İkranur Yılmaz
22003768
EE102-04

EE102 INTRODUCTION TO DIGITAL CICUIT DESIGN

TERM PROJECT FINAL REPORT

MOTION PAINTER

İKRANUR YILMAZ

22003768

EE102-04

İkranur Yılmaz
22003768
EE102-04

ABSTRACT

This project introduces a real-time movement visualization system utilizing the Basys3 development board interfaced with the OV7670 camera module and a wand-like device equipped with an RGB LED. The system captures user gestures in real-time and translates them into dynamic visual patterns through precise colour detection and adjustable thresholds. The integration of an effective reset mechanism ensures system reliability. The project highlights the potential of FPGA-based technologies in facilitating real-time interactive experiences and provides a versatile platform for creative expression.

İkranur Yılmaz
22003768
EE102-04

MOTION PAINTER

INTRODUCTION

The objective of this project is to design and implement an innovative movement visualization system that transforms user gestures into captivating real-time visualizations. Utilizing a camera module interfaced with a Basys3 development board, the system captures user gestures made with a wand equipped with an RGB LED. The RGB data from the camera feed is analyzed to identify and respond to pixels with selected colour values exceeding a predefined threshold, creating dynamic visual patterns based on user movements.

DESIGN METHODOLOGY

COMPONENTS:

Camera Module: Captures live video feed of user gestures. The OV7670 camera module will be configured to capture RGB data. The camera interface will be established with the Basys3 board to stream video data for processing.

Basys3: The Basys3 board will handle real-time data processing, utilizing its FPGA capabilities. VHDL code will be developed to process the RGB data and identify pixels based on the selected colour threshold.

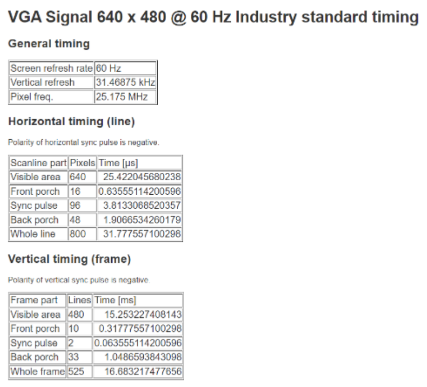
Wand: The wand will be equipped with an RGB LED, capable of changing colors based on user input. User movements with the wand will be captured by the camera and processed in real-time.

VGA Display: The processed visual data will be transmitted to a VGA display module. The display will render dynamic visual patterns reflecting the user's gestures.

User Input Controls: switches for users to set the desired color, threshold, and negative threshold values. These inputs dynamically affect the data processing logic on the Basys3 board.

INITIALIZATION:

Firstly, VGA display was initialized. According to VGA Timing Stardards for 640x480 resolution, relevant v_sync -represents the vertinal line- and h_sync -represents the horizontal line- signals were initilized and clock with with 25.175 MHz was connected.


VGA Signal 640 x 480 @ 60 Hz Industry standard timing

General timing

Screen refresh rate	60 Hz
Vertical refresh	31.46875 kHz
Pixel freq.	25.175 MHz

Horizontal timing (line)

Polarity of horizontal sync pulse is negative.

Scanline part	Pixels	Time [µs]
Visible area	640	25.422045680238
Front porch	16	0.63555114200596
Sync pulse	96	3.8133068520357
Back porch	48	1.9066534260179
Whole line	800	31.777557100298

Vertical timing (frame)

Polarity of vertical sync pulse is negative.

Frame part	Lines	Time [ms]
Visible area	480	15.253227408143
Front porch	10	0.31777557100298
Sync pulse	2	0.063555114200596
Back porch	33	1.0486593843098
Whole frame	525	16.683217477656

Figure 1 VGA Timing Standard for 640x480

Combined with RGB data, it was ready to display any image. In progress demo, the ability to draw on the Basys3 board using the VGA display and selected pixels was demonstrated.

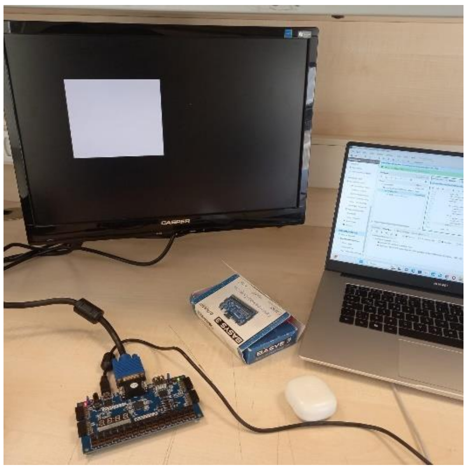


Figure 2 Progress Demo Photo Capture

Secondly, the OV7670 camera module was initialized. The camera module has 18 pins in total. These are:

- Vdd(3v3)
- GND
- SIOC/SIOD pins (they are SCL and SDA in VHDL code)
- VSYNC
- HREF
- PCLK
- XCLK
- 8 data pins
- RESET
- PWDN (active low)

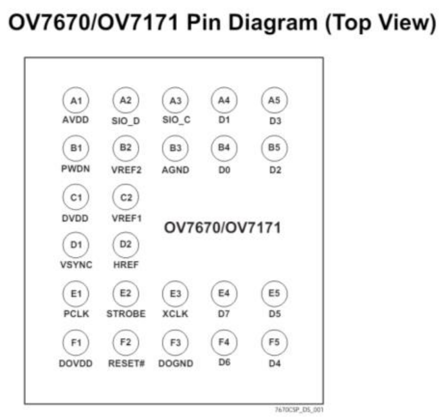


Figure 3 OV7670 Pin Diagram

İkranur Yılmaz
22003768
EE102-04

The VGA cannot read the RGB data coming from the camera module directly mostly because their clock frequencies are different. To solve this problem, the dual port memory in Basys 3 -it is called frameram- was used. However, there was not enough space in Basys 3 to store all of the data coming from camera module. My solution was to drop one horizontal line of data and store the other one for each 2 horizontal lines of RGB data. With this, the dual port memory was able to store RGB data (9 bits) of 320x480 pixels and there was a little bit of space left. The RGB data storage was completed according to RGB565 Timing Diagram. Basically, it sends 2 bytes of data for each pixel. It stores the most significant 3 bits of each colour data for each pixel. For 'R', first byte's 7, 6 and 5th bits; for 'G, first byte's 2, 1 and 0th bits and for 'B, second byte's 4, 3 and 2th bits are stored in framebuffer (for each pixel). Since one of two lines of pixel data was deleted due to lack of memory, VGA will display each line twice to fill the screen.

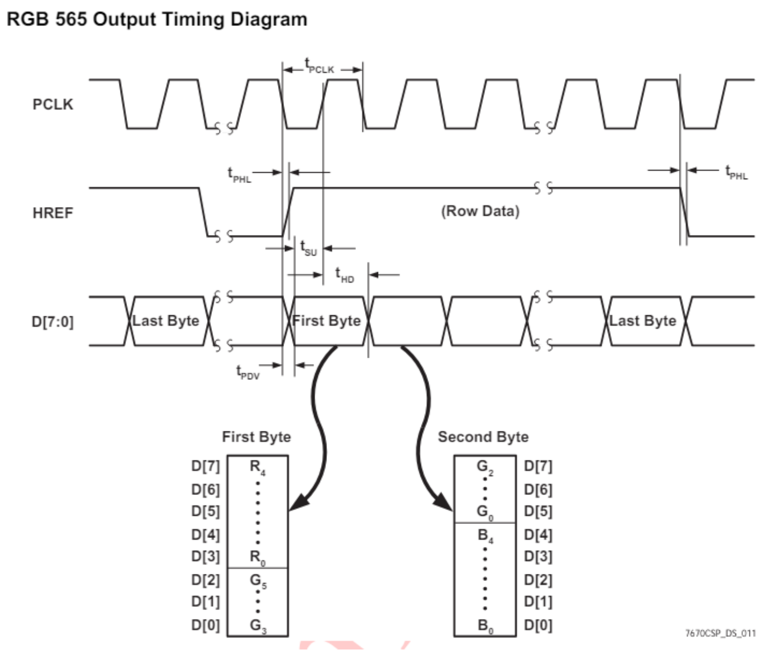


Figure 4 RGB 565 Timing Diagram

SIOC and SIOD pins(SDA and SCL in the code) are used for I^2C communication with the camera for setting it up. The Basys3 development board interfaces with the OV7670 camera module using the I^2C protocol to facilitate communication. This involves initializing the I^2C bus by generating a clock signal and managing start and stop conditions. The Basys3 sends the 7-bit address of the OV7670 with a write bit to configure the camera's internal registers. Data transmission occurs in 8-bit packets, with acknowledgments ensuring successful communication. Using this protocol is essential for real-time image processing and visualization since the default values for addresses in the OV7670 module are not suitable for my use and some of them need adjustment. After all of the connections were done, the camera and VGA was ready to display any real-time footage.

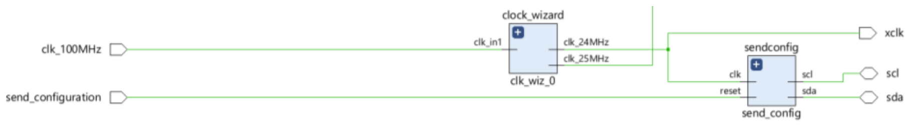


Figure 5 I2C schematic

İkranur Yılmaz
22003768
EE102-04

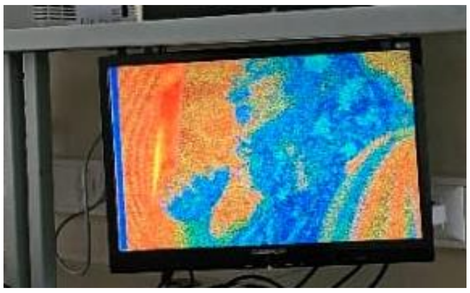


Figure 6 Before I2C

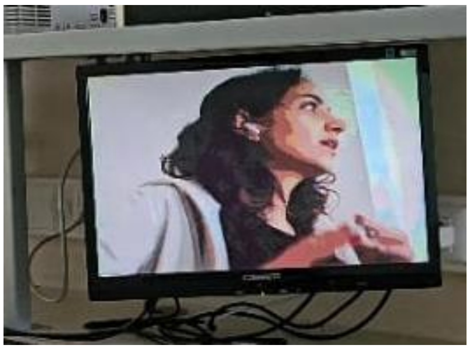


Figure 7 After I2C

After connecting the VGA and the camera module, it was time to create the painting system. To do this, another dual port memory is needed but this time it will only store one bit for each pixel. There is enough space left in Basys 3 to do this. After instantiating this new dual port memory called paintram, it is time to create “paint machine”. It is module that has 6 inputs: a clock, colour (user selects it manually), positive threshold (user selects it manually), negative threshold (user selects it manually), reset (user selects it manually) and the RGB data; and 2 outputs, wrepaint and dataforpaint. Here, colour, positive threshold, negative threshold and reset were connected to Basys 3 switches and buttons and RGB data was the same data going to frameram. This module examines each pixel for if the selected colour data is above the positive threshold and the other two colour data are below the negative threshold. If that is the case, the relevant address in paintram will be set to 1, otherwise it remains 0. Using a logical OR Gate, when paintram is high that pixel is overridden by white until resetted.

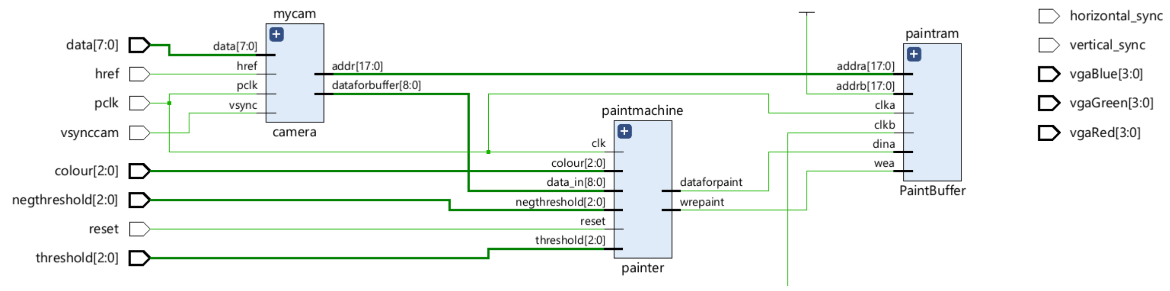


Figure 8 RTL Schematic of Painting System

İkranur Yılmaz
22003768
EE102-04

When the reset input is '1', the outputs wrepaint and dataforpaint are set to "1" and "0" respectively. The reset condition takes precedence over other logical conditions within the process block. This means that as soon as the reset signal is detected, the outputs are set to their reset values regardless of the current state or input values. Basically, it deletes all of the paintings on the screen if reset is kept being pressed for less than a second. All of these inputs mechanism ensures that the painting process can be controlled.

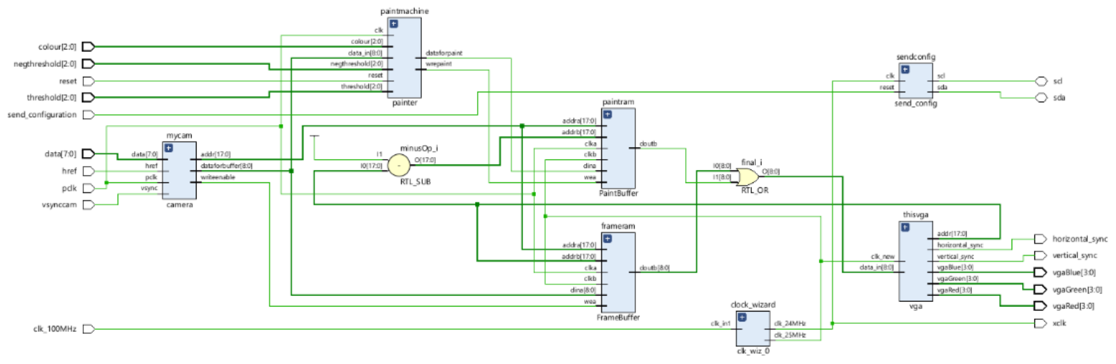


Figure 9 RTL Schematic of the whole system

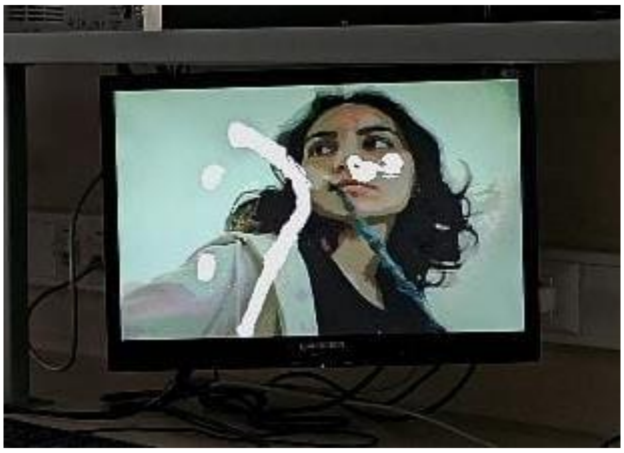


Figure 10 Painting on the screen

Last step was to design the wand. A simple circuit on a Elder Wand was constructred. A battery, a RGB led, 3 resistors, a button, a 4 pin dip switch and bunch of cables was enough to do so.

İkranur Yılmaz
22003768
EE102-04

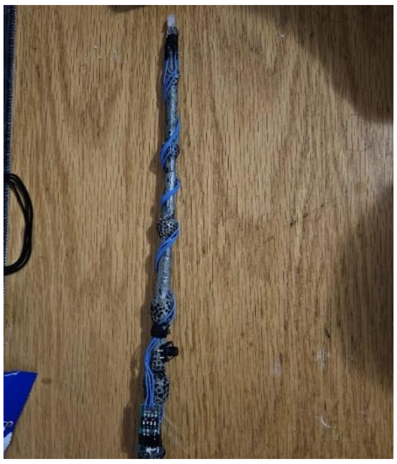


Figure 11 The Wand

Here are different colour displays on the wand:

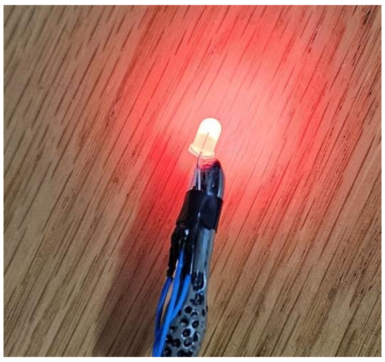


Figure 12 RGB Red on the wand

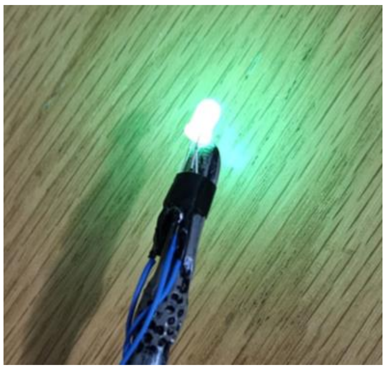


Figure 13 RGB Green on the wand

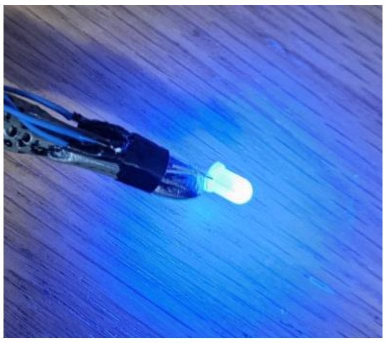


Figure 14 RGB Blue on the wand

RESULTS

The implementation of the movement visualization system on the Basys3, interfaced with the OV7670 camera module met expectations. The system successfully captures user gestures in real-time using the OV7670 camera module. The Basys 3 processes the captured data efficiently, leveraging its FPGA capabilities to handle parallel data processing. This allows for smooth and immediate translation of user movements into visual patterns. By analysing the RGB data from the camera feed, the system identifies pixels based on user-defined colour, threshold, and negative threshold values. This flexible threshold mechanism enables the creation of dynamic and expressive visual patterns. Users can easily adjust these values to change the visual output, enhancing the interactivity and customization of the system.

The system demonstrates accurate detection and differentiation of colours based on the specified thresholds. For instance, when the colour input is set to red, the system correctly identifies pixels where the red component exceeds the threshold and both green and blue components are below the negative threshold. Similar accuracy is observed for green and blue colour settings, ensuring precise control over the visualizations. This leads that if colour selection is red, then when the led on the wand set to blue, it won’t paint the screen.

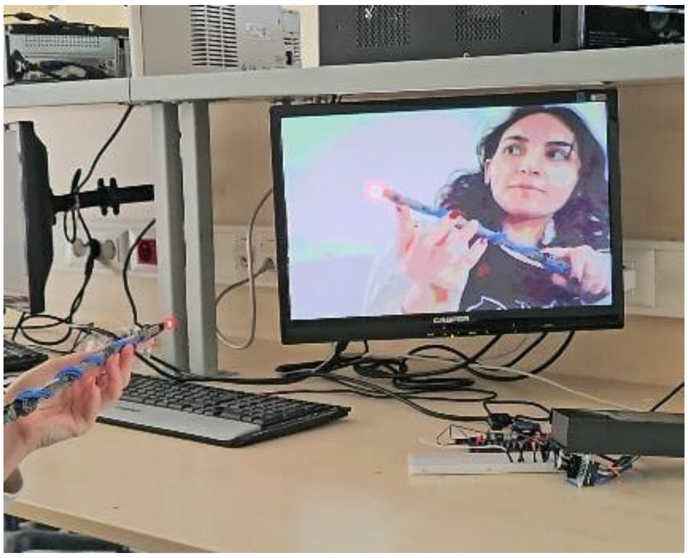


Figure 15 An attempt to paint 1

In Figure 12, the colour input was set to blue and RGB led on the wand was set to red. As it can be seen, there was no painting on the screen as expected.

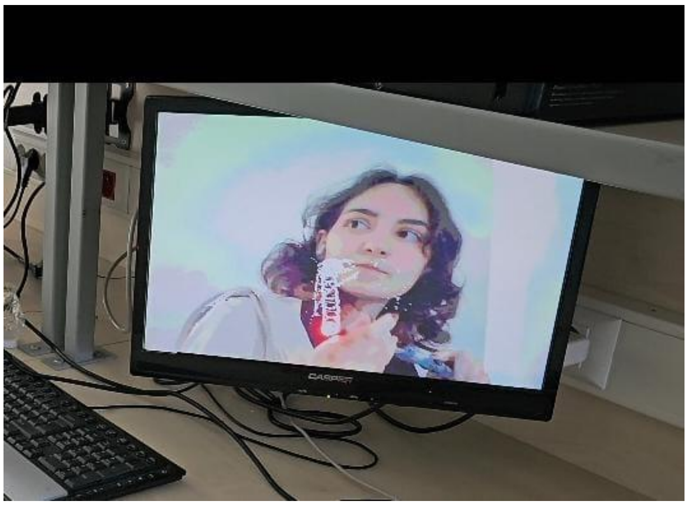


Figure 16 An attempt to paint 2

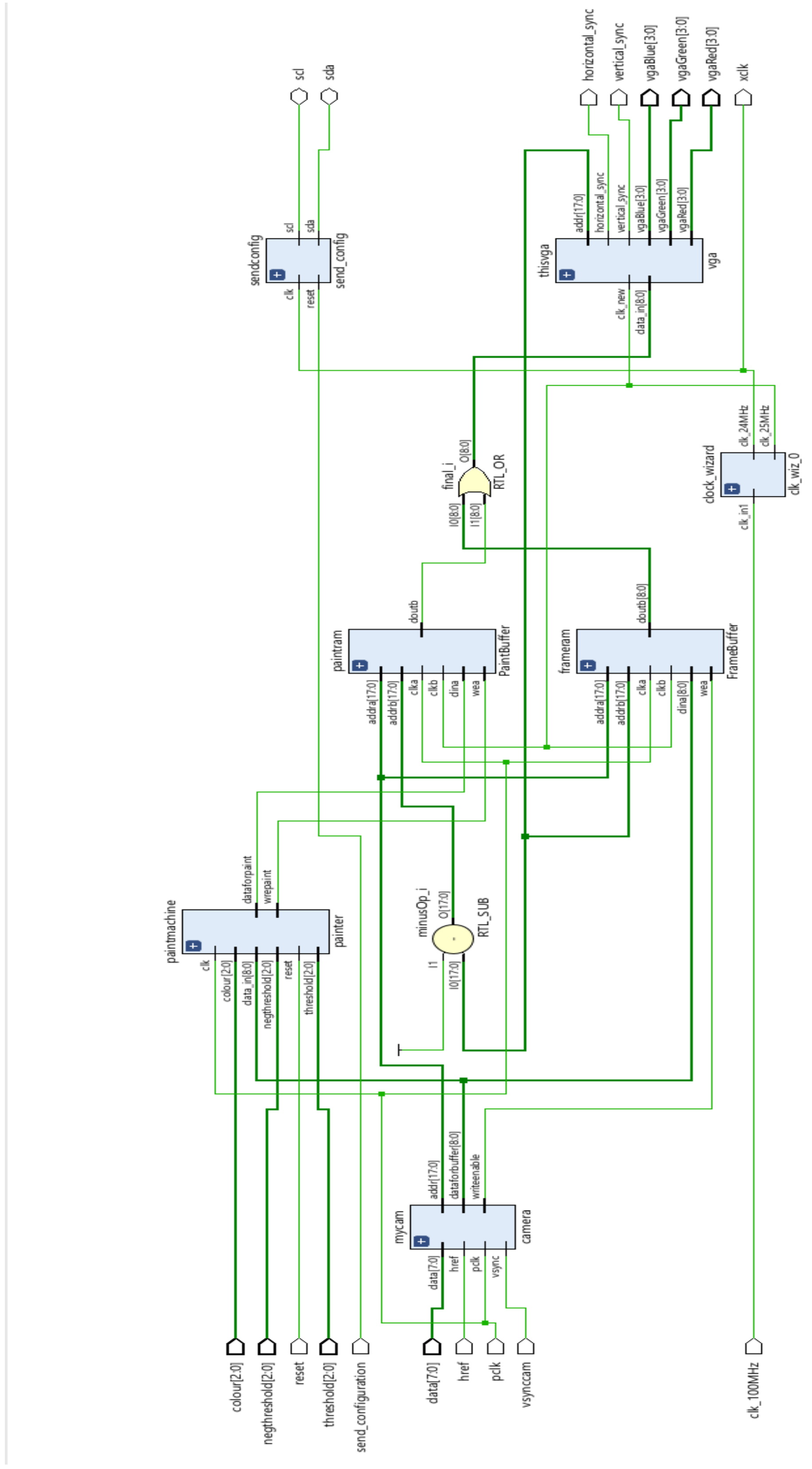
In figure 13, the colour input was set to red and the RGB led on the wand also set to red. As it can be seen, it painted the screen as expected.

Also, the reset functionality is effectively integrated, allowing the system to be initialized to a known state. This is crucial for reliable operation, especially during power-up or after encountering errors. When the reset signal is asserted, the outputs wrepaint and dataforpaint are set to "1" and "0" respectively, preparing the system for subsequent operations.

Overall, the project successfully captures user gestures in real-time and translates them into dynamic visual patterns. Through precise colour detection and adjustable thresholds, users can customize their visualizations with ease, enhancing system interactivity. The integration of an effective reset mechanism ensures the system's reliability and prepares it for subsequent operations. So, the system provides a versatile platform for creative expression, underscoring the potential of FPGA-based technologies in real-time interactive experiences.

İkranur Yılmaz
22003768
EE102-04
APPENDICES

Appendix A. Full RTL Schematic

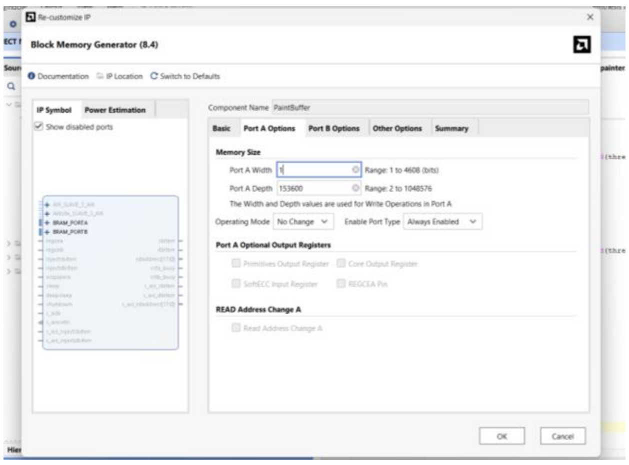


Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

Appendix B. Used Ips

Paintram (dual port memory)



Re-customize IP

Block Memory Generator (8.4)

Documentation | IP Location | Switch to Defaults

IP Symbol | Power Estimation

Show disabled ports

Component Name: PaintBuffer

Basic | Port A Options | Port B Options | Other Options | Summary

Memory Size

Port A Width: 1 — Range: 1 to 4608 (bits)

Port A Depth: 153600 — Range: 2 to 1048576

The Width and Depth values are used for Write Operations in Port A

Operating Mode: No Change — Enable Port Type: Always Enabled

Port A Optional Output Registers

Primitives Output Register | Core Output Register

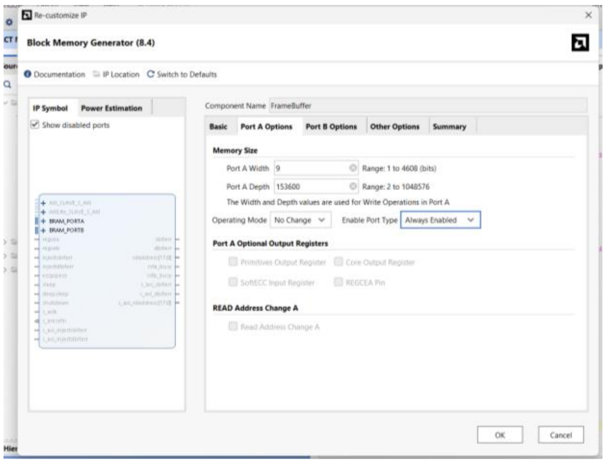
SoftECC Input Register | REGCEA Pin

READ Address Change A

Read Address Change A

OK | Cancel

Frameram (dual port memory)



Re-customize IP

Block Memory Generator (8.4)

Documentation | IP Location | Switch to Defaults

IP Symbol | Power Estimation

Show disabled ports

Component Name: FrameBuffer

Basic | Port A Options | Port B Options | Other Options | Summary

Memory Size

Port A Width: 9 — Range: 1 to 4608 (bits)

Port A Depth: 153600 — Range: 2 to 1048576

The Width and Depth values are used for Write Operations in Port A

Operating Mode: No Change — Enable Port Type: Always Enabled

Port A Optional Output Registers

Primitives Output Register | Core Output Register

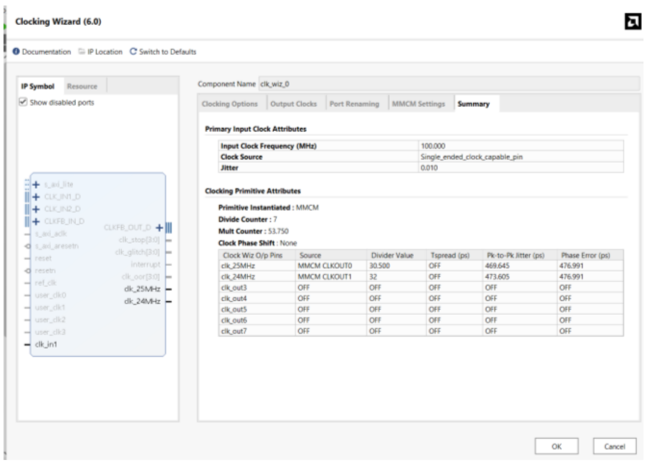
SoftECC Input Register | REGCEA Pin

READ Address Change A

Read Address Change A

OK | Cancel

Clocking Wizard



Clocking Wizard (6.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Resource

Show disabled ports

Component Name: clk_wiz_0

Clocking Options | Output Clocks | Port Renaming | MMCM Settings | Summary

Primary Input Clock Attributes

Input Clock Frequency (MHz)	100.000
Clock Source	Single_ended_clock_capable_pin
Jitter	0.010

Clocking Primitive Attributes

Primitive Instantiated : MMCM
Divide Counter : 7
Mult Counter : 53.750
Clock Phase Shift : None

Clock Wiz O/p Pins	Source	Divider Value	Tspread (ps)	Pk-to-Pk Jitter (ps)	Phase Error (ps)
clk_25MHz	MMCM CLKOUT0	30.500	OFF	469.645	476.991
clk_24MHz	MMCM CLKOUT1	32	OFF	473.605	476.991
clk_out3	OFF	OFF	OFF	OFF	OFF
clk_out4	OFF	OFF	OFF	OFF	OFF
clk_out5	OFF	OFF	OFF	OFF	OFF
clk_out6	OFF	OFF	OFF	OFF	OFF
clk_out7	OFF	OFF	OFF	OFF	OFF

OK | Cancel

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

Appendix C. VHDL Code

camera.vhd

```
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 camera is
   Port (
         pclk : in STD_LOGIC;
         vsync : in std_logic;
         href : in std_logic;
         data : in STD_LOGIC_VECTOR (7 downto 0);
         addr: out STD_LOGIC_VECTOR (17 downto 0);
         writeenable : out std_logic_vector(0 downto 0);
         dataforbuffer : out std_logic_vector(8 downto 0)
         );
end camera;

architecture Behavioral of camera is
signal counter,drop :  std_logic:='0';
signal R,G,B : std_logic_vector (2 downto 0):= (others=>'0');
signal address : integer :=0;
signal check, rgb_ready : std_logic:='0';
begin
   addr<= std_logic_vector(to_unsigned(address, addr'length));
   process (pclk)
   begin
      if rising_edge(pclk) then
         writeenable<="0";

         check<=href;
         if( vsync='1') then
            address<=0;
            drop<='0';
         end if;

         if((check='0') and (href='1')) then
            drop<= not drop;

         end if;
```

Codes written in red were found on the internet

```
            if ((href='1') and (drop='0')) then
                counter<=not counter;
                if counter='0' then
                    R(0)<=data(5);
                    R(1)<=data(6);
                    R(2)<=data(7);

                    G(0)<=data(0);
                    G(1)<=data(1);
                    G(2)<=data(2);
                else

                    B(0)<=data(2);
                    B(1)<=data(3);
                    B(2)<=data(4);

                    rgb_ready<='1';
                end if;

            end if;
            if rgb_ready='1' then
                dataforbuffer ( 2 downto 0)<= R;
                dataforbuffer ( 5 downto 3)<= G;
                dataforbuffer ( 8 downto 6)<= B;
                writeenable<="1";
                rgb_ready<='0';
                address<=address+1;
            end if;
        end if;

    end process;

end Behavioral;
```

vga.vhd

```
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;
```

İkranur Yılmaz
22003768
EE102-04

```
entity vga is
   Port ( clk_new : in STD_LOGIC;
          addr: out STD_LOGIC_VECTOR (17 downto 0);
          data_in : in std_logic_vector(8 downto 0);
          vgaRed: out std_logic_vector(3 downto 0);
          vgaBlue: out std_logic_vector(3 downto 0);
          vgaGreen: out std_logic_vector(3 downto 0);
          vertical_sync : out STD_LOGIC;
          horizontal_sync : out STD_LOGIC);
end vga;

architecture Behavioral of vga is
signal s_horizontal : STD_LOGIC:='0';
signal s_vertical : STD_LOGIC:='0';
signal counterHorizontal: unsigned (31 downto 0):= (others=>'0');
signal counterVertical: unsigned (31 downto 0):= (others=>'0');
signal addresscounter: integer:=0;
signal repaint : std_logic:='0';

begin
vertical_sync<=s_vertical;
horizontal_sync<=s_horizontal;

addr<= std_logic_vector(to_unsigned(addresscounter, addr'length));

        process(clk_new)
        begin
        if rising_edge(clk_new) then
          counterHorizontal<=counterHorizontal+1;
          s_horizontal<='1';
          if counterHorizontal>656 and counterHorizontal<752+1  then
              s_horizontal<='0';
          end if;
          if counterHorizontal=800-1 then
              counterHorizontal<=(others=>'0');
              counterVertical<=counterVertical+1;
              repaint <= not repaint;
              if ((repaint = '0') and (addresscounter>=640)) then
                  addresscounter<=addresscounter-642;
              end if;

          end if;
          s_vertical<='1';
          if counterVertical> 490 and counterVertical<492+1 then
              s_vertical<='0';
          end if;
          if counterVertical=525 then
          counterVertical<=(others=>'0');
          addresscounter<=0;
```

Codes written in red were found on the internet

```
                end if;

                if counterVertical>=0 and counterVertical<=480 and counterHorizontal>=0 and
counterHorizontal<=640 then
                   vgaRed<= data_in(2 downto 0)&"0";
                   vgaGreen<=data_in(5 downto 3)&"0";
                   vgaBlue<=data_in(8 downto 6)&"0";
                   addresscounter<=addresscounter+1;
                else
                   vgaRed<="0000";
                   vgaBlue<="0000";
                   vgaGreen<="0000";
                end if;

             end if;

             end process;
         end Behavioral;
```

painter.vhd

```
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 painter is
   Port ( clk : in STD_LOGIC;
   threshold : in std_logic_vector(2 downto 0);
   negthreshold: in std_logic_vector( 2 downto 0);
   colour : in std_logic_vector(2 downto 0);
   reset : in std_logic;
   data_in : in std_logic_vector( 8 downto 0);
   dataforpaint : out std_logic_vector(0 downto 0);
   wrepaint : out std_logic_vector(0 downto 0)
         );

end painter;

architecture Behavioral of painter is
```

İkranur Yılmaz
22003768
EE102-04

```
    begin
    process(clk)
    begin
       if rising_edge(clk) then
       wrepaint<="0";
          if colour="001" then --RED
             if (unsigned(data_in(2 downto 0)) >= unsigned(threshold) and unsigned(data_in(5
downto 3))<=unsigned(negthreshold) and unsigned(data_in(8 downto 6 ))<=unsigned(negthreshold) )
then
                wrepaint<="1";
                dataforpaint<="1";
             end if;

          end if;

          if colour="010" then --GREEN
             if (unsigned(data_in(5 downto 3)) >= unsigned(threshold) and unsigned(data_in(2
downto 0))<=unsigned(negthreshold) and unsigned(data_in(8 downto 6))<=unsigned(negthreshold)
)then
                wrepaint<="1";
                dataforpaint<="1";
             end if;


          end if;

          if colour="100" then --BLUE
             if (unsigned(data_in(8 downto 6)) >= unsigned(threshold) and unsigned(data_in(5
downto 3))<=unsigned(negthreshold) and unsigned(data_in(2 downto 0))<=unsigned(negthreshold)
)then
                wrepaint<="1";
                dataforpaint<="1";

             end if;

          end if;

          if reset='1' then
             wrepaint<="1";
             dataforpaint<="0";
          end if;

       end if;
    end process;

    end Behavioral;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

FOLLOWING CODE WAS FOUND FROM INTERNET

send_config.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 03.05.2024 16:46:24
-- Design Name:
-- Module Name: send_config - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 send_config is
    Port ( clk : in STD_LOGIC;
           reset: in STD_LOGIC;
           scl : inout STD_LOGIC;
           sda : inout STD_LOGIC);
end send_config;

architecture Behavioral of send_config is
    component i2c_master is
        GENERIC(
            input_clk : INTEGER := 24_000_000; --input clock speed from user logic in Hz
            bus_clk   : INTEGER := 100_000   --speed the i2c bus (scl) will run at in Hz
          );
          PORT(
            clk       : IN     STD_LOGIC;                    --system clock
            reset_n   : IN     STD_LOGIC;                    --active low reset
            ena       : IN     STD_LOGIC;                    --latch in command
            addr      : IN     STD_LOGIC_VECTOR(6 DOWNTO 0); --address of target slave
            rw        : IN     STD_LOGIC;                    --'0' is write, '1' is read
            data_wr   : IN     STD_LOGIC_VECTOR(7 DOWNTO 0); --data to write to slave
            busy      : OUT    STD_LOGIC;                    --indicates transaction in progress
            data_rd   : OUT    STD_LOGIC_VECTOR(7 DOWNTO 0); --data read from slave
            ack_error : BUFFER STD_LOGIC;                    --flag if improper acknowledge from
slave
            sda       : INOUT  STD_LOGIC;                    --serial data output of i2c bus
            scl       : INOUT  STD_LOGIC                     --serial clock output of i2c bus
          );
        end component;
        signal i2c_ena,i2c_rw,i2c_busy, busy_prev : STD_LOGIC :='0';
        signal i2c_addr : STD_LOGIC_VECTOR(6 downto 0) := (others=>'0');
        signal i2c_data_wr : STD_LOGIC_VECTOR(7 downto 0):=(others=>'0');
        signal busy_cnt,delay_ctr : integer := 0;
        signal reset_sync,reset_sync2 : STD_LOGIC := '0';
        signal slave_addr : STD_LOGIC_VECTOR(6 downto 0) := "0100001";
      begin
        i2c_module: i2c_master
        port map(
          clk => clk,
          reset_n => reset_sync2,
          ena => i2c_ena,
          addr => i2c_addr,
          rw => i2c_rw,
          data_wr => i2c_data_wr,
          busy =>i2c_busy,
          sda => sda,
          scl => scl
        );
        process(clk)
        begin
          if rising_edge(clk) then
            i2c_rw<='0';i2c_addr <= slave_addr;
            if reset='1' then
              delay_ctr <= delay_ctr+1;
              if delay_ctr = 10_000_000 then
                delay_ctr<=0;
                reset_sync<='1';
                reset_sync2<='1';
              end if;
            else
              delay_ctr<=0;
```

İkranur Yılmaz
22003768
EE102-04

```
                  reset_sync<='0';
                  reset_sync2<='0';
               end if;

               busy_prev <= i2c_busy;                --capture the value of the previous i2c busy
signal
               if reset_sync='1' then
                  IF(busy_prev = '0' AND i2c_busy = '1') THEN  --i2c busy just went high
                    busy_cnt <= busy_cnt + 1;              --counts the times busy has gone from low
to high during transaction
                  END IF;
                                   CASE busy_cnt IS
                                         WHEN 0 => i2c_ena <= '1'; i2c_data_wr <= x"00";
                                         WHEN 1 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                                 if delay_ctr = 1_000_000 then
                                                         busy_cnt <= busy_cnt+1;
                                                         delay_ctr<=0;
                                                 end if;

                                         WHEN 2 => i2c_ena <= '1'; i2c_data_wr <= x"12";
                                         WHEN 3 =>i2c_ena <= '1'; i2c_data_wr <= x"80";
                                         WHEN 4 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                                 if delay_ctr = 7_500_000 then
                                                         busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                                 end if;
                                         WHEN 5 => i2c_ena <= '1'; i2c_data_wr <= x"11";
                                         WHEN 6 =>i2c_ena <= '1'; i2c_data_wr <= x"02";
                                         WHEN 7 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                                 if delay_ctr = 250_000 then
                                                         busy_cnt <= busy_cnt+1;
delay_ctr<=0;

                                                 end if;
                                         WHEN 8 => i2c_ena <= '1'; i2c_data_wr <= x"3A";
                                         WHEN 9 =>i2c_ena <= '1'; i2c_data_wr <= x"04";
                                         WHEN 10 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                                 if delay_ctr = 250_000 then
                                                         busy_cnt <= busy_cnt+1;
delay_ctr<=0;

                                                 end if;
                                         WHEN 11 => i2c_ena <= '1'; i2c_data_wr <= x"12";
                                         WHEN 12 =>i2c_ena <= '1'; i2c_data_wr <= x"00";
                                         WHEN 13 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                                 if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                end if;
                        WHEN 14 => i2c_ena <= '1'; i2c_data_wr <= x"17";
                        WHEN 15 =>i2c_ena <= '1'; i2c_data_wr <= x"13";
                        WHEN 16 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                if delay_ctr = 250_000 then
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                end if;
                        WHEN 17 => i2c_ena <= '1'; i2c_data_wr <= x"18";
                        WHEN 18 =>i2c_ena <= '1'; i2c_data_wr <= x"01";
                        WHEN 19 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                if delay_ctr = 250_000 then
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                end if;
                        WHEN 20 => i2c_ena <= '1'; i2c_data_wr <= x"32";
                        WHEN 21 =>i2c_ena <= '1'; i2c_data_wr <= x"B6";
                        WHEN 22 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                if delay_ctr = 250_000 then
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                end if;
                        WHEN 23 => i2c_ena <= '1'; i2c_data_wr <= x"19";
                        WHEN 24 =>i2c_ena <= '1'; i2c_data_wr <= x"02";
                        WHEN 25 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                if delay_ctr = 250_000 then
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                end if;
                        WHEN 26 => i2c_ena <= '1'; i2c_data_wr <= x"1A";
                        WHEN 27 =>i2c_ena <= '1'; i2c_data_wr <= x"7A";
                        WHEN 28 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                if delay_ctr = 250_000 then
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                end if;
                        WHEN 29 => i2c_ena <= '1'; i2c_data_wr <= x"03";
                        WHEN 30 =>i2c_ena <= '1'; i2c_data_wr <= x"0A";
                        WHEN 31 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                if delay_ctr = 250_000 then
                                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
```

Codes written in red were found on the internet

```
                    end if;
                WHEN 32 => i2c_ena <= '1'; i2c_data_wr <= x"0C";
                WHEN 33 =>i2c_ena <= '1'; i2c_data_wr <= x"00";
                WHEN 34 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 35 => i2c_ena <= '1'; i2c_data_wr <= x"3E";
                WHEN 36 =>i2c_ena <= '1'; i2c_data_wr <= x"00";
                WHEN 37 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 38 => i2c_ena <= '1'; i2c_data_wr <= x"70";
                WHEN 39 =>i2c_ena <= '1'; i2c_data_wr <= x"3A";
                WHEN 40 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 41 => i2c_ena <= '1'; i2c_data_wr <= x"71";
                WHEN 42 =>i2c_ena <= '1'; i2c_data_wr <= x"35";
                WHEN 43 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 44 => i2c_ena <= '1'; i2c_data_wr <= x"72";
                WHEN 45 =>i2c_ena <= '1'; i2c_data_wr <= x"11";
                WHEN 46 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 47 => i2c_ena <= '1'; i2c_data_wr <= x"73";
                WHEN 48 =>i2c_ena <= '1'; i2c_data_wr <= x"F0";
                WHEN 49 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 50 => i2c_ena <= '1'; i2c_data_wr <= x"A2";
```

İkranur Yılmaz
22003768
EE102-04

```
                                WHEN 51 =>i2c_ena <= '1'; i2c_data_wr <= x"02";
                                WHEN 52 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                WHEN 53 => i2c_ena <= '1'; i2c_data_wr <= x"15";
                                WHEN 54 =>i2c_ena <= '1'; i2c_data_wr <= x"00";
                                WHEN 55 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                WHEN 56 => i2c_ena <= '1'; i2c_data_wr <= x"7a";
                                WHEN 57 =>i2c_ena <= '1'; i2c_data_wr <= x"20";
                                WHEN 58 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                WHEN 59 => i2c_ena <= '1'; i2c_data_wr <= x"7b";
                                WHEN 60 =>i2c_ena <= '1'; i2c_data_wr <= x"10";
                                WHEN 61 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                WHEN 62 => i2c_ena <= '1'; i2c_data_wr <= x"7c";
                                WHEN 63 =>i2c_ena <= '1'; i2c_data_wr <= x"1e";
                                WHEN 64 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                WHEN 65 => i2c_ena <= '1'; i2c_data_wr <= x"7d";
                                WHEN 66 =>i2c_ena <= '1'; i2c_data_wr <= x"35";
                                WHEN 67 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                WHEN 68 => i2c_ena <= '1'; i2c_data_wr <= x"7e";
                                WHEN 69 =>i2c_ena <= '1'; i2c_data_wr <= x"5a";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                                   WHEN 70 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                            if delay_ctr = 250_000 then
                                                     busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                            end if;
                                   WHEN 71 => i2c_ena <= '1'; i2c_data_wr <= x"7f";
                                   WHEN 72 =>i2c_ena <= '1'; i2c_data_wr <= x"69";
                                   WHEN 73 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                            if delay_ctr = 250_000 then
                                                     busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                            end if;
                                   WHEN 74 => i2c_ena <= '1'; i2c_data_wr <= x"80";
                                   WHEN 75 =>i2c_ena <= '1'; i2c_data_wr <= x"76";
                                   WHEN 76 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                            if delay_ctr = 250_000 then
                                                     busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                            end if;
                                   WHEN 77 => i2c_ena <= '1'; i2c_data_wr <= x"81";
                                   WHEN 78 =>i2c_ena <= '1'; i2c_data_wr <= x"80";
                                   WHEN 79 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                            if delay_ctr = 250_000 then
                                                     busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                            end if;
                                   WHEN 80 => i2c_ena <= '1'; i2c_data_wr <= x"82";
                                   WHEN 81 =>i2c_ena <= '1'; i2c_data_wr <= x"88";
                                   WHEN 82 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                            if delay_ctr = 250_000 then
                                                     busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                            end if;
                                   WHEN 83 => i2c_ena <= '1'; i2c_data_wr <= x"83";
                                   WHEN 84 =>i2c_ena <= '1'; i2c_data_wr <= x"8f";
                                   WHEN 85 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                            if delay_ctr = 250_000 then
                                                     busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                            end if;
                                   WHEN 86 => i2c_ena <= '1'; i2c_data_wr <= x"84";
                                   WHEN 87 =>i2c_ena <= '1'; i2c_data_wr <= x"96";
                                   WHEN 88 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                                        if delay_ctr = 250_000 then
                                              busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                  WHEN 89 => i2c_ena <= '1'; i2c_data_wr <= x"85";
                                  WHEN 90 =>i2c_ena <= '1'; i2c_data_wr <= x"a3";
                                  WHEN 91 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                              busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                  WHEN 92 => i2c_ena <= '1'; i2c_data_wr <= x"86";
                                  WHEN 93 =>i2c_ena <= '1'; i2c_data_wr <= x"af";
                                  WHEN 94 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                              busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                  WHEN 95 => i2c_ena <= '1'; i2c_data_wr <= x"87";
                                  WHEN 96 =>i2c_ena <= '1'; i2c_data_wr <= x"c4";
                                  WHEN 97 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                              busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                  WHEN 98 => i2c_ena <= '1'; i2c_data_wr <= x"88";
                                  WHEN 99 =>i2c_ena <= '1'; i2c_data_wr <= x"d7";
                                  WHEN 100 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                              busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                  WHEN 101 => i2c_ena <= '1'; i2c_data_wr <=
x"89";
                                  WHEN 102 =>i2c_ena <= '1'; i2c_data_wr <= x"e8";
                                  WHEN 103 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                              busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                                  WHEN 104 => i2c_ena <= '1'; i2c_data_wr <=
x"13";
                                  WHEN 105 =>i2c_ena <= '1'; i2c_data_wr <=
x"E0";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                WHEN 106 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 107 => i2c_ena <= '1'; i2c_data_wr <=
x"00";
                WHEN 108 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
                WHEN 109 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 110 => i2c_ena <= '1'; i2c_data_wr <=
x"10";
                WHEN 111 =>i2c_ena <= '1'; i2c_data_wr <= x"00";
                WHEN 112 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 113 => i2c_ena <= '1'; i2c_data_wr <=
x"0d";
                WHEN 114 =>i2c_ena <= '1'; i2c_data_wr <= x"40";
                WHEN 115 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 116 => i2c_ena <= '1'; i2c_data_wr <=
x"14";
                WHEN 117 =>i2c_ena <= '1'; i2c_data_wr <= x"18";
                WHEN 118 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 119 => i2c_ena <= '1'; i2c_data_wr <=
x"a5";
                WHEN 120 =>i2c_ena <= '1'; i2c_data_wr <=
x"05";
                WHEN 121 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
delay_ctr<=0;


x"ab";

x"07";

delay_ctr+1;


delay_ctr<=0;


x"24";

x"95";

delay_ctr+1;


delay_ctr<=0;


x"25";

x"33";

delay_ctr+1;


delay_ctr<=0;


x"26";

x"E3";

delay_ctr+1;


delay_ctr<=0;


x"9f";

x"78";

delay_ctr+1;
```

```
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 122 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 123 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 124 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 125 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 126 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 127 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 128 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 129 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 130 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 131 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 132 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 133 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 134 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 135 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 136 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 137 => i2c_ena <= '1'; i2c_data_wr <=
x"a0";
        WHEN 138 =>i2c_ena <= '1'; i2c_data_wr <=
x"68";
        WHEN 139 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 140 => i2c_ena <= '1'; i2c_data_wr <=
x"a1";
        WHEN 141 =>i2c_ena <= '1'; i2c_data_wr <=
x"03";
        WHEN 142 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 143 => i2c_ena <= '1'; i2c_data_wr <=
x"a6";
        WHEN 144 =>i2c_ena <= '1'; i2c_data_wr <=
x"d8";
        WHEN 145 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 146 => i2c_ena <= '1'; i2c_data_wr <=
x"a7";
        WHEN 147 =>i2c_ena <= '1'; i2c_data_wr <=
x"d8";
        WHEN 148 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 149 => i2c_ena <= '1'; i2c_data_wr <=
x"a8";
        WHEN 150 =>i2c_ena <= '1'; i2c_data_wr <= x"f0";
        WHEN 151 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 152 => i2c_ena <= '1'; i2c_data_wr <=
x"a9";
WHEN 153 =>i2c_ena <= '1'; i2c_data_wr <=
x"90";
WHEN 154 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 155 => i2c_ena <= '1'; i2c_data_wr <=
x"aa";
WHEN 156 =>i2c_ena <= '1'; i2c_data_wr <=
x"94";
WHEN 157 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 158 => i2c_ena <= '1'; i2c_data_wr <=
x"13";
WHEN 159 =>i2c_ena <= '1'; i2c_data_wr <= x"e5";
WHEN 160 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 161 => i2c_ena <= '1'; i2c_data_wr <=
x"0e";
WHEN 162 =>i2c_ena <= '1'; i2c_data_wr <=
x"61";
WHEN 163 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 164 => i2c_ena <= '1'; i2c_data_wr <=
x"0f";
WHEN 165 =>i2c_ena <= '1'; i2c_data_wr <=
x"4b";
WHEN 166 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                              WHEN 167 => i2c_ena <= '1'; i2c_data_wr <=
x"16";
                              WHEN 168 =>i2c_ena <= '1'; i2c_data_wr <=
x"02";
                              WHEN 169 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                              WHEN 170 => i2c_ena <= '1'; i2c_data_wr <=
x"1e";
                              WHEN 171 =>i2c_ena <= '1'; i2c_data_wr <=
x"27";
                              WHEN 172 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                              WHEN 173 => i2c_ena <= '1'; i2c_data_wr <=
x"21";
                              WHEN 174 =>i2c_ena <= '1'; i2c_data_wr <=
x"02";
                              WHEN 175 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                              WHEN 176 => i2c_ena <= '1'; i2c_data_wr <=
x"22";
                              WHEN 177 =>i2c_ena <= '1'; i2c_data_wr <=
x"91";
                              WHEN 178 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
                                                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                        end if;
                              WHEN 179 => i2c_ena <= '1'; i2c_data_wr <=
x"29";
                              WHEN 180 =>i2c_ena <= '1'; i2c_data_wr <=
x"07";
                              WHEN 181 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                        if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
        WHEN 182 => i2c_ena <= '1'; i2c_data_wr <=
x"33";
        WHEN 183 =>i2c_ena <= '1'; i2c_data_wr <=
x"0b";
        WHEN 184 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
        WHEN 185 => i2c_ena <= '1'; i2c_data_wr <=
x"35";
        WHEN 186 =>i2c_ena <= '1'; i2c_data_wr <=
x"0b";
        WHEN 187 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
        WHEN 188 => i2c_ena <= '1'; i2c_data_wr <=
x"37";
        WHEN 189 =>i2c_ena <= '1'; i2c_data_wr <=
x"1d";
        WHEN 190 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
        WHEN 191 => i2c_ena <= '1'; i2c_data_wr <=
x"38";
        WHEN 192 =>i2c_ena <= '1'; i2c_data_wr <=
x"71";
        WHEN 193 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
        WHEN 194 => i2c_ena <= '1'; i2c_data_wr <=
x"39";
        WHEN 195 =>i2c_ena <= '1'; i2c_data_wr <= x"2a";
        WHEN 196 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
delay_ctr<=0;


x"3c";

x"78";

delay_ctr+1;


delay_ctr<=0;


x"4d";

x"40";

delay_ctr+1;


delay_ctr<=0;


x"4e";

x"20";

delay_ctr+1;


delay_ctr<=0;


x"69";

x"00";

delay_ctr+1;


delay_ctr<=0;


x"6b";


delay_ctr+1;
```

```
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 197 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 198 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 199 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 200 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 201 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 202 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 203 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 204 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 205 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 206 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 207 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 208 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 209 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 210 =>i2c_ena <= '1'; i2c_data_wr <= x"4a";
WHEN 211 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
delay_ctr<=0;


x"74";

x"10";

delay_ctr+1;


delay_ctr<=0;


x"8d";


delay_ctr+1;


delay_ctr<=0;


x"8e";

x"00";

delay_ctr+1;


delay_ctr<=0;


x"8f";

x"00";

delay_ctr+1;


delay_ctr<=0;


x"90";

x"00";

delay_ctr+1;
```

```
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 212 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 213 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 214 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 215 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 216 =>i2c_ena <= '1'; i2c_data_wr <= x"4f";
WHEN 217 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 218 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 219 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 220 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 221 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 222 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 223 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 224 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 225 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 226 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
delay_ctr<=0;


x"91";

x"00";

delay_ctr+1;


delay_ctr<=0;


x"96";

x"00";

delay_ctr+1;


delay_ctr<=0;


x"9a";

x"00";

delay_ctr+1;


delay_ctr<=0;


x"b0";

x"84";

delay_ctr+1;


delay_ctr<=0;


x"b1";


delay_ctr+1;
```

```
                        busy_cnt <= busy_cnt+1;

                end if;
WHEN 227 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 228 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 229 => i2c_ena <= '0'; delay_ctr <=

                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;

                end if;
WHEN 230 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 231 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 232 => i2c_ena <= '0'; delay_ctr <=

                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;

                end if;
WHEN 233 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 234 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 235 => i2c_ena <= '0'; delay_ctr <=

                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;

                end if;
WHEN 236 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 237 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 238 => i2c_ena <= '0'; delay_ctr <=

                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;

                end if;
WHEN 239 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 240 =>i2c_ena <= '1'; i2c_data_wr <= x"0c";
WHEN 241 => i2c_ena <= '0'; delay_ctr <=

                if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 242 => i2c_ena <= '1'; i2c_data_wr <=
x"b2";
WHEN 243 =>i2c_ena <= '1'; i2c_data_wr <= x"0e";
WHEN 244 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 245 => i2c_ena <= '1'; i2c_data_wr <=
x"b3";
WHEN 246 =>i2c_ena <= '1'; i2c_data_wr <=
x"82";
WHEN 247 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 248 => i2c_ena <= '1'; i2c_data_wr <=
x"b8";
WHEN 249 =>i2c_ena <= '1'; i2c_data_wr <= x"0a";
WHEN 250 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 251 => i2c_ena <= '1'; i2c_data_wr <=
x"43";
WHEN 252 =>i2c_ena <= '1'; i2c_data_wr <= x"0a";
WHEN 253 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 254 => i2c_ena <= '1'; i2c_data_wr <=
x"44";
WHEN 255 =>i2c_ena <= '1'; i2c_data_wr <= x"f0";
WHEN 256 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 257 => i2c_ena <= '1'; i2c_data_wr <=
x"45";
WHEN 258 =>i2c_ena <= '1'; i2c_data_wr <=
x"34";
WHEN 259 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 260 => i2c_ena <= '1'; i2c_data_wr <=
x"46";
WHEN 261 =>i2c_ena <= '1'; i2c_data_wr <=
x"58";
WHEN 262 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 263 => i2c_ena <= '1'; i2c_data_wr <=
x"47";
WHEN 264 =>i2c_ena <= '1'; i2c_data_wr <=
x"28";
WHEN 265 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 266 => i2c_ena <= '1'; i2c_data_wr <=
x"48";
WHEN 267 =>i2c_ena <= '1'; i2c_data_wr <= x"3a";
WHEN 268 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 269 => i2c_ena <= '1'; i2c_data_wr <=
x"59";
WHEN 270 =>i2c_ena <= '1'; i2c_data_wr <=
x"88";
WHEN 271 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
x"5a";

x"88";

delay_ctr+1;


delay_ctr<=0;


x"5b";

x"44";

delay_ctr+1;


delay_ctr<=0;


x"5c";

x"67";

delay_ctr+1;


delay_ctr<=0;


x"5d";

x"49";

delay_ctr+1;


delay_ctr<=0;


x"5e";


delay_ctr+1;


delay_ctr<=0;
```

```
WHEN 272 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 273 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 274 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 275 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 276 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 277 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 278 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 279 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 280 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 281 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 282 =>i2c_ena <= '1'; i2c_data_wr <=

WHEN 283 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
WHEN 284 => i2c_ena <= '1'; i2c_data_wr <=

WHEN 285 =>i2c_ena <= '1'; i2c_data_wr <= x"0e";
WHEN 286 => i2c_ena <= '0'; delay_ctr <=

          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;

          end if;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 287 => i2c_ena <= '1'; i2c_data_wr <=
x"6c";
WHEN 288 =>i2c_ena <= '1'; i2c_data_wr <= x"0a";
WHEN 289 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 290 => i2c_ena <= '1'; i2c_data_wr <=
x"6d";
WHEN 291 =>i2c_ena <= '1'; i2c_data_wr <=
x"55";
WHEN 292 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 293 => i2c_ena <= '1'; i2c_data_wr <=
x"6e";
WHEN 294 =>i2c_ena <= '1'; i2c_data_wr <= x"11";
WHEN 295 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 296 => i2c_ena <= '1'; i2c_data_wr <=
x"6f";
WHEN 297 =>i2c_ena <= '1'; i2c_data_wr <= x"9f";
WHEN 298 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 299 => i2c_ena <= '1'; i2c_data_wr <=
x"6a";
WHEN 300 =>i2c_ena <= '1'; i2c_data_wr <=
x"40";
WHEN 301 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 302 => i2c_ena <= '1'; i2c_data_wr <=
x"01";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 303 =>i2c_ena <= '1'; i2c_data_wr <=
x"40";
WHEN 304 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 305 => i2c_ena <= '1'; i2c_data_wr <=
x"02";
WHEN 306 =>i2c_ena <= '1'; i2c_data_wr <=
x"60";
WHEN 307 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 308 => i2c_ena <= '1'; i2c_data_wr <=
x"13";
WHEN 309 =>i2c_ena <= '1'; i2c_data_wr <= x"e7";
WHEN 310 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 311 => i2c_ena <= '1'; i2c_data_wr <=
x"4f";
WHEN 312 =>i2c_ena <= '1'; i2c_data_wr <=
x"80";
WHEN 313 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 314 => i2c_ena <= '1'; i2c_data_wr <=
x"50";
WHEN 315 =>i2c_ena <= '1'; i2c_data_wr <=
x"80";
WHEN 316 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 317 => i2c_ena <= '1'; i2c_data_wr <=
x"51";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                              WHEN 318 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
                              WHEN 319 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 320 => i2c_ena <= '1'; i2c_data_wr <=
x"52";
                              WHEN 321 =>i2c_ena <= '1'; i2c_data_wr <=
x"22";
                              WHEN 322 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 323 => i2c_ena <= '1'; i2c_data_wr <=
x"53";
                              WHEN 324 =>i2c_ena <= '1'; i2c_data_wr <= x"5e";
                              WHEN 325 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 326 => i2c_ena <= '1'; i2c_data_wr <=
x"54";
                              WHEN 327 =>i2c_ena <= '1'; i2c_data_wr <=
x"80";
                              WHEN 328 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 329 => i2c_ena <= '1'; i2c_data_wr <=
x"58";
                              WHEN 330 =>i2c_ena <= '1'; i2c_data_wr <= x"9e";
                              WHEN 331 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 332 => i2c_ena <= '1'; i2c_data_wr <=
x"41";
                              WHEN 333 =>i2c_ena <= '1'; i2c_data_wr <=
x"08";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                              WHEN 334 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 335 => i2c_ena <= '1'; i2c_data_wr <=
x"3f";
                              WHEN 336 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
                              WHEN 337 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 338 => i2c_ena <= '1'; i2c_data_wr <=
x"75";
                              WHEN 339 =>i2c_ena <= '1'; i2c_data_wr <=
x"05";
                              WHEN 340 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 341 => i2c_ena <= '1'; i2c_data_wr <=
x"76";
                              WHEN 342 =>i2c_ena <= '1'; i2c_data_wr <= x"e1";
                              WHEN 343 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 344 => i2c_ena <= '1'; i2c_data_wr <=
x"4c";
                              WHEN 345 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
                              WHEN 346 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                  if delay_ctr = 250_000 then
                                      busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                  end if;
                              WHEN 347 => i2c_ena <= '1'; i2c_data_wr <=
x"77";
                              WHEN 348 =>i2c_ena <= '1'; i2c_data_wr <=
x"01";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 349 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 350 => i2c_ena <= '1'; i2c_data_wr <=
x"3d";
WHEN 351 =>i2c_ena <= '1'; i2c_data_wr <= x"c3";
WHEN 352 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 353 => i2c_ena <= '1'; i2c_data_wr <=
x"4b";
WHEN 354 =>i2c_ena <= '1'; i2c_data_wr <=
x"09";
WHEN 355 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 356 => i2c_ena <= '1'; i2c_data_wr <=
x"c9";
WHEN 357 =>i2c_ena <= '1'; i2c_data_wr <=
x"60";
WHEN 358 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 359 => i2c_ena <= '1'; i2c_data_wr <=
x"41";
WHEN 360 =>i2c_ena <= '1'; i2c_data_wr <=
x"38";
WHEN 361 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 362 => i2c_ena <= '1'; i2c_data_wr <=
x"56";
WHEN 363 =>i2c_ena <= '1'; i2c_data_wr <=
x"40";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 364 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
WHEN 365 => i2c_ena <= '1'; i2c_data_wr <=
x"34";
WHEN 366 =>i2c_ena <= '1'; i2c_data_wr <= x"11";
WHEN 367 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
WHEN 368 => i2c_ena <= '1'; i2c_data_wr <=
x"3b";
WHEN 369 =>i2c_ena <= '1'; i2c_data_wr <=
x"12";
WHEN 370 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
WHEN 371 => i2c_ena <= '1'; i2c_data_wr <=
x"a4";
WHEN 372 =>i2c_ena <= '1'; i2c_data_wr <=
x"88";
WHEN 373 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
WHEN 374 => i2c_ena <= '1'; i2c_data_wr <=
x"96";
WHEN 375 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
WHEN 376 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
WHEN 377 => i2c_ena <= '1'; i2c_data_wr <=
x"97";
WHEN 378 =>i2c_ena <= '1'; i2c_data_wr <=
x"30";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 379 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 380 => i2c_ena <= '1'; i2c_data_wr <=
x"98";
WHEN 381 =>i2c_ena <= '1'; i2c_data_wr <=
x"20";
WHEN 382 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 383 => i2c_ena <= '1'; i2c_data_wr <=
x"99";
WHEN 384 =>i2c_ena <= '1'; i2c_data_wr <=
x"30";
WHEN 385 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 386 => i2c_ena <= '1'; i2c_data_wr <=
x"9a";
WHEN 387 =>i2c_ena <= '1'; i2c_data_wr <=
x"84";
WHEN 388 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 389 => i2c_ena <= '1'; i2c_data_wr <=
x"9b";
WHEN 390 =>i2c_ena <= '1'; i2c_data_wr <=
x"29";
WHEN 391 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 392 => i2c_ena <= '1'; i2c_data_wr <=
x"9c";
WHEN 393 =>i2c_ena <= '1'; i2c_data_wr <=
x"03";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
        WHEN 394 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 395 => i2c_ena <= '1'; i2c_data_wr <=
x"9d";
        WHEN 396 =>i2c_ena <= '1'; i2c_data_wr <= x"4c";
        WHEN 397 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 398 => i2c_ena <= '1'; i2c_data_wr <=
x"9e";
        WHEN 399 =>i2c_ena <= '1'; i2c_data_wr <= x"3f";
        WHEN 400 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 401 => i2c_ena <= '1'; i2c_data_wr <=
x"78";
        WHEN 402 =>i2c_ena <= '1'; i2c_data_wr <=
x"04";
        WHEN 403 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 404 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
        WHEN 405 =>i2c_ena <= '1'; i2c_data_wr <=
x"01";
        WHEN 406 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
        WHEN 407 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
        WHEN 408 =>i2c_ena <= '1'; i2c_data_wr <= x"f0";
        WHEN 409 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
            WHEN 410 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
            WHEN 411 =>i2c_ena <= '1'; i2c_data_wr <= x"0f";
            WHEN 412 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
            WHEN 413 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
            WHEN 414 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
            WHEN 415 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
            WHEN 416 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
            WHEN 417 =>i2c_ena <= '1'; i2c_data_wr <=
x"10";
            WHEN 418 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
            WHEN 419 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
            WHEN 420 =>i2c_ena <= '1'; i2c_data_wr <= x"7e";
            WHEN 421 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
            WHEN 422 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
            WHEN 423 =>i2c_ena <= '1'; i2c_data_wr <= x"0a";
            WHEN 424 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                end if;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 425 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
WHEN 426 =>i2c_ena <= '1'; i2c_data_wr <=
x"80";
WHEN 427 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 428 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
WHEN 429 =>i2c_ena <= '1'; i2c_data_wr <=
x"0b";
WHEN 430 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 431 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
WHEN 432 =>i2c_ena <= '1'; i2c_data_wr <=
x"01";
WHEN 433 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 434 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
WHEN 435 =>i2c_ena <= '1'; i2c_data_wr <= x"0c";
WHEN 436 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 437 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
WHEN 438 =>i2c_ena <= '1'; i2c_data_wr <= x"0f";
WHEN 439 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
        if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
        end if;
WHEN 440 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
                WHEN 441 =>i2c_ena <= '1'; i2c_data_wr <=
x"0d";
                WHEN 442 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 443 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
                WHEN 444 =>i2c_ena <= '1'; i2c_data_wr <=
x"20";
                WHEN 445 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 446 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
                WHEN 447 =>i2c_ena <= '1'; i2c_data_wr <=
x"09";
                WHEN 448 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 449 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
                WHEN 450 =>i2c_ena <= '1'; i2c_data_wr <=
x"80";
                WHEN 451 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 452 => i2c_ena <= '1'; i2c_data_wr <=
x"79";
                WHEN 453 =>i2c_ena <= '1'; i2c_data_wr <=
x"02";
                WHEN 454 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                    if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                    end if;
                WHEN 455 => i2c_ena <= '1'; i2c_data_wr <=
x"c8";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 456 =>i2c_ena <= '1'; i2c_data_wr <= x"c0";
WHEN 457 => i2c_ena <= '0'; delay_ctr <= delay_ctr+1;
            if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
                        delay_ctr<=0;
            end if;
WHEN 458 => i2c_ena <= '1'; i2c_data_wr <= x"79";
WHEN 459 =>i2c_ena <= '1'; i2c_data_wr <= x"03";
WHEN 460 => i2c_ena <= '0'; delay_ctr <= delay_ctr+1;
            if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
                        delay_ctr<=0;
            end if;
WHEN 461 => i2c_ena <= '1'; i2c_data_wr <= x"c8";
WHEN 462 =>i2c_ena <= '1'; i2c_data_wr <= x"40";
WHEN 463 => i2c_ena <= '0'; delay_ctr <= delay_ctr+1;
            if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
                        delay_ctr<=0;
            end if;
WHEN 464 => i2c_ena <= '1'; i2c_data_wr <= x"79";
WHEN 465 =>i2c_ena <= '1'; i2c_data_wr <= x"05";
WHEN 466 => i2c_ena <= '0'; delay_ctr <= delay_ctr+1;
            if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
                        delay_ctr<=0;
            end if;
WHEN 467 => i2c_ena <= '1'; i2c_data_wr <= x"c8";
WHEN 468 =>i2c_ena <= '1'; i2c_data_wr <= x"30";
WHEN 469 => i2c_ena <= '0'; delay_ctr <= delay_ctr+1;
            if delay_ctr = 250_000 then
                        busy_cnt <= busy_cnt+1;
                        delay_ctr<=0;
            end if;
WHEN 470 => i2c_ena <= '1'; i2c_data_wr <= x"79";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 471 =>i2c_ena <= '1'; i2c_data_wr <=
x"26";
WHEN 472 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
WHEN 473 => i2c_ena <= '1'; i2c_data_wr <=
x"12";
WHEN 474 =>i2c_ena <= '1'; i2c_data_wr <=
x"04";
WHEN 475 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
WHEN 476 => i2c_ena <= '1'; i2c_data_wr <=
x"8C";
WHEN 477 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
WHEN 478 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
WHEN 479 => i2c_ena <= '1'; i2c_data_wr <=
x"04";
WHEN 480 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
WHEN 481 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
WHEN 482 => i2c_ena <= '1'; i2c_data_wr <=
x"40";
WHEN 483 =>i2c_ena <= '1'; i2c_data_wr <=
x"10";
WHEN 484 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
            if delay_ctr = 250_000 then
                busy_cnt <= busy_cnt+1;
delay_ctr<=0;
            end if;
WHEN 485 => i2c_ena <= '1'; i2c_data_wr <=
x"14";
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN 486 =>i2c_ena <= '1'; i2c_data_wr <=
x"38";
WHEN 487 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 488 => i2c_ena <= '1'; i2c_data_wr <=
x"4F";
WHEN 489 =>i2c_ena <= '1'; i2c_data_wr <=
x"B3";
WHEN 490 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 491 => i2c_ena <= '1'; i2c_data_wr <=
x"50";
WHEN 492 =>i2c_ena <= '1'; i2c_data_wr <=
x"B3";
WHEN 493 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 494 => i2c_ena <= '1'; i2c_data_wr <=
x"51";
WHEN 495 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
WHEN 496 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 497 => i2c_ena <= '1'; i2c_data_wr <=
x"52";
WHEN 498 =>i2c_ena <= '1'; i2c_data_wr <=
x"3D";
WHEN 499 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
          if delay_ctr = 250_000 then
                    busy_cnt <= busy_cnt+1;
delay_ctr<=0;
          end if;
WHEN 500 => i2c_ena <= '1'; i2c_data_wr <=
x"53";
```

Codes written in red were found on the internet

```
                                        WHEN 501 =>i2c_ena <= '1'; i2c_data_wr <=
x"A7";
                                        WHEN 502 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                             if delay_ctr = 250_000 then
                                                  busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                             end if;
                                        WHEN 503 => i2c_ena <= '1'; i2c_data_wr <=
x"54";
                                        WHEN 504 =>i2c_ena <= '1'; i2c_data_wr <=
x"E4";
                                        WHEN 505 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                             if delay_ctr = 250_000 then
                                                  busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                             end if;
                                        WHEN 506 => i2c_ena <= '1'; i2c_data_wr <=
x"3D";
                                        WHEN 507 =>i2c_ena <= '1'; i2c_data_wr <=
x"C0";
                      WHEN 508 => i2c_ena <= '0'; delay_ctr <= delay_ctr+1;
                                             if delay_ctr = 250_000 then
                                                  busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                             end if;
                                        WHEN 509 => i2c_ena <= '1'; i2c_data_wr <=
x"13";
                                        WHEN 510 =>i2c_ena <= '1'; i2c_data_wr <=
x"00";
                                        WHEN 511 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                             if delay_ctr = 250_000 then
                                                  busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                             end if;
                                        WHEN 512 => i2c_ena <= '1'; i2c_data_wr <=
x"11";
                                        WHEN 513 =>i2c_ena <= '1'; i2c_data_wr <=
x"02";
                                        WHEN 514 => i2c_ena <= '0'; delay_ctr <=
delay_ctr+1;
                                             if delay_ctr = 250_000 then
                                                  busy_cnt <= busy_cnt+1;
delay_ctr<=0;
                                             end if;
                                        WHEN OTHERS => i2c_ena <= '0';
                                   END CASE;
                   else
```

Codes written in red were found on the internet

```
                    i2c_ena <='0';
                    busy_cnt <= 0;
                end if;
            end if;
        end process;
    end Behavioral;
```

i2c_master.vhd

```
--------------------------------------------------------------------------------
--
--   FileName:         i2c_master.vhd
--   Dependencies:     none
--   Design Software:  Quartus II 64-bit Version 13.1 Build 162 SJ Full Version
--
--   HDL CODE IS PROVIDED "AS IS."  DIGI-KEY EXPRESSLY DISCLAIMS ANY
--   WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
--   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
--   PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DIGI-KEY
--   BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL
--   DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF
--   PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS
--   BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),
--   ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
--
--   Version History
--   Version 1.0 11/01/2012 Scott Larson
--     Initial Public Release
--   Version 2.0 06/20/2014 Scott Larson
--     Added ability to interface with different slaves in the same transaction
--     Corrected ack_error bug where ack_error went 'Z' instead of '1' on error
--     Corrected timing of when ack_error signal clears
--   Version 2.1 10/21/2014 Scott Larson
--     Replaced gated clock with clock enable
--     Adjusted timing of SCL during start and stop conditions
--   Version 2.2 02/05/2015 Scott Larson
--     Corrected small SDA glitch introduced in version 2.1
--
--------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
```

İkranur Yılmaz
22003768
EE102-04

```
ENTITY i2c_master IS
  GENERIC(
    input_clk : INTEGER := 50_000_000; --input clock speed from user logic in Hz
    bus_clk   : INTEGER := 400_000);   --speed the i2c bus (scl) will run at in Hz
  PORT(
    clk       : IN     STD_LOGIC;                    --system clock
    reset_n   : IN     STD_LOGIC;                    --active low reset
    ena       : IN     STD_LOGIC;                    --latch in command
    addr      : IN     STD_LOGIC_VECTOR(6 DOWNTO 0); --address of target slave
    rw        : IN     STD_LOGIC;                    --'0' is write, '1' is read
    data_wr   : IN     STD_LOGIC_VECTOR(7 DOWNTO 0); --data to write to slave
    busy      : OUT    STD_LOGIC;                    --indicates transaction in progress
    data_rd   : OUT    STD_LOGIC_VECTOR(7 DOWNTO 0); --data read from slave
    ack_error : BUFFER STD_LOGIC;                    --flag if improper acknowledge from slave
    sda       : INOUT  STD_LOGIC;                    --serial data output of i2c bus
    scl       : INOUT  STD_LOGIC);                   --serial clock output of i2c bus
END i2c_master;

ARCHITECTURE logic OF i2c_master IS
  CONSTANT divider  :  INTEGER := (input_clk/bus_clk)/4; --number of clocks in 1/4 cycle 
of scl
  TYPE machine IS(ready, start, command, slv_ack1, wr, rd, slv_ack2, mstr_ack, stop); --
needed states
  SIGNAL state         : machine;                        --state machine
  SIGNAL data_clk      : STD_LOGIC;                      --data clock for sda
  SIGNAL data_clk_prev : STD_LOGIC;                      --data clock during previous system 
clock
  SIGNAL scl_clk       : STD_LOGIC;                      --constantly running internal scl
  SIGNAL scl_ena       : STD_LOGIC := '0';               --enables internal scl to output
  SIGNAL sda_int       : STD_LOGIC := '1';               --internal sda
  SIGNAL sda_ena_n     : STD_LOGIC;                      --enables internal sda to output
  SIGNAL addr_rw       : STD_LOGIC_VECTOR(7 DOWNTO 0);   --latched in address and 
read/write
  SIGNAL data_tx       : STD_LOGIC_VECTOR(7 DOWNTO 0);   --latched in data to write 
to slave
  SIGNAL data_rx       : STD_LOGIC_VECTOR(7 DOWNTO 0);   --data received from slave
  SIGNAL bit_cnt       : INTEGER RANGE 0 TO 7 := 7;      --tracks bit number in transaction
  SIGNAL stretch       : STD_LOGIC := '0';               --identifies if slave is stretching scl
BEGIN

  --generate the timing for the bus clock (scl_clk) and the data clock (data_clk)
  PROCESS(clk, reset_n)
    VARIABLE count  :  INTEGER RANGE 0 TO divider*4;  --timing for clock generation
  BEGIN
    IF(reset_n = '0') THEN                --reset asserted
      stretch <= '0';
      count := 0;
    ELSIF(clk'EVENT AND clk = '1') THEN
      data_clk_prev <= data_clk;          --store previous value of data clock
      IF(count = divider*4-1) THEN        --end of timing cycle
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
        count := 0;                     --reset timer
      ELSIF(stretch = '0') THEN           --clock stretching from slave not detected
        count := count + 1;               --continue clock generation timing
      END IF;
      CASE count IS
        WHEN 0 TO divider-1 =>            --first 1/4 cycle of clocking
          scl_clk <= '0';
          data_clk <= '0';
        WHEN divider TO divider*2-1 =>    --second 1/4 cycle of clocking
          scl_clk <= '0';
          data_clk <= '1';
        WHEN divider*2 TO divider*3-1 =>  --third 1/4 cycle of clocking
          scl_clk <= '1';                 --release scl
          IF(scl = '0') THEN              --detect if slave is stretching clock
            stretch <= '1';
          ELSE
            stretch <= '0';
          END IF;
          data_clk <= '1';
        WHEN OTHERS =>                    --last 1/4 cycle of clocking
          scl_clk <= '1';
          data_clk <= '0';
      END CASE;
    END IF;
  END PROCESS;

  --state machine and writing to sda during scl low (data_clk rising edge)
  PROCESS(clk, reset_n)
  BEGIN
    IF(reset_n = '0') THEN                 --reset asserted
      state <= ready;                      --return to initial state
      busy <= '1';                         --indicate not available
      scl_ena <= '0';                      --sets scl high impedance
      sda_int <= '1';                      --sets sda high impedance
      ack_error <= '0';                    --clear acknowledge error flag
      bit_cnt <= 7;                        --restarts data bit counter
      data_rd <= "00000000";               --clear data read port
    ELSIF(clk'EVENT AND clk = '1') THEN
      IF(data_clk = '1' AND data_clk_prev = '0') THEN  --data clock rising edge
        CASE state IS
          WHEN ready =>                    --idle state
            IF(ena = '1') THEN             --transaction requested
              busy <= '1';                 --flag busy
              addr_rw <= addr & rw;        --collect requested slave address and command
              data_tx <= data_wr;          --collect requested data to write
              state <= start;              --go to start bit
            ELSE                           --remain idle
              busy <= '0';                 --unflag busy
              state <= ready;              --remain idle
            END IF;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
WHEN start =>                    --start bit of transaction
  busy <= '1';                   --resume busy if continuous mode
  sda_int <= addr_rw(bit_cnt);     --set first address bit to bus
  state <= command;                --go to command
WHEN command =>                      --address and command byte of transaction
  IF(bit_cnt = 0) THEN               --command transmit finished
    sda_int <= '1';               --release sda for slave acknowledge
    bit_cnt <= 7;                 --reset bit counter for "byte" states
    state <= slv_ack1;              --go to slave acknowledge (command)
  ELSE                          --next clock cycle of command state
    bit_cnt <= bit_cnt - 1;       --keep track of transaction bits
    sda_int <= addr_rw(bit_cnt-1); --write address/command bit to bus
    state <= command;               --continue with command
  END IF;
WHEN slv_ack1 =>                    --slave acknowledge bit (command)
  IF(addr_rw(0) = '0') THEN        --write command
    sda_int <= data_tx(bit_cnt);   --write first bit of data
    state <= wr;                  --go to write byte
  ELSE                          --read command
    sda_int <= '1';               --release sda from incoming data
    state <= rd;                  --go to read byte
  END IF;
WHEN wr =>                       --write byte of transaction
  busy <= '1';                   --resume busy if continuous mode
  IF(bit_cnt = 0) THEN               --write byte transmit finished
    sda_int <= '1';               --release sda for slave acknowledge
    bit_cnt <= 7;                 --reset bit counter for "byte" states
    state <= slv_ack2;              --go to slave acknowledge (write)
  ELSE                          --next clock cycle of write state
    bit_cnt <= bit_cnt - 1;       --keep track of transaction bits
    sda_int <= data_tx(bit_cnt-1); --write next bit to bus
    state <= wr;                  --continue writing
  END IF;
WHEN rd =>                       --read byte of transaction
  busy <= '1';                   --resume busy if continuous mode
  IF(bit_cnt = 0) THEN               --read byte receive finished
    IF(ena = '1' AND addr_rw = addr & rw) THEN  --continuing with another read at
same address
      sda_int <= '0';             --acknowledge the byte has been received
    ELSE                        --stopping or continuing with a write
      sda_int <= '1';             --send a no-acknowledge (before stop or repeated start)
    END IF;
    bit_cnt <= 7;                 --reset bit counter for "byte" states
    data_rd <= data_rx;             --output received data
    state <= mstr_ack;             --go to master acknowledge
  ELSE                          --next clock cycle of read state
    bit_cnt <= bit_cnt - 1;       --keep track of transaction bits
    state <= rd;                  --continue reading
  END IF;
WHEN slv_ack2 =>                    --slave acknowledge bit (write)
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
          IF(ena = '1') THEN              --continue transaction
            busy <= '0';                  --continue is accepted
            addr_rw <= addr & rw;         --collect requested slave address and command
            data_tx <= data_wr;           --collect requested data to write
            IF(addr_rw = addr & rw) THEN   --continue transaction with another write
              sda_int <= data_wr(bit_cnt); --write first bit of data
              state <= wr;                --go to write byte
            ELSE                          --continue transaction with a read or new slave
              state <= start;             --go to repeated start
            END IF;
          ELSE                            --complete transaction
            state <= stop;                --go to stop bit
          END IF;
        WHEN mstr_ack =>                    --master acknowledge bit after a read
          IF(ena = '1') THEN              --continue transaction
            busy <= '0';                  --continue is accepted and data received is available on bus
            addr_rw <= addr & rw;         --collect requested slave address and command
            data_tx <= data_wr;           --collect requested data to write
            IF(addr_rw = addr & rw) THEN   --continue transaction with another read
              sda_int <= '1';             --release sda from incoming data
              state <= rd;                --go to read byte
            ELSE                          --continue transaction with a write or new slave
              state <= start;             --repeated start
            END IF;
          ELSE                            --complete transaction
            state <= stop;                --go to stop bit
          END IF;
        WHEN stop =>                      --stop bit of transaction
          busy <= '0';                    --unflag busy
          state <= ready;                 --go to idle state
      END CASE;
    ELSIF(data_clk = '0' AND data_clk_prev = '1') THEN  --data clock falling edge
      CASE state IS
        WHEN start =>
          IF(scl_ena = '0') THEN                  --starting new transaction
            scl_ena <= '1';                       --enable scl output
            ack_error <= '0';                     --reset acknowledge error output
          END IF;
        WHEN slv_ack1 =>                          --receiving slave acknowledge (command)
          IF(sda /= '0' OR ack_error = '1') THEN  --no-acknowledge or previous no-
acknowledge
            ack_error <= '1';                     --set error output if no-acknowledge
          END IF;
        WHEN rd =>                                --receiving slave data
          data_rx(bit_cnt) <= sda;                --receive current slave data bit
        WHEN slv_ack2 =>                          --receiving slave acknowledge (write)
          IF(sda /= '0' OR ack_error = '1') THEN  --no-acknowledge or previous no-
acknowledge
            ack_error <= '1';                     --set error output if no-acknowledge
          END IF;
```

Codes written in red were found on the internet

İkranur Yılmaz
22003768
EE102-04

```
          WHEN stop =>
            scl_ena <= '0';                      --disable scl
          WHEN OTHERS =>
            NULL;
        END CASE;
      END IF;
    END IF;
  END PROCESS;

  --set sda output
  WITH state SELECT
    sda_ena_n <= data_clk_prev WHEN start,     --generate start condition
                 NOT data_clk_prev WHEN stop,  --generate stop condition
                 sda_int WHEN OTHERS;          --set to internal sda signal

  --set scl and sda outputs
  scl <= '0' WHEN (scl_ena = '1' AND scl_clk = '0') ELSE 'Z';
  sda <= '0' WHEN sda_ena_n = '0' ELSE 'Z';

END logic;
```

Codes written in red were found on the internet