A PROJECT REPORT

ON

"IMAGE PROCESSING OPERATION USING HLS"

SOFTWARE HARDWARE CO-DESIGN TO PERFORM CONTRAST ADJUSTMENT OPERATION ON ZEDBOARD USING HLS AND XILINK SDK

Submitted By:

Gogireddy Ravi Kiran Reddy **(MT2022515)**
Sanampudi Gopala Krishna Reddy **(MT2022527)**

Under the Guidance of

Dr. Nanditha Rao

Professor, IIIT-B
Bengaluru- 560100

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY, BANGALORE.

26/C, Hosur Rd,
Electronics City Phase 1, Electronic City,
Bengaluru, Karnataka 560100
Academic Year 2022-23

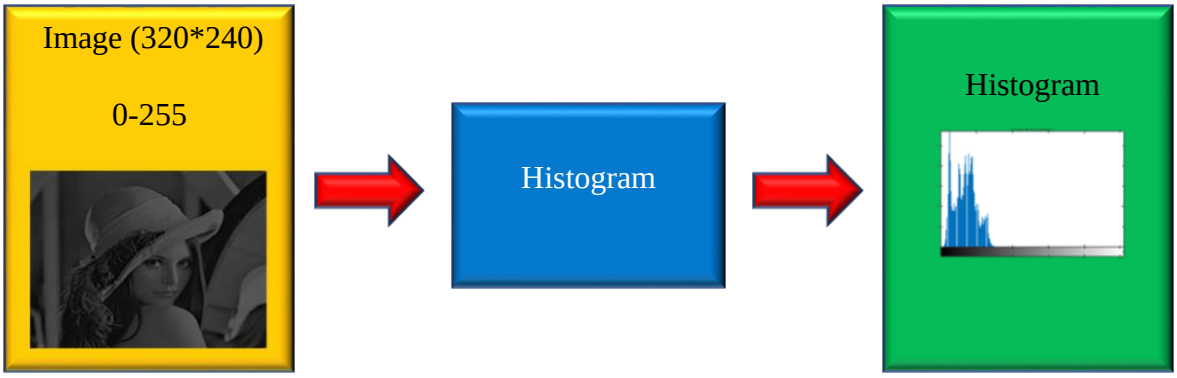
This Project is on

GitHub

https://github.com/sanampudig/IMAGE-PROCESSING-OPERATION-USING-HLS

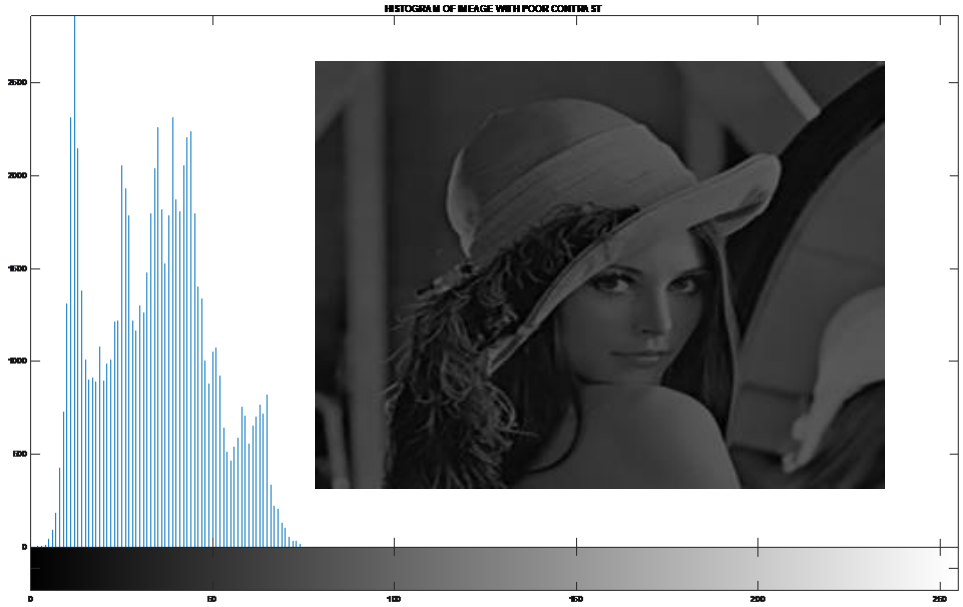
SOFTWARE HARDWARE CO-DESIGN TO PERFORM CONTRAST ADJUSTMENT OPERATION ON ZEDBOARD USING HLS AND XILINK SDK

The histogram of an image is a function that maps each grey level of an image to the number of times it occurs in the image. From Histogram we extract maximum and minimum pixel value.

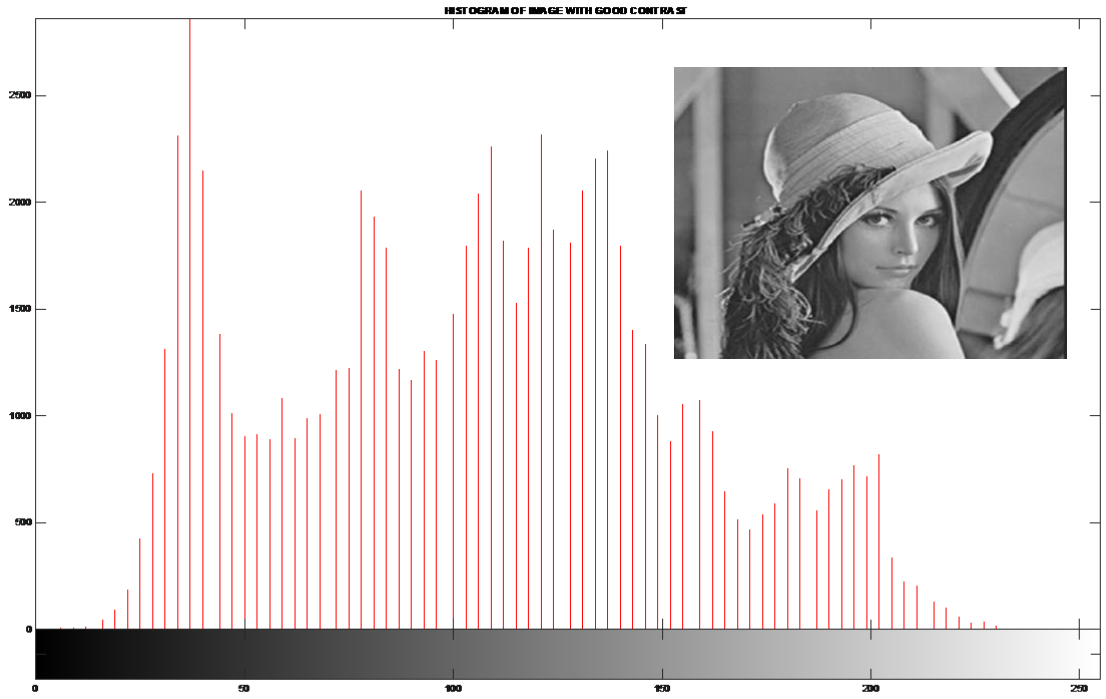


If histogram of image is not sparce or limited to small range of values then contrast of image is very poor. To such images we can perform histogram stretch to increase quality and visibility of image.

EXAMPLE of Image with poor contrast and its histogram



EXAMPLE of Image with good contrast and its histogram



Formula to calculate Histogram Stretch:

$$Y = \frac{X - Xmin}{Xmax - Xmin} * Ymax$$

Xmin/Xmax – Minimum/Maximum Value of Input Image

Ymax – Maximum Possible Value(i.e.255)

UNDERSTANDING:

INPUTS/OUTPUTS of "doHist"- function which calculates histogram of given image.

- A image converted into pixel values stored in array(size:320*240=76800) which can be sent over AXI4 Stream.
- Output is array of 256 Values which can used to plot Histogram

From this array we calculate max and minimum pixel value in the image. These values can be used to perform Histogram Stretch Operation.

INPUTS/OUTPUTS of "doHistStreach"- function which is used to perform histogram stretch and adjust the contrast of image.

- A image converted into pixel values stored in array(size:320*240=76800) which can be sent over AXI4 Stream.
- Another input is minimum and maximum values of pixels intensity which are sent over AXI4-LITE
- Output is array of processed pixels transmitted of AXI4 Stream to PS(Which can extracted from DDR Memory and converted into enhanced image)

IMPLEMENTING AN IP TO CALCULATE HISTOGRAM OF AN IMAGE USING HLS

Protocol signals in AXI

- **TLAST** indicates the boundary of a packet.
- **TKEEP** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as part of the data stream
- **TUSER** is user defined sideband information that can be transmitted alongside the data stream
- **TID** is the data stream identifier that indicates different streams of data

Code to generate IP and its functionality (in comments) is given below

Header File:

```
#include <hls_stream.h>
// Use the axi stream side-channel (TLAST, TKEEP, TUSR, TID)
#include <ap_axi_sdata.h>
typedef ap_axiu<8,2,5,6> uint_8_side_channel;
void doHist(hls::stream<uint_8_side_channel> &inStream,int histo[256]);
```

Source Code:

```
#include "core.h"
void doHist(hls::stream<uint_8_side_channel> &inStream,int histo[256]){
#pragma HLS INTERFACE axis port=inStream
#pragma HLS INTERFACE s_axilite port=return bundle=CTRL_BUS
#pragma HLS INTERFACE bram port=histo

// Assigning zeros to all vales in array so that even if we perform two histogram
operations consequently, both histograms should not merge.
    for(int idxHist = 0;idxHist < 256; idxHist++)
    {
#pragma HLS PIPELINE
        histo[idxHist] = 0;
    }


//iterate on a stream of(320*240)
//MAIN LOGIC TO CALUCLATE HISTOGRAM. IT WILL ITERATES THROUGH ALL PIXEL
VALUES AND INCREMENT COUNT CORROSPONDING THAT PARTICULAR PIXEL INTENSITY
for(int idxPixel =0; idxPixel < (320*240); idxPixel++){

        uint_8_side_channel currPixelSideChannel = inStream.read();
        //uint_8_side_channel dataOutSideChannel;

 histo[currPixelSideChannel.data]+=1;

    }
}
```

Test Bench:

```
#include <stdio.h>
#include "core.h"
#include <opencv2/core/core.hpp>
#include <hls_opencv.h>//To convert RGB Image to Grey Scale

#include  "hls_video.h"


#define INPUT_IMAGE_CORE "leenawithoutcontrast.bmp"//Input Image
#define FILE_HISTOGRAM "Histogram.txt"//
char outImage[320][240];
int histo[256];
int lutMult[256];

void saveImage(const std::string path, cv::InputArray inArr)
{
    double min;
    double max;
    cv::minMaxIdx(inArr, &min, &max);
    cv::Mat adjMap;
    cv::convertScaleAbs(inArr, adjMap, 255 / max);
    cv::imwrite(path,adjMap);
}
void saveHistogram(const char* filename, int *histPointer){


    FILE *pFile;
    pFile = fopen(filename,"w");
    if(pFile != NULL)
    {
        for (int idx=0; idx<256; idx++)
        {
            fprintf(pFile,"Bin[%d]=%d\n",idx, histPointer[idx]);
        }
    }
}

int main()
{

    printf("Load image %s\n",INPUT_IMAGE_CORE);
    cv::Mat imageSrc;
    imageSrc = cv::imread(INPUT_IMAGE_CORE);

    cv::cvtColor(imageSrc, imageSrc, CV_BGR2GRAY);
//  hls::CvtColor<HLS_BGR2GRAY>(imageSrc, imageSrc);

    printf("Image Rows:%d Cols:%d\n",imageSrc.rows,imageSrc.cols);
hls::stream<uint_8_side_channel>inputStream;
for(int idxRows=0; idxRows < imageSrc.rows;idxRows++){
    for(int idxCols=0;idxCols < imageSrc.cols; idxCols++){
        uint_8_side_channel valIn;
```

```
        valIn.data = imageSrc.at<unsigned char>(idxRows,idxCols);
        valIn.keep = 1; valIn.strb = 1; valIn.user =1; valIn.id = 0; valIn.dest = 0;
        inputStream << valIn;
    }
}
doHist(inputStream, histo);


saveHistogram(FILE_HISTOGRAM,histo);


return 0;
}
```

OUTPUT File:

```
Bin[0]=2
Bin[1]=7
Bin[2]=11
Bin[3]=17
Bin[4]=47
Bin[5]=71
Bin[6]=99
Bin[7]=160
Bin[8]=230
Bin[9]=297
Bin[10]=371
Bin[11]=525
Bin[12]=676
Bin[13]=1068
Bin[14]=1462
Bin[15]=1707
Bin[16]=1601
Bin[17]=1311
Bin[18]=941
Bin[19]=764
Bin[20]=650
Bin[21]=621
Bin[22]=540
Bin[23]=543
Bin[24]=525
Bin[25]=585
Bin[26]=638
Bin[27]=634
Bin[28]=675
Bin[29]=605
Bin[30]=586
```

Performance Estimate and Resource Estimate for Calculation of Histogram on Hardware

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	5.806	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
230659	230659	230659	230659	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	123
FIFO	-	-	-	-
Instance	0	-	36	40
Memory	-	-	-	-
Multiplexer	-	-	-	149
Register	-	-	111	-
Total	0	0	147	312
Available	280	220	106400	53200
Utilization (%)	0	0	~0	~0

IMPLEMENTING AN IP TO PERFORM HISTOGRAM STRETCH TO IMPROVE CONTRAST OF IMAGE USING HLS

Header File:

```
#include <hls_stream.h>
#include <ap_axi_sdata.h>
typedef ap_axiu<8,2,5,6> uint_8_side_channel;
void doHistStrech(hls::stream<uint_8_side_channel>&inStream,
hls::stream<uint_8_side_channel> &outStream,unsigned char xMin, unsigned char xMax);
```

Source Code:

```
#include "core.h"
void doHistStrech(hls::stream<uint_8_side_channel>&inStream,
hls::stream<uint_8_side_channel> &outStream,unsigned char xMin, unsigned char xMax)
{
#pragma HLS INTERFACE axis port =inStream
#pragma HLS INTERFACE axis port =outStream
#pragma HLS INTERFACE s_axilite port =return bundle=CRTL_BUS
#pragma HLS INTERFACE s_axilite port =xMin bundle=CRTL_BUS
#pragma HLS INTERFACE s_axilite port =xMax bundle=CRTL_BUS
       float xMax_minus_xMin = xMax-xMin;
       for (int idxPixel = 0; idxPixel <(320*240); idxPixel++ ){
#pragma HLS PIPELINE

              uint_8_side_channel currPixelSideChannel =inStream.read();
              uint_8_side_channel dataOutSideChannel;
              unsigned char x_t = currPixelSideChannel.data;
              float y_t_float = ((x_t - xMin) / (xMax_minus_xMin)) * 255;
       unsigned char y_t = y_t_float;

       dataOutSideChannel.data = y_t;
       dataOutSideChannel.keep = currPixelSideChannel.keep ;
       dataOutSideChannel.strb = currPixelSideChannel.strb;
       dataOutSideChannel.user = currPixelSideChannel.user;
       dataOutSideChannel.last = currPixelSideChannel.last;
       dataOutSideChannel.id = currPixelSideChannel.id;
       dataOutSideChannel.dest = currPixelSideChannel.dest;
       outStream.write(dataOutSideChannel);
       }
       }
```

Test Bench:

```
#include <stdio.h>
#include "core.h"
#include <opencv2/core/core.hpp>
#include <hls_opencv.h>
#define INPUT_IMAGE_CORE "Lennawithoutstr.bmp"
#define OUTPUT_IMAGE_CORE "LennaStr.bmp"
char outImage[320][240];
void saveImage(const std::string path, cv::InputArray inArr){
    double min;
    double max;
    cv::minMaxIdx(inArr, &min, &max);
    cv::Mat adjMap;
    cv::convertScaleAbs(inArr, adjMap, 255/ max);
    cv::imwrite(path,adjMap);
}
int main(){
printf("Load image %s\n",INPUT_IMAGE_CORE);
cv::Mat imageSrc;
imageSrc = cv::imread(INPUT_IMAGE_CORE);
cv::cvtColor(imageSrc, imageSrc, CV_BGR2GRAY)
printf("Image Rows:%d Cols:%d\n",imageSrc.rows,imageSrc.cols);
hls::stream<uint_8_side_channel>inputStream;
hls::stream<uint_8_side_channel>outputStream;
cv::Mat   imgCvOut(cv::Size(imageSrc.cols,   imageSrc.rows),   CV_8UC1,   outImage,
cv::Mat::AUTO_STEP);
for(int idxRows=0; idxRows < imageSrc.rows;idxRows++){
    for(int idxCols=0;idxCols < imageSrc.cols; idxCols++){
        uint_8_side_channel valIn;
        valIn.data = imageSrc.at<unsigned char>(idxRows,idxCols);
       valIn.keep = 1; valIn.strb = 1; valIn.user =1; valIn.id = 0; valIn.dest = 0;
        inputStream << valIn;
    }
}
doHistStrech(inputStream, outputStream,40,111);
for(int idxRows=0; idxRows < imageSrc.rows;idxRows++){
    for(int idxCols=0;idxCols < imageSrc.cols; idxCols++){
        uint_8_side_channel valOut;
        outputStream.read(valOut);
        outImage[idxRows][idxCols] = valOut.data;
    }
}
saveImage(std::string(OUTPUT_IMAGE_CORE),imgCvOut);
return 0;
}
```

Performance Estimate and Resource Estimate to perform HISTOGRAM STRETCH on Hardware

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.328	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
76834	76834	76834	76834	none

Detail

Instance

Loop

Utilization Estimates

Summary

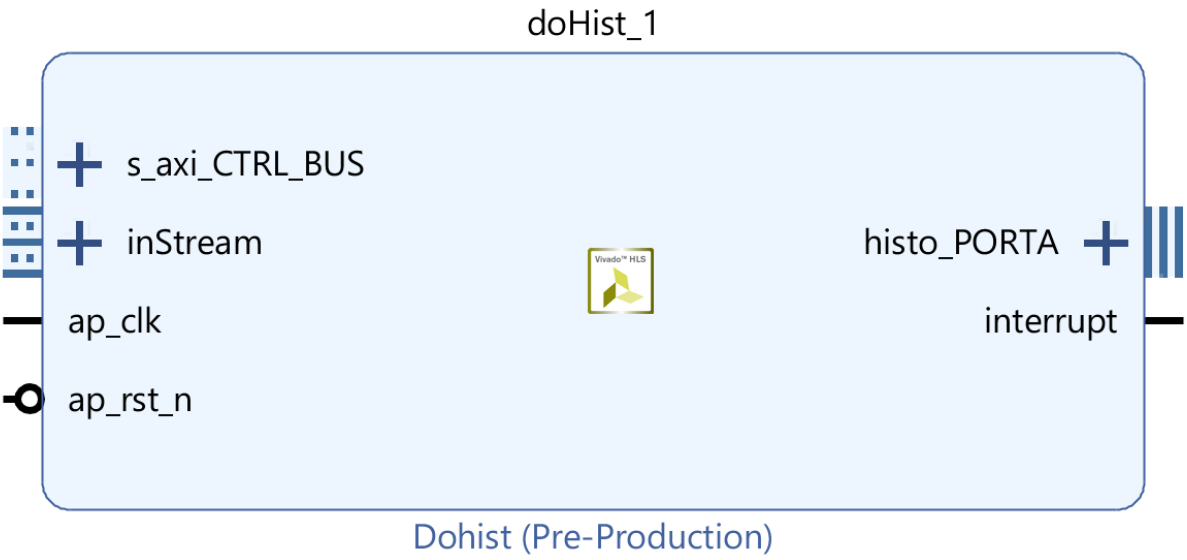
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	540
FIFO	-	-	-	-
Instance	0	3	1308	1941
Memory	-	-	-	-
Multiplexer	-	-	-	443
Register	0	-	822	224
Total	0	3	2130	3148
Available	280	220	106400	53200
Utilization (%)	0	1	2	5

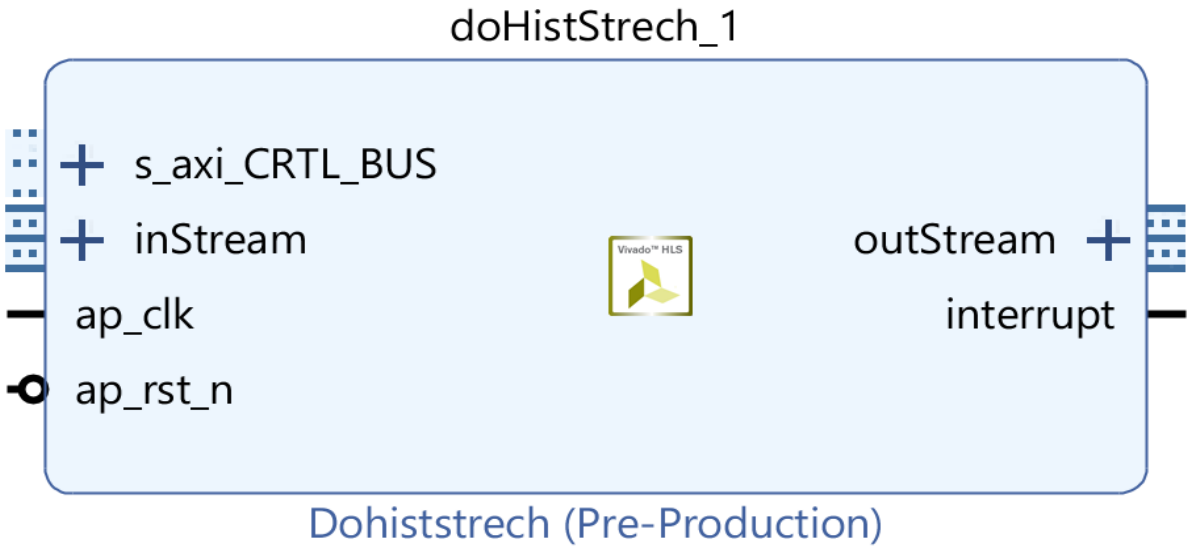
OUTPUT

INPUT

OUTPUT

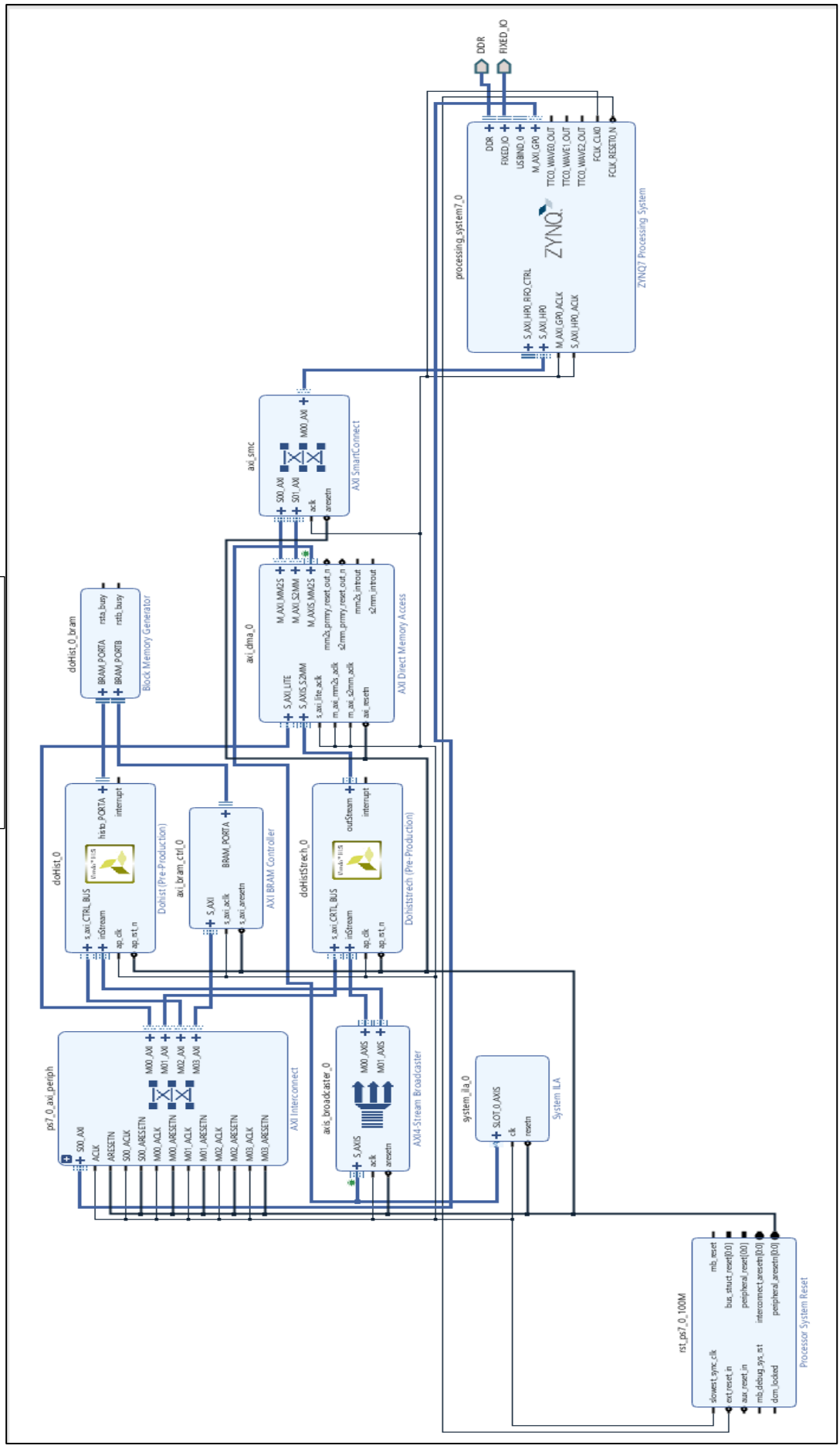
The above two IP's generated using HLS should be imported it to VIVADO.





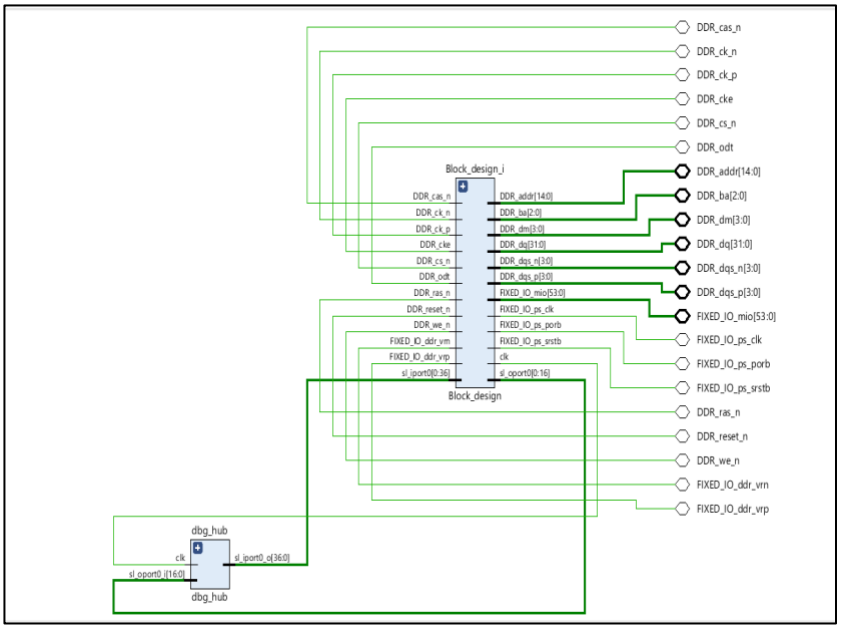
We need to create a block design to integrate these blocks with PS in Zed Board.

BLOCK DESIGN

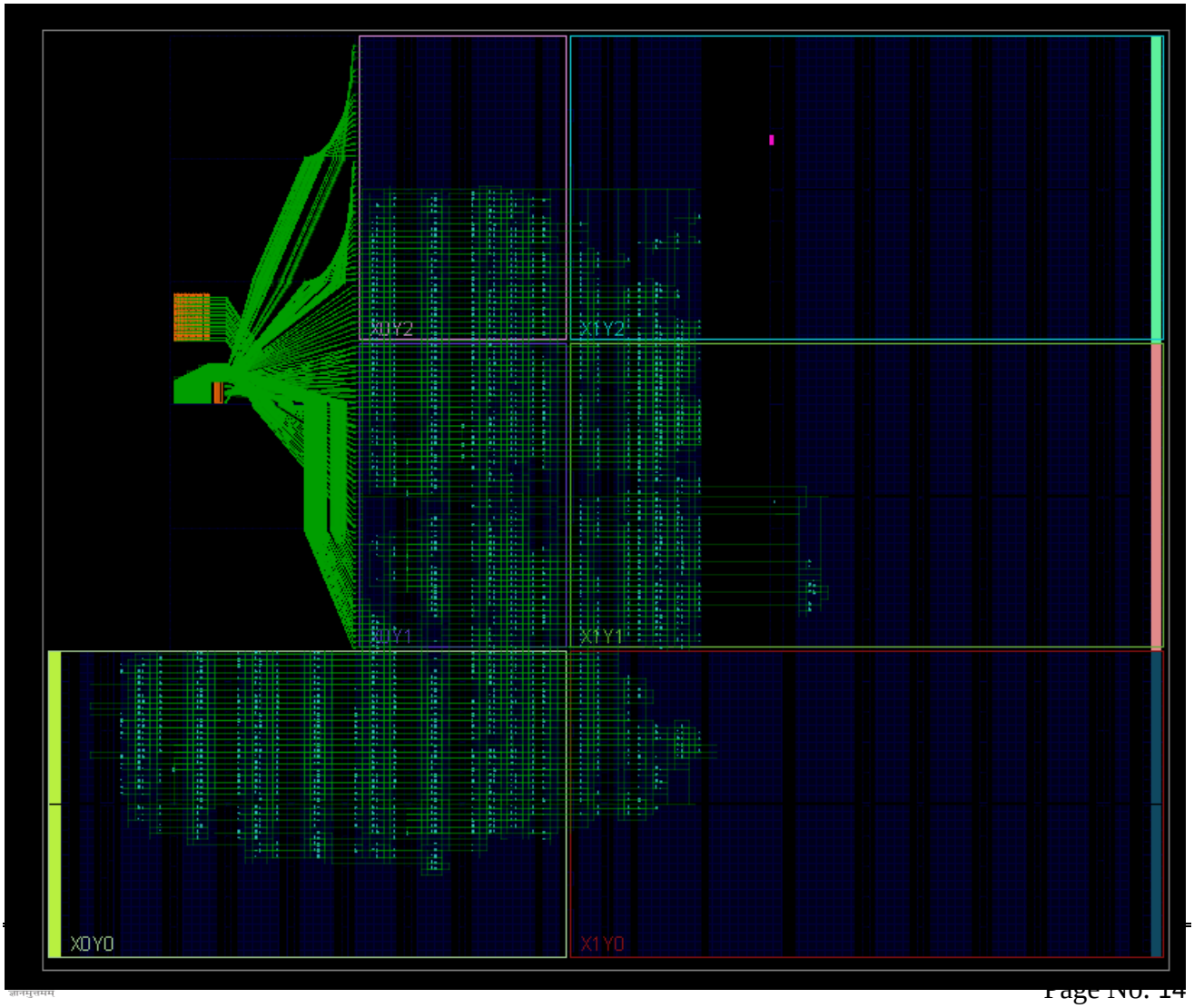


BROADCAST BLOCK: This block is used to send AXI stream data to two IP’s parallelly or we can say it repeats the stream to two IP’ cores (stream is synchronised).

OVERALL SCHEMATIC

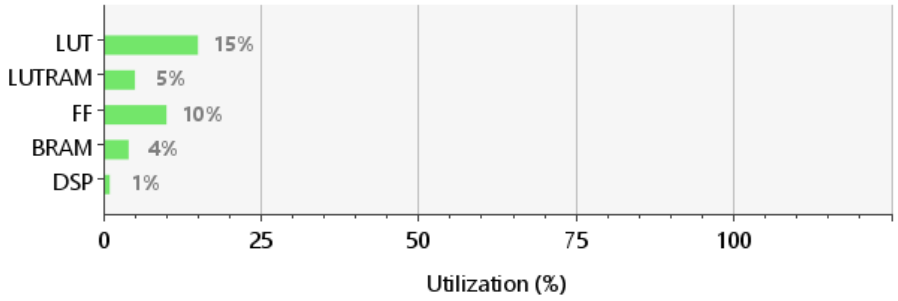


FLOOR PLAN



REPORT UTILIZATION

Resource	Utilization	Available	Utilization %
LUT	8171	53200	15.36
LUTRAM	876	17400	5.03
FF	10787	106400	10.14
BRAM	5.50	140	3.93
DSP	3	220	1.36



REPORT OF TIMING

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.052 ns	Worst Hold Slack (WHS):	0.009 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	31028	Total Number of Endpoints:	31012	Total Number of Endpoints:	12316

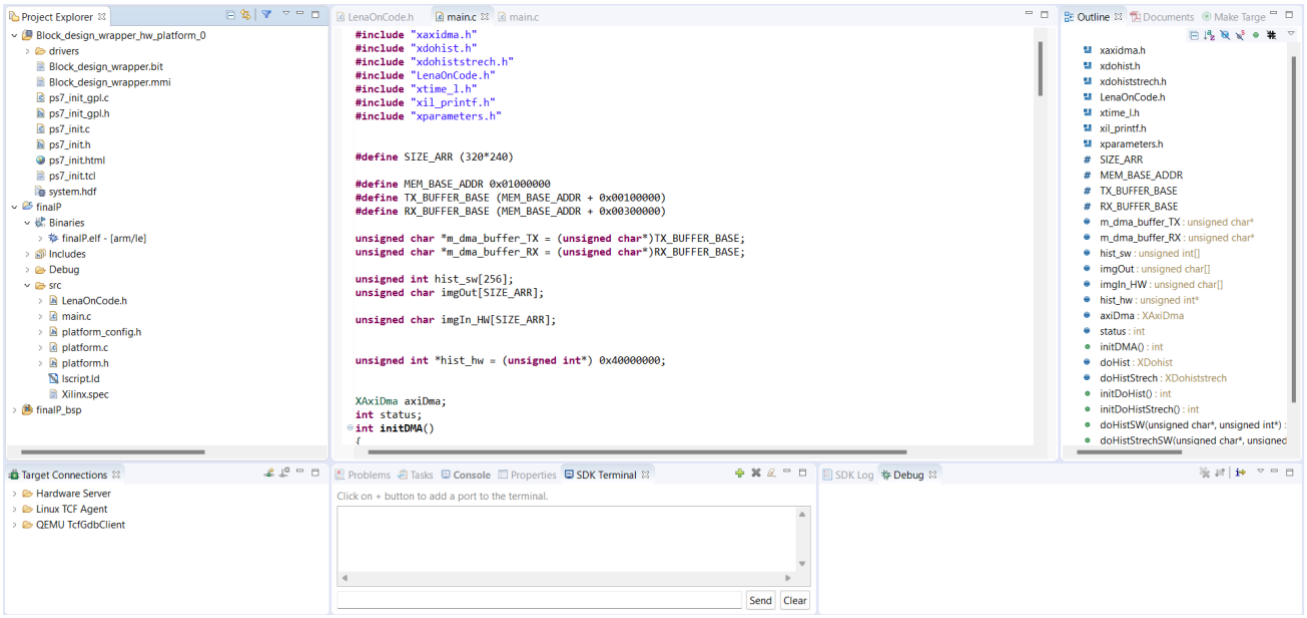
All user specified timing constraints are met.

After creating HDL Wrapper, we generated the bitstream and Exported hardware including bitstream then jumped to SDK.

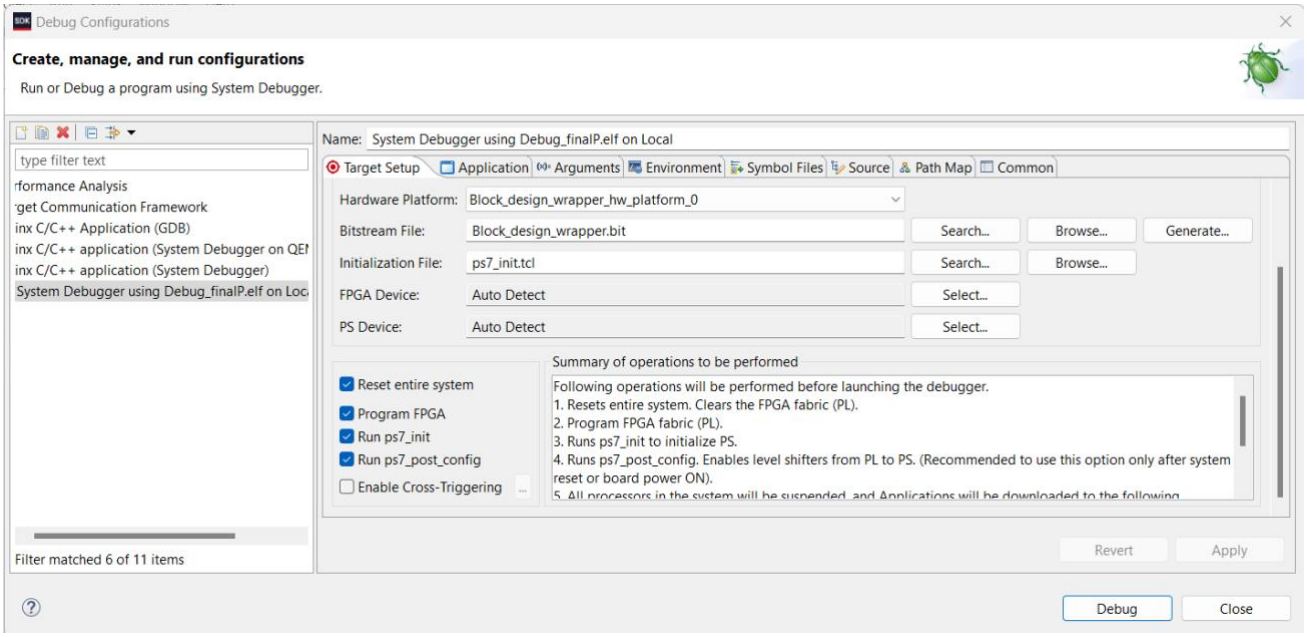
SDK PART:

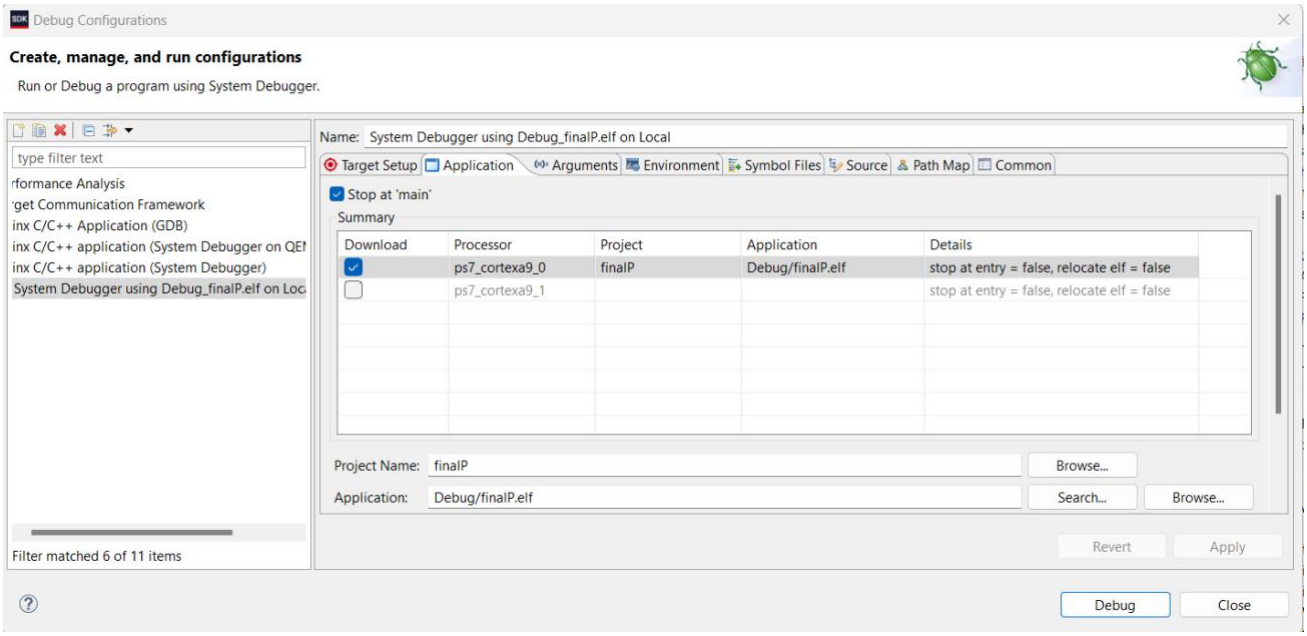
We have to convert the image as header file and include it in code. The process of converting image is included in MATLAB Section.

We have to create new application project and create main.c (code is included in sources)



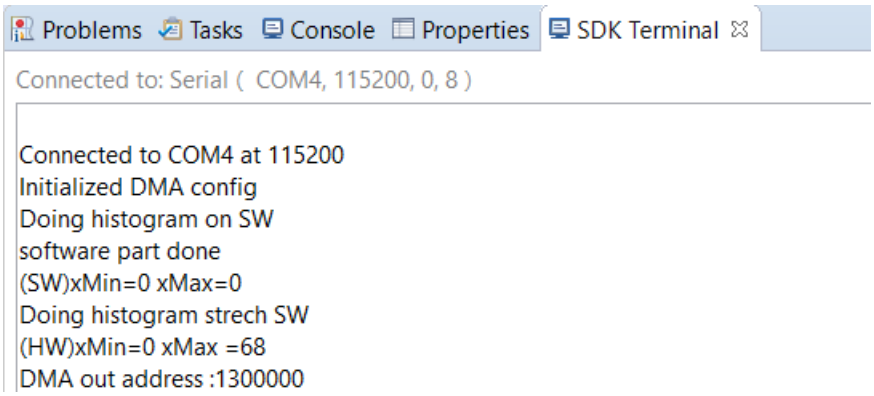
Go to Debug Configurations – Target Setup – Stand Alone Application Debug– Check Program FPGA – Application – Select Core- Then Click Debug





Then Switch to VIVADO and goto hardware manager choose auto connect and program device.

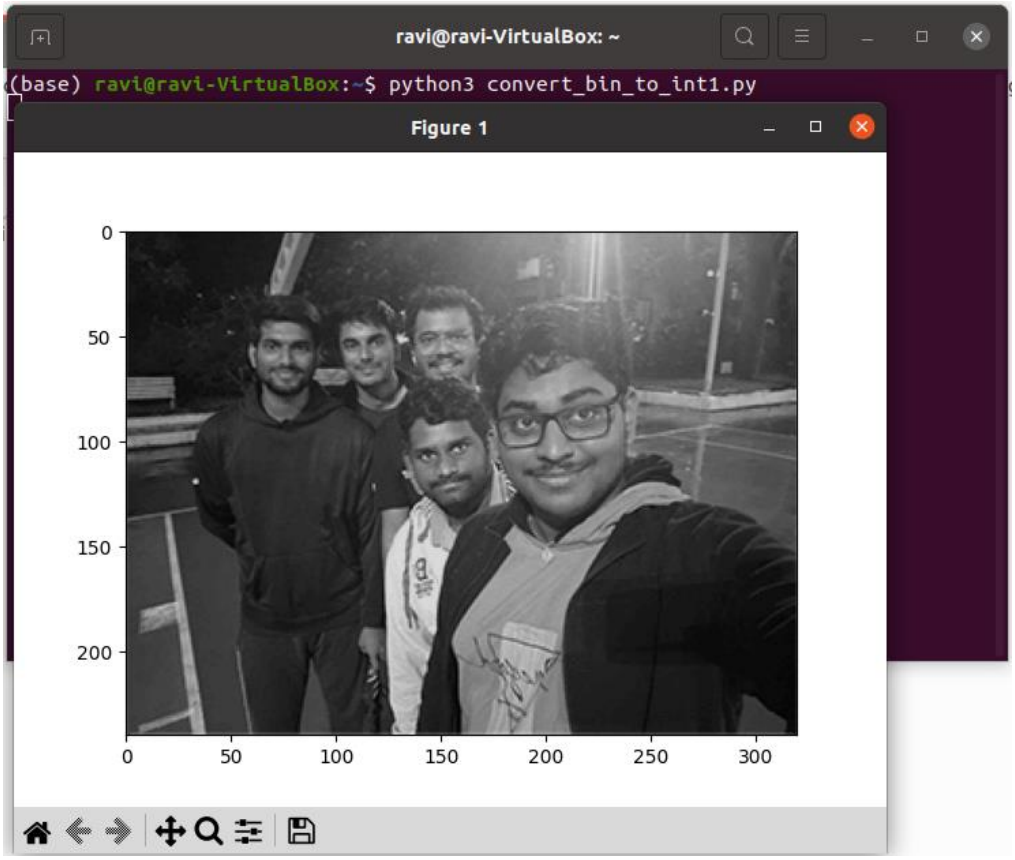
Switch back to SDK and go to previous line to return 0 of main.c and click run to line.



Then Go to XSCT terminal and use the following command to extract the data from DDR to BIN file

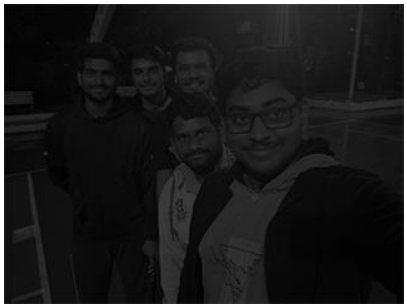
```
xsct%mrd -bin -file project.bin -size b -value 0x01300000 76800
```

A python code is provided in sources to view the "extracted image in BIN format".



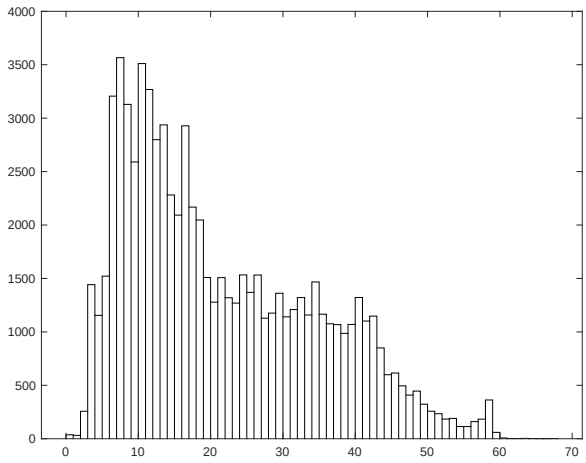
Output:

INPUT

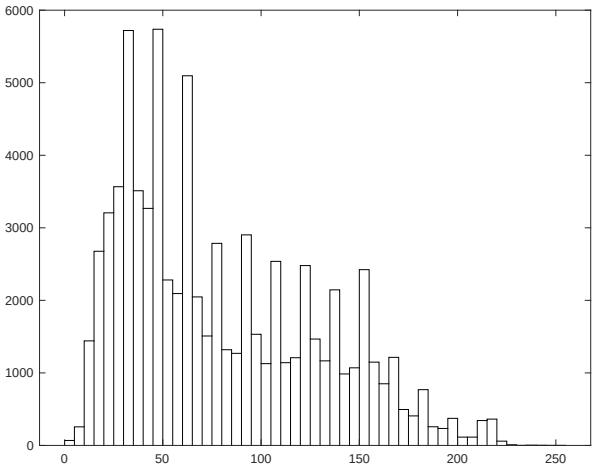


OUTPUT

Histogram of Input:



Histogram of Output:



Making the Project Hardware Software Co-design:

Normally we are performing two major operations

1) Finding Histogram to find maximum and minimum pixel values
2) Performing Histogram Stretch (which requires maximum and minimum pixel values)

We Developed IP's to perform both of above operations(Using HLS)

If we perform both operations using developed IP's then it becomes Pure hardware design.

In SDK we have facility to write C-code which RUNS on PS

We are calculating maximum and minimum pixel values with help of C code and passing these values along with image to IP implemented in PL makes this design as HARDWARE – SOFTWARE CODESIGN.

(We are passing maximum and minimum values over AXI4-LITE and Image pixel values over AXI4-STREAM)

SDK C-code which contains both normal hardware design and hardware-software code design in single code which is included in source files.

MATLAB Section:

Code to Perform Histogram Stretch:

Save this file as .m file

```
----------------------------------------
function [imgOut] = strechHisto(imgIn)
yMax = single(255);
xMin = single(min(min(imgIn)));
xMax = single(max(max(imgIn)));
imgOut = imgIn;
for idx=1:numel(imgIn)
    x = single(imgIn(idx));
    imgOut(idx)=((x-xMin)/(xMax-xMin))*yMax;
end
end
----------------------------------------
```

Commands:

Go to Working directory

To read image into variable

```
>> img = imread('Lennawithoutstr.bmp')
```

To see image stored in variable

```
>> imtool(img)
```

To find max and minimum pixel intensity of image

```
>> min(min(img))
```

```
>> max(max(img))
```

To check histogram

```
>> histogram(img)
```

To Perform histogram stretch

```
>> imtool(strechHisto(img));
```

To Convert data into csv file which can in used as input (in header file format) in SDK

```
>> reshape(img,1,[])
```

```
>> csvwrite('grp.csv',ans)
```

PYTHON CODE:

```
import matplotlib.pyplot as plt
import numpy as np

filename = "project.bin"

with open(filename, "rb") as datafile:
    data = datafile.read()

decimal = np.array([x for x in data]).reshape(320, 240)
decimal = np.transpose(decimal)

plt.imshow(decimal, cmap='gray')
plt.show()
```

References:

Youtube: The Vertex - What is Histogram of an Image: Dr Manjusha Deshmukh

https://www.youtube.com/watch?v=xLwpPm8Q1Ic

YouTube: The Development Channel - Vivado HLS Course Training

https://www.youtube.com/playlist?list=PLo7bVbJhQ6qzK6ELKCm8H_WEzzcr5YXHC

- END OF REPORT-