EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

Abstract

Multiple-choice examinations are indispensable method to measure the performance of a student throughout his/her education period. The teachers have a great responsibility for grading their students. Reading the correct answer for each student, giving corresponding grade to exams and uploading it to a computer takes too much time. The problem is that every teacher doesn't have their own optical mark reader (OMR) and they must examine all exam sheets with visual inspection. The purpose in this project is to build a simple optical mark reader, it can be very useful since it saves a lot of time. The algorithm that we built can recognize the contour of the answer sheet from an image, but it also needs to upload all answer sheets to a computer but using real time camera will make this process faster and more convenient. So, the camera will be integrated to the FPGA and the results will appear faster. The most important advantage of using an FPGA is to execute the compute-intensive image processing algorithms way faster than a conventional processor-based solution. In the OMR based applications, sometimes reading thousands of answer sheets takes days, even can be a week. The purpose in this project is to improve the algorithms by designing its specific hardware via HLS and this allows anyone to grade the answer sheets instantly.

Key Words- **OMR, Python, OpenCV, Optical Mark Reader, Optical Mark Recognition.**

1. Introduction

Optical mark recognition (OMR) is an application that scans paper forms to detect marked bubbles at predetermined positions. The idea of such a system was proposed by R. B. Johnson, a high school science teacher in Michigan, who devised a machine for recording students test answers and to compare them to an answer key [1]. It has been using for 80 years to transfer the form data to a computer for analysis [2]. The popularization of personal computers in 1980s caused an increase in OMR which allows anyone with a laser printer and scanner to custom design and print out their own forms. OMR has been widely used in education since the 1960s and is still popular today.

Many traditional OMR devices use laser scanner that illuminates a sheet. The reflectivity at those predetermined locations of the bubble patterns is used to detect the marked areas because those marked areas reflect less light than the blank areas of the sheet. Some of the OMR devices use preprinted forms onto trans-optic sheet and measure the amount of light that passes through the paper.

A software optical mark recognition, Image Processing Based OMR Sheet Scanning [3], can read the OMR sheet which contains maximum 50 questions by finding the region of interest and applying template matching algorithm. The implementation was done using C#.net and the image processing has done in Open CV.

OMR is a convenient way to read data from OMR sheets that are specialized forms with bubble pattern on them. Our aim is to detect those patterns that are marked by the students by using OMR algorithm and to adapt it to a hw/sw co-design. The main purpose of this project is to read and grade optical markers in an effective way, especially in time. Because in today's education systems many competitions, as well as government exams, are multiple-choice. So, OMR is the most cost-effective method when the number of answer sheets to be evaluated is very large.

The main advantages of OMR technology are its speed, accuracy and cost. OMR is less expensive than hardware-based OMR devices because it requires only ordinary paper and printer toner. It can maintain a throughput of 2000 to 10000 form per hour which makes it fast. Also, it consistently provides unmatched accuracy when reading data, exceeding the accuracy of expert key-entry clerks because it eliminates transcription errors.

This project can be used for other applications such as surveys, lotteries, elections, assessments, by making small changes in the form design and software algorithm.

2. Methodology [4] [5] [6] [7] [8] [9]

At the beginning, we were planning to use PYNQ-Z2 board. After the COVID-19 pandemic, our order was delayed due to restrictions. Then, our advisor suggested a website that allows us to access a PYNQ-Z2 board online [10]. But the website was in Chinese and very slow. Because of these reasons, we decided to work on our ZedBoard Zynq-7020 development kit. The project that we are working on is about image processing. We wrote all image processing functions by ourselves because we had trouble using OpenCV library on SDK side. We cannot run OpenCV library on an ARM processor. For now, we are dealing with processing on image for testing stage, later we will be working with real time camera.

EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

2.1. Uploading the Input Image Through an Array

First, we converted our RGB input image to one-dimensional array on Python. And we saved this array as a header file on SDK. Then, we did necessary morphological operations on that array.

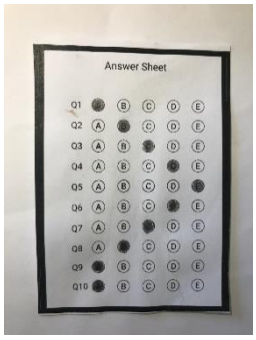

Fig.1: Input Image

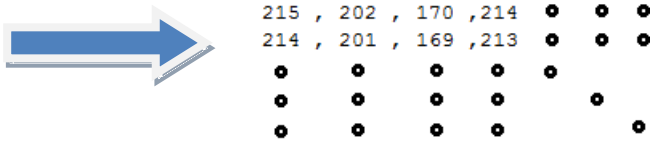

Fig.2: One-dimensional array represents the input image

2.1.1. Gray Scaling the Image

The answer sheet may have different colors on it like red, green and blue. It is a little bit hard to work with colorful images, but black and white colors are enough to detecting the right answers. According to this function, for each pixel we multiplied R, G and B values with their parameters and sum them up to obtain a new pixel value. On a digital image the colors represented by numbers. With the gray scaled image, the colors represented by numbers between 0-255 (0 is taken to be black, and 255 is taken to be white). And the image is gray scaled for next step.

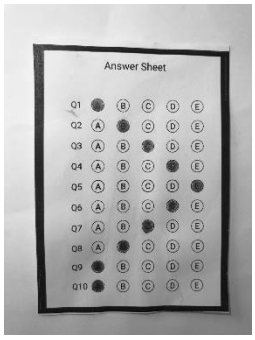

Fig. 3: Gray Scale Image

2.1.2. Blurring the Gray Image

It is used to reduce the image noise and reduce detail. It is a low pass filter; thus, it has the effect of reducing the image's high frequency components [14]. The reason that we blur the image is to make transition at the edges of the circles softer. At this step, we made some tests with different kernels and we noticed that the best result is obtained with a 5x5 kernel [15]. To get a better result we padded the gray scaled image with zeros.

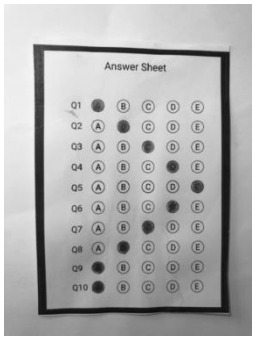

Fig. 4: Blurred Image (5x5 kernel)

2.2. Thresholding the Image

Gray scaling the image is not a clean way to check the right answers by itself because there are lots of gray tints which can be close the black or white. So; the image must include only white (255) or black (0) pixels. The simplest thresholding method replaces each pixel in an image with a black pixel if the image intensity is less than some fixed constant value or a white pixel if the image intensity is greater than that constant value. In the end the image includes only white (255) and black (0) intensity level [16].

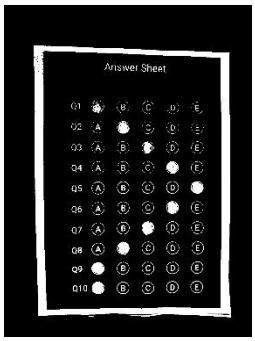

Fig. 5: Threshold Applied Image

Now the image is ready to count the circles by applying a mask that contains only the true answers. The mask works like a torch. The threshold applied image will be masked for each circle and the other all pixels outside of the mask is black. So, on the image there is only one choice. On the right answer there will be lots of white pixels and counting those pixels detect us the right answer.

2.3. Applying Canny Edge Detection

We used this function to detect the outermost black frame. If we can detect this frame, it will be easier for us to deal with the inside of the frame where the answered choices are marked after this stage. We convolved 2 different kernels to detect the horizontal and vertical edges. Then, we calculated gradient and angle values for each pixel. After, we checked each neighbor pixel to know whether there is an edge line or not. Finally, we applied a double-threshold to get rid of the weak edges.

EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

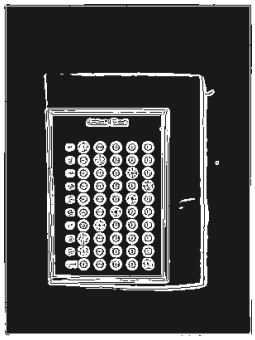

Fig. 6: Edge Detected Image

3. Empirical Setup

First, we designed our own OMR form to meet the requirements in the sample code and we printed it out. Then, we marked question bubbles on that form randomly. We left some of them blank to test the response of the code when it faced with an empty answer. Next, we took a picture of this filled OMR form and we uploaded it to a personal computer. After that, we run our code by importing the picture we have taken. We are planning to integrate a camera to the FPGA to make this OMR process faster and convenient. We used C++ as programming language, OpenCV for image processing, and we used Vivado HLS 2017.4 environment to design our IP core that handles with the gaussian blurring step.

4. Empirical Results

We run a part of our OMR code that contains some image processing functions such as gray scale, gaussian blur, threshold etc. When we run C simulation of this code on Vivado HLS, we did not get any error or warning. It run successfully.

As a next step we synthesized the C++ code. After synthesizing, a report file named "imgproc_csynth.rpt" is created. Its name is imgproc because it takes after its name from top function. According to this report file:

1. PERFORMANCE ESTIMATES

a) Synthesis Timing (ns) Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

b) Synthesis Latency (clock cycles) Loop Detail

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- Loop 1	1200800	1200800	3002	-	-	400	no
+ Loop 1.1	3000	3000	10	-	-	300	no
- Loop 2	246440	246440	610	-	-	404	no
+ Loop 2.1	608	608	2	-	-	304	no
- Loop 3	?	?	?	-	-	400	no
+ Loop 3.1	?	?	?	-	-	300	no
++ Loop 3.1.1	?	?	?	-	-	?	no
+++ Loop 3.1.1.1	?	?	3	-	-	?	no
- Loop 4	360800	360800	902	-	-	400	no
+ Loop 4.1	900	900	3	-	-	300	no

2. UTILIZATION ESTIMATES

a) Summary

EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	3	-	-
Expression	-	0	0	1137
FIFO	-	-	-	-
Instance	4	-	1136	1328
Memory	128	-	6	3
Multiplexer	-	-	-	462
Register	-	-	912	-
Total	132	3	2054	2930
Available	280	220	106400	53200
Utilization (%)	47	1	1	5

b) Details

- ***Instance***

Instance	Module	BRAM_18K	DSP48E	FF	LUT
imageproc_CTRL_s_axi_U	imageproc_CTRL_s_axi	0	0	112	168
imageproc_INPUT_r_m_axi_U	imageproc_INPUT_r_m_axi	2	0	512	580
imageproc_OUTPUT_r_m_axi_U	imageproc_OUTPUT_r_m_axi	2	0	512	580
Total	3	4	0	1136	1328

- ***DSP48***

Instance	Module	Expression
imageproc_am_addmcud_U1	imageproc_am_addmcud	(i0 + i1) * i2
imageproc_mac_muleOg_U3	imageproc_mac_muleOg	i0 * i1 + i2
imageproc_mul_muldEe_U2	imageproc_mul_muldEe	i0 * i1

- ***Memory***

Memory	Module	BRAM_18K	FF	LUT	Words	Bits	Banks	W*Bits*Banks
arr1_U	imageproc_arr1	64	0	0	120000	8	1	960000
kernel1_U	imageproc_kernel1	0	6	3	25	6	1	150
zpadded_img4_U	imageproc_zpaddedbkb	64	0	0	122816	8	1	982528
Total	3	128	6	3	242841	22	3	1942678

- ***Expression***

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
tmp_9_fu_784_p2	*	0	0	51	9	9
a_1_fu_934_p2	+	0	0	39	32	1
a_fu_760_p2	+	0	0	16	9	3
b_1_fu_928_p2	+	0	0	39	1	32
b_fu_796_p2	+	0	0	16	9	3
i_1_fu_516_p2	+	0	0	16	9	1
i_2_fu_599_p2	+	0	0	16	9	1
i_3_fu_816_p2	+	0	0	16	9	1
i_4_fu_994_p2	+	0	0	16	9	1
image_in2_sum_fu_544_p2	+	0	0	38	31	31
j_1_fu_534_p2	+	0	0	16	9	1
j_2_fu_615_p2	+	0	0	16	9	1
j_3_fu_1006_p2	+	0	0	16	9	1
j_4_fu_883_p2	+	0	0	16	9	1
k_2_fu_522_p2	+	0	0	24	17	9
m_1_fu_827_p2	+	0	0	39	32	1

n_1_fu_894_p2	+	0	0	39	32	1
next_mul6_fu_583_p2	+	0	0	24	17	9
next_mul8_fu_982_p2	+	0	0	24	17	9
next_mul_fu_504_p2	+	0	0	24	17	9
tmp_20_fu_705_p2	+	0	0	17	10	3
tmp_24_fu_739_p2	+	0	0	24	17	17
tmp_25_fu_806_p2	+	0	0	16	9	2
tmp_29_fu_715_p2	+	0	0	25	18	18
tmp_30_fu_729_p2	+	0	0	24	17	17
tmp_31_fu_1016_p2	+	0	0	24	17	17
tmp_37_fu_849_p2	+	0	0	15	6	6
tmp_45_fu_878_p2	+	0	0	25	18	18
tmp_47_fu_904_p2	+	0	0	15	6	6
tmp_49_fu_918_p2	+	0	0	25	18	18
tmp_4_fu_553_p2	+	0	0	24	17	17
tmp_5_fu_559_p2	+	0	0	24	1	17
tmp_7_fu_770_p2	+	0	0	16	9	2
p_neg_fu_859_p2	-	0	0	39	1	32
tmp_34_fu_954_p2	-	0	0	15	1	8
ap_block_state16_io	and	0	0	8	1	1
exitcond1_fu_988_p2	icmp	0	0	13	9	8
exitcond2_fu_790_p2	icmp	0	0	13	9	9
exitcond3_fu_754_p2	icmp	0	0	13	9	8
exitcond4_fu_609_p2	icmp	0	0	13	9	9
exitcond5_fu_593_p2	icmp	0	0	13	9	8
exitcond6_fu_528_p2	icmp	0	0	13	9	9
exitcond7_fu_510_p2	icmp	0	0	13	9	8
exitcond_fu_1000_p2	icmp	0	0	13	9	9
tmp_10_fu_621_p2	icmp	0	0	13	9	8
tmp_11_fu_627_p2	icmp	0	0	13	9	8
tmp_13_fu_639_p2	icmp	0	0	13	9	1
tmp_15_fu_651_p2	icmp	0	0	13	9	1
tmp_17_fu_663_p2	icmp	0	0	13	9	9
tmp_18_fu_669_p2	icmp	0	0	13	9	9
tmp_21_fu_681_p2	icmp	0	0	13	9	1
tmp_23_fu_693_p2	icmp	0	0	13	9	1
tmp_26_fu_822_p2	icmp	0	0	18	32	32
tmp_36_fu_889_p2	icmp	0	0	18	32	32
tmp_12_fu_633_p2	or	0	0	8	1	1
tmp_14_fu_645_p2	or	0	0	8	1	1
tmp_16_fu_657_p2	or	0	0	8	1	1
tmp_19_fu_675_p2	or	0	0	8	1	1
tmp_22_fu_687_p2	or	0	0	8	1	1
tmp_27_fu_699_p2	or	0	0	8	1	1
tmp_43_fu_969_p3	select	0	0	8	1	8
Total	61	0	0	1137	676	508

- ***Multiplexer***

Name	LUT	Input Size	Bits	Total Bits
INPUT_r_blk_n_AR	9	2	1	2
INPUT_r_blk_n_R	9	2	1	2
OUTPUT_r_blk_n_AW	9	2	1	2
OUTPUT_r_blk_n_B	9	2	1	2
OUTPUT_r_blk_n_W	9	2	1	2
a1_reg_382	9	2	32	64
ap_NS_fsm	141	31	1	31
ap_sig_ioackin_INPUT_r_ARREADY	9	2	1	2

ap_sig_ioackin_OUTPUT_r_AWREADY	9	2	1	2
ap_sig_ioackin_OUTPUT_r_WREADY	9	2	1	2
arr1_address0	27	5	17	85
arr1_d0	15	3	8	24
b1_reg_415	9	2	32	64
i1_reg_312	9	2	9	18
i3_reg_347	9	2	9	18
i5_reg_436	9	2	9	18
i_reg_256	9	2	9	18
j2_reg_336	9	2	9	18
j4_reg_359	9	2	9	18
j6_reg_459	9	2	9	18
j_reg_291	9	2	9	18
k_1_reg_302	9	2	17	34
k_reg_267	9	2	17	34
m_reg_371	9	2	32	64
n_reg_404	9	2	32	64
phi_mul5_reg_324	9	2	17	34
phi_mul7_reg_447	9	2	17	34
phi_mul_reg_279	9	2	17	34
tmp_2_reg_392	9	2	32	64
tmp_3_reg_424	9	2	32	64
zpadded_img4_address0	21	4	17	68
zpadded_img4_d0	15	3	8	24
Total	462	100	408	946

- ***Register***

Name	FF	LUT	Bits	Const Bits
OUTPUT_addr_reg_1053	30	0	32	2
a1_reg_382	32	0	32	0
a_cast_reg_1163	9	0	32	23
ap_CS_fsm	30	0	30	0
ap_reg_ioackin_INPUT_r_ARREADY	1	0	1	0
ap_reg_ioackin_OUTPUT_r_AWREADY	1	0	1	0
ap_reg_ioackin_OUTPUT_r_WREADY	1	0	1	0
arr1_load_1_reg_1303	8	0	8	0
b1_reg_415	32	0	32	0
b_1_reg_1252	32	0	32	0
b_cast_reg_1186	9	0	32	23
b_reg_1181	9	0	9	0
i1_reg_312	9	0	9	0
i3_reg_347	9	0	9	0
i5_reg_436	9	0	9	0
i_1_reg_1072	9	0	9	0
i_2_reg_1124	9	0	9	0
i_4_reg_1285	9	0	9	0
i_reg_256	9	0	9	0
image_in2_sum_reg_1090	31	0	31	0
j2_reg_336	9	0	9	0
j4_reg_359	9	0	9	0
j6_reg_459	9	0	9	0
j_1_reg_1085	9	0	9	0
j_2_reg_1137	9	0	9	0
j_3_reg_1293	9	0	9	0
j_4_reg_1229	9	0	9	0
j_reg_291	9	0	9	0
k_1_reg_302	17	0	17	0

k_2_reg_1077	17	0	17	0
k_reg_267	17	0	17	0
kernel1_load_reg_1262	6	0	6	0
m_1_reg_1204	32	0	32	0
m_reg_371	32	0	32	0
n_1_reg_1237	32	0	32	0
n_reg_404	32	0	32	0
next_mul6_reg_1116	17	0	17	0
next_mul8_reg_1277	17	0	17	0
next_mul_reg_1064	17	0	17	0
phi_mul5_reg_324	17	0	17	0
phi_mul7_reg_447	17	0	17	0
phi_mul_reg_279	17	0	17	0
tmp_10_cast_reg_1059	30	0	31	1
tmp_13_cast_reg_1168	9	0	32	23
tmp_16_reg_1142	1	0	1	0
tmp_1_reg_1111	8	0	8	0
tmp_25_cast_reg_1191	9	0	32	23
tmp_27_reg_1146	1	0	1	0
tmp_2_reg_392	32	0	32	0
tmp_30_reg_1155	17	0	17	0
tmp_33_reg_1219	8	0	8	0
tmp_37_reg_1209	6	0	6	0
tmp_39_reg_1214	18	0	18	0
tmp_3_reg_424	32	0	32	0
tmp_45_reg_1224	18	0	18	0
tmp_4_reg_1095	17	0	17	0
tmp_5_reg_1100	17	0	17	0
tmp_9_reg_1173	16	0	18	2
tmp_s_reg_1129	18	0	18	0
zpadded_img4_load_reg_1267	8	0	8	0
Total	912	0	1009	97

3. INTERFACE

a) Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_CTRL_AWVALID	in	1	s_axi	CTRL	scalar
s_axi_CTRL_AWREADY	out	1	s_axi	CTRL	scalar
s_axi_CTRL_AWADDR	in	5	s_axi	CTRL	scalar
s_axi_CTRL_WVALID	in	1	s_axi	CTRL	scalar
s_axi_CTRL_WREADY	out	1	s_axi	CTRL	scalar
s_axi_CTRL_WDATA	in	32	s_axi	CTRL	scalar
s_axi_CTRL_WSTRB	in	4	s_axi	CTRL	scalar
s_axi_CTRL_ARVALID	in	1	s_axi	CTRL	scalar
s_axi_CTRL_ARREADY	out	1	s_axi	CTRL	scalar
s_axi_CTRL_ARADDR	in	5	s_axi	CTRL	scalar
s_axi_CTRL_RVALID	out	1	s_axi	CTRL	scalar
s_axi_CTRL_RREADY	in	1	s_axi	CTRL	scalar
s_axi_CTRL_RDATA	out	32	s_axi	CTRL	scalar
s_axi_CTRL_RRESP	out	2	s_axi	CTRL	scalar
s_axi_CTRL_BVALID	out	1	s_axi	CTRL	scalar
s_axi_CTRL_BREADY	in	1	s_axi	CTRL	scalar
s_axi_CTRL_BRESP	out	2	s_axi	CTRL	scalar
ap_clk	in	1	ap_ctrl_hs	imageproc	return value
ap_rst_n	in	1	ap_ctrl_hs	imageproc	return value

EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

interrupt	out	1	ap_ctrl_hs	imageproc	return value
m_axi_INPUT_r_AWVALID	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWREADY	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWADDR	out	32	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWID	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWLEN	out	8	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWSIZE	out	3	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWBURST	out	2	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWLOCK	out	2	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWCACHE	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWPROT	out	3	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWQOS	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWREGION	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_AWUSER	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WVALID	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WREADY	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WDATA	out	32	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WSTRB	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WLAST	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WID	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_WUSER	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARVALID	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARREADY	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARADDR	out	32	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARID	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARLEN	out	8	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARSIZE	out	3	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARBURST	out	2	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARLOCK	out	2	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARCACHE	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARPROT	out	3	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARQOS	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARREGION	out	4	m_axi	INPUT_r	pointer
m_axi_INPUT_r_ARUSER	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RVALID	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RREADY	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RDATA	in	32	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RLAST	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RID	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RUSER	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_RRESP	in	2	m_axi	INPUT_r	pointer
m_axi_INPUT_r_BVALID	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_BREADY	out	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_BRESP	in	2	m_axi	INPUT_r	pointer
m_axi_INPUT_r_BID	in	1	m_axi	INPUT_r	pointer
m_axi_INPUT_r_BUSER	in	1	m_axi	INPUT_r	pointer
m_axi_OUTPUT_r_AWVALID	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWREADY	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWADDR	out	32	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWID	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWLEN	out	8	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWSIZE	out	3	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWBURST	out	2	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWLOCK	out	2	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWCACHE	out	4	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWPROT	out	3	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWQOS	out	4	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_AWREGION	out	4	m_axi	OUTPUT_r	pointer

m_axi_OUTPUT_r_AWUSER	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WVALID	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WREADY	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WDATA	out	32	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WSTRB	out	4	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WLAST	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WID	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_WUSER	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARVALID	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARREADY	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARADDR	out	32	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARID	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARLEN	out	8	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARSIZE	out	3	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARBURST	out	2	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARLOCK	out	2	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARCACHE	out	4	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARPROT	out	3	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARQOS	out	4	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARREGION	out	4	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_ARUSER	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RVALID	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RREADY	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RDATA	in	32	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RLAST	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RID	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RUSER	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_RRESP	in	2	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_BVALID	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_BREADY	out	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_BRESP	in	2	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_BID	in	1	m_axi	OUTPUT_r	pointer
m_axi_OUTPUT_r_BUSER	in	1	m_axi	OUTPUT_r	pointer

Finally, we need to export RTL to be able to use this IP in Vivado Design Suite. At the end of this exporting process, it is created a report file called “imgproc_export.rpt”. According to this report file timing met:

4. RESOURCE USAGE

	VHDL
SLICE	550
LUT	1256
FF	1745
DSP	3
BRAM	130
SRL	45

5. FINAL TIMING

	VHDL
CP required	10.000
CP achieved post-synthesis	7.563
CP achieved post-implementation	9.496

EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

We compared the fully software code that we have developed on SDK and the code that we have used an IP core only for gaussian blurring function. We could design an IP core only for the gaussian blurring function because BRAM was not enough for the rest. Data memory was also limited because we used arrays size of 400x300. As a result, the fully software code takes 0,20 seconds and the accelerated code takes 0,09 seconds. Thus, the speedup is about 2,223.

5. Project Block Design

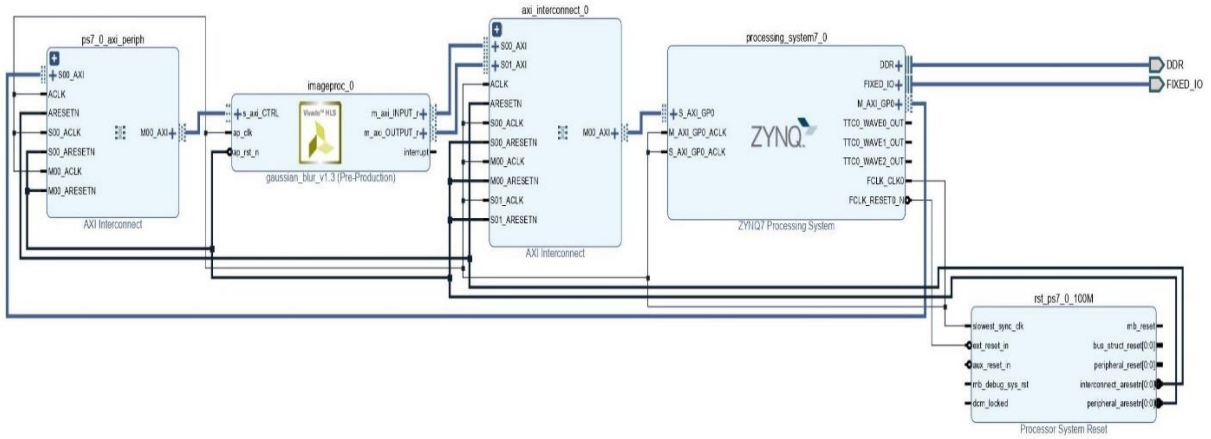

Fig. 7: Block design with the custom IP core designed on Vivado HLS

6. Conclusion

This method of mark recognition in OMR represents an improvement over existing methods employed in commercial software as well as others developed in recent years. The main advantage of this OMR process is that it can be used in many applications by changing the form design and software algorithm because it is a software-based OMR. It can maintain high level of accuracy that is enough even for scientific applications.

7. References

[1] R. S, K. Atal, ve A. Arora, "Cost Effective Optical Mark Reader", *Int. J. Comput. Sci. Artif. Intell.*, c. 3, ss. 44-49, Haz. 2013, doi: 10.5963/IJCSAI0302002.
[2] S. C. Loke, K. A. Kasmiran, ve S. A. Haron, "A new method of mark detection for software-based optical mark recognition", *PLOS ONE*, c. 13, sy 11, s. e0206420, Kas. 2018, doi: 10.1371/journal.pone.0206420.
[3] S. B. Gaikwad, "Image Processing Based OMR Sheet Scanning", c. 4, sy 3, s. 4, 2015.
[4] "Xilinx Vivado HLS Beginners Tutorial : Custom IP Core Design for FPGA". https://medium.com/@chathura.abeyrathne.lk/xilinx-vivado-hls-beginners-tutorial-custom-ip-core-design-for-fpga-59876d5a4119 (erişim May. 28, 2020).
[5] "Using HLS on an FPGA-Based Image Processing Platform - Hackster.io". https://www.hackster.io/adam-taylor/using-hls-on-an-fpga-based-image-processing-platform-8f029f (erişim May. 28, 2020).
[6] A. Taylor, *ATaylorCEngFIET/Hackster*. 2020.
[7] "FPGA-Based Edge Detection Using HLS - Hackster.io". https://www.hackster.io/adam-taylor/fpga-based-edge-detection-using-hls-192ad2 (erişim May. 28, 2020).
[8] "Leveraging OpenCV and High Level Synthesis with Vivado (v2013.1) - YouTube". https://www.youtube.com/watch?v=Y2iHh-HtXn4 (erişim May. 28, 2020).
[9] *Vivado HLS - FPGA Image Processing Histogram #09.1*. .
[10] "OpenHEC - FPGA云". http://www.openhec.com/#!/app/index (erişim May. 28, 2020).
[11] *Xilinx/xfopencv*. Xilinx, 2020.
[12] "HLS Video Library - Xilinx Wiki - Confluence". https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841665/HLS+Video+Library (erişim May. 28, 2020).

EEM 464 – System on Chip Design: Project Final Report

List of group members: Yunus Emre KÜTÜK
Hasan KARACA
Team Name: Murphy
Youtube Link: https://youtu.be/bGizXE-Kc7k

[13] "HLS CvtColor - Xilinx Wiki - Confluence". https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841662/HLS+CvtColor (erişim May. 28, 2020).
[14] "Gaussian blur", *Wikipedia*. May. 20, 2020, Erişim: May. 28, 2020. [Çevrimiçi]. Erişim adresi: https://en.wikipedia.org/w/index.php?title=Gaussian_blur&oldid=957739017.
[15] "HLS GaussianBlur - Xilinx Wiki - Confluence". https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841814/HLS+GaussianBlur (erişim May. 28, 2020).
[16] "HLS Threshold - Xilinx Wiki - Confluence". https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842068/HLS+Threshold (erişim May. 28, 2020).