Restored Field image from noise level 0.006

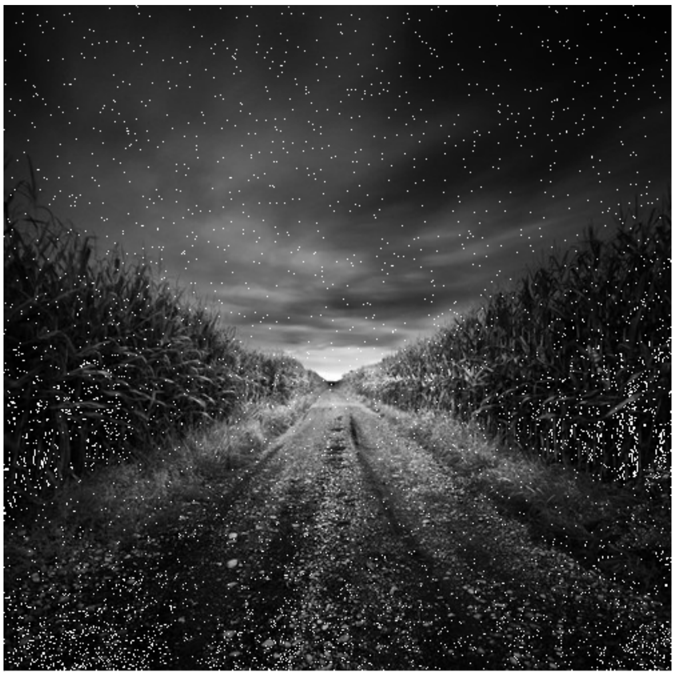


Figure 49: Field image with noise level 0.008

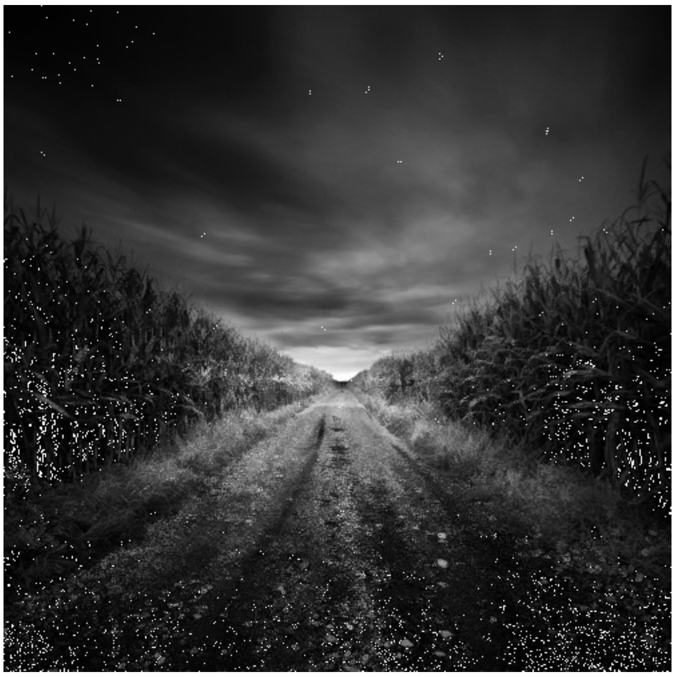


Figure 50: Restored Field image from noise level 0.008

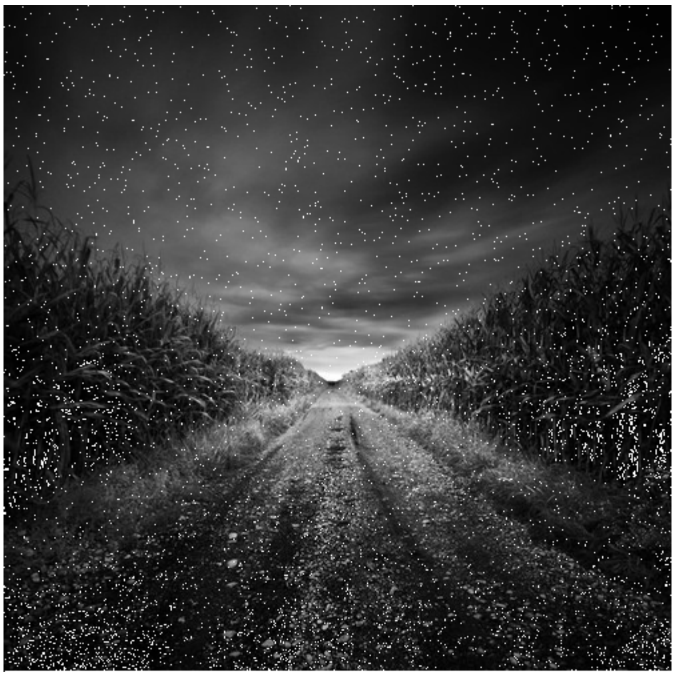


Figure 51: Field image with noise level 0.01

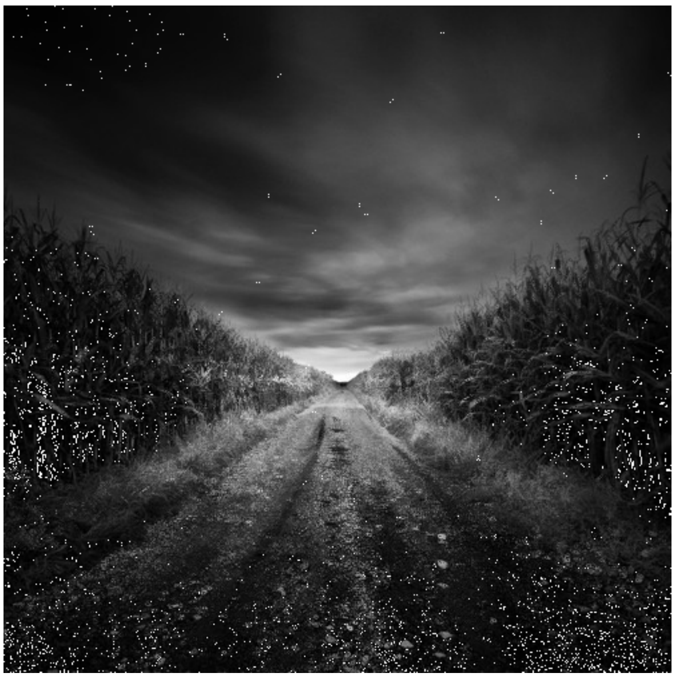


Figure 52: Restored Field image from noise level 0.01

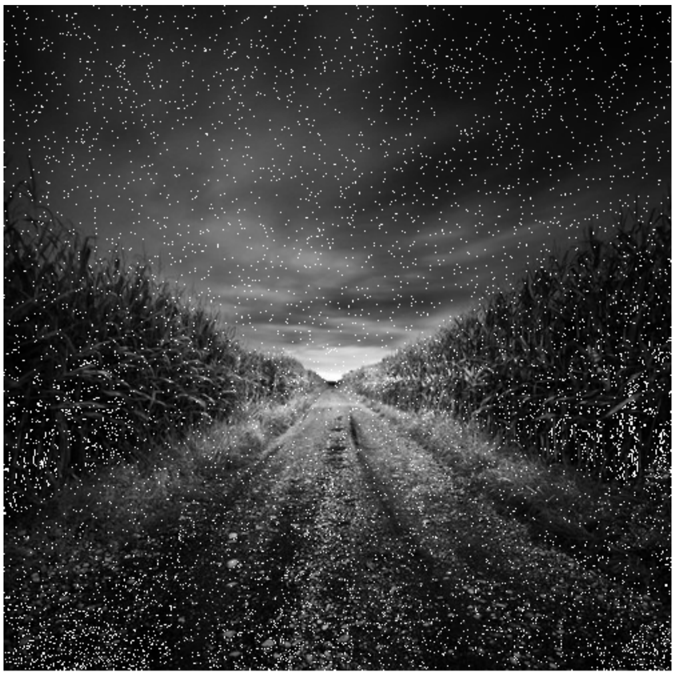


Figure 53: Field image with noise level 0.02

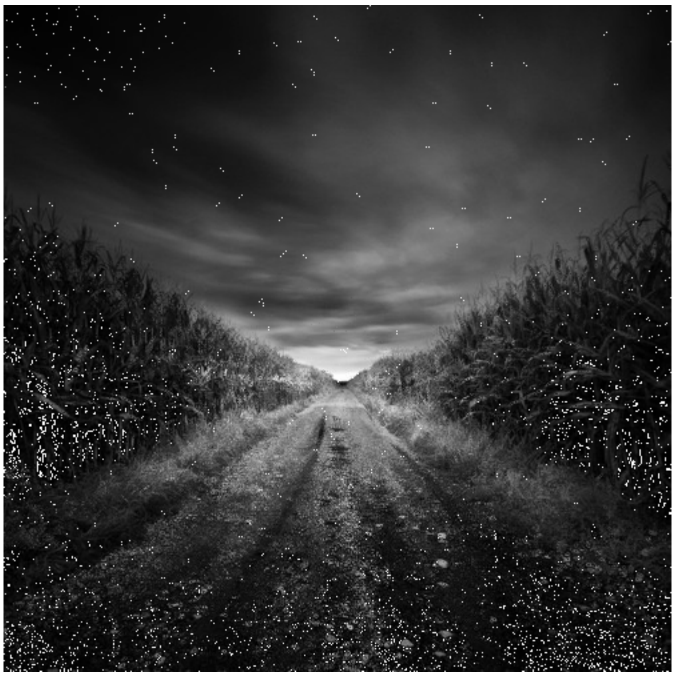


Figure 54: Restored Field image from noise level 0.02

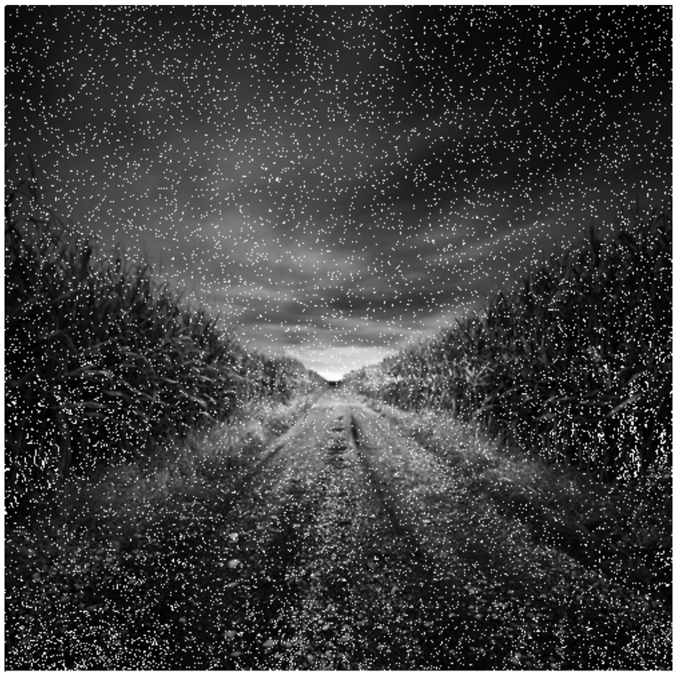


Figure 55: Field image with noise level 0.04

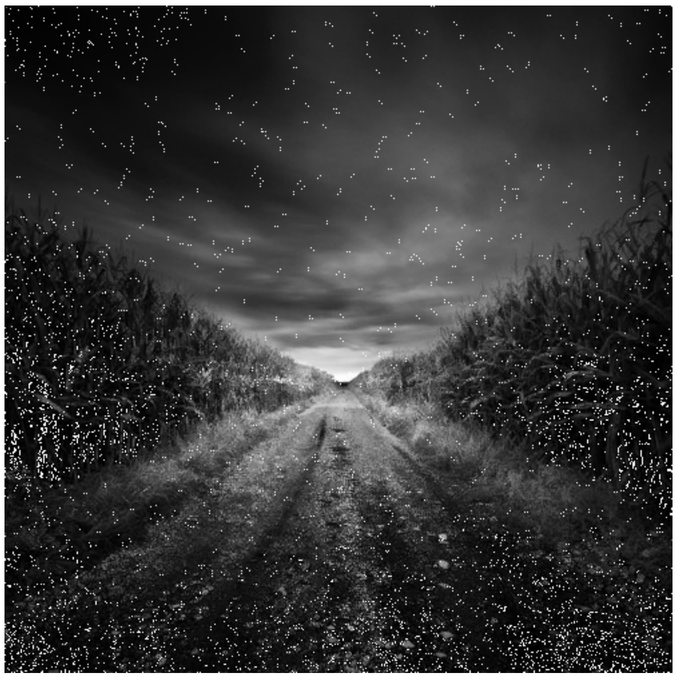


Figure 56: Restored Field image from noise level 0.04

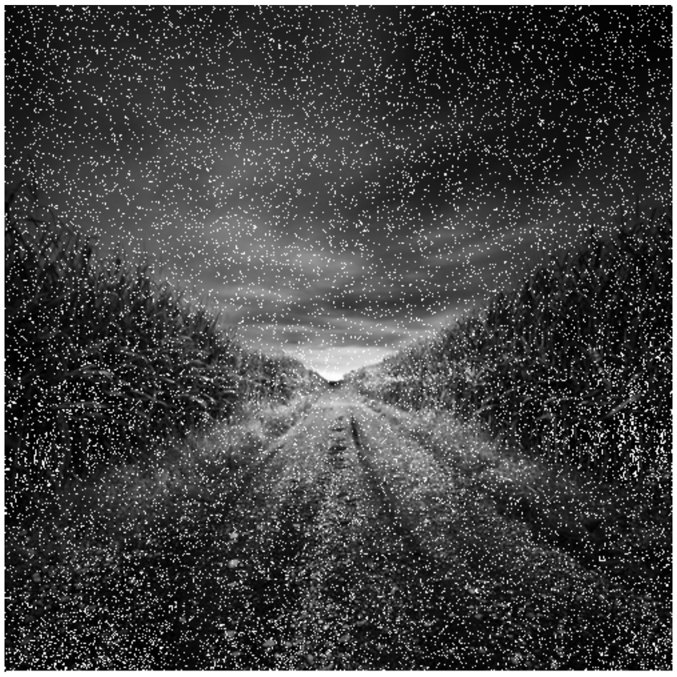


Figure 57: Field image with noise level 0.06

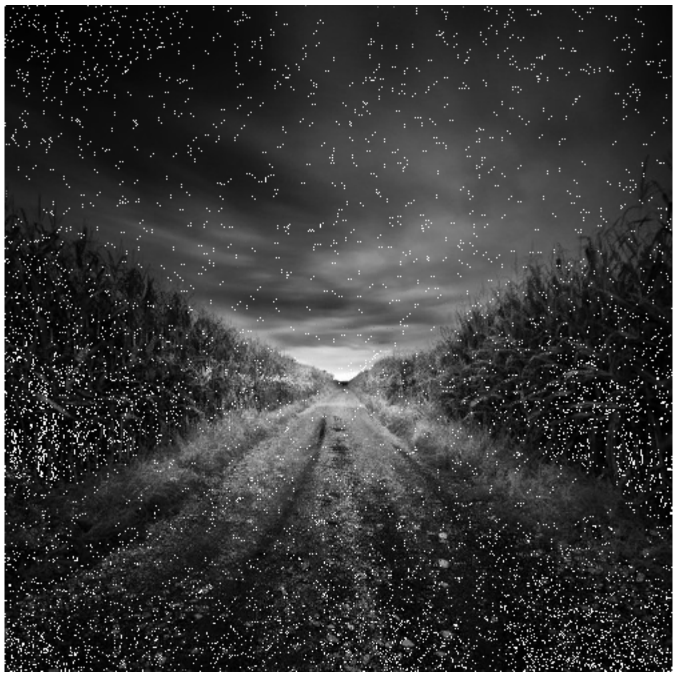


Figure 58: Restored Field image from noise level 0.06

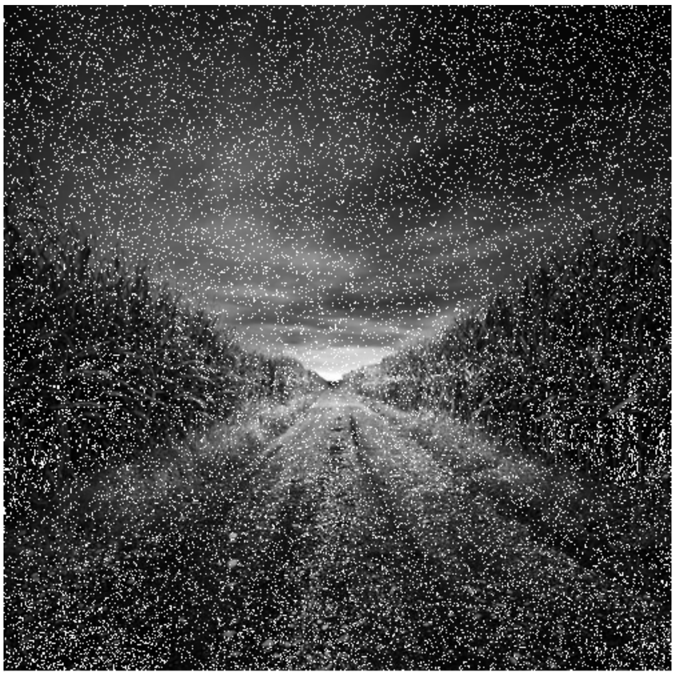


Figure 59: Field image with noise level 0.08

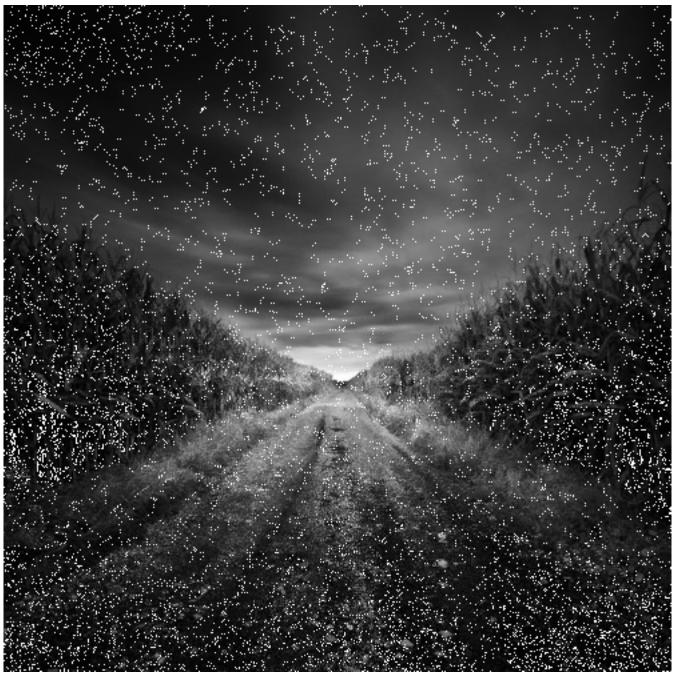


Figure 60: Restored Field image from noise level 0.08

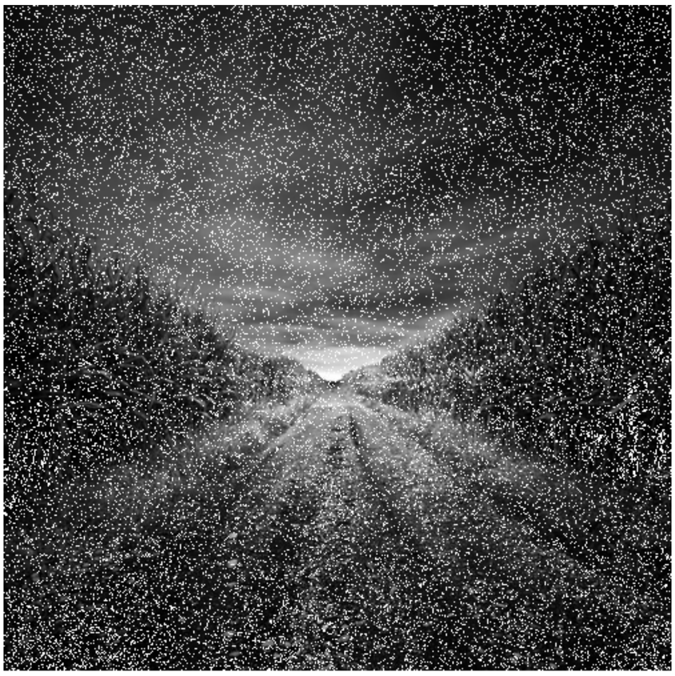


Figure 61: Field image with noise level 0.1

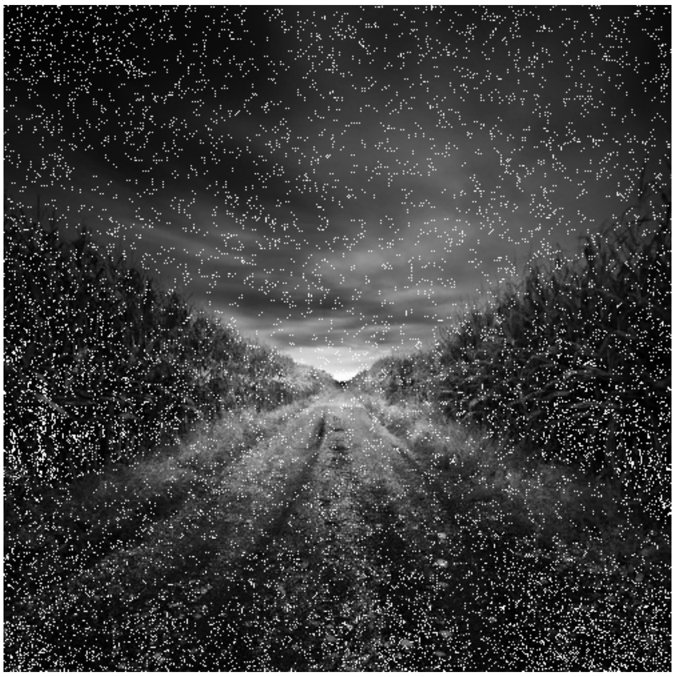


Figure 62: Restored Field image from noise level 0.1

.6.5 Field Images from different threshold levels

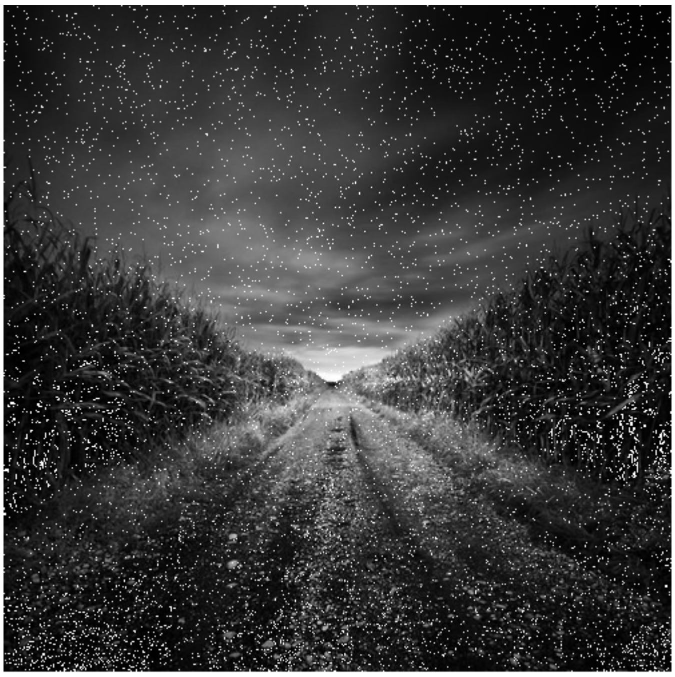


Figure 63: Field image with noise level 0.02

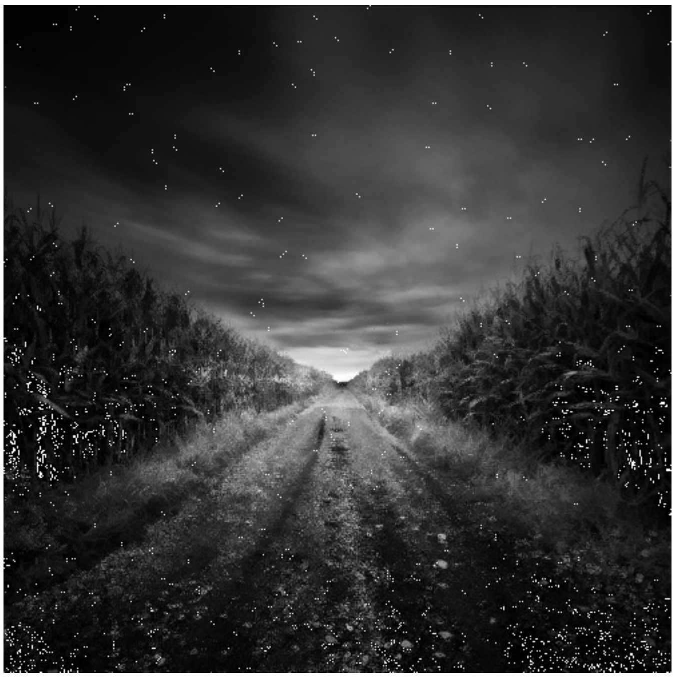


Figure 64: Restored Field image from threshold level 1

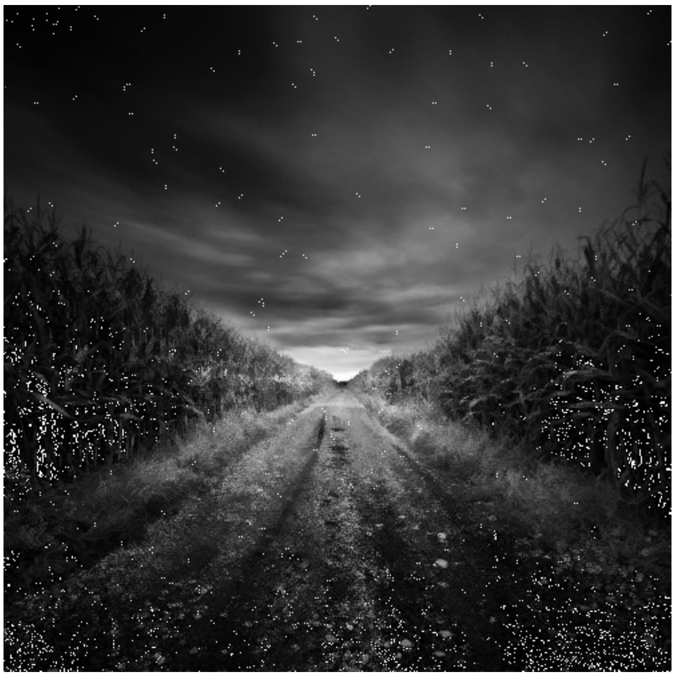


Figure 65: Restored Field image from threshold level 4

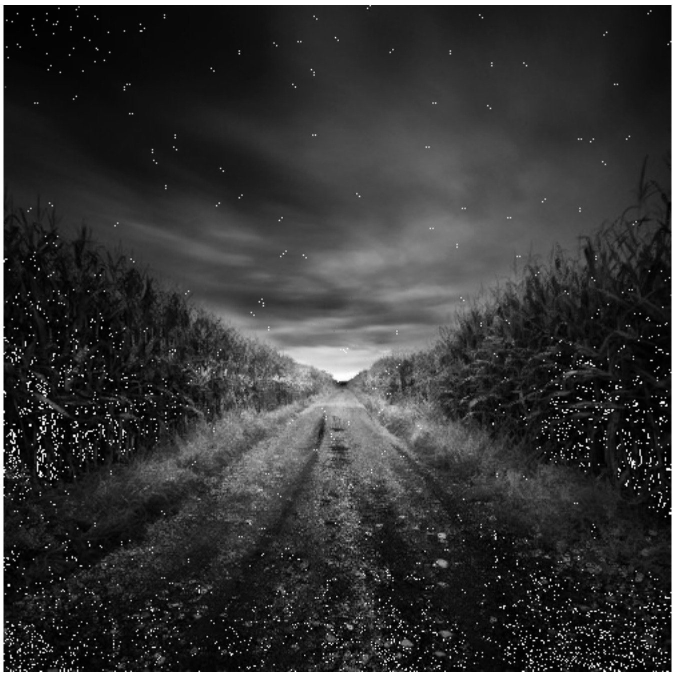


Figure 66: Restored Field image from threshold level 8

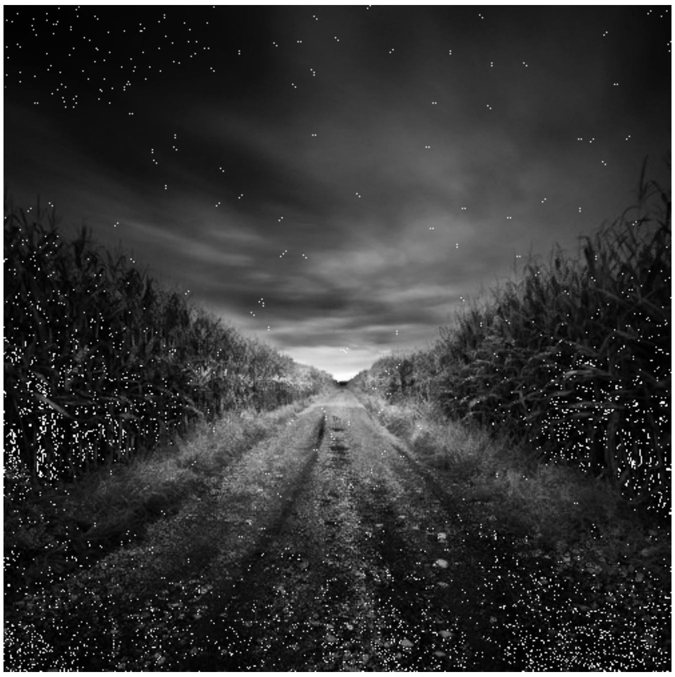


Figure 67: Restored Field image from threshold level 12

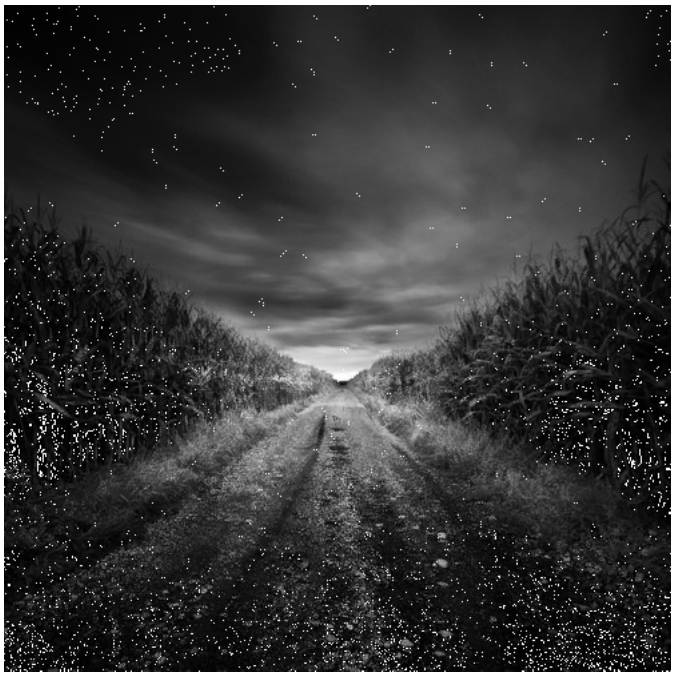


Figure 68: Restored Field image from threshold level 16

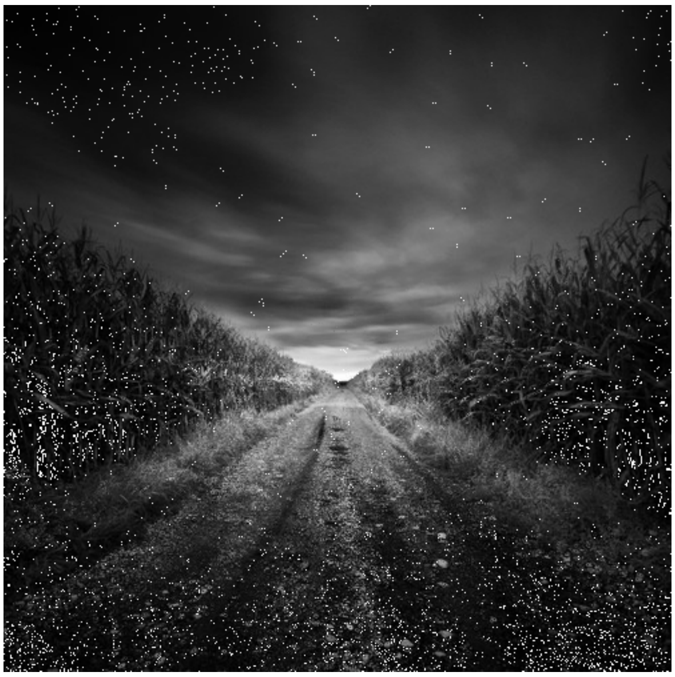


Figure 69: Restored Field image from threshold level 20

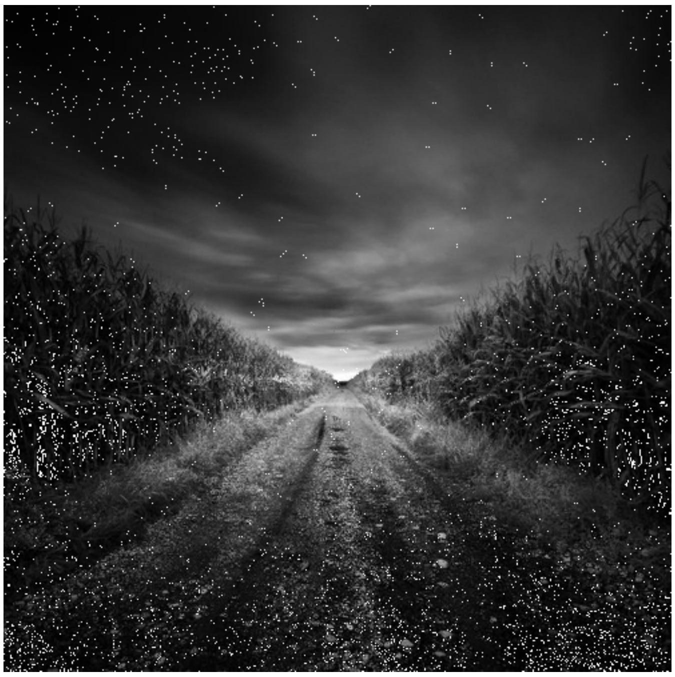


Figure 70: Restored Field image from threshold level 22

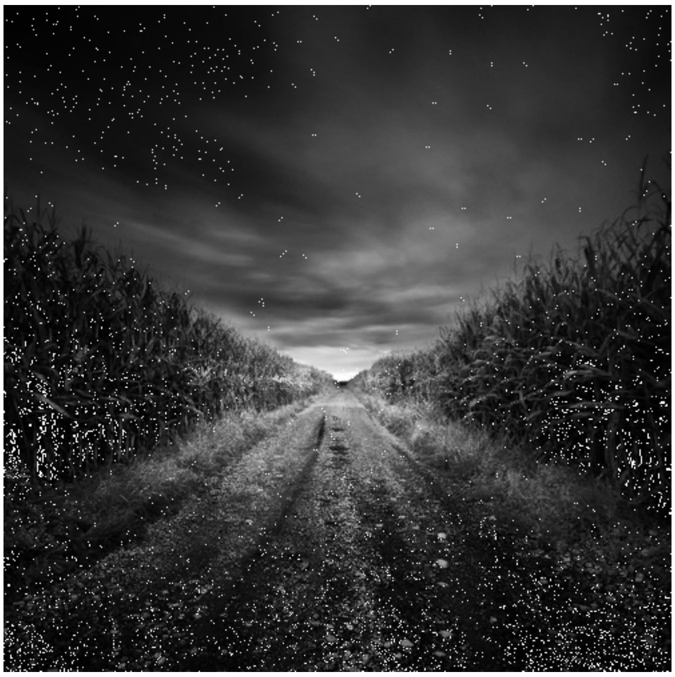


Figure 71: Restored Field image from threshold level 26

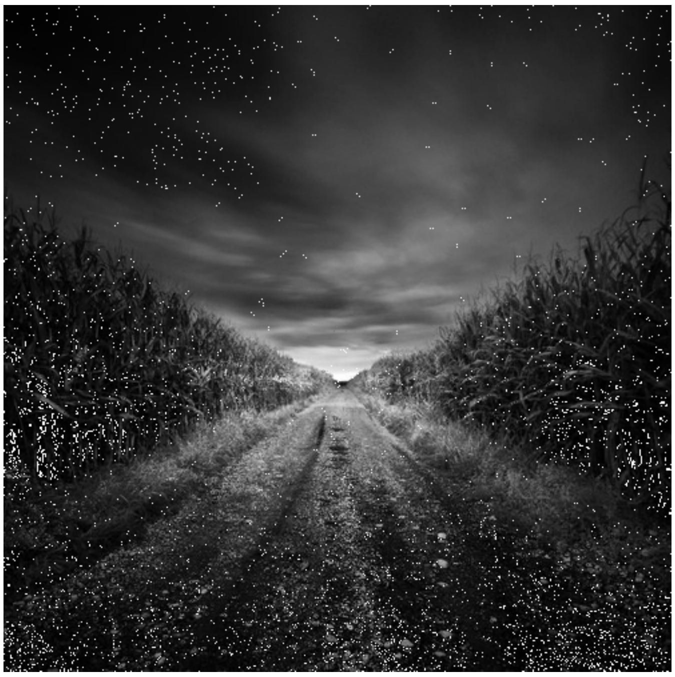


Figure 72: Restored Field image from threshold level 30

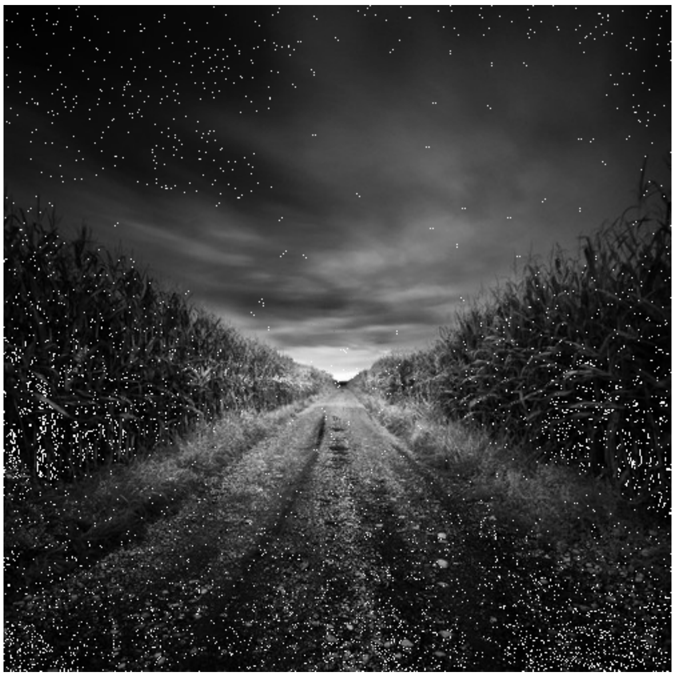


Figure 73: Restored Field image from threshold level 34

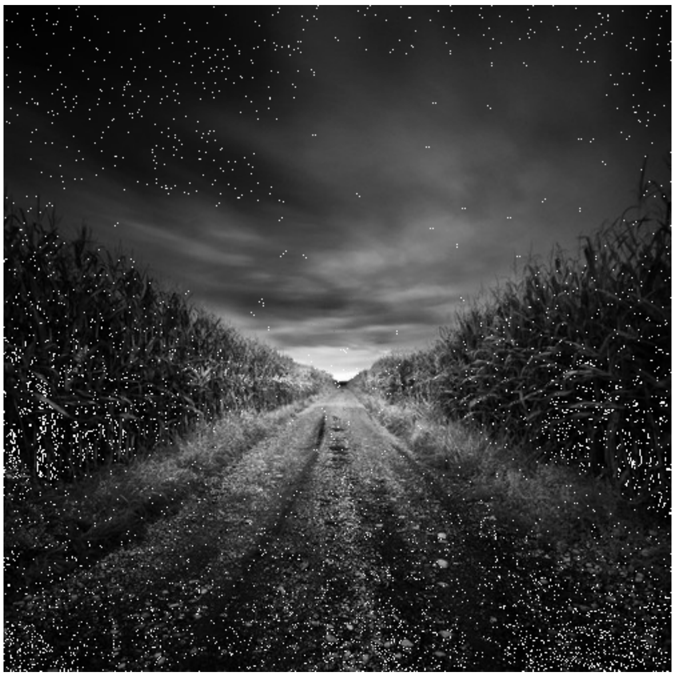


Figure 74: Restored Field image from threshold level 38

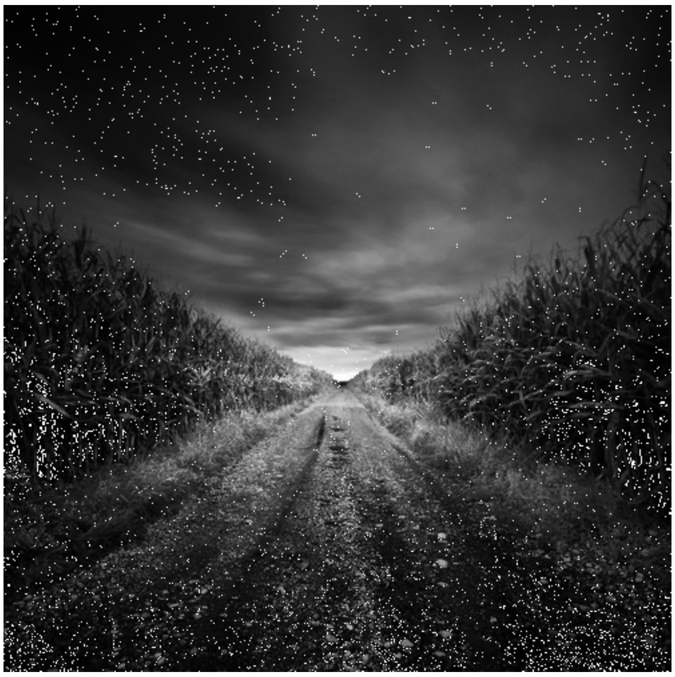


Figure 75: Restored Field image from threshold level 42

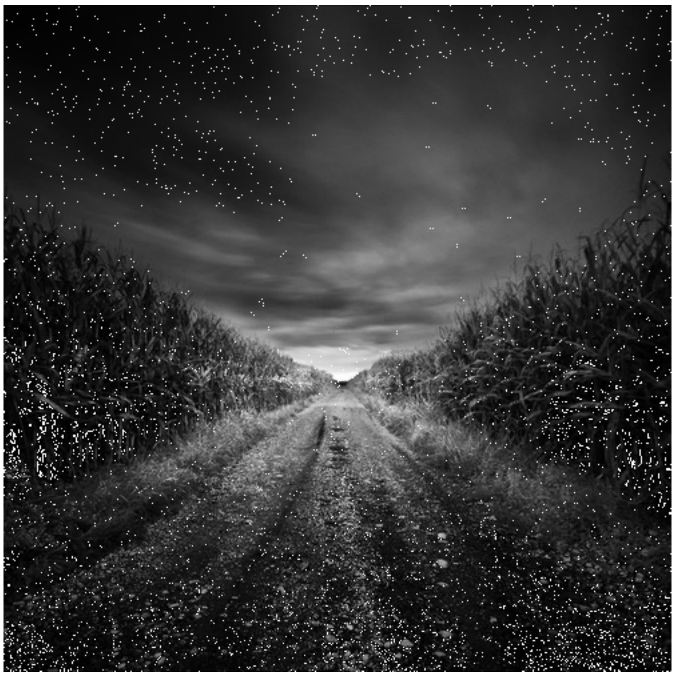


Figure 76: Restored Field image from threshold level 46

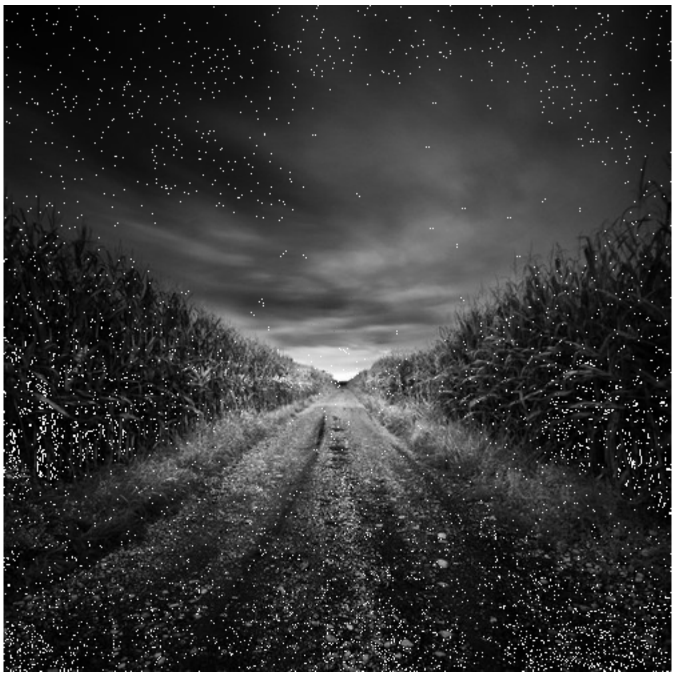


Figure 77: Restored Field image from threshold level 50

.6.6 Canyon Images from different noise levels

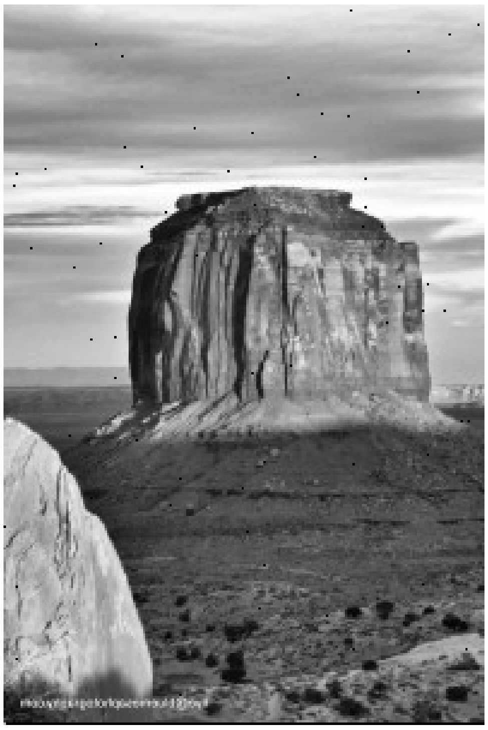


Figure 78: Canyon image with noise level 0.001

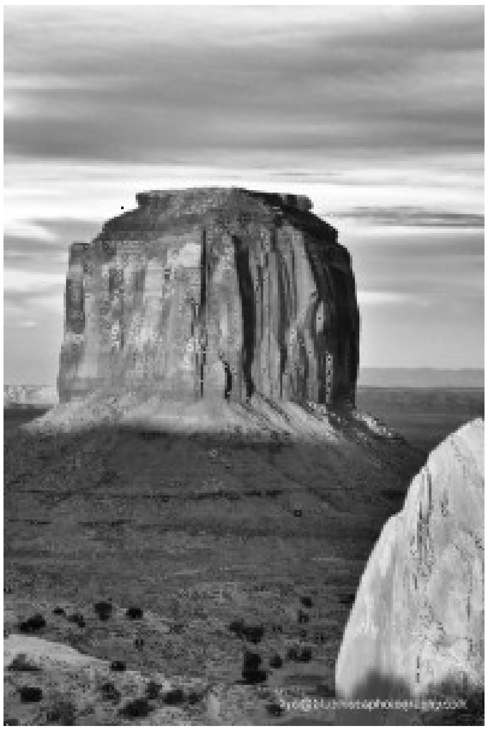


Figure 79: Restored Canyon image from noise level 0.001

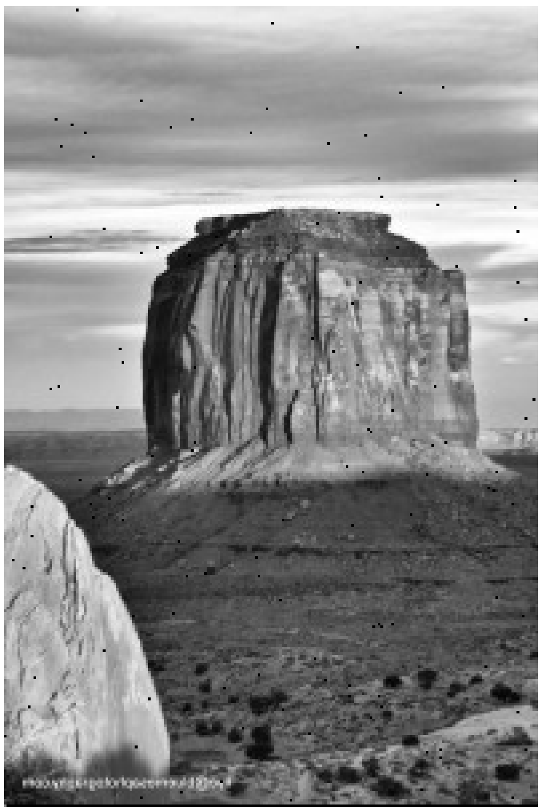


Figure 80: Canyon image with noise level 0.002

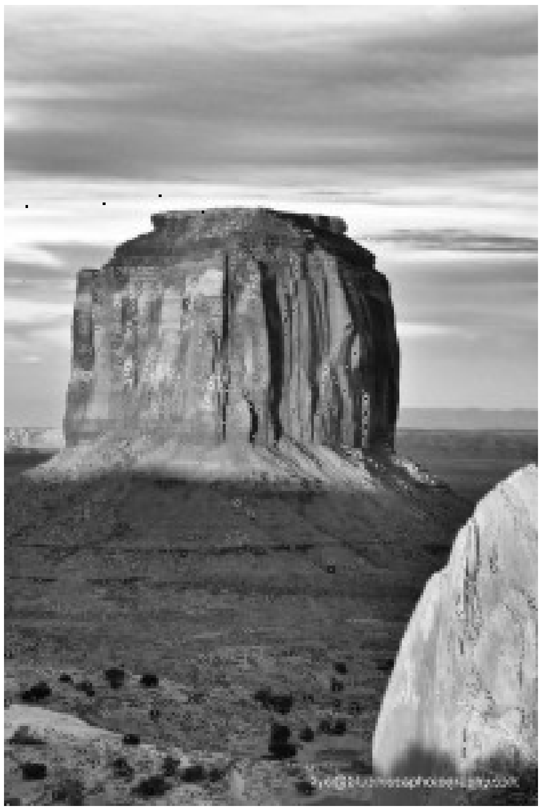


Figure 81: Restored Canyon image from noise level 0.002

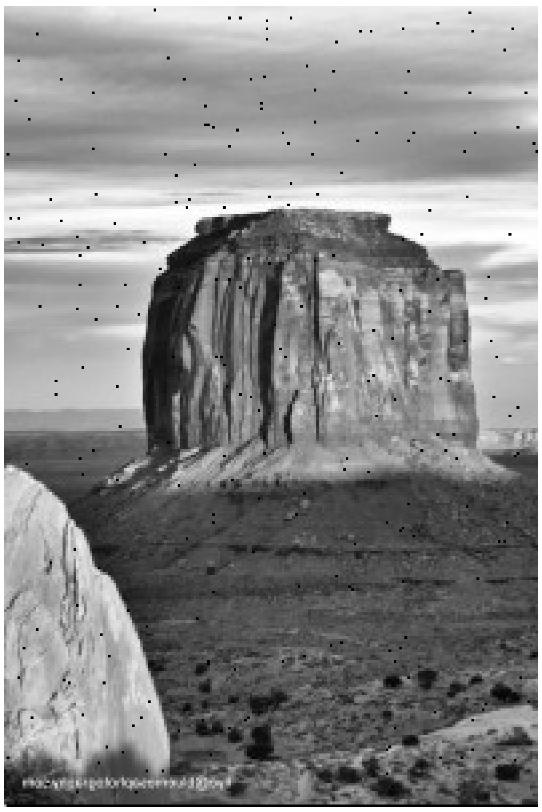


Figure 82: Canyon image with noise level 0.004

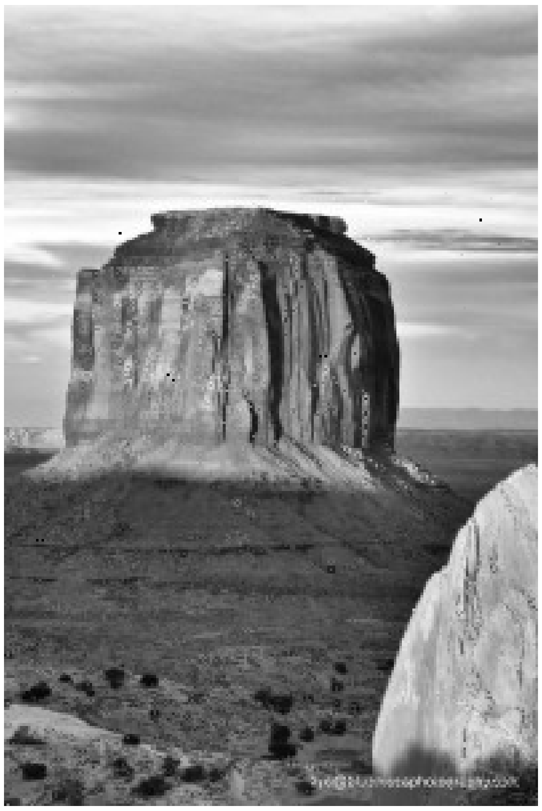


Figure 83: Restored Canyon image from noise level 0.004

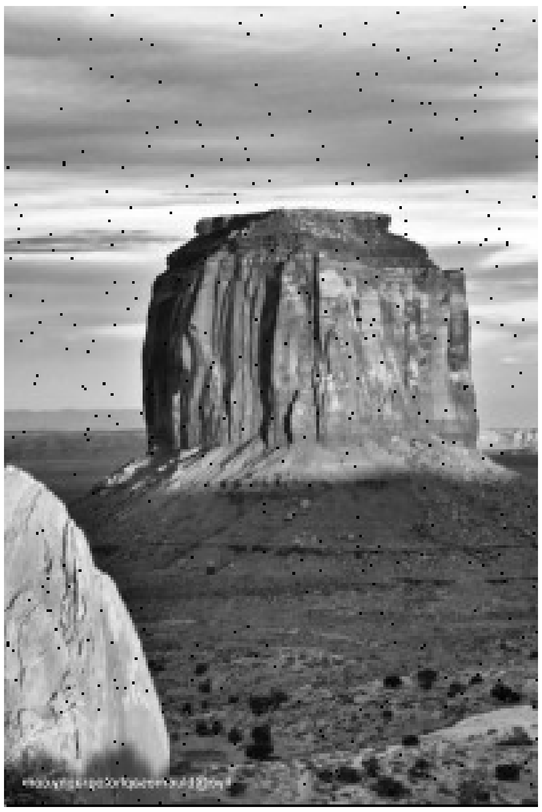


Figure 84: Canyon image with noise level 0.006

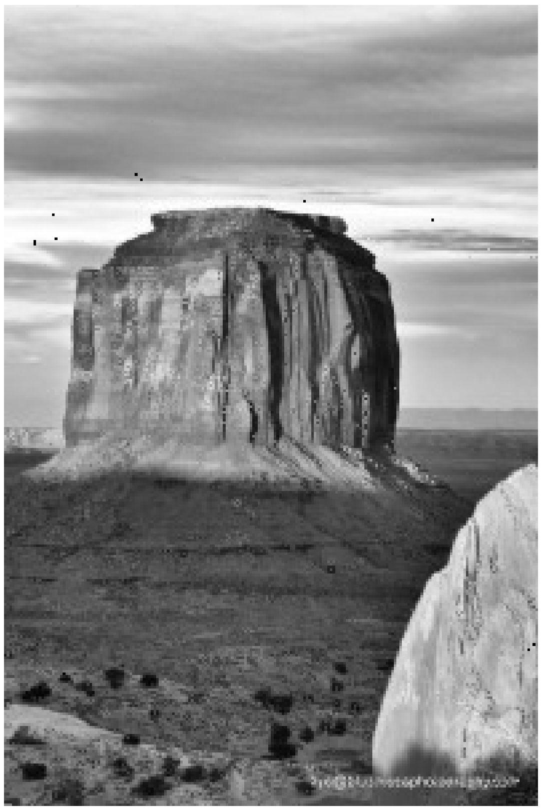


Figure 85: Restored Canyon image from noise level 0.006

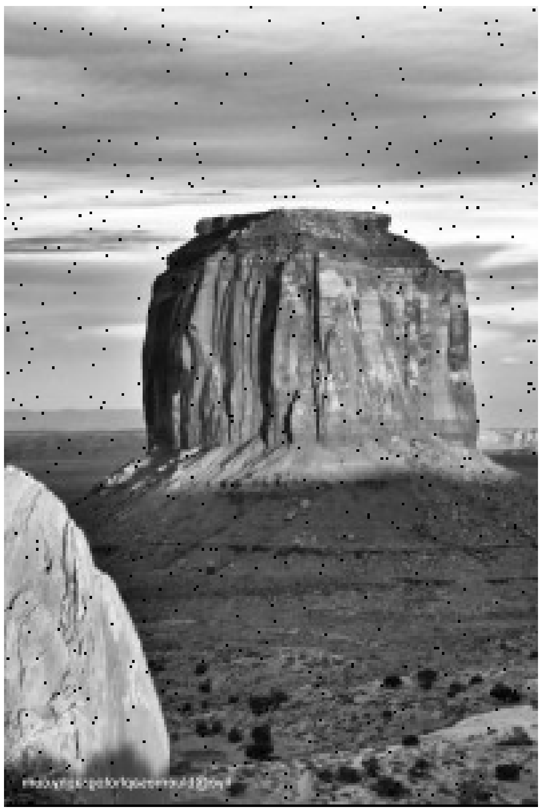


Figure 86: Canyon image with noise level 0.008

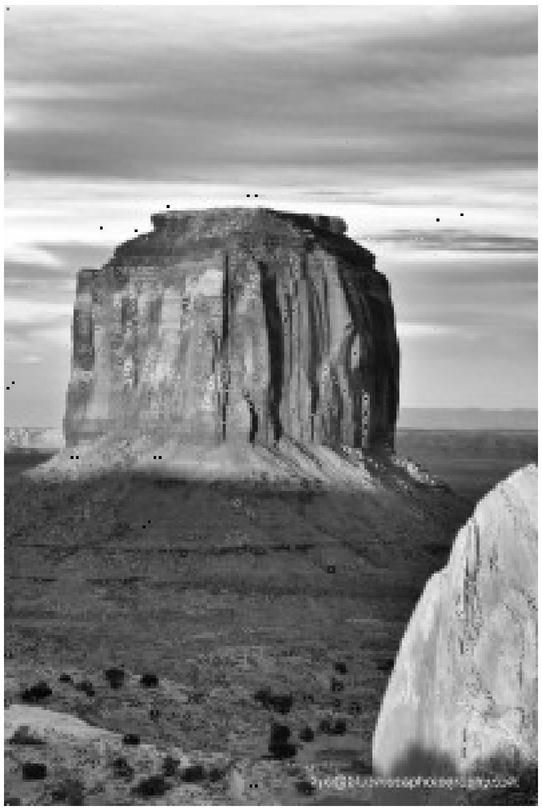


Figure 87: Restored Canyon image from noise level 0.008

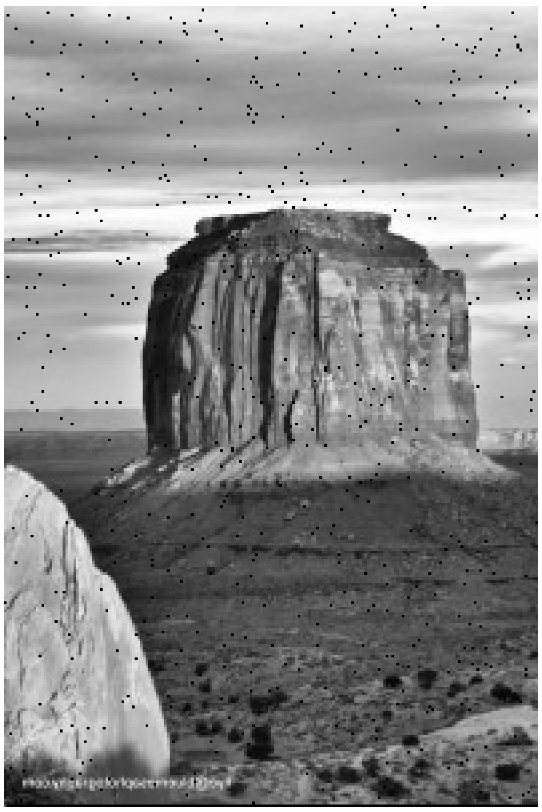


Figure 88: Canyon image with noise level 0.01

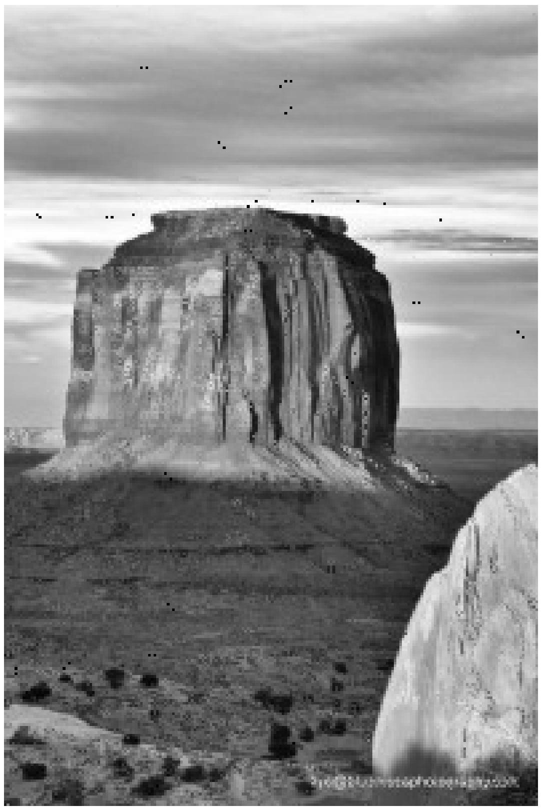


Figure 89: Restored Canyon image from noise level 0.01

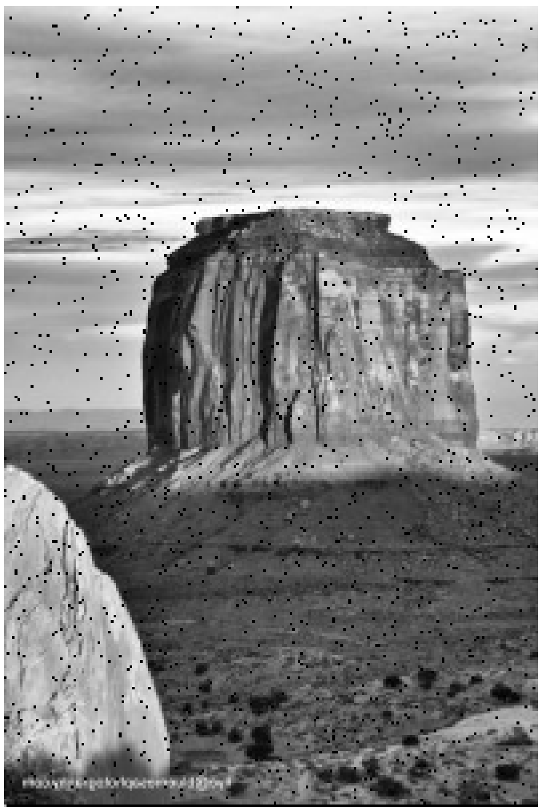


Figure 90: Canyon image with noise level 0.02

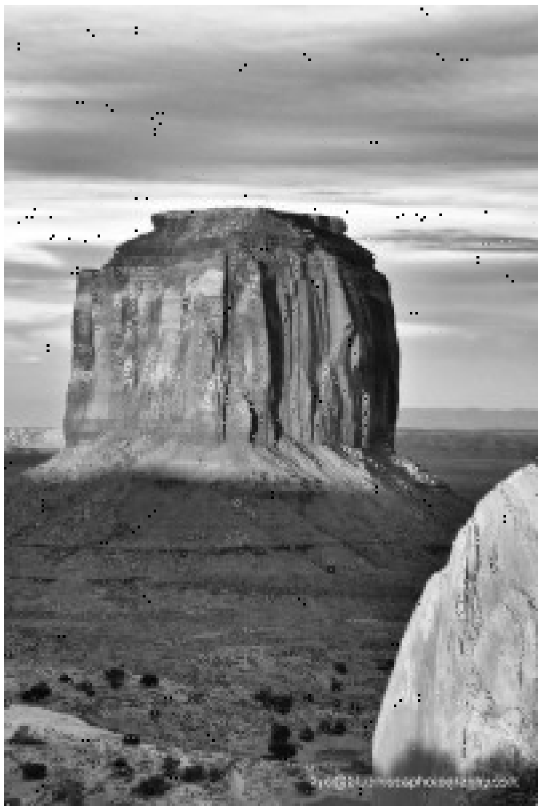


Figure 91: Restored Canyon image from noise level 0.02

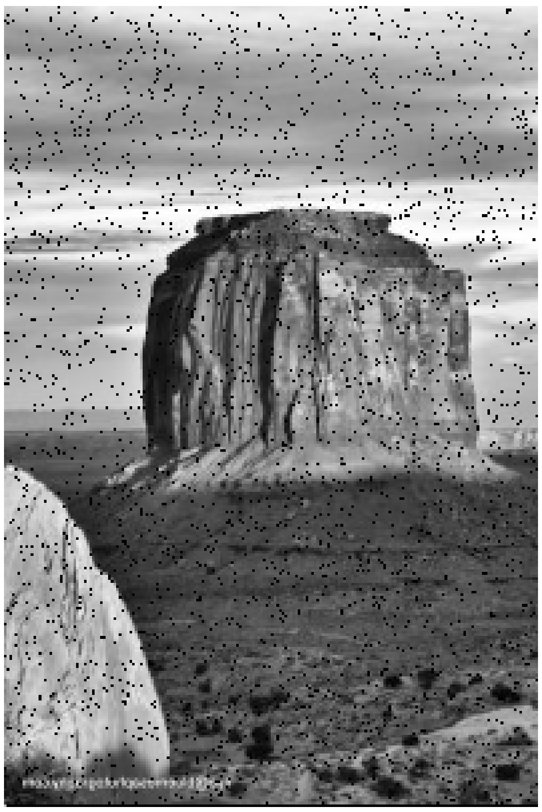


Figure 92: Canyon image with noise level 0.04

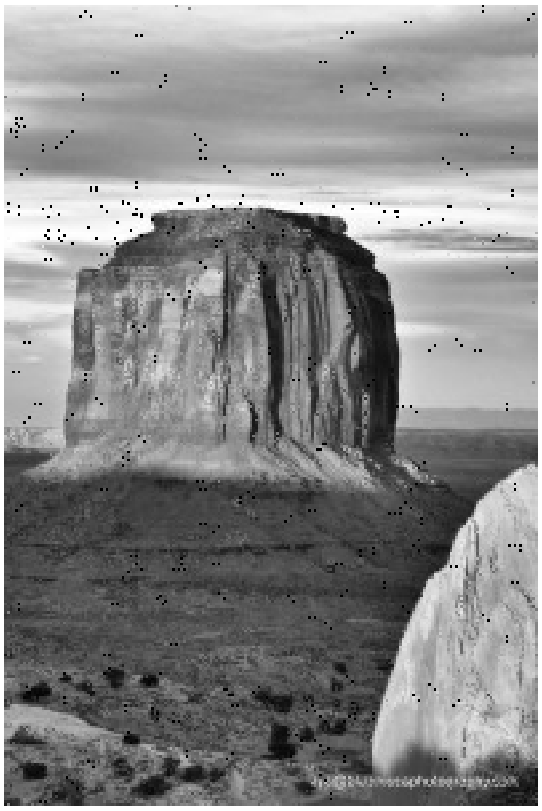


Figure 93: Restored Canyon image from noise level 0.04

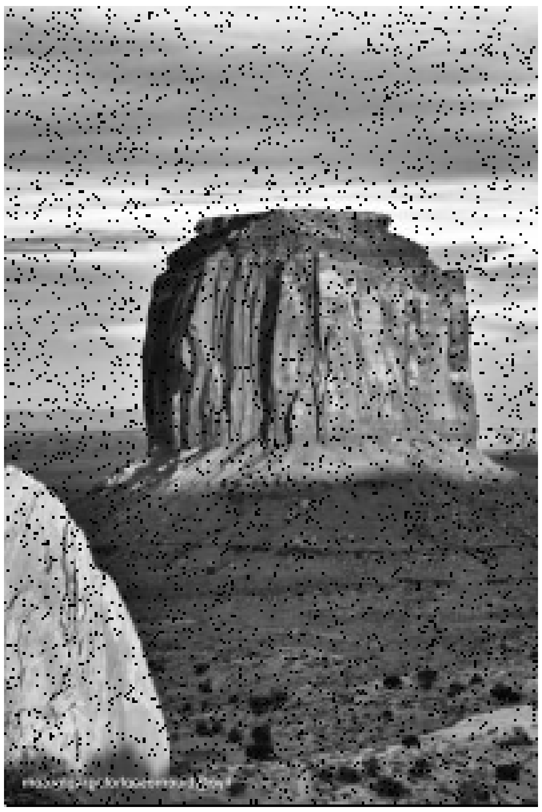


Figure 94: Canyon image with noise level 0.06

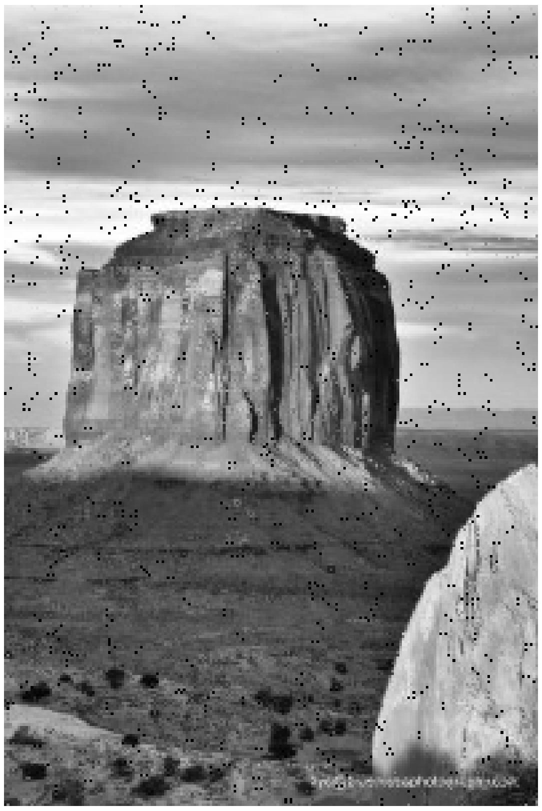


Figure 95: Restored Field image from noise level 0.06

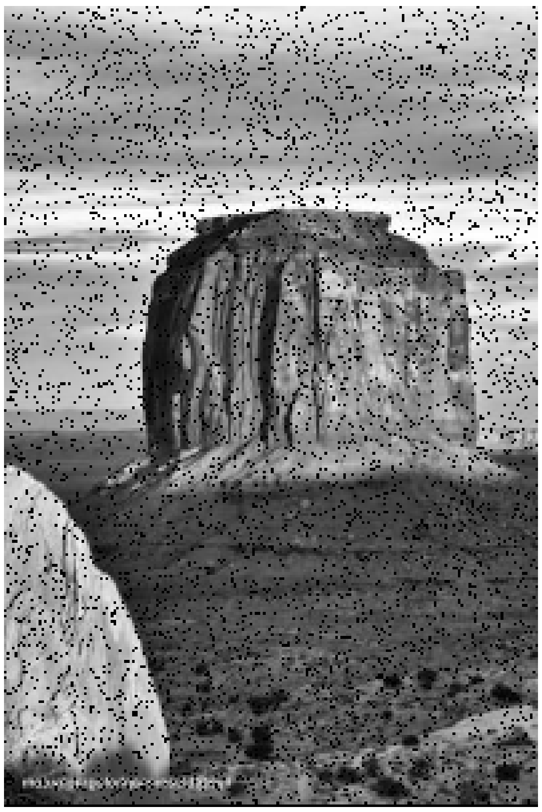


Figure 96: Canyon image with noise level 0.08

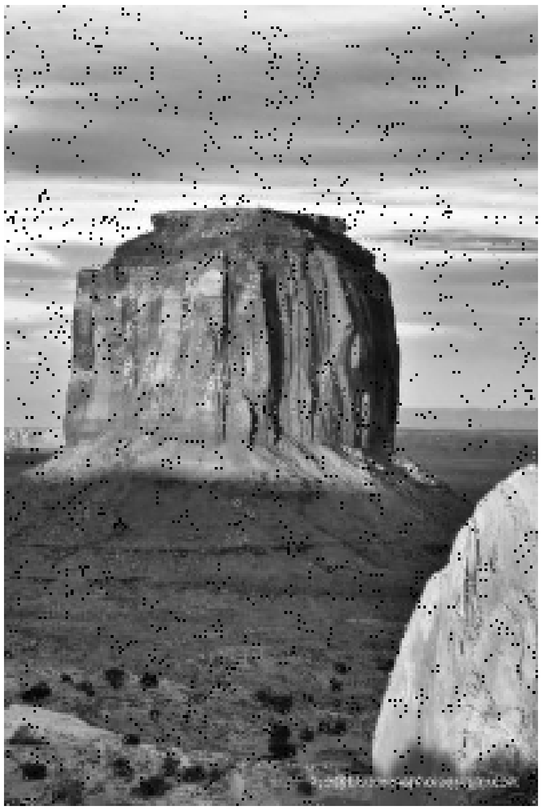


Figure 97: Restored Canyon image from noise level 0.08

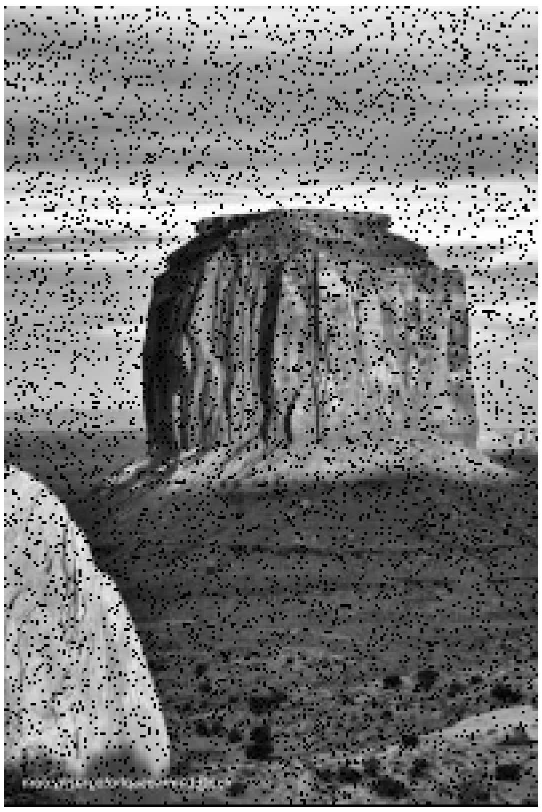


Figure 98: Canyon image with noise level 0.1

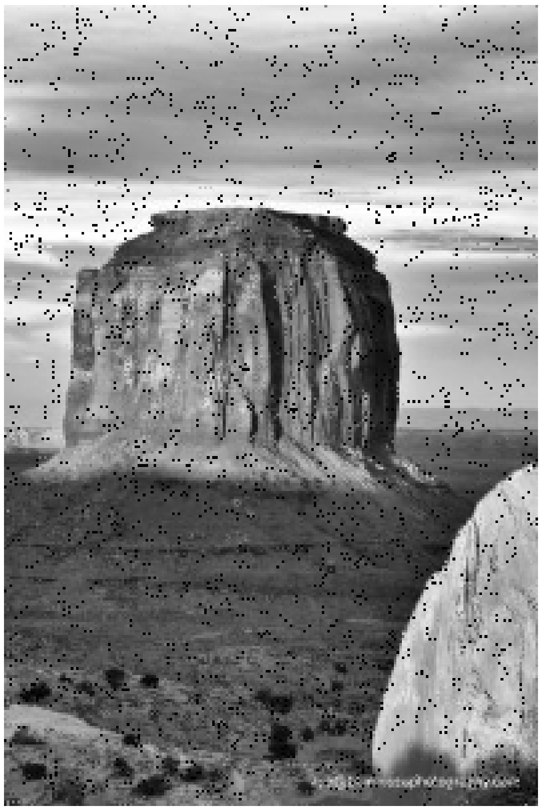


Figure 99: Restored Canyon image from noise level 0.1

.6.7 Canyon Images from different threshold levels

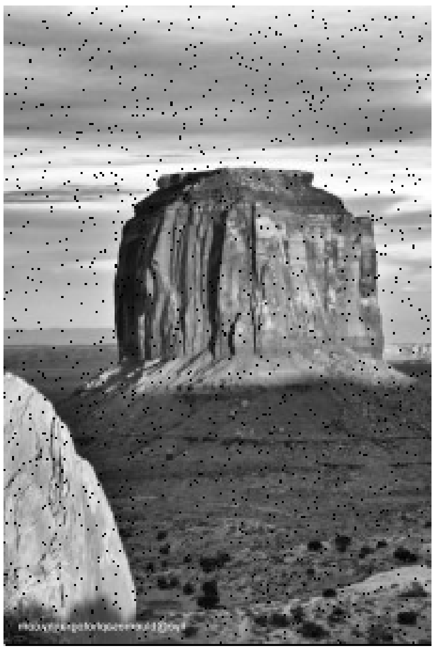


Figure 100: Canyon image with noise level 0.02

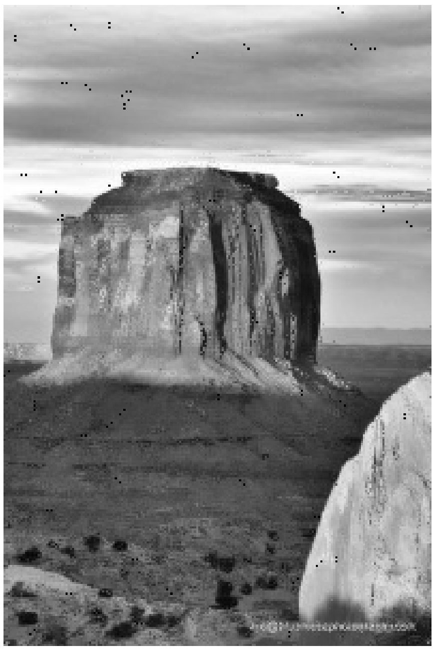


Figure 101: Restored Canyon image from threshold level 1

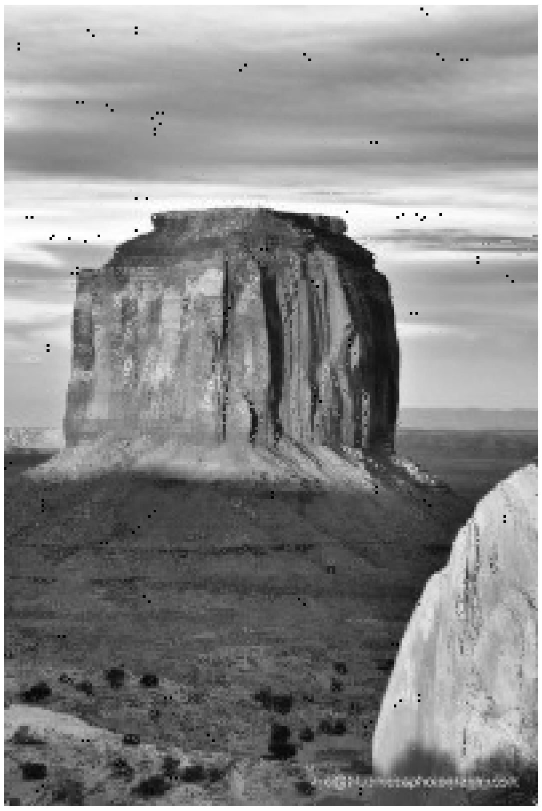


Figure 102: Restored Canyon image from threshold level 4

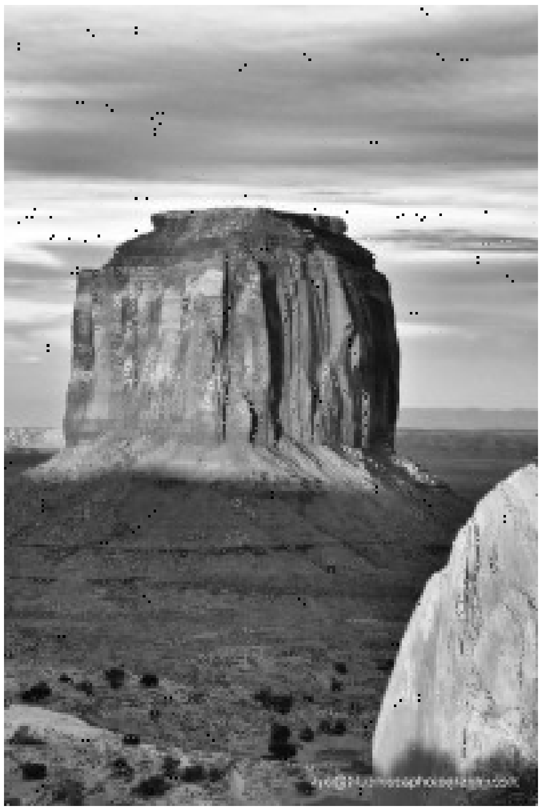


Figure 103: Restored Canyon image from threshold level 8

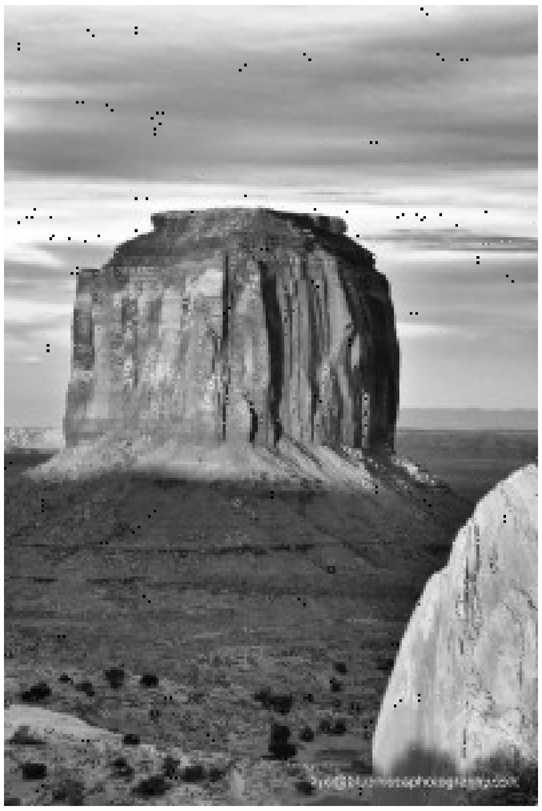


Figure 104: Restored Canyon image from threshold level 12

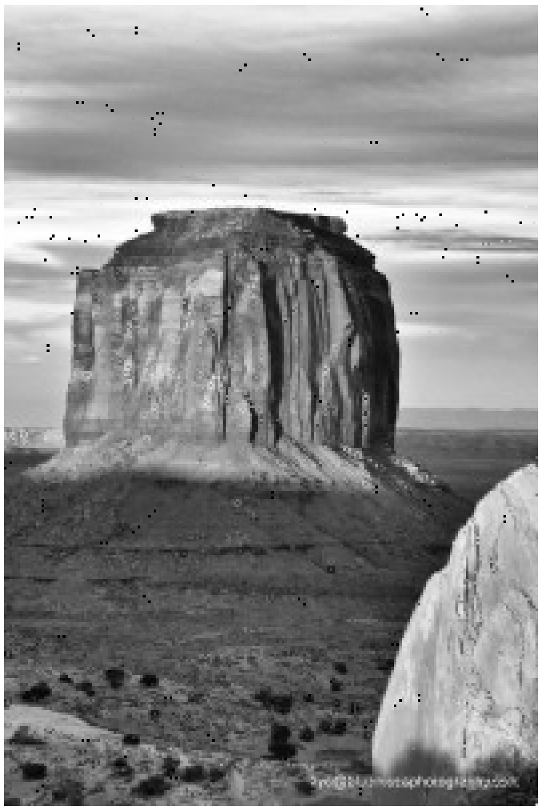


Figure 105: Restored Canyon image from threshold level 16

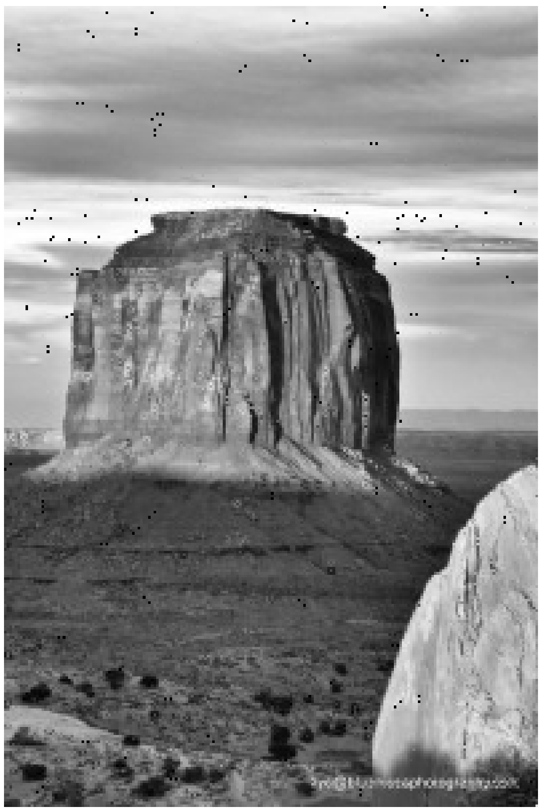


Figure 106: Restored Canyon image from threshold level 20

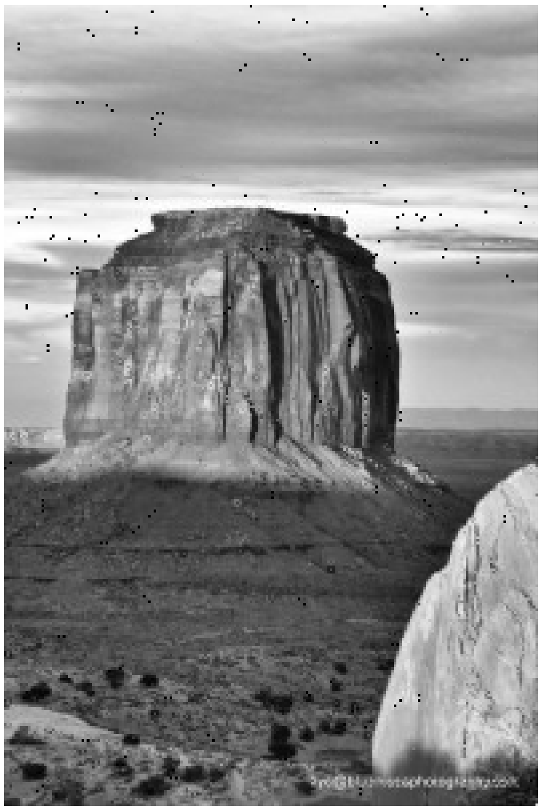


Figure 107: Restored Canyon image from threshold level 22

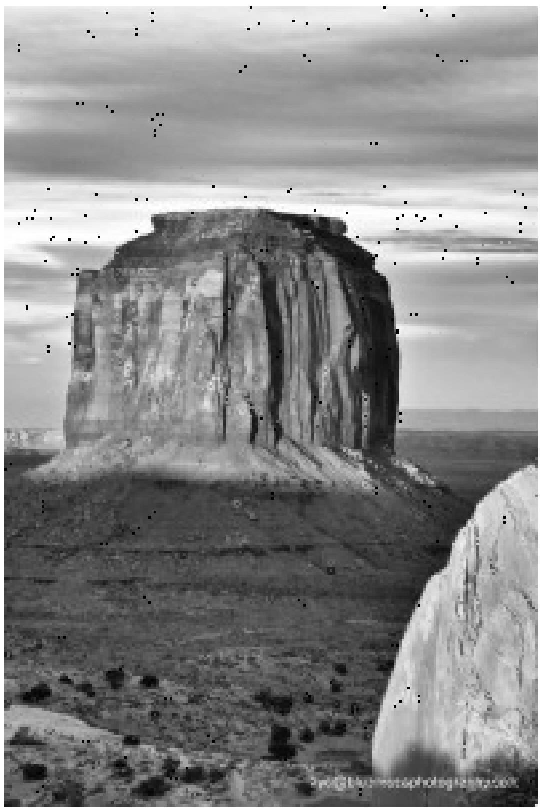


Figure 108: Restored Canyon image from threshold level 26

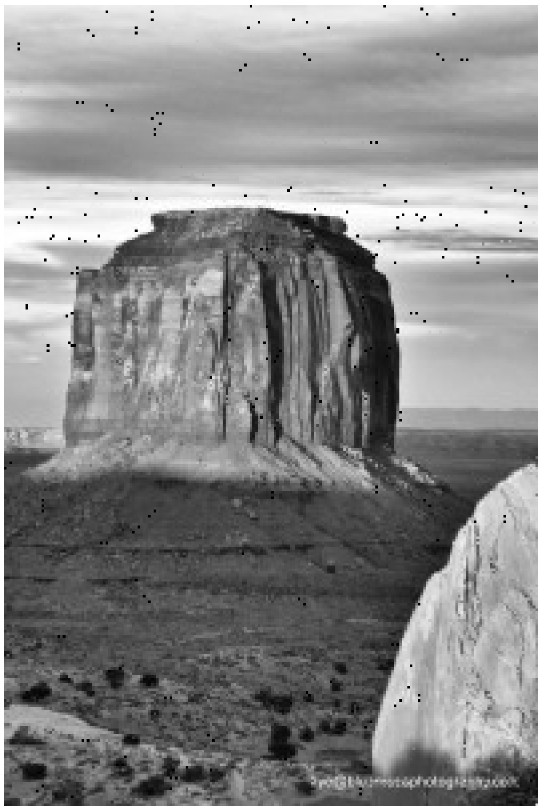


Figure 109: Restored Canyon image from threshold level 30

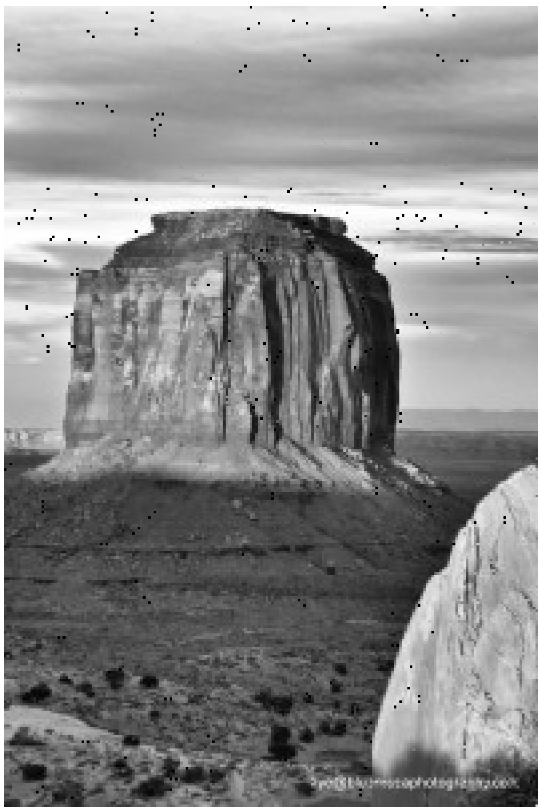


Figure 110: Restored Canyon image from threshold level 34

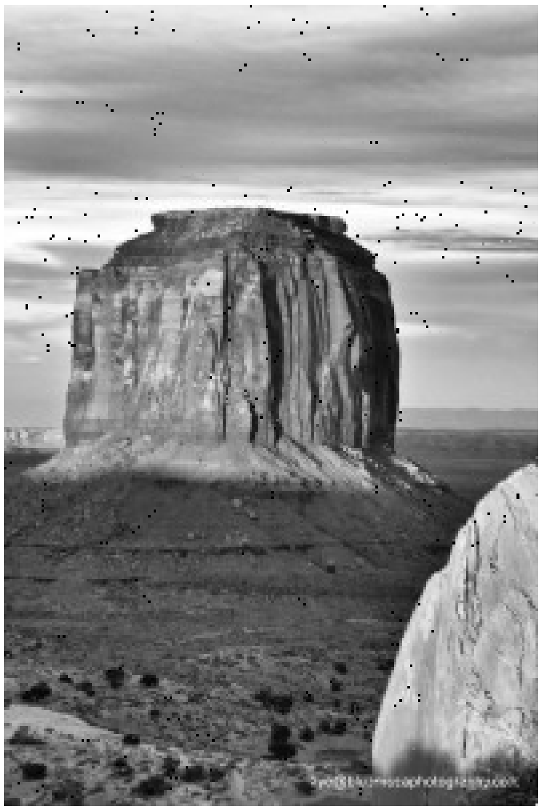


Figure 111: Restored Canyon image from threshold level 38

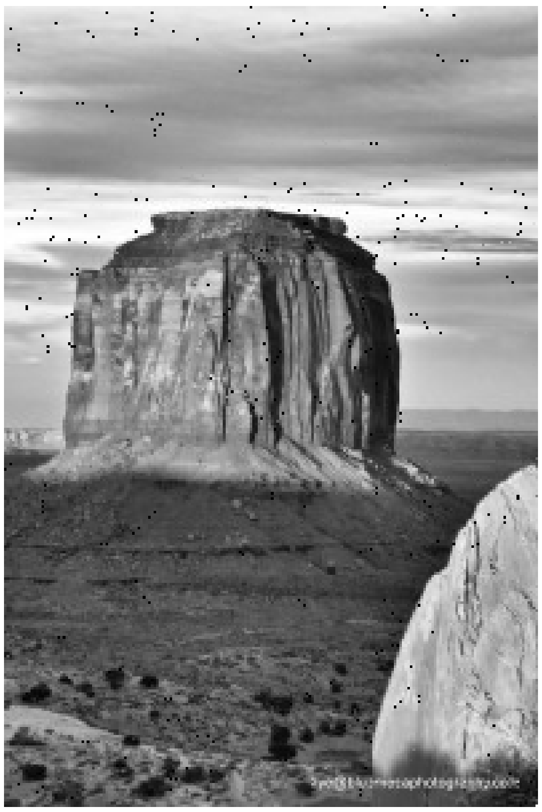


Figure 112: Restored Canyon image from threshold level 42

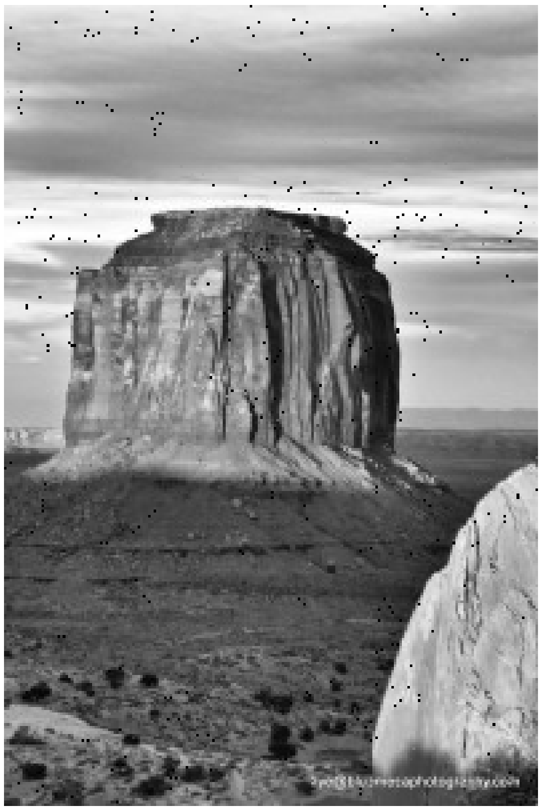


Figure 113: Restored Canyon image from threshold level 46

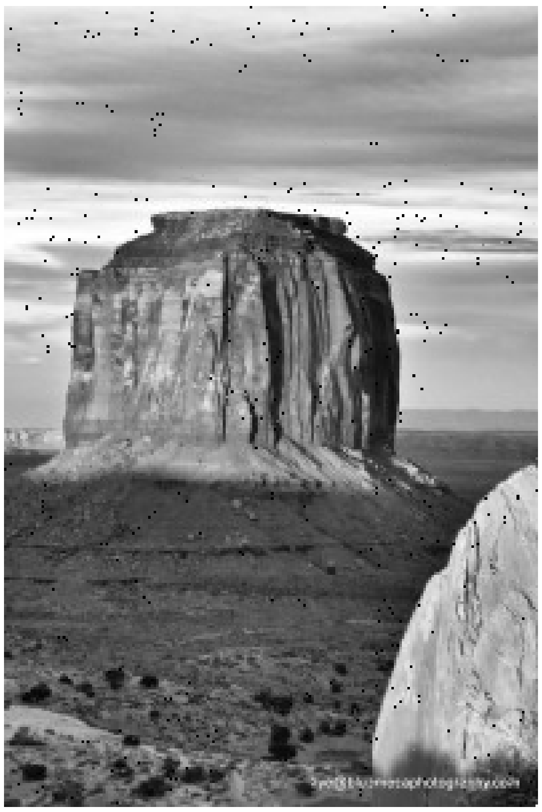


Figure 114: Restored Canyon image from threshold level 50

.6.8 Image from software simulation

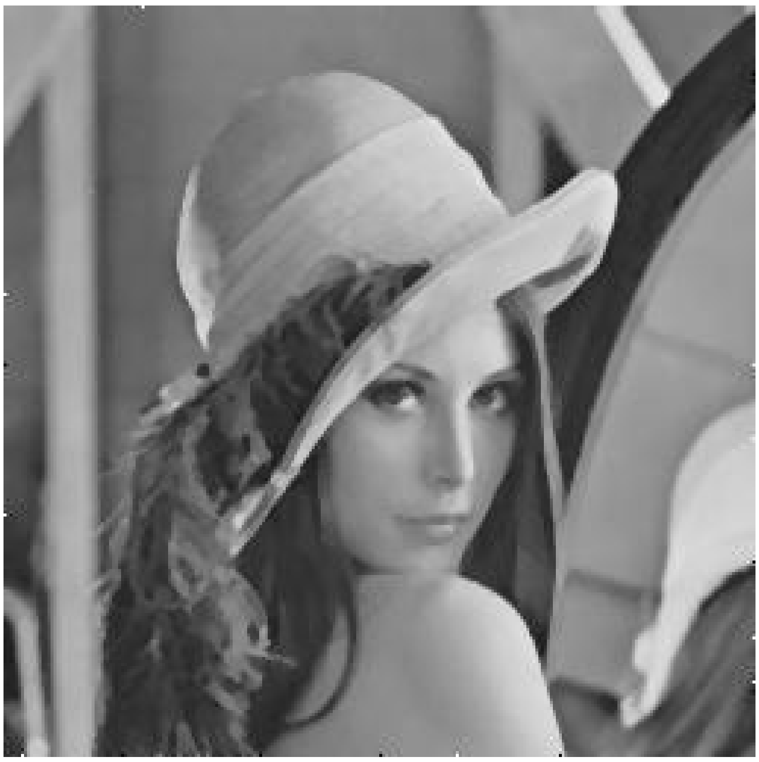


Figure 115: Restored Lena image when using the software algorithm designed in the 2009 fall project

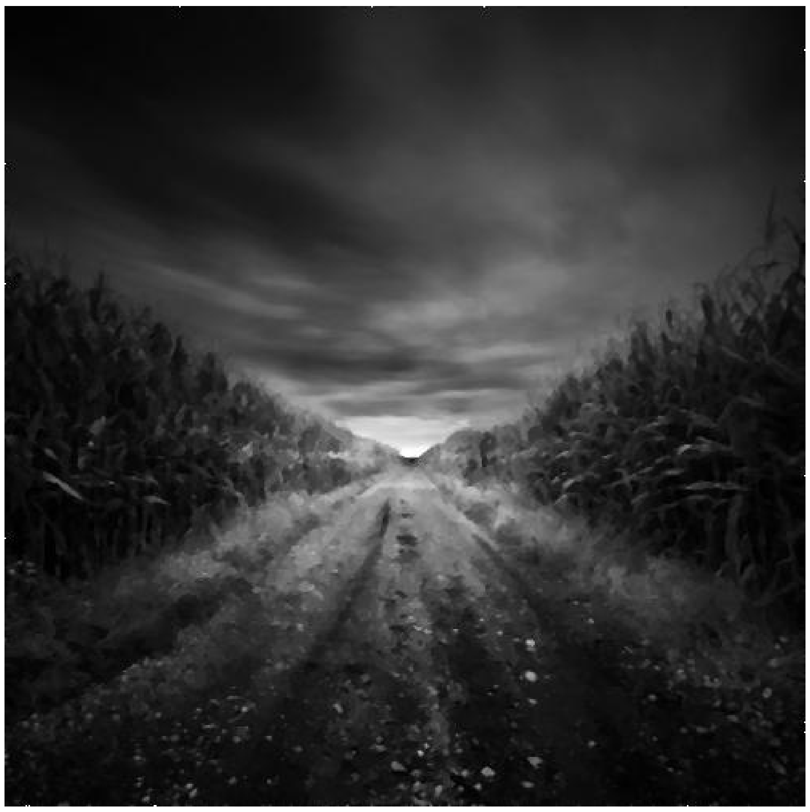


Figure 116: Restored field image when using the software algorithm designed in the 2009 fall project

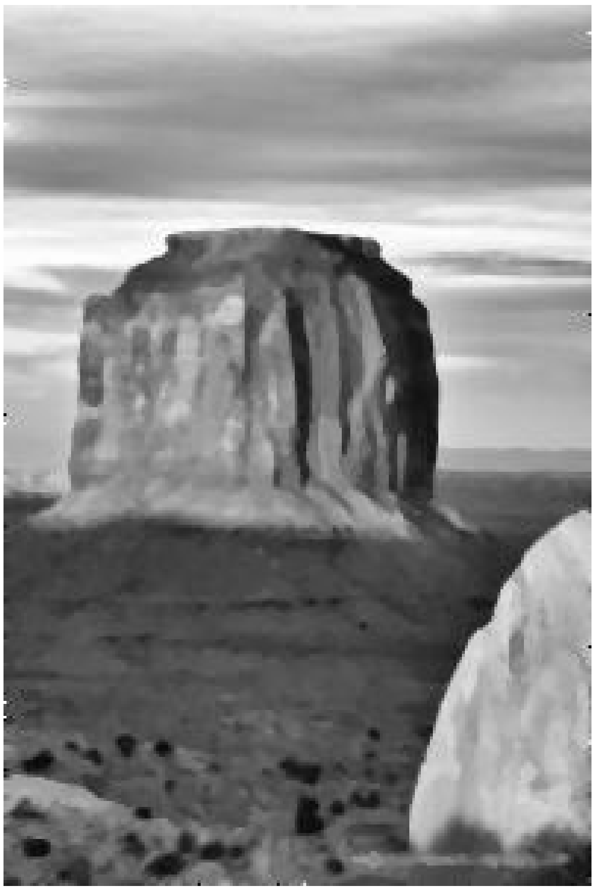


Figure 117: Restored canyon image when using the software algorithm designed in the 2009 fall project