Colour Recognizing Robot Arm Equipped with a CMOS Camera and an FPGA

Asma Taha Sadoon
College of Engineering
University of Baghdad

Dina Abdul Kareem Abdul Qader
College of Engineering
University of Baghdad

ABSTRACT
In this paper a system is designed on an FPGA using a Nios II soft-core processor, to detect the colour of a specific surface and moving a robot arm accordingly. The surface being detected is bounded by a starting mark and an ending mark, to define the region of interest. The surface is also divided into sections as rows and columns and each section can have any colour. Such a system has so many uses like for example warehouses or even in stores where their storing areas can be divided to sections and each section is coloured and a robot arm collects objects from these sections according to the section's colour also the robot arm can organize objects in sections according to the section's colour.

Keywords
Robot arm, sopc builder, colour reconition, FPGA, CMOS camera.

1. INTRODUCTION
Using robots in nowadays is very important for so many reasons which include accuracy and less time. Collecting, organizing, and sorting objects using robot arms are very useful in so many applications. In this paper a system is designed and implemented on an FPGA using a Nios II soft-core processor to detect the colour of a surface and moving a robot arm accordingly. So many previous work that involve colour recognition of objects in order for the robot arm to pick and place the objects according to their colour, especially if the objects move on a conveyer belt (only mentioning few). [1] designed and implemented a robot arm that can pick and sort objects depending on their colour, using a microcontroller and a colour sensor in the design. While [2] designed a pick and place robot arm that sorts objects according to their colour depending on a web cam connected to a computer on chip called eBox-3300MX which is responsible for the image processing and also connected to the microcontroller to control the robot arm. Also [3] designed a system that can sort objects according to their colour using a camera to capture the image that is sent to the computer to process the image using MATLAB and the computer sends signals to the microcontroller to move the robot. In this work the main objective is to design and implement a system that can organize objects in coloured spaces and also can be used for sorting objects according to their colour that is much simpler (although not tested).

The aim of this work is to design and implement a system that can detect the colour of a surface and pick an object from that particular colour position or organize objects in those desired positions. This system is implemented on an FPGA using a Nios ii soft-core processor and a CMOS camera.

2. THE SYSTEM DESIGN
Figure 1 shows a block diagram of the system, where the camera is connected to the FPGA, when the system is turned on the nios ii processor gives a start signal to the camera to start capturing and after 61 frames the nios ii processor gives a stop signal. The captured image is stored in the SDRAM located on the FPGA kit. The nios ii processor reads the image from the SDRAM and starts the image processing. Then the nios ii processor gives an output describing the coloured regions, this output is used by the FPGA to control the robot arm.

The system has two parts the hardware part and the software part:

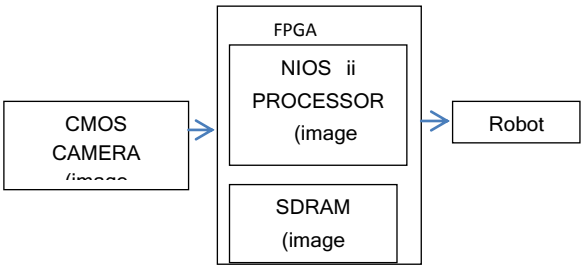

Figure 1. Block diagram of the system

2. 1The Hardware
The hardware part of the system is shown in figure 2, which is composed of the following:

2.1.1 DE2-115 FPGA.
The FPGA is chosen in this project as the programmable device to build the system due to its flexibility and the ability to use soft-core processors like Nios ii processor. The DE2-115 Cyclone IV EP4CE115F29C7 is chosen whose features are [4]:

- 114,480 LEs.
- 432 M9K memory blocks.
- 3,888 Kbits embedded memory.
- 4 PLLs.
- JTAG and AS mode configuration.
- EPCS64 serial configuration device.
- On-board USB Blaster circuitry.
- 128MB (32Mx32bit) SDRAM.
- 2MB (1Mx16) SRAM.
- 8MB (4Mx16) Flash with 8-bit mode.
- 32Kb EEPROM.

- 40-pin expansion port.(Configurable I/O standards (voltage levels:3.3/2.5/1.8/1.5V))
- VGA-out connector.
- Three 50MHz oscillator clock inputs.
- 18 slide switches and 4 push-buttons switches.
- 18 red and 9 green LEDs.
- Eight 7-segment displays.

Also other features not used in this project.

2.1.2 TRDB-D5M kit

The TRDB-D5M kit provides a 5 Mega pixel camera used with the DE2-115 FPGA.

2.1.3 Robot Arm

The robot arm has five degrees of freedom (five joints) each controlled by a servo motor. The servo motor can rotate from 0° to 180° and sometimes even more, by controlling the control signal of the servo motor. The servo motor has three wires, one for the control signal and one for the power (about 5v d.c) and one connected to ground. The control signal is a pulse waveform whose period is 20msec and the pulse width varies between 0.5 msec and 2msec in order to make the servo motor rotate from the 0 degree position to the 180 degree position.

2.1.4 A D.C Motor, Wheels, and a Motor Driver Circuit (L298d IC)

The wheels are connected to the D.C motor, and the D.C motor is controlled by a drive circuit shown in figure 3. The D.C motor's two wires are connected to OUT1 and OUT2. Table (1) shows the truth table of the motor drive circuit, if ENA is set to logic (1) then the D.C motor is enabled. To control the speed of the motor a pulse waveform is applied to the ENA input, this pulse waveform has a period of 1msec and as the duty cycle increases the speed of the motor increases. The pulse waveform that is applied to a D.C motor differs from that applied to a servo motor, for the first what matters is the duty cycle as for the latter what matters is the width of the pulse.

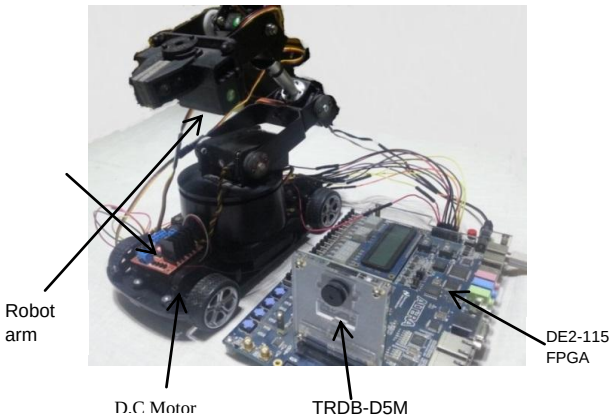

Figure 2. The Hardware Components

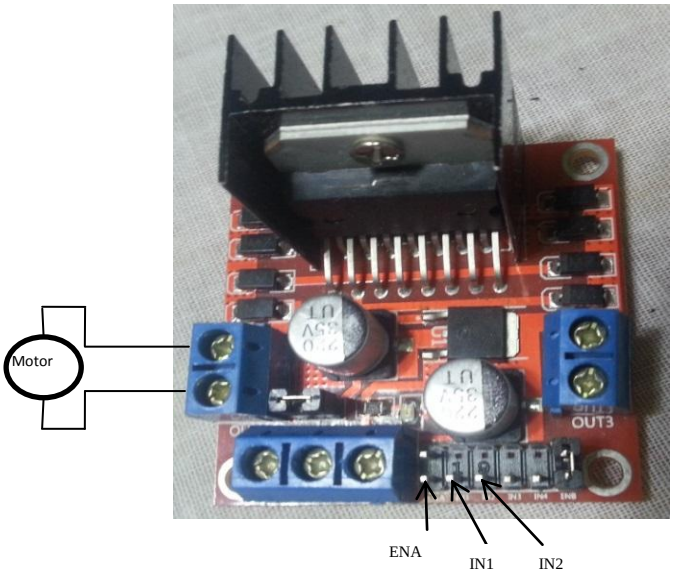

Figure 3. Motor Drive Circuit

Table 1 Motor drive circuit truth table

ENA	IN1	IN2	Motor status
1	0	0	Motor breaks
1	0	1	Motor moves
1	1	0	Motor moves in opposite Direction of case 2
1	1	1	Motor breaks
0	X	X	Motor is off

2. 2The Software

The software part is composed of the following:

2.2.1 The Camera Module

The camera module is composed of 4 main modules each provided by Altera and written in Verilog HDL:

- Cmos Sensor Data Capture (CCD_Capture)

This module gets raw data from the camera and converts it to raw data with X and Y coordinates to the next module (RAW2RGB).

- Bayer Color Pattern Data to 30-Bit RGB (RAW2RGB)

This module converts the raw data from the CCD_Capture module to a 30 bit RGB (Red, Green, and Blue) data, each colour has 10 bits.

- Multi-Port SDRAM Controller (Sdram_Control)

The SDRAM is the frame buffer of the images captured by the camera and this module controls the SDRAM chip. This module stores the Red, Green, and Blue data into the SDRAM chip available on the FPGA. The SDRAM is divided into two banks each with 16 bits, so the 10 bit Red data and the first five bits of the Green data are stored in bank 1, as for the 10 bit Blue data and the second 5 bit of the Green data are stored in bank 2.

- I2C Sensor configuration (I2C_CCD_Config)

This module controls camera settings like exposure time, resolution, and frame rate.

2.2.2 The SOPC (System On Programmable Chip)

The Nios ii system is designed using the SOPC (System On Programmable Chip) builder available in the Quartus ii package. The SOPC builder specifies the components of the system and its settings to build a complete computer system. The SOPC chooses a Nios ii processor (soft-core processor) and adds other components like peripherals, memories, bus connections and also creates components that are not available using the component editor (like the case of this paper) by adding the component's HDL. The SOPC creates a module written in Verilog HDL to describe each one of the components to be used in completing any system.

The SOPC of the colour recognizing robot arm is shown in Table (2). Each of the components is explained in the following:

- Cpu

This component is the Nios ii processor, in which the nios ii/f is chosen whose specifications are shown in **figure 4**. In this project the reset vector and the exception vector of the nios ii processor are set to the Flash memory and the SRAM memory respectively.

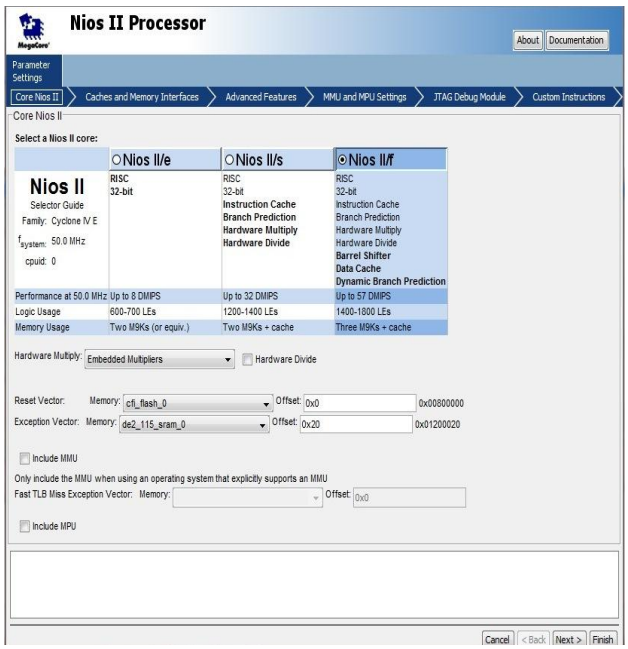

Figure (4) Nios ii Processor Specifications

- JTAG_UART

In this system a JTAG UART is used where the USB Blaster JTAG cable is used to configure the FPGA and also used as a UART device after the FPGA is configured [5].

- Cmos _Controller

This module is not available in the SOPC builder so its HDL module which was developed by [6] is added by the component editor to the SOPC builder. This module is responsible for reading image frames from the SDRAM and giving start or stop image capture signal to the camera.

- DE2_115_SRAM

This component is also not available in the sopc builder so it's HDL which was developed by [7] is added by the component editor to the SOPC builder. This module adds the 2MB DE2-115 SRAM to the system in order to set the exception vector of the nios ii processor to SRAM.

- Output_from_nios

This component is an input/output peripheral with width (1-32) bits. This peripheral is chosen as an output port with 32 bits width. This component is used to pass data between the Nios ii processor and the main Verilog block in a way that is explained in sections 2.2.3.

- Tri_state_bridge

This component is added to connect the Flash memory (also other memories and external components) to the main system bus [8].

- CFI_FLASH

This component adds the flash memory to the system in order to be the program memory device by setting the reset vector to FLASH memory. Each time the system is turned on the program works automatically without requiring operating the Nios ii software package.

Table 2 SOPC of the Colour Recognizing Robot Arm.

Module name	Module Description	Clock	Base address
Cpu	Nios ii processor	50 MHz	0x01400800
Jtag_uart	JTAG UART	50 MHz	0x01401020
Cmos_controller	Cmos_controller	50 MHz	0x01401000
De2_115_sram	De2_115_sram	50 MHz	0x01200000
Output_from_nios	PIO(Parallel I/O)	50 MHz	0x01401010
Tri_state_bridge	Avalon-MM Tristate Bridge	50 MHz	
Cfi_flash	Flash Memory Interface (CFI)	50 MHz	0x00800000

2.2.3 The Image Processing Using the Nios ii Package Provided By Altera.

The image processing is done by writing a program in C language using the nios ii SBT (Software Build Tools) for eclipse which describes the instruction set for the Nios ii processor. The Nios ii processor uses the FLASH memory and the SRAM as its program memories. The Nios ii processor accesses each peripheral and component in the system through their base addresses [5]. When turning the system on the Nios ii processor gives a start signal to the Cmos_Controller module to make the camera start capturing images and after 61 frames the processor gives a stop signal. In order for the Nios ii to write to or read from a peripheral or component a number of functions are employed for example IORD and IOWR. [6] developed the following functions so that the Nios ii communicates with the Cmos_Controller :

IOWR(base address, CAPTURE_START, 1) (1)

IOWR(base address, CAPTURE_STOP, 1) (2)

IORD(base address, CAPTURE_DATA) (3)

Where

CAPTURE_START = 0x0

CAPTURE_STOP = 0x1

CAPTURE_DATA = 0x2

These functions are added to the image processing program, where when eqn(1) and eqn(2) are executed the camera starts capturing and then stops after 61 frames, knowing that the base address is (0x01401000) from Table (2). The third function denoted by eqn(3) reads the image pixel by pixel so eqn(3) is executed as many times as the pixels and then all the pixels are stored in an array knowing that each pixel has 30 bits. The stored image is in RGB form where each pixel of the image has 30 bits (10 for R (RED), 10 for G (GREEN), and 10 for B (BLUE)).

The flow chart shown in **figure 5** shows the complete image processing steps. The image is converted from RGB to HSV colour model, where the HSV color model describes the color and brightness component respectively. For image processing issues the image is converted from RGB to HSV. The HSV colour model defines a colour space in terms of three constituent components [9]:

- Hue (H) is the colour type (such as red, magenta, blue, cyan, green or yellow). Hue ranges from 0-360 deg.
- Saturation (S) refers to the intensity of specific Hue. Saturation ranges are from 0 to 100%.
- Value (V) refers to the brightness of the colour. Value ranges are from 0-100%.

In this work Saturation and Value are chosen to be from 0-1.

Before converting RGB to HSV model it is required to normalize the pixel values which is done by dividing every red, green and blue values in a pixel by 255. The HSV transform function is shown in equations 4, 5, and 6 as follow [10]:

$$H = \begin{cases} 60 * [\left(\frac{G-B}{\alpha}\right) + 6\,] & if\ MAX = R \\ 60 * [\left(\frac{B-R}{\alpha}\right) + 2\,] & if\ MAX = G \\ 60 * [\left(\frac{R-G}{\alpha}\right) + 4] & if\ MAX = B \\ not\ defined & if\ MAX = 0 \end{cases} \quad (4)$$

$$S = \begin{cases} \frac{\alpha}{MAX} & if\ MAX \neq 0 \\ 0 & if\ MAX = 0 \end{cases} \quad (5)$$

$$V = MAX \quad (6)$$

Where $\alpha = (MAX - MIN),\ MAX = max(R, G, B),\ MIN = min(R, G, B)$

The resultant H, S and V values are manipulated with until the image read by the Nios ii processor is as accurate as possible then each pixel is assigned a colour class.

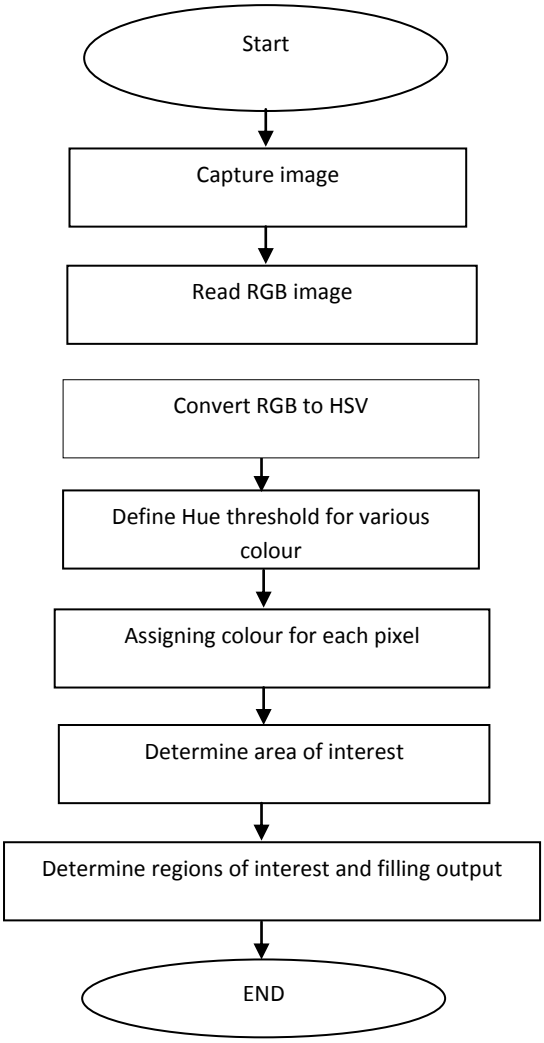

Figure 5. The Image Processing Steps Flowchart

The surface to be recognized should be organized in rows and columns with identical cells as shown in **figure 6**. The surface is marked with a starting mark and an ending mark which is chosen to be small green squares located at the top left and the bottom right. So the C program or the image processing program will search for the starting mark and the ending mark to define the area of interest. Then the program will find the colour of each region on the defined area and record that colour in an output register (peripheral) which is known from section 2.2.2 as the output_from_nios. This output has 32 bits and each bit will describe the colour of a region, for example if the regions have three colours then the first 9 bits of the output_from_nios describe the first colour and the second 9 bits describe the second colour and the third 9 bits describe the third colour so the number of regions will be 9. Each bit in each of the nine bits of a specific colour will describe a region. For example if the first colour is chosen to be recognized then the first 9 bits of the output_from_nios will be 1 if the region is that colour and 0 if not. These regions may have any colour but cannot exceed three colours, if the regions have two colours then there will be 16 regions.

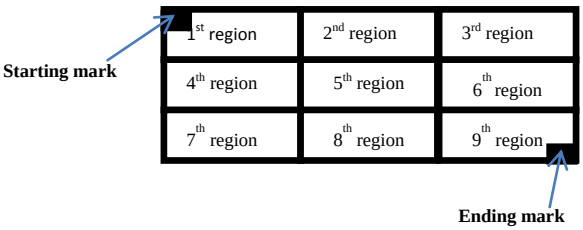

Figure 6. The Surface Organized as Rows and Columns

The flowchart in **figure 7** shows how the Output_from_nios will be filled, where:
Output_from_nios (O) = 0x00000000.
M1 = 0x00000001.
M2 = 0x00000200.
M3 = 0x00040000.
M1, M2 and M3 represent masking values.

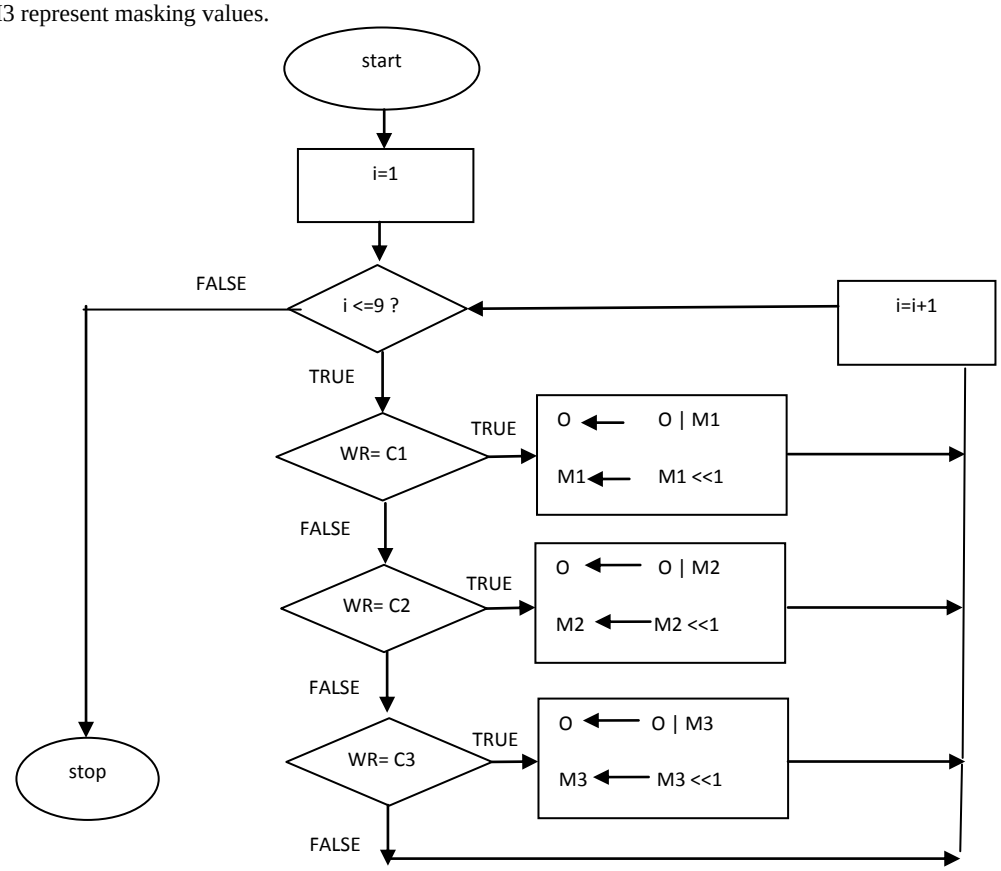

Figure (7) How the Output_from_nios is filled

WR means wanted region.
C1, C2, and C3 mean first, second and third colour respectively.
i counts from 1 to 9 which represent the number of regions.
So the program will detect each region if it is as the wanted colour region (this program can detect three, two or one colour as regions of interest) then the output will be 1 and if not it will be 0.
For example if the regions are as in **figure 8** then the output_from_nios is 0x00821495 in HEX or 0000 0000 1000 0010 0001 0100 1001 0101 in binary. The first nine LSB bits describe the first colour (RED) (0 1001 0101) and the second nine LSB bits describe the second colour (BLUE) (10 0001 010) and the third nine LSB bits describe the third colour (YELLOW) (000 1000 00), the remaining bits are ignored.

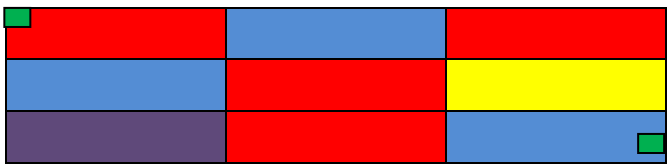

Figure 8. An Example of a Coloured Surface to be recognized

2.2.4 The robot movement part (servo motors moving the joints and D.C motor to move the wheels).

The robot movement has two parts the vertical movement and the horizontal movement. The vertical movement is done by the joints of the robot arm and the horizontal movement is done by the wheels. The joints of the robot arm are controlled by control signals which are provided by the FPGA by programming them using Verilog HDL. These HDL modules are added to the colour recognizing robot arm main HDL module that contains the camera module, the sopc, and the robot movement part.

The horizontal movement is done by the wheels, the D.C motor and the motor drive circuit which are all controlled by 3 signals ENA, IN1, and IN2 (mentioned in section 2.1.4). The pulse waveform applied to ENA and the inputs that are applied to IN1 and IN2 are provided by the FPGA by programming them using Verilog HDL. This HDL module is added to the colour recognizing robot arm main HDL module that contains the camera module, the sopc, and the robot movement part.

Now to achieve a specific movement, pulse waveforms are applied to all joints and a pulse waveform and input signals are applied to the motor drive circuit in order to make the wheels move forward or backward or stop. Each specific movement is programmed in the main HDL block.

3. IMPLEMENTATION OF THE PROPOSED SYSTEM AND RESULTS

Using the Quartus ii version 11.1 package, the main Verilog HDL program is downloaded on the DE2-115 EP4CE115F29C7 FPGA. Table (3) shows the compilation result and resource usage. Then the Nios ii Software Build Tools SBT for eclipse version 11.1 is operated, and using the flash programmer the flash memory is programmed with the C program for the Nios ii processer. This C program is responsible for the Image processing part that controls the image capture, reading image frames from the SDRAM and analyzing the image to obtain specific information. Now the camera starts capturing images and then stops after 61 frames in order to get a clear image. The nios ii package is operated only once (can be edited) because after programming the Flash memory there is no need to operate the nios ii package each time. Now that the image is read from the SDRAM and processed, the desired colour to be recognized can be chosen from the three push buttons located on the FPGA kit. After choosing the desired colour the robot moves to that colour position and collects the object placed there or places an object in that same colour position.

For example if in a warehouse or store there are coloured shelves organized in rows and columns, which all of the shelves have the same height and the same width as shown in **figure 9**. It is required to organize or pick objects in or from the shelves according to the shelves colour. First, the camera captures the image of the shelves and the nios ii processor detects the colours of the shelves. An output (output_from_nios) will be passed from nios to the main Verilog module; this output describes the colours of all regions (shelves). By pressing a KEY (push button) on the FPGA then a specific colour is chosen, in this prototype KEY1 is for choosing the colour RED (it can be any other colour). The robot will move to those chosen positions, the wheels move the robot horizontally and the joints move the robot vertically and the grip of the arm catches the objects.

The robot is placed at a starting position marked X in **figure 9**, it moves from the starting position to any region required and returns back to the starting position. The robot then rotates away from the shelves by moving the fifth joint to drop or to collect an object to or from the desired location. The movement is done by knowing in advance the exact dimensions of the entire shelves which means the maximum height and the minimum height (which is limited by the arm itself) and the entire width of the shelves. By knowing the dimensions of the shelves and all of the shelves are identical, all of the movements are programmed in Verilog HDL and according to the output_from_nios the robot goes to a specific region.

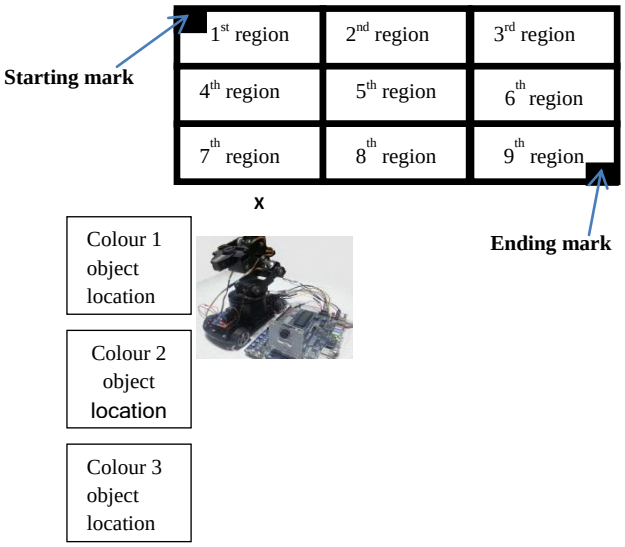

Figure 9. An Actual System with regions as rows and coloumns

Each movement requires 5 pwm (pulse width modulated) signals to control the 5 joints of the robot arm in order to move the robot vertically and to hold or release an object. All regions in the same row have the same pwm signals but when the row changes the pwm signals also change. Also each movement requires 3 signals to control the wheels (the D.C motor) which are 1 pwm signal for the speed and two signals to make the wheels move forward or backward or stop. To control the wheels or the horizontal movement, the speed of the motor is made constant at full speed (or any chosen speed) and knowing in advance the distance between any region and the starting position then the time it requires moving from the starting position to any region is known. This time is used to make the wheels reach the desired region and then stop for an enough time for the object to be collected or placed and then moves backward to the starting position and stops again.

The system is tested on different surfaces with different colours, they all have been recognized and the robot moves to the desired regions at different light intensities. The number of regions depends on the number of colours because the output_from_nios has 32 bits (any I/O peripheral cannot exceed 32 bits). These regions may have any colour but cannot exceed three colours, so if the regions have three colours then there will be only 9 regions and if the regions have two colours then there will be 16 regions.

To increase the number of regions it is proposed to divide all regions into groups of 9 regions (3 rows X 3 columns) each 9 regions has a starting mark and an ending mark and a starting position. Each nine regions are placed next to each other, when a specific colour is chosen then the robot goes to the first 9 regions collects or places objects from or to the desired colour and then goes to the starting position of the next 9 regions and does the same if the desired colour is available and then moves to the other 9 and so on. At each new starting position the camera captures the image of the new 9 regions by resetting the entire system automatically.

Table 3 The compilation result and resource usage.

Flow Status	Successful - Fri Nov 26 03:06:19 2015
Quartus II 32-bit Version	11.1 Build Web Edition
Revision Name	Colour_Recognizing_Robot
Top-level Entity Name	Colour_Recognizing_Robot
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	5,653 / 114,480 (5 %)
Total combinational functions	4,738 / 114,480 (4 %)
Dedicated logic registers	3,219 / 114,480 (3 %)
Total registers	3269
Total pins	425 / 529 (80 %)
Total virtual pins	0
Total memory bits	126,392 / 3,981,312 (3 %)
Embedded Multiplier 9-bit elements	4 / 532 (< 1 %)
Total PLLs	1 / 4 (25 %)

4. CONCLUSIONS AND DISCUSSION

In this paper a colour recognizing robot arm is designed and implemented using an FPGA and a CMOS camera. This system detects the colour of a surface and moves a robot arm accordingly. The surface is organized in rows and columns forming regions that are identical, the surface is bounded by a starting mark and an ending mark. In this prototype the starting mark and the ending mark are chosen to be small green squares so the surface must not have the colour green in its regions. The dimensions of the surface should be known and the robot is placed at a starting position. This system can recognize any colour (but cannot exceed 3 colours) and at any light intensity except at darkness. This system cannot be compared with the systems mentioned in the related work because they differ from each other by the objective, where the related work systems sort objects according to the objects colour while this system organizes any objects (whatever their colour) according to the colour of the storing area. This system can be very useful in warehouses and stores to organize objects easily according to the colour of the storing area.

5. REFERENCES

[1] Joy, A., 2014 *Object Sorting Robotic Arm Based on Colour Sensing* International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (ISSN (Online): 2278 – 8875) Vol. 3, Issue 3, pp 7741-7746.

[2] Chandramohan, A., Murthy, K. K. R., Sowmya, G., Prasad, S. P. A., Krishna, V. V., and Peeyush, K. P., 2014 Cost Effective Object Recognition and Sorting Robot Using Embedded Image Processing Techniques, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-3, Issue-11,pp-29-32.

[3] Devalla, V., Singh, R., Mondal, A. K., and Kaundal, V., 2012, Design and Development of Object Recognition and Sorting Robot for Material Handling in Packaging and Logistic Industries, International Journal of Science and Advanced Technology (ISSN 2221-8386) Volume 2 No 9 , pp 30-35.

[4] Altera corporation 2009 DE2_115 user manual.

[5] Hamblen, J. O., Hall, T. S., and Furman, M. D., 2005, Rapid Prototyping of Digital Systems: Quartus® II Edition Springer Science & Business Media, Computers - 371 pages.

[6] Cnblogs 2008 How to read CMOS from the Nios II's image on the SDRAM?, URL: http://www.cnblogs.com/oomusou/archive/2008/08/31/de2_70_cmos_controller.html

[7] Cnblogs 2010 How to customize the interface IP SRAM of Avalon, for Nios II use URL: http://www.cnblogs.com/yuphone/archive/2010/09/27/1836519.html

[8] Hamblen, J. O., Hall, T. S., and Furman, M. D., 2007, Rapid Prototyping of Digital Systems: SOPC Edition Springer Science & Business Media, - Technology & Engineering - 411 pages.

[9] Georgieva, L, Dimitrova, T, and Angelov, N., 2005 RGB and HSV color models in color identification of digital traumas images. International conference on computer systems and technologies. Bulgaria, 12-1-6.

[10] Chen, W., Shi, Y. Q. and Xuan, G. 2007 Identifying computer graphics using HSV color model and statistical moments of characteristic functions, Proc. IEEE Int. Conf. Multimedia and Expo (ICME), pp.1123 -1126.