**Technion**

*Electrical Engineering Department*

High Speed Digital System Lab

Image Manipulation Core for FPGA

**Version**:

**Last update**:

**Written By**: Ran Mizrahi , Uri Tsipin

Table of Contents

[1 Introduction 3](#_Toc307316071)

[1.1 Abstract 3](#_Toc307316072)

[1.2 Applications 3](#_Toc307316073)

[1.2.1 Pilot Helmets 3](#_Toc307316074)

[1.2.2 Intelligence Surveillance Devices 3](#_Toc307316075)

[1.2.3 Surgical Applications 4](#_Toc307316076)

[1.2.4 Printer 4](#_Toc307316077)

[2 Goals 4](#_Toc307316078)

[3 Project requirements 5](#_Toc307316079)

[4 The algorithm 5](#_Toc307316080)

[4.1 Crop Input Image 5](#_Toc307316081)

[4.2 Image Rotation 5](#_Toc307316082)

[4.2.1 Calculation of source pixel address 6](#_Toc307316083)

[4.2.2 Bilinear Interpolation 7](#_Toc307316084)

[4.3 Zoom 8](#_Toc307316085)

[4.3.1 Mathematical background 8](#_Toc307316086)

[4.3.2 Implementation 8](#_Toc307316087)

[4.4 User defined output resolution 9](#_Toc307316088)

Table of Figures

[Figure 1 3](#_Toc307316274)

[Figure 2 3](#_Toc307316275)

[Figure 3 4](#_Toc307316276)

[Figure 4 4](#_Toc307316277)

[Figure 5 – Example Of Image Cropping 5](#_Toc307316278)

[Figure 6 - Example of 30 degree rotation 6](#_Toc307316279)

[Figure 7 - A counterclockwise rotation of a vector through angle θ 6](#_Toc307316280)

[Figure 8 7](#_Toc307316281)

[Figure 9 8](#_Toc307316282)

[Figure 10 8](#_Toc307316283)

[Figure 11 8](#_Toc307316284)

# Introduction

## Abstract

Many modern digital devices excute different image manipulations. These manipulations requires image rotation and zooming.

Image Processing algorithms are "heavy consumers" of resources and therefor we would to boost the procces using hardware acceleration.

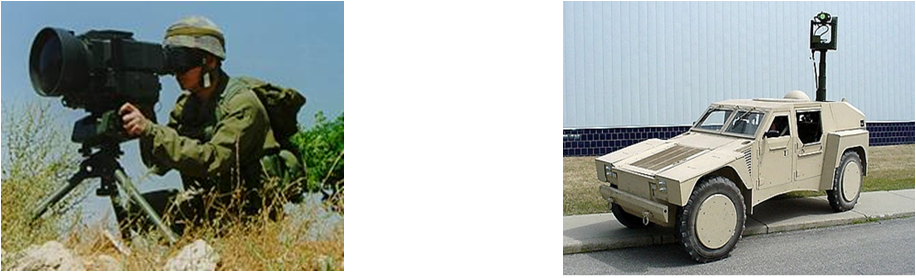
## Applications

### Pilot Helmets

Modern Day Pilot Helmets contains a digital display module. Due to the helmets geometry, being elliptic and not straight, there's a need to "deform" the displayed image.  
Part of the "deformation" algorithm requires to rotate the image and zooming in/out of the image.  


Figure

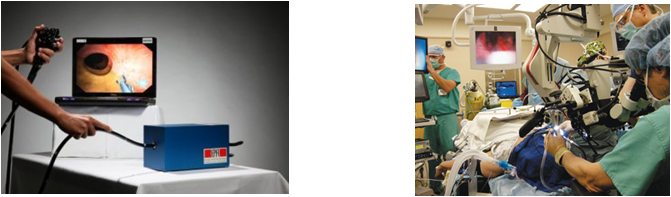
### Intelligence Surveillance Devices

Security forces use different surveillance devices, and sometimes due to terrain conditions there is a need to rotate the image in order to make make the surveillance more comfortable.  


Figure

### Surgical Applications

Tiny cameras and Optical fibers are commonly used in various medical procedures. In order to improve the image and ease the procedure image rotation and zooming is necessary.



Figure

### Printer

New age printers have a built-in LCD display, using our feature the user will have the ability to edit (crop, zoom, rotate) the image before it is sent out to print simply and fast  


Figure

# Goals

Implement a FPGA core using VHDL. The core will execute the following objectives

* Full panoramic rotation: 0 to 360 degrees.
* Support of Zoom function.
* Support of crop image by user defined coordinates.

Building a GUI (Graphical User Interface) using Matlab. The GUI will transfer data packets (including the image, required zoom/rotation parameters/crop coordinates) to the FPGA.  
Finally, the GUI will display the output image for Debug purposes.

# Project requirements

* Input Image resolution must be 600x800 and monochromatic.
* Output image resolution will be 480x640.
* Zoom factor must be an integer, power of 2 (i.e. 4,16,32…)
* The rotation angle must be an integer.
* System clock- 133 MHz.
* Vesa (display) clock- 40 MHz.

# The algorithm

להוסיף רקע הסבר על מה זה תמונה במחשב (מטריצה) וכדו'  
  
----

The Algorithm executes four operations, where "Crop Input Image"(4.1) is carried out separately before the other function which work simultaneously

## Crop Input Image

The user inputs (x\_start,y\_start) coordinates which describe the top left corner of the required cropped image, where (1,1) leaves the original picture intact.

The algorithm defines a new image which copies the original image matrix starting from  
(x\_start,y\_start) coordinates until the end of the matrix size.  
In case of colored picture, the matrix is three dimensional, the third dimension will be copied according to the same coordinates.

Figure 5 – Example Of Image Cropping

## Image Rotation

The rotation algorithm includes three main stages

* define a black picture with the required input dimensions
* Scanning the output image, pixel by pixel, and calculating the source coordinate of the pixel in original image.

Figure - Example of 30 degree rotation

* Evaluating the grey/color level of the pixel using bi-linear interpolation.

### Calculation of source pixel address

Assume the pixel coordinate in the orginal image matrix is   
For a rotation over an angle ,  in the original image is mapped onto the point in the output image.

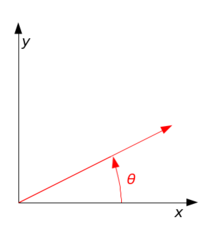


Figure - A counterclockwise rotation of a vector through angle θ

The relation between the points is:



During the algorithm we scan the output image and we evaluate the grey level of the original non-rotated image and therefore we need to use the inverse transform:



Since a matrix cell address is a positive integer we have a problem calculating the source address using this method because cosine and sine functions give real values. Hence we will round up and down the values and evaluate the grey level using bilinear interpolation.

### Bilinear Interpolation

Once we round up the address values a problem rises, we cannot restore the original pixel address in the source image, and therefore we evaluate the grey/color level using bilinear interpolation.  
Bilinear Interpolation performs a weighted average between the four sourrounding neighbours of the required "real" address according to the following method:

* R1, R2 are the weighted averages between the top pair and the bottom pair , accordingly.
* Output is the weighted average between R1 and R2

Figure

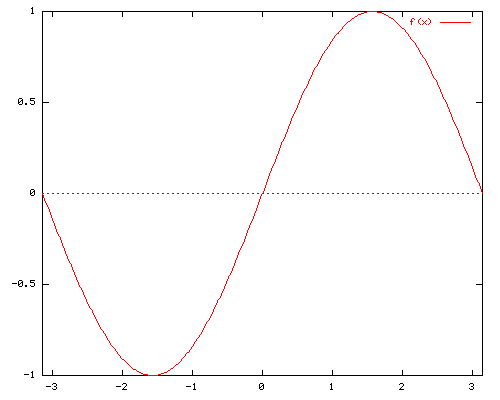
The following formula describes the bilinear interpolation (weighted average):



## Zoom

### Mathematical background

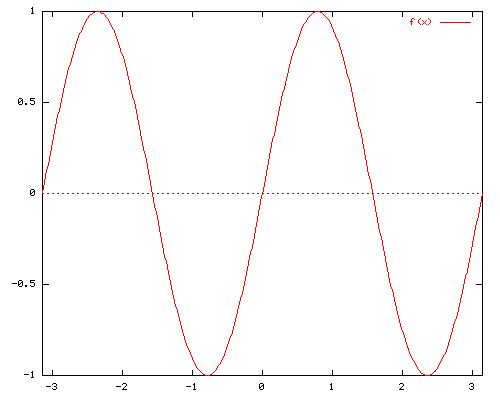
The zoom function is achieved by using the mathematical operation of scaling the axis. For example- in one dimension: 



Figure

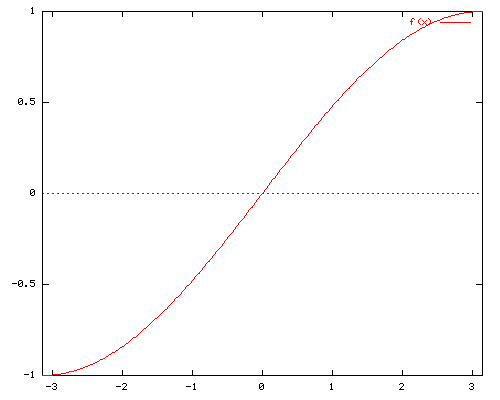
By scaling the X axis with the factor 'a', we receive the follow result:





Figure





Figure

We induct the same principle in two dimensions- (x,y) axis.

### Implementation

In order to zoom in during the scan of the source image, the algorithm multiplies the number of appearances of a source pixel in the output image, according to the zoom factor.

In order to zoom out, in the output image we "skip" every specific amount of pixels in the source image- according the zoom factor. The following formula describes both actions:



## User defined output resolution

The algorithm assumes that the output resolution is smaller or equal to the source resolution. In case the output resolution is smaller, the algorithm shrinks the source image in order to fit it in the new frame. The implementation is identical to the zoom function, only with different factors. The following formula describes the action:



And the same for Y axis.

# General description

## Data flow

**Host** transmits wrapped message of data packets, composed of bitmap image and user parameters to the **RX Path**. Message is decoded, transmitted to the **Memory Management** block, and stored into **Storage Device** (parameters into registers, image to the SDRAM). Data is read from the memory, sent to image manipulation block, and rewritten to the SDRAM. Then the manipulated image is transmitted to the **Display** through the **Display Controller**. ---Status and debug signals can be sent through the **TX Path** to the **Host**.

## Project's Top Blocks Scheme

## 

The following blocks were reused from a previous project:

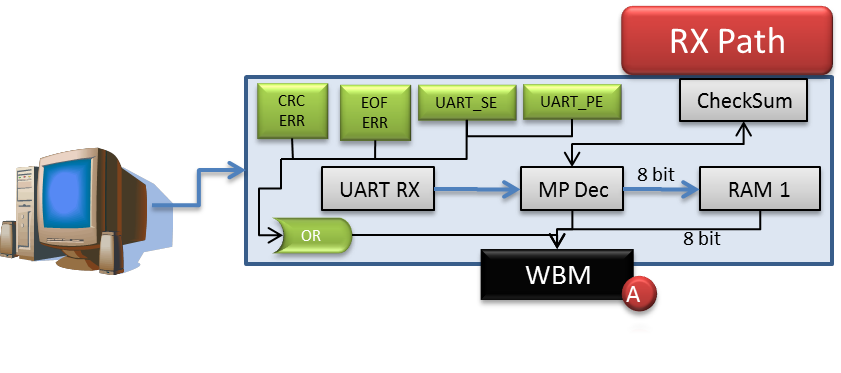
Rx Path, Tx Path, Memory Management, Wishbone Intercon, Display Controller.  
The blocks were adapted to fit our project's needs.

The previous project goal was to display a compressed image (using Run-Length coding), where our project is using a bitmap image so the following changes were made:

* Removing the Run-Length extractor
* Adapting of the Host – creating a GUI that will support our needs
* Creating a new block – Image Manipulation

## Block Descriptions

### Rx Path



This component is used for asynchronous serial data channel.

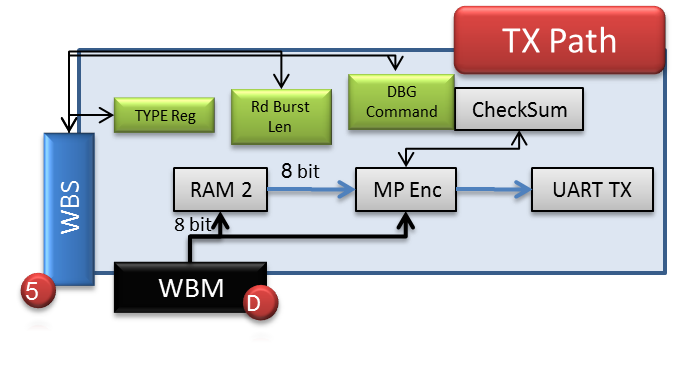
The receiver converts serial start bit, data, parity and stop bit to parallel data.

The data word length can be 5-8 bits, according to generic parameter. Parity bit can be odd or even or if decided can be inhibited, according to generic parameters.

All inputs and outputs are synchronized with the positive edge of the clock.

Any system clock and any baud rate are supported, according to generic parameter.

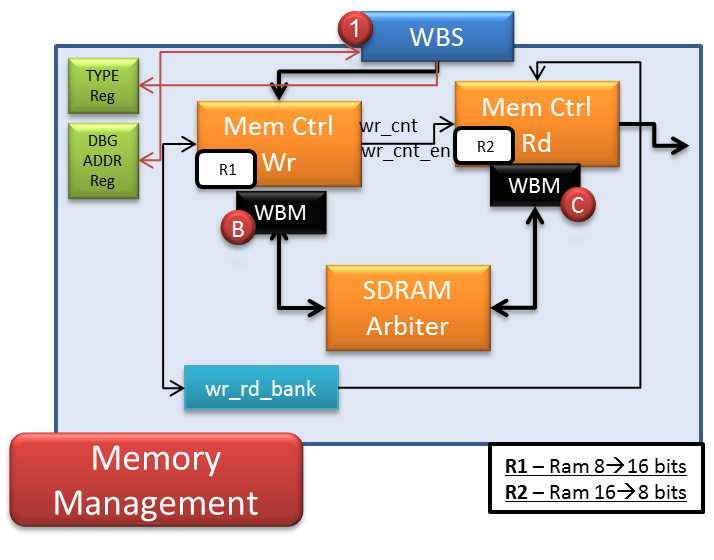
### Tx Path



The transmitter converts parallel data into serial form and automatically adds start bit, parity and stop bit.

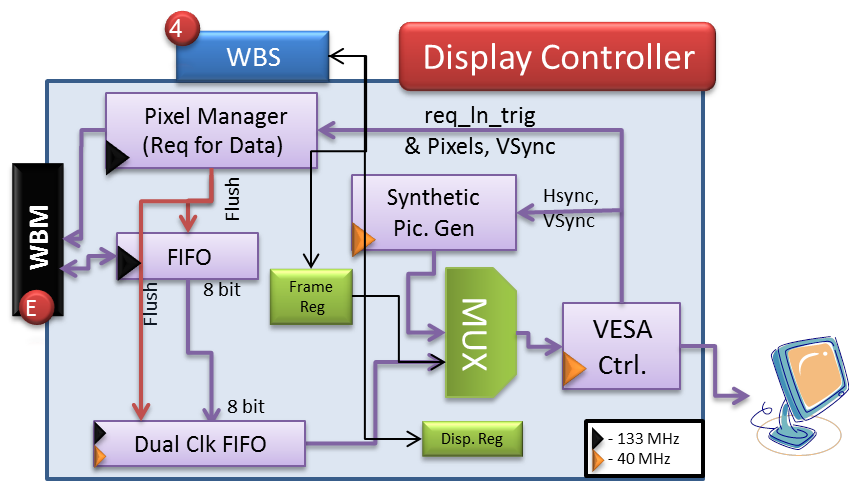
The main use of the component is for debug.

### Memory Management



The Memory Management block is in-charge of writing and reading data from the SDRAM.

### Display Controller



The Display Controller block transmits the data from the SDRAM / Synthetic Picture Generator, through the DAC on the DE2, to the VGA.

The FIFO's depth is 256X2X3 = 1536 (X8 bits), for 3 SDRAM read burst.

The Dual Clock FIFO's depth is 640X6 = 3,840 (X8 bits), for 6 lines.

### Wishbone Interconnector

complete

### Image Manipulation

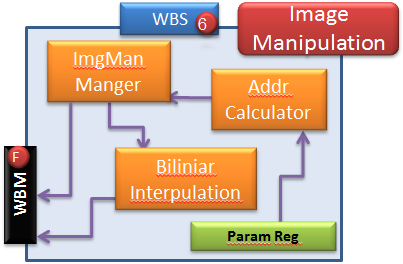


Image Manipulation block is in-charge of rotating, zooming and cropping the image. The blocks receives the data through the Wishbone Intercon, performs the required manipulation and writes it back to the SDRAM via the Wishbone Intercon.

The main components of the block are

* ImgMan Manager – controller for the image manipulation block
* Address Calculator – Calculates "matrix address" of 4 pixels that are required for the bilinear-interpolation and converts the "matrix address" into a "1D" SDRAM address.
* Bilinear Interpolator – Calculates a mean average between 4 pixels
* Parameters Register – holds the user parameters: angle, ROI indexes (Xstart, Ystart), Zoom.

## Abbreviations

* SDRAM – Synchronous Dynamic Random Access Memory
* RAM – Random Access Memory
* TX – Transmission
* RX – Receive
* FIFO – First in First out
* PLL – Phased Locked Loop
* TB – Test bench
* SOF – Start of frame
* EOF – End of frame
* CRC - Cyclic redundancy check
* MP – Message Pack
* TB – Test Bench
* UART – Universal Asynchronous Receiver Transmitter
* VESA - Video Electronics Standards Association
* VGA - Video Graphics Array
* DVI - Digital Visual Interface
* IP – Intellectual Property