YouTube link:

https://www.youtube.com/watch?v=4UEmLFAkHLs&feature=youtu.be

İdil Karagöz

21902050

27.12.2020

THE CORONA ASSISTANT

1.Abstract

In this Project, I have aimed to construct a device such that it can be helpful during these times of Corona Virus and it can have further applications. There is a camera to determine the current total number of people in the room, an ultrasonic distance sensor to find the max number of people can be fitted in the room. The camera transfers the footage through UART of Basys3, and the aimed resolution of these transferred footage is 640x480. The footage was stored into the BRAM and was transferred bit by bit. The photos were grayscaled to avoid the memory issues that can arise from the Basys3. The distance module can be controlled using three switches: on for resetting, second for choosing between width and depth and the third one is for starting the process of getting the data. First were written the camera code and then transferred data from UART to be processed. Images were constructed by phyton and the total number of people were displayed. The device can find out the max number of people can be in the room and it also can tell how many people is in the room at current situation. The distance sensor works with a signal to trigger it and echoes back a signal which's pulse width is the distance measured. To accurately display the distance the measured there were some algebraic manipulations needed. The camera is OX7670 which is initially an Ardunio camera, but the application was possible. While the maximum number of people that can be in the room was calculating social distancing of 150 cm was used and the code was constructed in this way. Phyton does the operation of finding how manty people is in the room and gets the data from the USB port. The maximum number of people is determined using the social distancing value that should be provided in these times, so it is applicable to the times of pandemic, this is where it gets the name.

2.Design Specification Plan

While I was searching for sensors to use in this project, I thought about using radar and ultrasound sensors, but I have used the distance module JSN-SR04T because its range was suitable for rooms, it can measure from 25cm to 4,5 half meter which makes it suitable for smaller room, rooms in houses rather than crowded areas and bigger spaces. It is a waterproof module so it can be used outside in rainy conditions. The operating temperature of the module is 10 degrees Celsius to 80 degrees

Celsius which is suitable for rainy days and especially for indoor uses. Because of the limitations due to distance module, the max number of people that can be in the room would not be more than 9, so to display the value, I have used seven segment display. The question that bothered me about the camera setup was that how I can get the current frame from the camera and transfer it to the computer, and I have learned about the UART connection of Basys3. I have used UART for transferring the data obtained from the camera to the computer and the Baud Rate of the transformation is 115200 which transfers the 640x480 resolution frame that were obtained to the computer in 20 seconds. The camera is a bit zoomed in to get see the whole room because of the size limitations that I have because of the limitations of the distance module. The camera in not the best quality so the quality of the photos is not the best. The need of memory devices could have been solved if I used the OV7670-FIFO camera which can do a lot of operations in itself, has a white corrector, has its own clock and memory so it's easier to use and the signal that is coming from the camera is not raw as it was while in the OV7670, the signal coming from the camera I have putted it through some operations, converted it to grayscale and have implemented an BRAM to store the Frames. It is an ergonomic device due to the sizes of the sensors used and it is not hard to store or place in a room. It should be placed close to the edges of the room so the camera has max angle to observe the room and covers more area and the distance module can have more accurate calculation of the distance so there will be less errors involved. One of the things to consider is the overlapping of people, which was not a solvable issue, if there is two people standing behind each other there is no way to prevent the errors. The width and depth calculations of the room should be done separately and manually. The dedicated switch must be opened to get the other component of the calculation. I have used a breadboard for connecting the distance module to the Basys3 and I have used jumpers both male and female. In the initial plan, I thought about implementing a temperature sensor but the problem with that was I was not able to find a suitable temperature sensor so while executing I left that out.

3.Design Methodology

The first step of the project was to check how the sensors could have been implemented. I have checked the datasheets of the datasheets of both OV7670 camera and the JSN-SR04T distance sensor and learned how they work. The camera had 18 pins which I have connected to the JB and JC pins at most comfortable way. In my design there are modules for both camera and distance sensor, and they are linked in the top module of the design. The modules in the design are: Distance module, Clocking module, UART, Frame Buffer, Button Bouncer and Capture. The first module to be explained is the distance module.

3.A Distance Module

This module uses three switches and the 50 MHz clock that have been generated in the clocking module that I have used in the project. It takes input from the JA connection 2, sends a trigger signal

using JA connection 1. The three switches used are for activator, which activates the module and lets it do the calculation, selector which selects if we are measuring the width and the depth of the room. After successfully getting the two values of distance, the max number of people that possibly be in the room can be seen on the seven-segment display of the Basys3. All of those operations are done in the same module, “distancecalculator”. In this module, I have used an FSM for the generation of the trigger signal. In the state “trigsignal” the Basys3 generates a signal which has a width of 50 clock period. 50 clock period is generated using a counter which counts till 50 and makes the signal 0 again. The next state is the reading f the echo signal. The echo signal’s pulse width is the distance value that is needed for the calculation. Reading of the pulse signal is done in a similar fashion with the trigger signal generator, it uses a counter to count how many clock cycles have passed till the echo signal received from the distance sensor became 0. Then the value is multiplied by 2 and divided by 5000 to get the cm value of the given signal. Then the signal is stored using a register in the design and the states are going back to the ready state where both of the counters are set to zero and the state stays as ready state as long as the activator switch is not high. When activator switch is on, the state moves on to the trig state. The selector switch works at the “echoread” state, chooses which one of the registers we will have the value of the calculated distance stored in. the last switch that has been used in the design is for thee reset which sets the state to ready state and refreshes the system. When both values are successfully stored in the registers, the max number of people that can be in the room is defined by dividing the values that have been found by 150 cm which is the propriate social distancing value and multiplied with each other. After this algebraic manipulation, we add and additional zero to the calculation because if there is some distance that have been calculated, there can be one person guarantied. Then the received value is read and seven segment logic vector is determined with an if statement. The first digit of the seven segment is activated, and it is enough due to the limitations of the distance sensor mentioned as before. In figure 1 there can be seen the RTL schematic of the module and the inputs and outputs can be seen of the module. The module uses an 50MHz clock because the 100MHz clock of the Basys3 goes into a buffer in the clocking module and there generated two clocks with 25MHz and 50MHz frequencies. The max number of people value is also stored in a register so it will be displayed until the device gets a new value for the “maxnbp” signal.

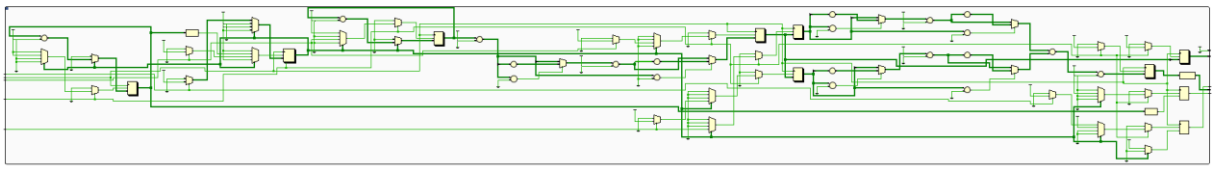


Figure 1) the RTL schematic of the Distance Module

The state register is determining the state that the FSM will be.

3.A) Clocking Module

In this module the 100MHz clock is used to generate two different clocks, 50MHz and 25MHz. the 50MHz clock signal is used by Frame buffer, controller and the UART module where the 25 MHz Clock is used by button debounce module. The writing and reading of addresses are done by 50MHz clock because they operate in the top module. In the clocking module, the 100 MHz clock goes into a buffer and there generated two different clocks. Since this is a different technique that I was not aware of, the code was taken from another source and applied to my needs.

3.B) Capture Module

In this module, the current frame is captured so it can be transferred to the frame buffer and be stored there. It takes 8-bit data, horizontal and vertical sync from the camera, the pclk generated by the camera and the signals from the resolution buttons. The pclk from the camera decreases the frame rate, so it is suitable for the transfer. It does the connection between horizontal and vertical sync, reading the lines and columns. Also, in this module, I have used latched signals so the reading and writing frames would be accurate and the data can be transferred to the computer using UART. When there used no latches, the data comes in a disturbed matter and no picture can be obtained but when the data is latched the transfer becomes valid. The capture is very crucial pint for my design because without the captured data, the transferring of the frames is not possible. The Capture module had a problem that I was not able to solve where the captured image was 320*240 rather than what e have aimed for, 640*480. The capturing command comes from a switch which will be turned on by the user. Here, we also generate the writing address for the frame buffer and the data that will go to the frame buffer is also obtained here.

3.C) Frame Buffer

In the frame buffer, the data from the capture module is stored, which is 12 bits RGB data. The core that dos the operation was generated by putting in needed values and attributes to the IP core generator in Vivado, so the code was constructed automatically. This core takes an address data but cannot take all the 19 bits produced by the capture module, so it takes only 17 bits from the module. It reads and writes the data and uses two different clocks while doing so, it takes two addresses one data in and data out, it also uses two different clocks. The 50MHz clock is used while reading the data and the 25MHz clock is used while the writing of the data is done. It takes the frame and sends it into the RGB to Grayscale module which converts the photo to grayscale so it can be send faster and more efficiently. There reading and writing of the frame buffer is synched with the UART with an additional

code in the top module that specifies the timing between both modules. The addresses for reading are generated using the writing address so the data can be transformed with a correct order.

3.D) RGB to Grayscale Module

In this module, the data that had been obtained and converted into 12-bit RGB is and the four bits indicating the red, green and blue are made 8-bit data to be able to convert the data to 8-bit grayscale. To achieve the conversion these 8-bit vectors which indicating the three colors red, green and blue were multiplied with the respected constants and then added up to be the grayscale output data which is 8 bits. The values that had been used to multiply the values are predetermined and are determined by how human eye can perceive colour. The obtained data from the calculations are initially 16 bits but the most significant 8 bits are used to determine the grayscale data.

3.E) Controller Module

In this module, there are two other modules are included which are i2C and Register modules. In the i2c, there are specifications of the camera, where SIOD is the data to the camera and the SIOC is the input clock that should be between 50 MHz and 10 MHz as specified on the datasheet of the OV7670 camera. The SIOD is in out where we receive a data from the camera as well. In registers module there are configurations that had been specified in the datasheet applied. In the datasheet there can be seen register values for different cases, and they are applied there which configurates the camera and puts an output to a led that the configuration had been completed successfully. In this module the FIFO is specified. The configuration and the clocking of the camera is done here. There is also an input signal of send, when 1, it refreshes the camera. This is coming from a button so the timing will be better and to obtain accurate results there is a button bouncer used to successfully output 1 because the Basys3 does not have a built-in button bouncer.

3.F) Button Debouncer Module

This module should eb included because Basys3 does not have a button debouncer built in. This debouncer takes in the 25MHz clock signal and the button input then outputs a resend signal which will be used by the controller module. When a button is pressed, it does not stay as high as switches, so it goes back to its original state. In this module there is a counter which sets the output of the button debouncer 1 when the highest value of it reached which means the button should be pressed long enough to avoid unnecessary changes. When button is pressed long enough the output resend becomes 1 and goes to the controller else the output of the bouncer is 0.

3.G) UART

In the UART module, the transferring of the data obtained by the RGB to Grayscale module is used and transferred serially using USB to the computer. For this purpose, first thing that is needed is a

BAUD rate clock that will transfer the data using the speed that max can be used for the receiving of the data. The other thing that we need is an index counter that will make sure the data is send in the correct order, so the transferring of the data is successful. We use the 50 MHz that had been generated by the clocking module and arrange the timer such that the baud rate will be 115200. When I the state, send_ready, the baud clock counter becomes zero and the bit index becomes zero, when in the load state the bit index and the clock is starting to count. If the ready indicator is one the data is loaded with the starting and the stopping bit. Then in the send_serial state the data is sent to the computer using USB connection.

3.H) Top Module

In this module the components are defined, and the port mapping of the signals are done. The modules: clocking, distance calculator, frame buffer, capture, controller, button debouncer and UART are written as components declaring the inputs and outputs of each module. The overall in-out connections that have been used as pins of camera, clock, LEDs, and switches are done in the entity of the Top Module. The instantiation of the signals and inputs-outputs are done by port mapping the given signals to the defined components. Also, at the end oof the top module there included a code that generates the reading address for the frame buffer using the writing address that had been provided by the capture module. There is also instantiation of the individual signals red, blue and green that goes into the RGB module, the size selection buttons that goes into the capture is done I the top module. There are lots of signals and in and out definitions in this module.

4.Results

I was able to see the frame that had been transferred to the computer using phyton but, the frames were more than one and not in the wanted resolution of 640*480. I think the error is coming from the capture module, there is something that I have missed there but else other parts of the project are working. The distance module is working well but when the switches are used too often the sensor stops calculating. The arrangement of the distance module is done manually by the user and the photographing part of the camera is done in the same manner. User is using the switches to do the operations needed and the wanted results can be obtained. The design still has few flaws like the frame is not being the wanted resolution and the error that can happen with the distance module. The error in the distance module can be solved using the reset switch and the led indicator on the sensor itself. Because of photos arriving with smaller resolution than anticipated, the human detection code that I have written in the phyton using OpenCV does not work well with the images. Because it detects body, when photo is taken in a closer manner to the objects, the detection system fails.

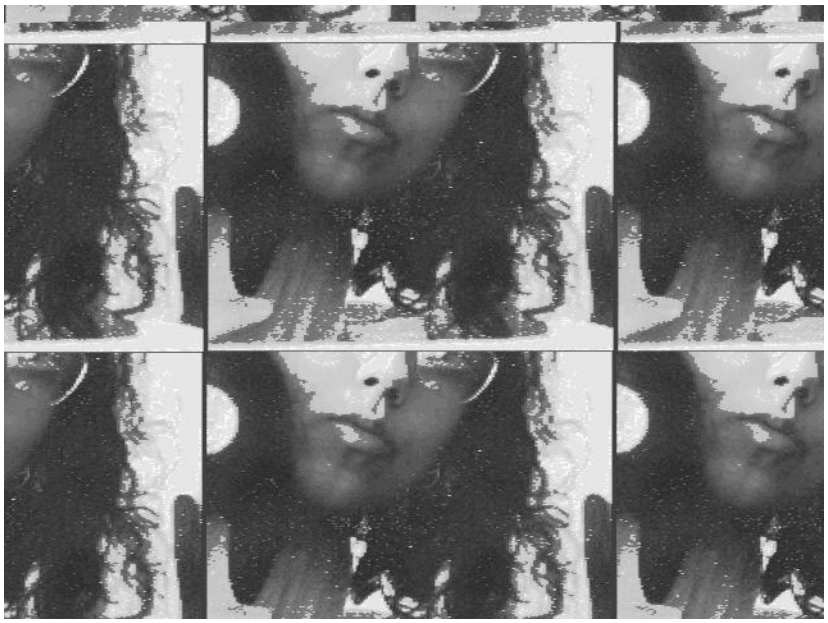
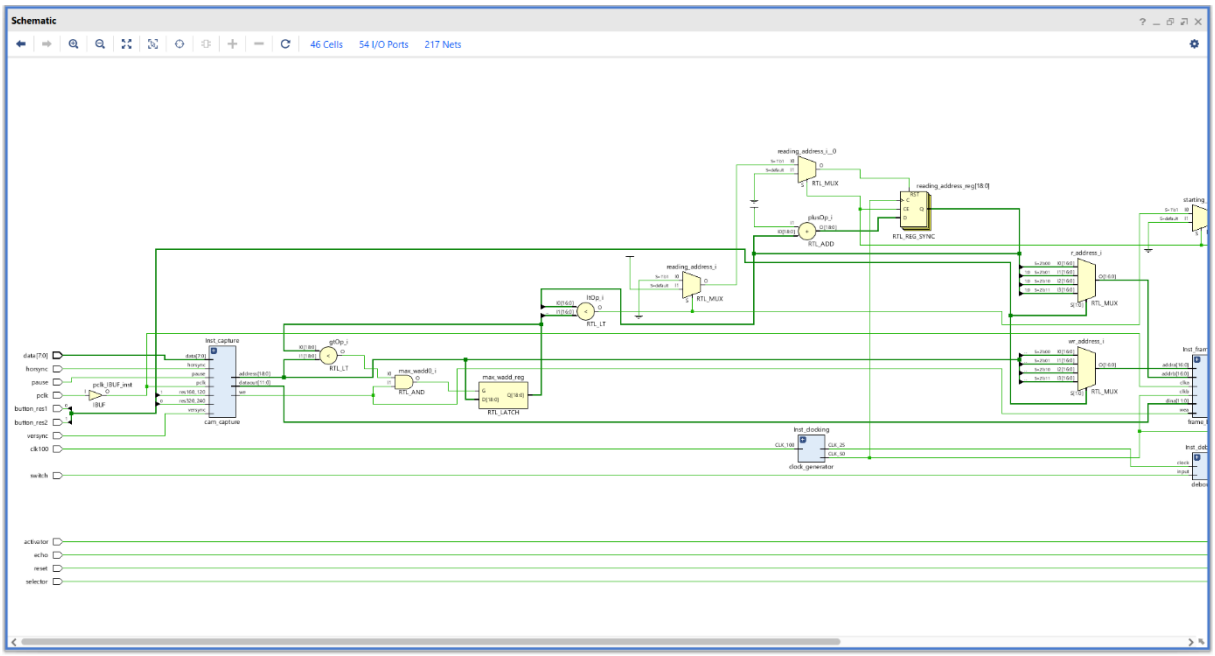
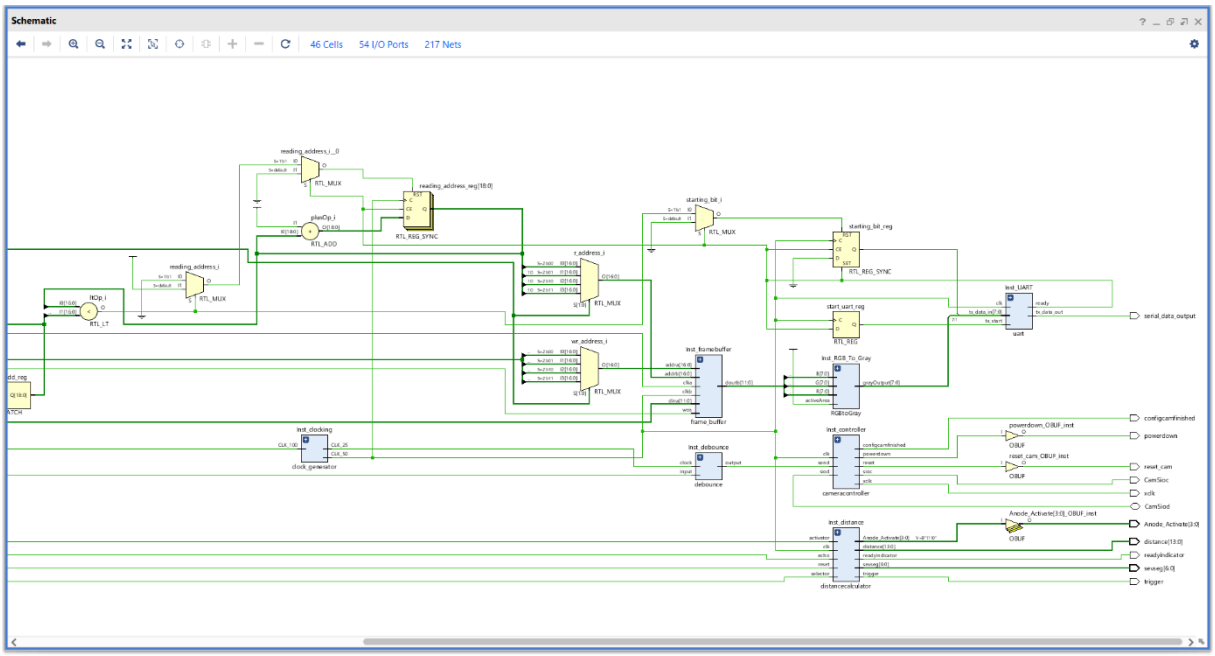


Figure 2) The Data from the Uart signal.

In the picture there can be seen the mentioned problem about the resolution and the frames. When the data is read assumed as 640*480, the repetition of frames can be observed.



In this figure there can be seen the inputs and the capture module of the design. Also, the operations with the addresses include the operators that can be seen in the schematic. There can be seen the address registers and the multiplexers for address determining using the buttons for the resolution change.



There can be seen the modules for the camera and the distance sensor, the inputs and output of each can eb seen using the schematic. Also, here there can be seen the outputs of the design.

Figures 3 and 4) The RTL Schematic of the design

5.Conclusion

I have learned a lot while I tried to prepare this project. The design was not easy to apply due to the complexity of the camera and the issues that I have had that I was not able to do while doing the project. I have learned how to implement a Camera and a distance module to FPGA, how data transformation works. I would have been very happy to see it working better especially the resolution issue that I have faced, but I was not able to solve the issue that I have had while doing the project. I have learned how to use BRAM works and can be generated and how the pulse width of a signal can be read. The Phyton application helped me to recall the knowledge that I had about the topic and learn more about the Phyton and OpenCV. The issue with transferring images, I was not able to solve and find the reason why this error occurs. I have worked on the UART code and the Phyton code, but I was not able to solve the issue. Despite the issues mentioned it was an experience which had thought me a lot about the VHDL and how FPGA's work.

Citations

"Basics of UART Communication," Circuit Basics, 11-Apr-2017. [Online]. Available: http://www.circuitbasics.com/basics-uart-communication/. [Accessed: 29-Dec-2018].

https://www.voti.nl/docs/OV7670.pdf

Subdin, Alexey. UART Interface in VHDL for Basys3 Board. 1.11.2020. [Online]. Available: https://www.hackster.io/alexey-sudbin/uart-interface-in-vhdl-for-basys3-board-eef170#toc-conclusion-11

Phongchit, Nikko. "What is Baud Rate? Why it is important?", 04.02.2016 [Online]. Available: https://www.setra.com/blog/what-is-baud-rate-and-what-cable-length-is-required-1

User Hshallcross, "FPGA + OV7670 = SUPER SLOW UART WEBCAM", 24.05.2017, [Online]. Available: https://hsel.co.uk/2017/fpga-ov7670-super-slow-uart-webcam/

Appendix

Top Module VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Top_level_file is
    Port ( clk100 : in std_logic;
           -- for and from the camera
           versync : in std_logic;
           horsync : in std_logic;
           pclk : in std_logic;
           xclk : out std_logic;
           CamSioc : out std_logic;
           CamSiod : inout std_logic;
           data : in std_logic_vector(7 downto 0);
           reset_cam : out std_logic;
           powerdown : out std_logic;
           configcamfinished: out std_logic;
           button_res1 : in std_logic;
           button_res2 : in std_logic;
           -- for the distance module
           echo : in std_logic;
```

```
        trigger : out std_logic;
        distance : out std_logic_vector(13 downto 0);
        activator : in std_logic;
        selector : in std_logic;
        readyindicator: out std_logic;
        reset : in std_logic;
        sevseg : out std_logic_vector(6 downto 0);
        Anode_Activate: out std_logic_vector(3 downto 0):= "1110";
        -- for the uart
        serial_data_output: out std_logic;
        pause : in std_logic;
        switch : in std_logic
       -- uart_sender : in std_logic
                );
end Top_level_file;

architecture Behavioral of Top_level_file is
--components for different parts of the project
--components for camera modules
component cameracontroller
         port(
            clk : in std_logic;
            siod: inout std_logic;
            sioc : out std_logic;
            powerdown : out std_logic;
            reset : out std_logic;
            xclk : out std_logic;
            configcamfinished: out std_logic;
            send : in std_logic );
end component;

component clock_generator
```

```
    port ( CLK_100          : in     std_logic;
    CLK_50          : out    std_logic;
    CLK_25          : out    std_logic);
            end component;


component cam_capture
    Port(
        pclk : in std_logic;
        res320_240 : in std_logic;
        res160_120 : in std_logic;
        versync : in std_logic;
        horsync : in std_logic;
        data : in std_logic_vector(7 downto 0);
        address : out std_logic_vector(18 downto 0);
        dataout : out std_logic_vector(11 downto 0);
        we : out std_logic;
        pause : in std_logic;
        start_send : out std_logic
        );
end component;

component frame_buffer
        port(
            clka : in std_logic;
            addra : in std_logic_vector(16 downto 0);
            dina : in std_logic_vector(11 downto 0);
            wea : in std_logic_vector(0 downto 0);
            clkb : in std_logic;
            addrb : in std_logic_vector(16 downto 0);
            doutb : out std_logic_vector(11 downto 0)
            );
```

```
end component;


COMPONENT uart is
      Port ( clk           : in  std_logic;
      ready         : out  std_logic;
      tx_start      : in  std_logic;
      tx_data_in    : in  std_logic_vector (7 downto 0);
      tx_data_out   : out std_logic
      );
END COMPONENT;


-- distance calculator components
Component distancecalculator is
   Port (
        echo : in std_logic;
        trigger : out std_logic;
        distance : out std_logic_vector(13 downto 0);
        clk : in std_logic;
        activator : in std_logic;
        selector : in std_logic;
        readyindicator: out std_logic;
        reset : in std_logic;
        sevseg : out std_logic_vector(6 downto 0);
        Anode_Activate: out std_logic_vector(3 downto 0):= "1110"
         );
 end component;


component RGBtoGray is
```

```
    port (    R : in STD_LOGIC_VECTOR (7 downto 0);
              G : in STD_LOGIC_VECTOR (7 downto 0);
              B : in STD_LOGIC_VECTOR (7 downto 0);
              grayOutput : out STD_LOGIC_VECTOR (7 downto 0);
              activeArea : in STD_LOGIC );
end component;

component debounce is
    port ( clock : in std_logic;
           input : in std_logic;
           output : out std_logic
           );
end component;

--signals for various components
-- signals for camera
signal camera_clk : std_logic;
signal clk_25 : std_logic;
signal wren : std_logic_vector(0 downto 0);
signal resend : std_logic;
signal writing_address, reading_address : std_logic_vector(18 downto 0);
signal wr_address, r_address : std_logic_vector(16 downto 0);
signal writing_data, reading_data : std_logic_vector(11 downto 0);
signal grayoutput : std_logic_vector(7 downto 0);
signal builtin_clk : std_logic;
signal camerared, camerablue, cameragreen : std_logic_vector(7 downto 0);
signal uartdatain : std_logic_vector(7 downto 0);
signal activeArea: std_logic;
signal starting_bit : std_logic;
signal max_wadd, min_wadd : std_logic_vector(18 downto 0);
signal res320_240, res160_120 : std_logic;
signal uartSignal, pauseSwitch : std_logic;
```

```
signal select_size : std_logic_vector(1 downto 0);
signal start_uart : std_logic;
signal ready : std_logic;
signal pause_camera : std_logic;


begin
    activeArea <= '1';
    builtin_clk <= clk100;
    res320_240 <= button_res1;
    res160_120 <= button_res2;
    uartdatain <= grayoutput(7 downto 1) & starting_bit;
    cameraRed <= reading_data(11 downto 8) & reading_data(11 downto 8);
    cameraGreen <= reading_data(7 downto 4)& reading_data(7 downto 4);
    cameraBlue <= reading_data(3 downto 0)& reading_data(3 downto 0) ;


Inst_distance : distancecalculator
                    Port Map( echo => echo,
                              trigger => trigger,
                              distance => distance,
                              clk => camera_clk,
                              activator => activator,
                              selector => selector,
                              readyindicator => readyindicator,
                              reset => reset,
                              sevseg => sevseg,
                              Anode_Activate => Anode_Activate
                              );

Inst_capture : cam_capture
    port Map (  pclk => pclk,
```

```
            res320_240 => res320_240,
            res160_120 => res160_120,
            versync => versync,
            horsync => horsync,
            data => data,
            address => writing_address,
            dataout => writing_data,
            we => wren(0),
            pause => pauseSwitch,
            start_send => pause_camera
            );
select_size <= button_res2 & button_res1;

 with select_size select
    wr_address <= writing_address(18 downto 2) when "00",
    writing_address(16 downto 0) when "01",
    writing_address(16 downto 0) when "10",
    writing_address(16 downto 0) when "11";

  with select_size select
    r_address <= reading_address(18 downto 2) when "00",
    reading_address(16 downto 0) when "01",
    reading_address(16 downto 0) when "10",
    reading_address(16 downto 0) when "11";

Inst_framebuffer: frame_buffer
    port map(  clka => pclk,
            addra => wr_address,
            dina => writing_data,
            wea => wren,
            clkb => camera_clk,
            addrb => r_address,
```

```
                    doutb=> reading_data
                    );

Inst_controller : cameracontroller
   Port Map ( clk => camera_clk,
              siod => Camsiod,
              sioc => Camsioc,
              powerdown => powerdown,
              reset => reset_cam,
              xclk => xclk,
              configcamfinished => configcamfinished,
              send => resend
               );



 Inst_clocking : clock_generator
    port map
    (
     CLK_100 => CLK100,
     CLK_50 => camera_clk,
     CLK_25 => clk_25);

 Inst_RGB_To_Gray : RGBtoGray
   port map (R => cameraRed,
             G => cameraGreen,
             B => cameraBlue,
             grayOutput => grayoutput,
             activeArea =>activeArea );

Inst_debounce : debounce
   port map ( clock => clk_25,
```

```
            input => switch,
            output => resend
            );

Inst_UART: uart
   Port Map ( clk => camera_clk,
                        ready => ready,
                        tx_start => start_uart,
                        tx_data_in => uartdatain,
                        tx_data_out => serial_data_output
                        );

--generating adresses
process(camera_clk)
begin
   if rising_edge( camera_clk) then
      if ready = '1' then
         start_uart <= '1';
          if reading_address(18 downto 2) < max_wadd(18 downto 2) then
            reading_address <= reading_address + 1 ;
            starting_bit <= '0';
          else
             starting_bit <= '1';
             reading_address <= (others => '0');
          end if;
      else
        start_uart <= '0';
      end if;
   end if;
end process;
pauseSwitch <= pause;
process( pclk)
```

```
begin
    if writing_address > max_wadd AND wren(0) = '1' then
        max_wadd <= writing_address;
    end if;
   if writing_address < min_wadd AND wren(0) = '1' then
        min_wadd <= writing_address;
    end if;
end process;


end Behavioral;
```

Clocking VHDL Code:

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

library unisim;
use unisim.vcomponents.all;

entity clocking is
port
 (-- Clock in ports
  CLK_100          : in     std_logic;
  -- Clock out ports
  CLK_50          : out    std_logic;
  CLK_25          : out    std_logic
 );
end clocking;

architecture xilinx of clocking is
```

```
    attribute CORE_GENERATION_INFO : string;
    attribute CORE_GENERATION_INFO of xilinx : architecture is "clocking,clk_wiz_v3_6,{component_name=clocking,use_phase_alignment=true,use_min_o_jitter=false,use_max_i_jitter=false,use_dyn_phase_shift=false,use_inclk_switchover=false,use_dyn_reconfig=false,feedback_source=FDBK_AUTO,primtype_sel=MMCM_ADV,num_out_clk=2,clkin1_period=10.000,clkin2_period=10.000,use_power_down=false,use_reset=false,use_locked=false,use_inclk_stopped=false,use_status=false,use_freeze=false,use_clk_valid=false,feedback_type=SINGLE,clock_mgr_type=MANUAL,manual_override=false}";
    -- Input clock buffering / unused connectors
    signal clkin1      : std_logic;
    -- Output clock buffering / unused connectors
    signal clkfbout         : std_logic;
    signal clkfbout_buf     : std_logic;
    signal clkfboutb_unused : std_logic;
    signal clkout0          : std_logic;
    signal clkout0b_unused  : std_logic;
    signal clkout1          : std_logic;
    signal clkout1b_unused  : std_logic;
    signal clkout2_unused   : std_logic;
    signal clkout2b_unused  : std_logic;
    signal clkout3_unused   : std_logic;
    signal clkout3b_unused  : std_logic;
    signal clkout4_unused   : std_logic;
    signal clkout5_unused   : std_logic;
    signal clkout6_unused   : std_logic;
    -- Dynamic programming unused signals
    signal do_unused        : std_logic_vector(15 downto 0);
    signal drdy_unused      : std_logic;
    -- Dynamic phase shift unused signals
    signal psdone_unused    : std_logic;
    -- Unused status signals
    signal locked_unused    : std_logic;
    signal clkfbstopped_unused : std_logic;
    signal clkinstopped_unused : std_logic;
```

```
begin


 -- Input buffering
 ------------------------------------
 clkin1_buf : IBUFG
 port map
  (O => clkin1,
   I => CLK_100);



 -- Clocking primitive
 ------------------------------------
 -- Instantiation of the MMCM primitive
 --    * Unused inputs are tied off
 --    * Unused outputs are labeled unused
 mmcm_adv_inst : MMCME2_ADV
 generic map
  (BANDWIDTH            => "OPTIMIZED",
   CLKOUT4_CASCADE      => FALSE,
   COMPENSATION         => "ZHOLD",
   STARTUP_WAIT         => FALSE,
   DIVCLK_DIVIDE        => 1,
   CLKFBOUT_MULT_F      => 10.000,
   CLKFBOUT_PHASE       => 0.000,
   CLKFBOUT_USE_FINE_PS => FALSE,
   CLKOUT0_DIVIDE_F     => 20.000,
   CLKOUT0_PHASE        => 0.000,
   CLKOUT0_DUTY_CYCLE   => 0.500,
   CLKOUT0_USE_FINE_PS  => FALSE,
   CLKOUT1_DIVIDE       => 40,
   CLKOUT1_PHASE        => 0.000,
```

```
 CLKOUT1_DUTY_CYCLE   => 0.500,
 CLKOUT1_USE_FINE_PS  => FALSE,
 CLKIN1_PERIOD        => 10.000,
 REF_JITTER1          => 0.010)
port map
 -- Output clocks
(CLKFBOUT            => clkfbout,
 CLKFBOUTB           => clkfboutb_unused,
 CLKOUT0             => clkout0,
 CLKOUT0B            => clkout0b_unused,
 CLKOUT1             => clkout1,
 CLKOUT1B            => clkout1b_unused,
 CLKOUT2             => clkout2_unused,
 CLKOUT2B            => clkout2b_unused,
 CLKOUT3             => clkout3_unused,
 CLKOUT3B            => clkout3b_unused,
 CLKOUT4             => clkout4_unused,
 CLKOUT5             => clkout5_unused,
 CLKOUT6             => clkout6_unused,
 -- Input clock control
 CLKFBIN             => clkfbout_buf,
 CLKIN1              => clkin1,
 CLKIN2              => '0',
 -- Tied to always select the primary input clock
 CLKINSEL            => '1',
 -- Ports for dynamic reconfiguration
 DADDR               => (others => '0'),
 DCLK                => '0',
 DEN                 => '0',
 DI                  => (others => '0'),
 DO                  => do_unused,
 DRDY                => drdy_unused,
```

```
    DWE                 => '0',
    -- Ports for dynamic phase shift
    PSCLK               => '0',
    PSEN                => '0',
    PSINCDEC            => '0',
    PSDONE              => psdone_unused,
    -- Other control and status signals
    LOCKED              => locked_unused,
    CLKINSTOPPED        => clkinstopped_unused,
    CLKFBSTOPPED        => clkfbstopped_unused,
    PWRDWN              => '0',
    RST                 => '0');

  -- Output buffering
  -------------------------------------
  clkf_buf : BUFG
  port map
   (O => clkfbout_buf,
    I => clkfbout);


  clkout1_buf : BUFG
  port map
   (O   => CLK_50,
    I   => clkout0);

  clkout2_buf : BUFG
  port map
   (O   => CLK_25,
    I   => clkout1);

end xilinx;
```

RGB to Gray VHDL Code:

```
library IEEE;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.all;
use ieee.std_logic_arith;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--use work.fixed_generic_pkg_mod.all;
--use IEEE.STD_LOGIC_ARITH.ALL;

entity RGBtoGray is
   Port (    R : in STD_LOGIC_VECTOR (7 downto 0);
             G : in STD_LOGIC_VECTOR (7 downto 0);
             B : in STD_LOGIC_VECTOR (7 downto 0);
          grayOutput : out STD_LOGIC_VECTOR (7 downto 0);
          activeArea : in STD_LOGIC);
end RGBtoGray;
architecture Behavioral of RGBtoGray is

signal Gray : std_logic_vector(15 downto 0);

begin



Gray <= (x"4C"*R) + (x"97"*G) +(x"1C"*B);
grayOutput <= Gray(15 downto 8) when activeArea ='1' else "00000000" ;
end Behavioral;
```

UART VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
entity uart is
   port(tx_start : in std_logic;
```

```
        tx_data_in : in std_logic_vector;
        clk : in std_logic;
        ready : out std_logic;
        tx_data_out : out std_logic
        );
end uart;

architecture Behavioral of uart is

type uartstate is (send_ready, send_serial, load);

constant bit_timer_max : std_logic_vector(13 downto 0) :="00000110110010";
constant bit_index_max : natural := 10;
signal bit_timer : std_logic_vector(13 downto 0) := (others => '0');
signal bit_done : std_logic;
signal bit_index : natural;
signal bit_serial : std_logic := '1';
signal data_serial : std_logic_vector(9 downto 0);
signal tx_state : uartstate := send_ready;

begin
process(clk)
begin
   if rising_edge(clk) then
      case tx_state is
         when send_ready =>
            if (tx_start = '1') then
               tx_state <= load;
            end if;
         when load =>
            tx_state <= send_serial;
         when send_serial =>
```

```
            if bit_done = '1' then
                if bit_index =bit_index_max then
                    tx_state<= send_ready;
                else
                    tx_state <= load;
                end if;
            end if;
        when others =>
            tx_state <= send_ready;
    end case;
  end if;
end process;

process(clk)
begin
  if rising_edge(clk) then
    if tx_state = send_ready then
      bit_timer <=(others => '0');
    else
      if bit_done ='1' then
        bit_timer <= (others => '0');
      else
        bit_timer<= bit_timer + 1;
      end if;
    end if;
  end if;
end process;
bit_done <= '1' when (bit_timer = bit_timer_max) else '0';

process(clk)
begin
  if rising_edge(clk) then
```

```
        if tx_state = send_ready then
            bit_index <= 0;
        elsif tx_state =load then
            bit_index <= bit_index +1;
        end if;
    end if;
end process;

process(clk)
begin
    if rising_edge(clk) then
        if  tx_start = '1' then
            data_serial <= '1' & tx_data_in & '0';
        end if;
    end if;
end process;

process(clk)
begin
    if rising_edge(clk) then
        if tx_state = send_ready then
            bit_serial <= '1';
        elsif tx_state <= load then
            bit_serial<=  data_serial(bit_index);
        end if;
    end if;
end process;

tx_data_out <= bit_serial;
ready <= '1' when (tx_state = send_ready) else '0';
end behavioral;
```

Controller VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity cameracontroller is
        port(
           clk : in std_logic;
           siod: inout std_logic;
           sioc : out std_logic;
           powerdown : out std_logic;
           reset : out std_logic;
           xclk : out std_logic;
           configcamfinished: out std_logic;
           send : in std_logic );
end cameracontroller;

architecture Behavioral of cameracontroller is
        COMPONENT ov7670_registers
        PORT(
                clk      : IN std_logic;
                advance  : IN std_logic;
                command  : OUT std_logic_vector(15 downto 0);
                finished : OUT std_logic;
                send_input : in std_logic
```

```
        );
    END COMPONENT;

    COMPONENT i2c_sender
    PORT(
        clk   : IN std_logic;
        send  : IN std_logic;
        taken : out std_logic;
        id    : IN std_logic_vector(7 downto 0);
        reg   : IN std_logic_vector(7 downto 0);
        value : IN std_logic_vector(7 downto 0);
        siod  : INOUT std_logic;
        sioc  : OUT std_logic
        );
    END COMPONENT;

    signal sys_clk  : std_logic := '0';
    signal command  : std_logic_vector(15 downto 0);
    signal finished : std_logic := '0';
    signal taken    : std_logic := '0';
    signal sending     : std_logic;

    constant camera_address : std_logic_vector(7 downto 0) := x"42";
begin
  configcamfinished <= finished;

    sending <= not finished;
    Inst_i2c_sender: i2c_sender PORT MAP(
        clk   => clk,
        taken => taken,
        siod  => siod,
        sioc  => sioc,
```

```
            send  => sending,
            id    => camera_address,
            reg   => command(15 downto 8),
            value => command(7 downto 0)
        );

        reset <= '1';
        powerdown  <= '0';
        xclk  <= sys_clk;

        Inst_ov7670_registers: ov7670_registers PORT MAP(
            clk      => clk,
            advance  => taken,
            command  => command,
            finished => finished,
            send_input => send
        );

        process(clk)
        begin
            if rising_edge(clk) then
                sys_clk <= not sys_clk;
            end if;
        end process;
end Behavioral;
```

Capture VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;
```

```
entity cam_capture is
    Port(
        pclk : in std_logic;
        res320_240 : in std_logic;
        res160_120 : in std_logic;
        versync : in std_logic;
        horsync : in std_logic;
        data : in std_logic_vector(7 downto 0);
        address : out std_logic_vector(18 downto 0);
        dataout : out std_logic_vector(11 downto 0);
        we : out std_logic;
        pause : in std_logic;
        start_send : out std_logic
        );
end cam_capture;

architecture Behavioral of cam_capture is

   signal d_latch      : std_logic_vector(15 downto 0) := (others => '0');
   signal address_write      : std_logic_vector(18 downto 0) := (others => '0');
   signal line : std_logic_vector(1 downto 0) := (others => '0');
   signal horsync_line : std_logic_vector(6 downto 0) := (others => '0');
   signal we_reg : std_logic := '0';
   signal horsync_value : std_logic := '0';
   signal ltch_versync : std_logic :='0';
   signal ltch_horsync : std_logic := '0';
   signal ltch_data_in : std_logic_vector(7 downto 0) := (others => '0');
   signal current_pause : std_logic := '0';

begin
   start_send <= current_pause;
   address <= address_write;
```

```
  we <= we_reg and not current_pause;
  dataout <= d_latch(15 downto 12) & d_latch(10 downto 7) & d_latch(4 downto 1);



capture_process: process(pclk)
  begin
  if rising_edge(pclk) then
    if we_reg = '1' then
      address_write <= std_logic_vector( unsigned(address_write) +1);
    end if;
    if horsync_value ='0' and ltch_horsync ='1' then
     case line is
        when "00" => line <= "01";
        when "01" => line <= "10";
        when "10" => line <= "11";
        when others => line <= "00";
     end case;
    end if;
    horsync_value <= ltch_horsync;
    if ltch_horsync = '1' then
     d_latch <= d_latch(7 downto 0) & ltch_data_in;
    end if;
    we_reg <= '0';
    if ltch_versync = '1' then
     if pause = '1' then
        current_pause <= '1';
     else
        current_pause <= '0';
     end if;
     address_write <= (others => '0');
     horsync_line <= (others => '0');
```

```
            line <= (others => '0');
          else
            if (res160_120 = '1' and horsync_line(6) ='1') or (res320_240 = '1' and horsync_line(2) ='1')
or
            (res160_120 = '0' and res320_240 ='0' and horsync_line(0) = '1') then
              if res160_120 = '1' then
                if line ="10" then
                  we_reg <= '1';
                end if;
              elsif res320_240 = '1' then
                if line(1) = '1' then
                  we_reg <= '1';
                end if;
              else
                we_reg <= '1';
              end if;
              horsync_line <= (others => '0');
            else
              horsync_line <= horsync_line(horsync_line'high-1 downto 0) & ltch_horsync;
            end if;
          end if;
        end if;
       if falling_edge(pclk) then
        ltch_data_in <= data;
        ltch_horsync <= horsync;
        ltch_versync <= versync;
       end if;
       end process;



end behavioral;
```

Distance Calculator VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;
library std;
--use std.textio.all;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity distancecalculator is
    --generic(ClockFreq : integer:= 100);
    Port (
          echo : in std_logic;
          trigger : out std_logic;
          distance : out std_logic_vector(13 downto 0);
          clk : in std_logic;
          activator : in std_logic;
          selector : in std_logic;
          readyindicator: out std_logic;
          reset : in std_logic;
          sevseg : out std_logic_vector(6 downto 0);
          Anode_Activate: out std_logic_vector(3 downto 0):= "1110"
           );
end distancecalculator;

architecture Behavioral of distancecalculator is
signal counter_echo : integer := 0;
```

```
signal counter_trig : integer :=0;
--signal echo2 : std_logic;
signal maxnbp : std_logic_vector(3 downto 0);
signal var1 : integer:=0;
signal var2 : integer:=0;


signal distancesig : integer;


type distancestates is (trigsignal, echoread, assign, readystate);
signal state : distancestates :=readystate;

begin


process(clk, reset, var1, var2, echo, selector)
begin
    if rising_edge(clk) then
        if reset = '1' then
            state <= readystate;
--            counter_echo <= 0;
--            counter_trig <= 0;
        else

            case state is
              when readystate =>
                  counter_echo <= 0;
                  counter_trig <= 0;
                  if activator ='1' then
                      state <= trigsignal;
```

```
            else
                state <=readystate;
                readyindicator <= '1';
            end if;
        when trigsignal =>
            readyindicator <= '0';
            counter_trig <= counter_trig + 1;
            trigger <= '1';
            if counter_trig = 50  then
                --counter_trig <= 0;
                trigger <= '0';
                state <= echoread;
            end if;
        when echoread =>
            if echo = '1' then
                counter_echo <= counter_echo +1;
            elsif echo = '0' then
                distancesig <= (counter_echo*2)/5000;
                state <= assign;
            end if;
        when assign =>
            if selector = '1' then
                var1 <= distancesig;
            else
                var2 <= distancesig;
            end if;
            state <= readystate;
    end case;
maxnbp <= std_logic_vector(to_unsigned((((var1/150)*(var2/150))+1),4));
distance <= std_logic_vector(to_unsigned(distancesig,14));
end if;
```

```
    end if;


end process;


ssevenseg : process(maxnbp)
begin
case maxnbp is
    when "0000" => sevseg <= "0000001"; -- "0"
    when "0001" => sevseg <= "1001111"; -- "1"
    when "0010" => sevseg <= "0010010"; -- "2"
    when "0011" => sevseg <= "0000110"; -- "3"
    when "0100" => sevseg <= "1001100"; -- "4"
    when "0101" => sevseg <= "0100100"; -- "5"
    when "0110" => sevseg <= "0100000"; -- "6"
    when "0111" => sevseg <= "0001111"; -- "7"
    when "1000" => sevseg <= "0000000"; -- "8"
    when "1001" => sevseg <= "0000100"; -- "9"
    when "1010" => sevseg <= "0000010"; -- a
    when "1011" => sevseg <= "1100000"; -- b
    when "1100" => sevseg <= "0110001"; -- C
    when "1101" => sevseg <= "1000010"; -- d
    when "1110" => sevseg <= "0110000"; -- E
    when "1111" => sevseg <= "0111000"; -- F
    end case;
end process;
```

```
end Behavioral;
```

Button Debouncer VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity debounce is
    Port ( clk : in  STD_LOGIC;
           in_1 : in  STD_LOGIC;
           out_1 : out  STD_LOGIC);
end debounce;

architecture Behavioral of debounce is
           signal c_1 : unsigned(23 downto 0);
begin
           process(clk)
           begin
                    if rising_edge(clk) then
                      if in_1 = '1' then
                        if c_1 = x"FFFFFF" then
                                          out_1 <= '1';
                                else
                                          out_1 <= '0';
                                end if;
                                c_1 <= c_1 +1;
                           else
                                c_1 <= (others => '0');
                                out_1 <= '0';
                           end if;
                    end if;
           end process;
```

```
end Behavioral;
```

Controller Registers VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ov7670_registers is
    Port ( clk      : in  STD_LOGIC;
           advance  : in  STD_LOGIC;
           command  : out  std_logic_vector(15 downto 0);
           finished : out  STD_LOGIC;
           send_input : in std_logic
           );
end ov7670_registers;

architecture Behavioral of ov7670_registers is
           signal Register_cam   : std_logic_vector(15 downto 0);
           signal address : std_logic_vector(7 downto 0) := (others => '0');
begin
           command <= Register_cam;
           with Register_cam select finished  <= '1' when x"FFFF", '0' when others;

           process(clk)
           begin
                    if rising_edge(clk) then
                             if send_input = '1' then
                                      address <= (others => '0');
                             elsif advance = '1' then
                                      address <= std_logic_vector(unsigned(address)+1);
                             end if;

                             case address is
```

```
                    when x"00" => Register_cam <= x"1280";
                    when x"01" => Register_cam <= x"1280";
                    when x"02" => Register_cam <= x"1204";
                    when x"03" => Register_cam <= x"1100";
                    when x"04" => Register_cam <= x"0C00";
                    when x"05" => Register_cam <= x"3E00";

                when x"06" => Register_cam <= x"8C00";
                    when x"07" => Register_cam <= x"0400";
                    when x"08" => Register_cam <= x"4010";
                    when x"09" => Register_cam <= x"3a04";
                    when x"0A" => Register_cam <= x"1438";
                    when x"0B" => Register_cam <= x"4fb3";
                    when x"0C" => Register_cam <= x"50b3";
                    when x"0D" => Register_cam <= x"5100";
                    when x"0E" => Register_cam <= x"523d";
                    when x"0F" => Register_cam <= x"53a7";
                    when x"10" => Register_cam <= x"54e4";
                    when x"11" => Register_cam <= x"589e";
                    when x"12" => Register_cam <= x"3dc0";
                    when x"13" => Register_cam <= x"1100";

                    when x"14" => Register_cam <= x"1711";
                    when x"15" => Register_cam <= x"1861";
                    when x"16" => Register_cam <= x"32A4";

                    when x"17" => Register_cam <= x"1903";
                    when x"18" => Register_cam <= x"1A7b";
                    when x"19" => Register_cam <= x"030a";

when x"1A" => Register_cam <= x"0e61";
when x"1B" => Register_cam <= x"0f4b";
```

```
when x"1C" => Register_cam <= x"1602";
when x"1D" => Register_cam <= x"1e37";

when x"1E" => Register_cam <= x"2102";
when x"1F" => Register_cam <= x"2291";

when x"20" => Register_cam <= x"2907";
when x"21" => Register_cam <= x"330b";

when x"22" => Register_cam <= x"350b";
when x"23" => Register_cam <= x"371d";

when x"24" => Register_cam <= x"3871";
when x"25" => Register_cam <= x"392a";

when x"26" => Register_cam <= x"3c78";
when x"27" => Register_cam <= x"4d40";

when x"28" => Register_cam <= x"4e20";
when x"29" => Register_cam <= x"6900";

when x"2A" => Register_cam <= x"6b4a";
when x"2B" => Register_cam <= x"7410";

when x"2C" => Register_cam <= x"8d4f";
when x"2D" => Register_cam <= x"8e00";

when x"2E" => Register_cam <= x"8f00";
when x"2F" => Register_cam <= x"9000";

when x"30" => Register_cam <= x"9100";
```

```
            when x"31" => Register_cam <= x"9600";

            when x"32" => Register_cam <= x"9a00";
            when x"33" => Register_cam <= x"b084";

            when x"34" => Register_cam <= x"b10c";
            when x"35" => Register_cam <= x"b20e";

            when x"36" => Register_cam <= x"b382";
            when x"37" => Register_cam <= x"b80a";
   --
            when others => Register_cam <= x"ffff";
                        end case;
                end if;
        end process;
end Behavioral;
```

i2C Protocol VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity i2c_sender is
   Port ( clk   : in  STD_LOGIC;
          siod  : inout  STD_LOGIC;
          sioc  : out  STD_LOGIC;
                    taken : out  STD_LOGIC;
                    send  : in  STD_LOGIC;
          id    : in  STD_LOGIC_VECTOR (7 downto 0);
          reg   : in  STD_LOGIC_VECTOR (7 downto 0);
          value : in  STD_LOGIC_VECTOR (7 downto 0));
end i2c_sender;
```

```
architecture Behavioral of i2c_sender is
    signal   divider  : unsigned (7 downto 0) := "00000001"; -- this value gives a 254 cycle pause before the initial frame is sent
    signal   busy_sr  : std_logic_vector(31 downto 0) := (others => '0');
    signal   data_sr  : std_logic_vector(31 downto 0) := (others => '1');
begin
    process(busy_sr, data_sr(31))
    begin
        if busy_sr(11 downto 10) = "10" or
          busy_sr(20 downto 19) = "10" or
          busy_sr(29 downto 28) = "10"  then
            siod <= 'Z';
        else
            siod <= data_sr(31);
        end if;
    end process;

    process(clk)
    begin
        if rising_edge(clk) then
            taken <= '0';
            if busy_sr(31) = '0' then
                SIOC <= '1';
                if send = '1' then
                    if divider = "00000000" then
                        data_sr <= "100" &   id & '0'  &   reg & '0' & value & '0' & "01";
                        busy_sr  <=  "111"  &  "111111111"  &  "111111111" & "111111111" & "11";
                        taken <= '1';
                    else
                        divider <= divider+1; -- this only happens on powerup
                    end if;
```

```
            end if;
        else

            case busy_sr(32-1 downto 32-3) & busy_sr(2 downto 0) is
                when "111"&"111" => -- start seq #1
                    case divider(7 downto 6) is
                        when "00"   => SIOC <= '1';
                        when "01"   => SIOC <= '1';
                        when "10"   => SIOC <= '1';
                        when others => SIOC <= '1';
                    end case;
                when "111"&"110" => -- start seq #2
                    case divider(7 downto 6) is
                        when "00"   => SIOC <= '1';
                        when "01"   => SIOC <= '1';
                        when "10"   => SIOC <= '1';
                        when others => SIOC <= '1';
                    end case;
                when "111"&"100" => -- start seq #3
                    case divider(7 downto 6) is
                        when "00"   => SIOC <= '0';
                        when "01"   => SIOC <= '0';
                        when "10"   => SIOC <= '0';
                        when others => SIOC <= '0';
                    end case;
                when "110"&"000" => -- end seq #1
                    case divider(7 downto 6) is
                        when "00"   => SIOC <= '0';
                        when "01"   => SIOC <= '1';
                        when "10"   => SIOC <= '1';
                        when others => SIOC <= '1';
                    end case;
```

```
                when "100"&"000" => -- end seq #2
                        case divider(7 downto 6) is
                                when "00"   => SIOC <= '1';
                                when "01"   => SIOC <= '1';
                                when "10"   => SIOC <= '1';
                                when others => SIOC <= '1';
                        end case;
                when "000"&"000" => -- Idle
                        case divider(7 downto 6) is
                                when "00"   => SIOC <= '1';
                                when "01"   => SIOC <= '1';
                                when "10"   => SIOC <= '1';
                                when others => SIOC <= '1';
                        end case;
                when others      =>
                        case divider(7 downto 6) is
                                when "00"   => SIOC <= '0';
                                when "01"   => SIOC <= '1';
                                when "10"   => SIOC <= '1';
                                when others => SIOC <= '0';
                        end case;
        end case;

        if divider = "11111111" then
                busy_sr <= busy_sr(32-2 downto 0) & '0';
                data_sr <= data_sr(32-2 downto 0) & '1';
                divider <= (others => '0');
        else
                divider <= divider+1;
        end if;
    end if;
end if;
```

```
        end process;
end Behavioral;
```

Constraints:

```
    ##clock
set_property PACKAGE_PIN W5 [get_ports clk100]

        set_property IOSTANDARD LVCMOS33 [get_ports clk100]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk100]


##Pmod Header JB
##Sch name = JB1
set_property PACKAGE_PIN A14 [get_ports {powerdown}]

        set_property IOSTANDARD LVCMOS33 [get_ports {powerdown}]
##Sch name = JB2
set_property PACKAGE_PIN A16 [get_ports {data[0]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[0]}]
##Sch name = JB3
set_property PACKAGE_PIN B15 [get_ports {data[2]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[2]}]
##Sch name = JB4
set_property PACKAGE_PIN B16 [get_ports {data[4]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[4]}]
##Sch name = JB7
set_property PACKAGE_PIN A15 [get_ports {reset_cam}]

        set_property IOSTANDARD LVCMOS33 [get_ports {reset_cam}]
##Sch name = JB8
set_property PACKAGE_PIN A17 [get_ports {data[1]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[1]}]
##Sch name = JB9
```

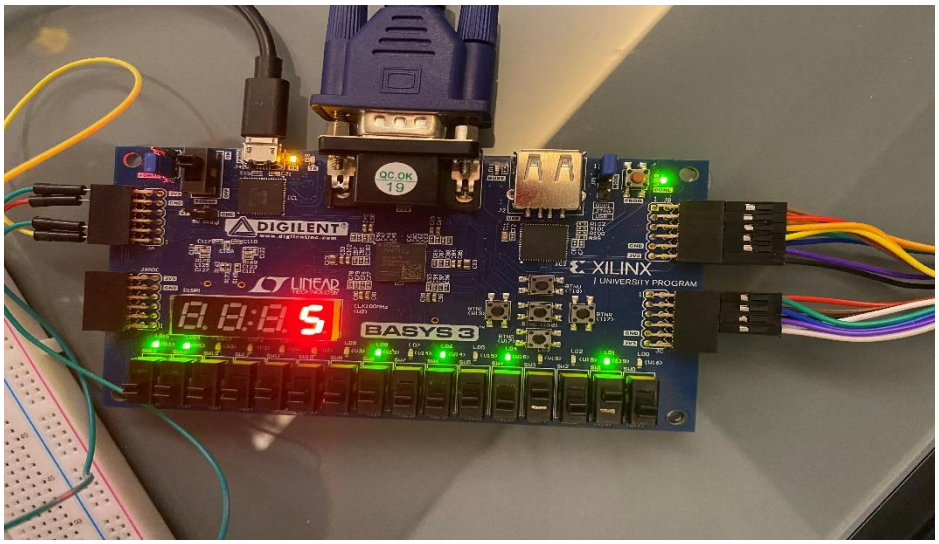
```
set_property PACKAGE_PIN C15 [get_ports {data[3]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[3]}]
##Sch name = JB10
set_property PACKAGE_PIN C16 [get_ports {data[5]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[5]}]



##Pmod Header JC
##Sch name = JC1
set_property PACKAGE_PIN K17 [get_ports {data[6]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[6]}]
##Sch name = JC2
set_property PACKAGE_PIN M18 [get_ports xclk]
        set_property IOSTANDARD LVCMOS33 [get_ports xclk]
##Sch name = JC3
set_property PACKAGE_PIN N17 [get_ports horsync]
        set_property IOSTANDARD LVCMOS33 [get_ports horsync]
##Sch name = JC4
set_property PACKAGE_PIN P18 [get_ports CamSiod]
        set_property IOSTANDARD LVCMOS33 [get_ports CamSiod]
        set_property PULLUP TRUE [get_ports CamSiod]
##Sch name = JC7
set_property PACKAGE_PIN L17 [get_ports {data[7]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {data[7]}]
##Sch name = JC8
set_property PACKAGE_PIN M19 [get_ports pclk]
        set_property IOSTANDARD LVCMOS33 [get_ports pclk]
   set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {pclk_IBUF}]
##Sch name = JC9
set_property PACKAGE_PIN P17 [get_ports versync]
```

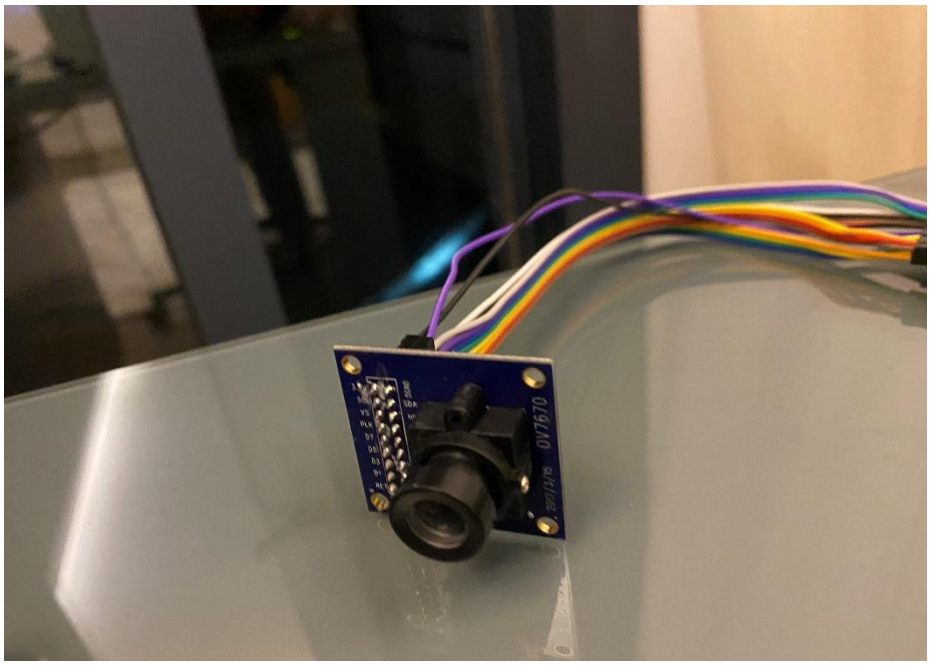
```
        set_property IOSTANDARD LVCMOS33 [get_ports versync]
##Sch name = JC10
set_property PACKAGE_PIN R18 [get_ports CamSioc]

        set_property IOSTANDARD LVCMOS33 [get_ports CamSioc]

set_property PACKAGE_PIN A18 [get_ports serial_data_output]
   set_property IOSTANDARD LVCMOS33 [get_ports serial_data_output]


set_property PACKAGE_PIN U16 [get_ports {distance[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[0]}]
set_property PACKAGE_PIN E19 [get_ports {distance[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[1]}]
set_property PACKAGE_PIN U19 [get_ports {distance[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[2]}]
set_property PACKAGE_PIN V19 [get_ports {distance[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[3]}]
set_property PACKAGE_PIN W18 [get_ports {distance[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[4]}]
set_property PACKAGE_PIN U15 [get_ports {distance[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[5]}]
set_property PACKAGE_PIN U14 [get_ports {distance[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[6]}]
set_property PACKAGE_PIN V14 [get_ports {distance[7]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[7]}]
set_property PACKAGE_PIN V13 [get_ports {distance[8]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[8]}]
set_property PACKAGE_PIN V3 [get_ports {distance[9]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[9]}]
set_property PACKAGE_PIN W3 [get_ports {distance[10]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[10]}]
set_property PACKAGE_PIN U3 [get_ports {distance[11]}]
```

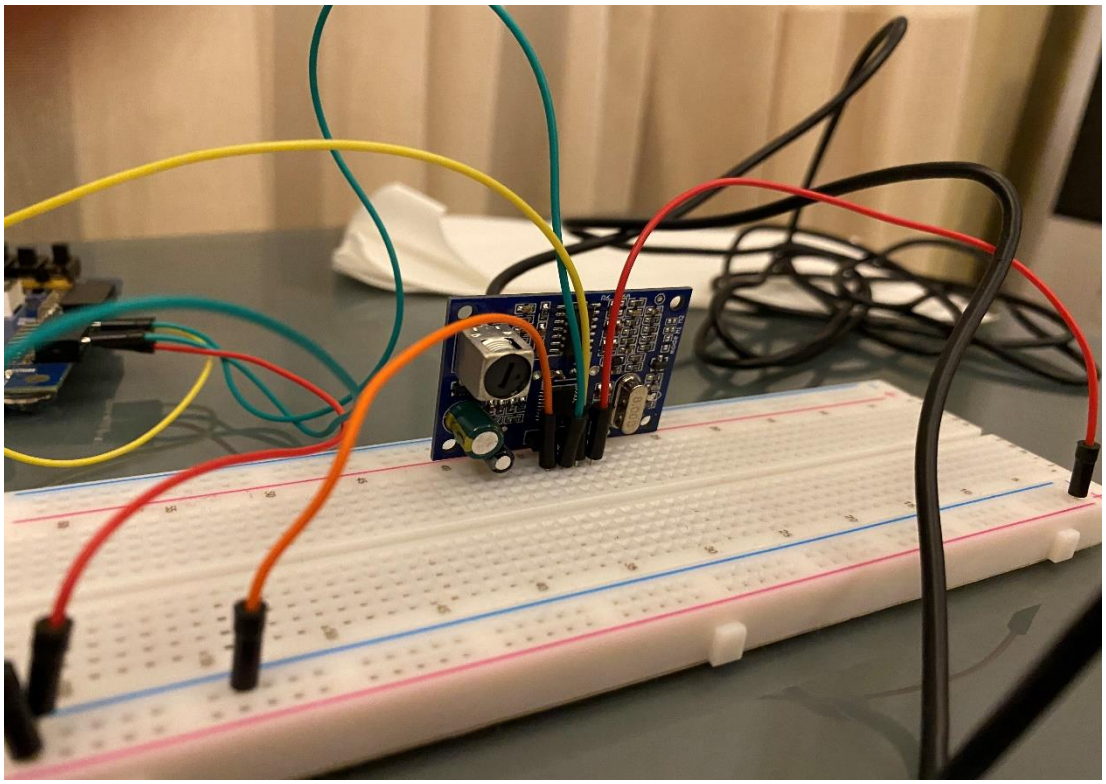
```
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[11]}]
set_property PACKAGE_PIN P3 [get_ports {distance[12]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[12]}]
set_property PACKAGE_PIN N3 [get_ports {distance[13]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {distance[13]}]
set_property PACKAGE_PIN P1 [get_ports {configcamfinished}]
   set_property IOSTANDARD LVCMOS33 [get_ports {configcamfinished}]
set_property PACKAGE_PIN L1 [get_ports {readyindicator}]
   set_property IOSTANDARD LVCMOS33 [get_ports {readyindicator}]

 ##Pmod Header JA
#Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {trigger}]
    set_property IOSTANDARD LVCMOS33 [get_ports {trigger}]
##Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {echo}]
    set_property IOSTANDARD LVCMOS33 [get_ports {echo}]
## Switches
set_property PACKAGE_PIN V17 [get_ports {reset}]
    set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN V16 [get_ports {selector}]
    set_property IOSTANDARD LVCMOS33 [get_ports {selector}]
set_property PACKAGE_PIN W16 [get_ports {activator}]
    set_property IOSTANDARD LVCMOS33 [get_ports {activator}]
set_property PACKAGE_PIN W17 [get_ports {pause}]
    set_property IOSTANDARD LVCMOS33 [get_ports {pause}]
#set_property PACKAGE_PIN W15 [get_ports {uart_sender}]
#    set_property IOSTANDARD LVCMOS33 [get_ports {uart_sender}]
set_property PACKAGE_PIN T18 [get_ports {switch}]
    set_property IOSTANDARD LVCMOS33 [get_ports {switch}]

set_property PACKAGE_PIN T17 [get_ports {button_res1}]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports {button_res1}]

set_property PACKAGE_PIN U18 [get_ports {button_res2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {button_res2}]
 set_property PACKAGE_PIN W7 [get_ports {sevseg[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[6]}]
set_property PACKAGE_PIN W6 [get_ports {sevseg[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[5]}]
set_property PACKAGE_PIN U8 [get_ports {sevseg[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[4]}]
set_property PACKAGE_PIN V8 [get_ports {sevseg[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[3]}]
set_property PACKAGE_PIN U5 [get_ports {sevseg[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[2]}]
set_property PACKAGE_PIN V5 [get_ports {sevseg[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[1]}]
set_property PACKAGE_PIN U7 [get_ports {sevseg[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sevseg[0]}]
set_property PACKAGE_PIN U2 [get_ports {Anode_Activate[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[0]}]
set_property PACKAGE_PIN U4 [get_ports {Anode_Activate[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[1]}]
set_property PACKAGE_PIN V4 [get_ports {Anode_Activate[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[2]}]
set_property PACKAGE_PIN W4 [get_ports {Anode_Activate[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[3]}]
```

Photos of The Device and Parts


QC.OK
19
DIGILENT
XILINX
UNIVERSITY PROGRAM
BASYS 3


OV7670



Datasheets

Distance Sensor JSN-SR04T: https://www.makerguides.com/wp-content/uploads/2019/02/JSN-SR04T-Datasheet.pdf

Camera OV7670 : https://datasheetspdf.com/pdf-file/555220/OmniVisionTechnologies/OV7670/1