Camera Video Capture FPGA Description

June 2012

CuteDigi Technologies, Inc.

1. Function:

   Camera captures the image and the images are displayed on the LCD. There is no need for SD card and SD card controller.

2. Development Environment

   Software: QUARTUSII 9 + NIOSII IDE 9

   Hardware: FPGA core board, LCD bridge board, LCD core board, OV7670 camera board, 5V wall adapter.

3. Design Description:

   Create NIOSII /F in FPGA, configure CFI Flash controller, SDRAM controller, and TFT and OV7670 pin assignment.

   The project should configure such that the FPGA configuration file is stored in EPCS4, and NIOSII IDE firmware is stored in CFI Flash. When powered up, FPGA will read FPGA configuration from EPCS4, and after FPGA is configured properly, the reset pointer is pointed to external CFI flash. Because in NIOSII IDE, the program reset run starts from SDRAM, after NIOSII/F reset, the program is copied from CFI Flash to SDRAM, then the program runs from SDRAM.

   The firmware will first output FPGA, TFT and OV7670 initialization information. After that, it will display the embedded image. It the camera is initialized correctly, the captured live image 320*240 QVGA

RGB565 will be displayed on the TFT.

In the initialization stage, NIOSII controls TFT. I2C_CCD_Config.v module controls OV7670 IIC bus to do the initialization. In NIOSII IDE code, OV7670 initialization code is still kept, but the top module didn't assign the NIOSII IIC bus to the IIC bus of OV7670.

After the embedded image is displayed on the LCD, the control of OV7670 and TFT is moved to cmos_top.v. In NIOSII IDE, the code is as follows:

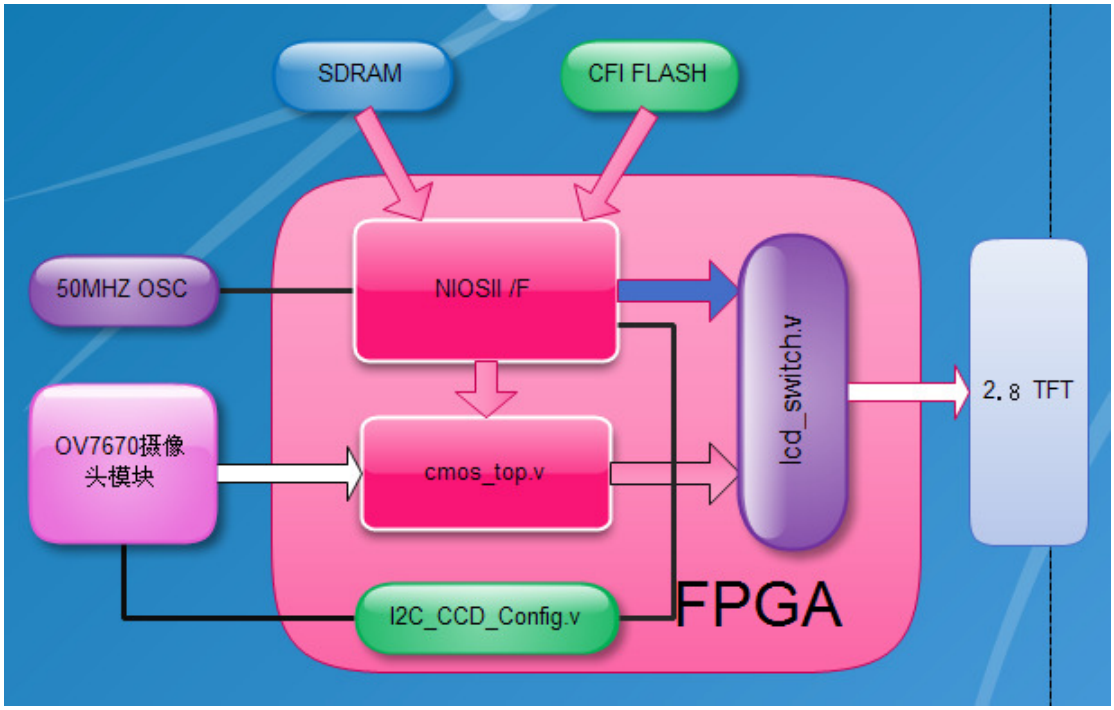
```
    LCD_Init();
    Address_set(0,0,239,319);
    IOWR_ALTERA_AVALON_PIO_DATA(LCD_CS_BASE,0);
    IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE,1);
    IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE,0);
//enable hw lcd control
    while(1);
    return 0;
```

After the control is moved, NIOSII CPU is idle.

4. FPGA download steps:

1. Connect the download cable to AS header of the FPGA board. Install camera module to OV760 header. Please be careful with pin 1 alignment. Apply 5V wall adapter to the FPGA board, and turn on the power switch.
2. Download pof file using AS mode header in Quartus II.
3. Turn off power, plug programmer cable to jtag and turn on power.
4. Open NIOSII IDE project.
5. Use flash program to download the program to external flash. Please don't download SOF to external flash. External flash is only for NIOSII firmware.
6. Power cycle the board. It will show the live image after the initialization message.

5. Code Description:



1. Program Structure:

In this program, the initialization of OV7670 and TFT is controlled by NIOSII. After the initialization is done, the control is handed over to hardware driver.

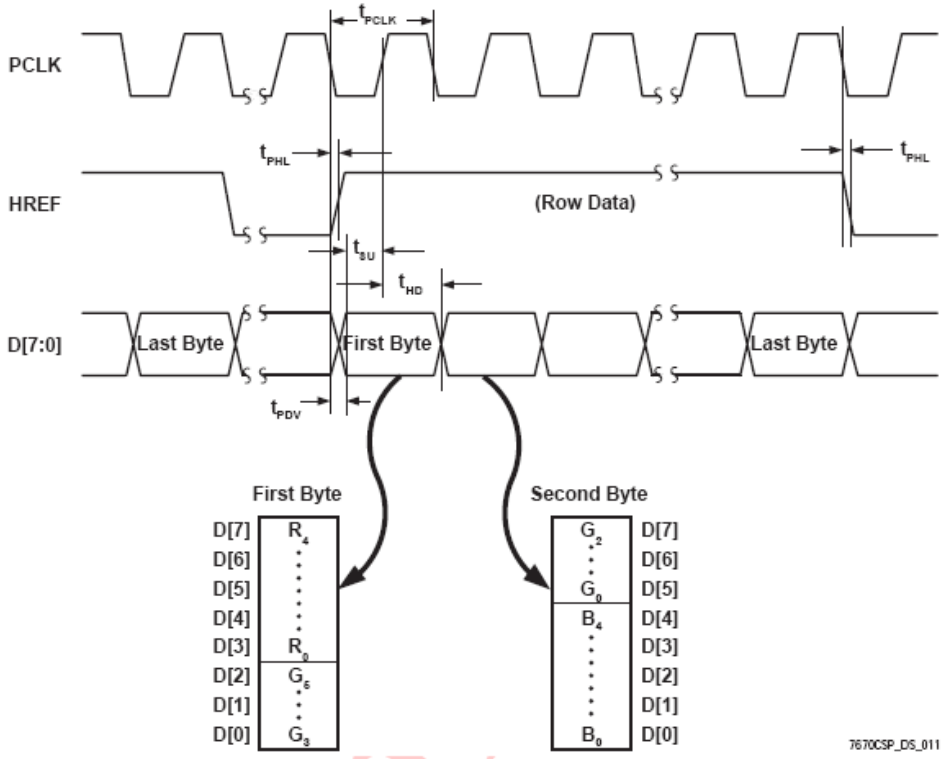
2. Cmos_top.v description:

1) Function: Implement the camera video capture, and transform the data to TFT format, and display on TFT.

2) Camera output format

In the initialization, the format is set to RGB565. The timing sequence is shown as follows:

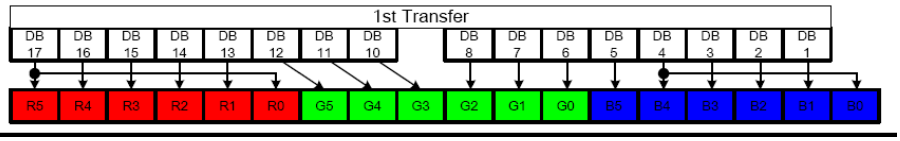
Figure 11 RGB 565 Output Timing Diagram



In the timing sequence, PCLK is provided externally. HREF is output by the camera. The 16-bit data is output twice and grouped by BYTE, R4,R3, R2, R1, R0, G5,G4,G3, G2, G1, G0, B4, B3, B2, B1, B0.

3) LCD input format

TRI	DFM	16-bit MPU System Interface Data Format
0	*	system 16-bit interface (1 transfers/pixel) 65,536 colors



The LCD input is 16-bit format. TRI is configured to 0.

The purpose of the program is output the camera data to TFT without the processing of NIOS II.

```
always@(posedge iCLK or negedge iRST)
	if(!iRST)
		begin
			state1 <= 1'b0;
		end
	else
	begin
	if(iLVAL & mCCD_FVAL)
		case(state1)
		1'b0 : begin
			pre_i0v7660_data_8bit[7:0] <= i0v7660_data_8bit[7:0];
			state1 <= 1'b1;
			end
		1'b1 : begin
			ov7660_data_16bit[15:0] <= {pre_i0v7660_data_8bit[7:0],i0v7660_data_8bit[7:0]};
			state1 <= 1'b0;
			end
		default : state1 <= 1'b0;
		endcase
	end
```

4)

```
186   always@(posedge iCLK or negedge iRST)
187   begin
188   	if(!iRST)
189   		Frame_Cont <= 0;
190   	else
191   	begin
192   		if({Pre_FVAL,iFVAL}==2'b10)
193   			if(Frame_Cont < 7) 为什么前7个像素周期不进行显示
194   			begin
195   				Frame_Cont <= Frame_Cont+1;
196   				tft_outenable <= 1'b0;
197   				end
198   			else
199   				tft_outenable <= 1'b1;
200   	end
201   end
202
203   endmodule
```

查阅了相关数据手册也没看到

The code here is to control when tft_outenable is valid.

The control signal is used in the following:

```
always@(negedge iCLK or negedge iRST)
        if(!iRST)
                begin
                        reg_lcd2_wr <= 1'b1;
                        write_state <= 1'b0;
                        lcd_data_valid <= 1'b0;
                end
        else
                case(write_state)
                        0 : if(temp2&(Y_Cont == 0)&tft_outenable)
                                begin
                                        reg_lcd2_wr <= ~reg_lcd2_wr;
                                        write_state <= 1'b1;
                                        lcd_data_valid <= 1'b1;
                                end
                                else
                                        begin
                                        reg_lcd2_wr <= 1'b1;
                                        lcd_data_valid <= 1'b0;
                                        end
                        1 : begin
```

如果 tft_outenable 无效，则不进行 TFT 的写操作。

Note: In the test, it is found that the camera cannot output complete frame in the beginning. If at the beginning the data is written to TFT, the screen will roll. So the first several frames need to be dropped.

5）

```
        end
else                    这个时序过程看的很郁闷，
        case(write_state)
                0 : if(temp2&(Y_Cont == 0)&tft_outenable)
                        begin                           temp2比行同步信号慢一个时钟。Y_cont==0就是X_cont计数未到640。tft_outenable=1,以上条件完全
                                reg_lcd2_wr <= ~reg_lcd2_wr;
                                write_state <= 1'b1;            满足着开启写信号，为什么不是，直接对写信号赋0，而是取反
                                lcd_data_valid <= 1'b1;
                        end
                        else
                                begin
                                reg_lcd2_wr <= 1'b1;
                                lcd_data_valid <= 1'b0;
                                end
```

```
169                 end
170             1 : begin
171                 if(temp2)
172                     reg_lcd2_wr <= ~reg_lcd2_wr;
173                 else
174                     reg_lcd2_wr <= 1'b1;
175                 if(Y_Cont < 240)
176                     write_state <= 1'b1;
177                 else
178                     begin
179                     write_state <= 1'b0;
180                     lcd_data_valid <= 1'b0;
181                     end
182                 end
```

到第二个状态后为什么对写信号取反？？？？？

为什么Y_cont计数240次后，停止写信号？？？