The Design of IP Core for LCD Controller Based on SOPC

Mingfeng Li[1,a], Xifeng Zhou[2] and Qiangang Guo[1]

[1]Automated institute, Nangjing University of Posts and Telecommunications, Nanjing, China
[2]Automated institute, Nangjing University of Posts and Telecommunications, Nanjing, China

Abstract. In this paper, it introduces the design and implementation of custom IP core for liquid crystal display (LCD) controller based on system on a programmable chip (SOPC) technology and Verilog hardware description language (HDL). The IP core can be put up as a general-purpose peripheral module to the Avalon Bus, communicating with the NIOS II through registers, and the underlying sequential logic for communicating with the external LCD liquid crystal module is automatically completed by the module. The driver does not need to care about the details of the underlying timing, it only needs to implement the read and write operations of the LCD data through register access, thus achieve the purpose of controlling the display content of the LCD. In addition to reducing the complexity of driver development, the design can also take full advantage of the parallel operation of hardware logic to improve the execution efficiency of the entire system and the throughput of processing tasks, especially in some occasions where high-speed image real-time processing is required. Simulations and experiments show that the IP core for LCD controller has good stability, versatility and compatibility.

1 Introduction

The LCD controller design scheme using a piece of complex programmable logic device (CPLD) and a piece of frame storage is introduced in detail [1]. The field programmable gate array (FPGA) is more flexible in programming than CPLD; the programming method of CPLD is to modify the logic function with fixed internal circuit, and the programming mode of FPGA is mainly to change the wiring of internal connections; CPLD has more advantages on completing various algorithms and combinational logic and FPGA is more suitable for tasks that require higher timing logic; therefore, FPGA is more suitable than CPLD for the design of LCD controller with relatively high timing requirements. In addition, due to the properties of open structure, parallel processing design with FPGA is possible and easily designed [2-4]. It describes the design and implementation of a multi-function LCD controller based on FPGA, introduces the overall structure of the controller, and then describes in detail the design and implementation methods of each module and the pivotal technologies involved [2]. The FPGA-based TFT-LCD controller is designed, the module design using Verilog HDL solves the problem of real-time image display of the embedded system, saves development cost and development time cost, and expands the application range [5]. It is the first complete hardware evolution (CHE) based on SOPC for EH. It is completely built on the configurable logic blocks (CLB) of a single commercial off the shelf (COTS) FPGA [6]. The system on a programmable chip (SOPC) is a special embedded system: first of all, it is the system-on-a-chip (SOC), which is the main logic function of the whole system by a single chip; secondly, it is a programmable system; The simple design of the embedded system using MCU is obviously unsatisfactory, and utilizing SOPC technology is an efficient and economical approach. Today, SOPC can be considered as an FPGA-based solution; it integrates the functional modules required for system design such as memory, processor, I/O port and CDR into a programmable module to form a programmable system-on-chip. The advantages and process of SOPC software and hardware collaborative design are expounded in detail [7]; in addition to reducing the complexity of driver development, hardware and software collaborative design can also make full use of the advantages of hardware logic to run in parallel; thereby improving the overall system execution efficiency and processing task throughput, especially in some situations where high-speed image real-time processing is required. In-depth study of the hardware system design and software development method of SOPC system, successfully transplanted uClinux embedded system on DE2 hardware development board, designed the LCD controller of TFT-LCD which is only suitable for DE2 development board matching LCD screen and display format [8], but it has limitations and does not apply to other devices.

The NIOS II-based SOPC embedded system can design IP cores such as digital frequency meter, seven-segment digital tube driver, PID, AD controller, etc [9-12]. The following describes several user-defined LCD controller IP cores. Adding the IP core for the LCD module developed by the SOPC design and integrating it into the system, realizes the interface design of the

a Corresponding author: 924514342@qq.com

embedded NIOS II soft core processor and the liquid crystal display module, and writes the driver, which can connect with the system; independent development of the LCD display can be realized without knowing the principle of the LCD [13]. Aiming at the shortcomings of poor compatibility among existing LCD controller models, a design method of parameterized TFT-LCD controller IP core based on SOPC Builder tool is proposed; this method has good versatility and compatibility [14]. The IP core of TFT-LCD controller based on Avalon Bus is achieved, and the display resolution and pixel depth can be configured [15]. The IP core of the LCD controller is based on MIPI interface, combined with MIPI technology and SOPC embedded system, it can realize image display with different resolutions [16], but the IP core can only be used for LCD with MIPI interfaces and it is not universal. In this paper, a design of IP core for industrial LCD controller based on SOPC is introduced.

2 LCD

Two operational timing circuits are the 80 series and 68 series, MPUs can be selected through hardware settings. It has 4-bit display data line, fast transfer data speed and powerful mapping function; it can support text display, graphic display and graphic and text mixed display; simple MPU interface and full-featured control instruction set. Made with COB technology, the structure is stable and the service life is long.

2.1 Schematic diagram of LCD

The LCD screen selected for this design is YB320240A, using the powerful S1D13305 (compatible with RaiO 8835) as the controller. The display mode is a 320*240 graphic matrix. The schematic diagram of LCD is shown in Figure 1.

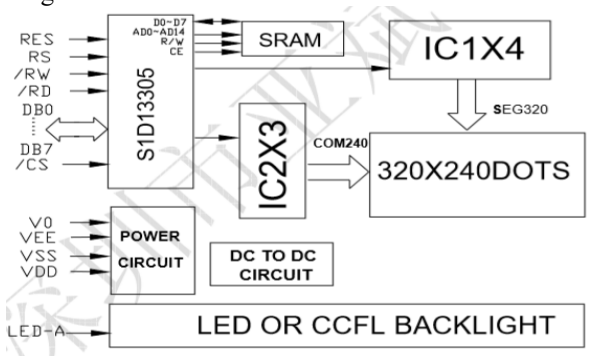

Figure 1. Schematic diagram of LCD

2.2 The interface signals of Intel8080

Pull sel1 low can select Intel8080 timing, when we set sel1 high, M6800 timing is selected. In this design, Intel8080 timing is selected.

2.2.1 Read and write timing

The Intel8080 read and write timing diagram is shown in Figure 2, Tcyc8 is the whole period of /RD and /WR. When A0 is pulled low, the data register is selected; it needs to wait Taw8 to pull /RD and /WR low; the low level hold time of /RD and /WR is Tcc; Tah8 must be kept after the rising edges of the two signals. For the write signal, Tds8 is the data setup time, Tdh8 is the data hold time, in order to write data on the rising edge of /WR; for the read signal, during the time period of Tacc8, we need to read the data from the video memory and put it into the register; Toh8 is the hold time of the data, the data should be kept during Toh8 period so that it can be read as soon as the rising edge of /RD occurs.

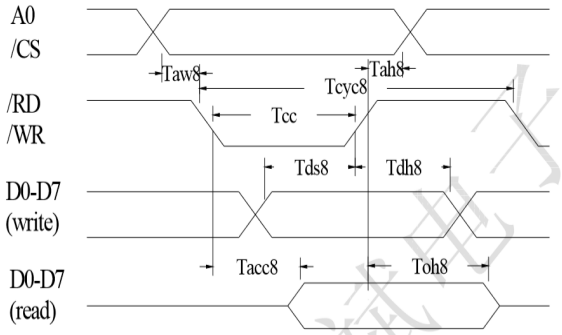

Figure 2. Read and write timing diagram of Intel8080

2.2.2 Combination functions of signals

As is shown in Table 1, there are four signals: /CS, A0, /RD and /WR, different combinations determine different functions. Once /CS is high, it executes prohibit operation; when /CS is low, it executes the following operations depending on the combination of the other three signals.

Table 1 Interface signal combination functions of Intel8080

/CS	A0	/RD	/WR	Fuction
1	X	X	X	Prohibit operation
0	0	0	1	Read status flag
0	0	1	0	Write command parameters and display data
0	1	0	1	Read display data and cursor pointer
0	1	1	0	Write instruction code

2.3 LCD controller design

2.3.1 Ports design

LCD controller interface diagram is shown in Figure 3, it shows the ports and a few internal registers which LCD controller is contained. Some are input ports, some are output ports, and there also exists an inout port.

Apart from the input and output ports, there is another port called inout ports in FPGA. If full-duplex communication required, then two channels will be required. In other words, you need to connect two FPGA pins to an external device, while sometimes half-duplex communication can ofen meet our requirements. There is

only one channel, the pin that implements this function on FPGA is the inout port. When the input port is in a high-impedance state, its state is determined by another circuit connected to it and it can be considered as an input

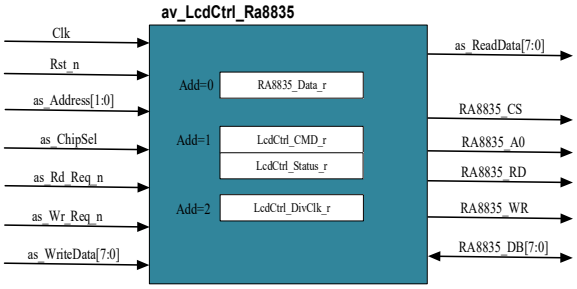

Figure 3. LCD controller interface diagram

port. The RA8835_DB signal is bidirectional, it is an inout port. The DB_OE signal is defined to choose the state of the inout port. When DB_OE = 1, RA8835_DB is as output signal. When DB_OE = 0, RA8835_DB is in a high-impedance state, which is as input signal, thus achieving half-duplex communication.

2.3.2 Internal registers

For the description of several internal registers in the figure:

(1) Data register: ADD = 0;

(a) Write data register will send data to RA8835 (LCD driver);

(b) If you want to read the data of LCD video memory, you must write the custom command USER_RD_REQ = 8'h81 to the command register firstly, and then query the busy flag. When the command is completed, the video memory will be stored in the data register; then read the data register can obtain the data of the LCD video memory;

(2) Command register: ADDR = 1;

When writing this register, it is to send a control command;

(3) Status register: ADDR = 1;

When reading this register, it is to obtain the state of the LCD controller.

As is shown in Table 2, the status register has 8 bits, of which Bit0 and Bit1 respectively serve as error flag and busy flag of the LCD controller.

(a) Busy_Flag: This flag is set when the LCD controller is communicating with the LCD module; the user program must ensure that the flag is in the idle state before the LCD controller can be operated;

Table 2. Status register format

Bit7~Bit2	Bit1	Bit0
	Busy_Flag	Err_Flag

(b) Err_Flag: When the LCD controller could not get the LCD module idle status flag (idle state D6 = 0), Err_flag will be set; the LCD controller will try 500 times to get the idle flag; when the LCD controller obtains the idle flag, it will automatically clear the flag;

Note: If the LCD module is not connected, D6 = 0 is not obtained (the weak pull-up should be set for the output signal of the LCD module). Err_Flag will be set at this time. You can use this flag to judge whether the LCD module is connected.

(4) Frequency division counter: ADDR = 2;

This register is used to divide the system clock. The clock that communicates with the LCD module is expected to be 50MHz. When the system clock is greater than 50M,

Table 3. Frequency division counter register format

Bit7~Bit3	Bit2	Bit1	Bit0
	Frequency division coefficient = system clock/50M		

the frequency division is required; the frequency division counter register format is shown in Table 3; it shows the calculation formula of the frequency division coefficient. When you want to read the data stored in the LCD module, the user's specific operations are as follows:

(a) Send the command: CMD_CSRW;

(b) Setting the memory address;

(c) Set the cursor movement direction: CMD_CSRDIR_RIGHT;

(d) Send read mode command: CMD_MREAD;

(e) Send read data command: USER_RD_REQ = 8'h81; This command is the command of the LCD controller, it will not be sent to the LCD module, just to start the operation reading the LCD video memory data;

(f) If you want to read the current memory data, you can send the command (USER_RD_REQ = 8'h81) directly.

3 Simulation results

In this experiment, in addition to using Verilog HDL to implement the module function, the corresponding testbench file is also compiled. Timing simulation waveform of LCD controller is shown in Figure 4, RA8835_CS, RA8835_A0, RA8835_RD and RA8835_WR are the signals that the LCD controller needs to be exported. They are connected accordingly to the /CS, A0, /RD and /WR interfaces of the LCD display. The functions implemented by the combinations of the signals which are shown in Table 2. When /CS = 1, the operation is prohibited. As is shown in Figure 4, when Rst_n = 1, the system works normally.

(a) When as_address = 1, as_WriteData = 40h, as_Chipsel = 1, as_Wr_Req_n = 0; then look at the combinations of RA8835_CS, RA8835_A0, RA8835_RD and RA8835_WR, it is 0001 firstly, and then it is 0110, the operation is to read status flag firstly to ensure the LCD controller is in idle state; then write the instruction code, 40h is the instruction for system set.

(b) When as_address = 0, as_WriteData = 10h, as_Chipsel = 1, as_Wr_Req_n = 0; then look at the combinations of the four signals: it is 0001 firstly, and then it is 0010, the operation is to read status flag firstly to ensure the LCD controller is idle; then write command parameters and display the data 10h.

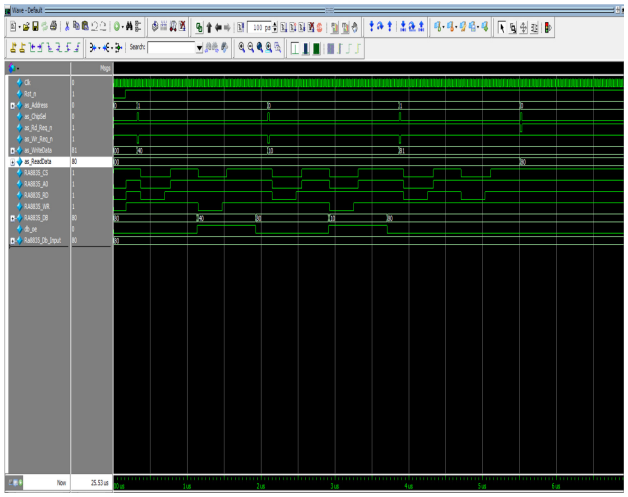

Figure 4. Timing simulation waveform of LCD controller

(c) When as_address = 1, as_WriteData = 81h, as_Chipsel = 1, as_Wr_Req_n = 0; then look at the combinations of the four signals, it is 0001 firstly, and then it is 0101, the operation is to read the status flag, and then display data and cursor pointers for reading. To Write 81h to the command register and then to query the busy flag. When the command is executed, the LCD video memory data will be stored in the data register.

(d) When as_address = 0, as_Chipsel = 1, as_Rd_Req_n = 0; the contents of the data register will be read, so the data of the LCD video memory can be obtained.

In summary, through the simulation waveform, it can be clearly seen that the module is well realized and connected, and then the IP core of the LCD controller is encapsulated in the Qsys software.

4 Software and hardware collaborative design

4.1 The construction of SOPC system

Structure diagram of SOPC system is shown in Figure 5, LCD connects with LCD controller, SDRAM connects with SDRAM controller and the two controllers communication with NIOS II processor through Avalon memory mapped interface (Avalon-MM).

Controlling multiple clocks generated by the same PLL can output clocks with different frequencies or phases. The Avalon ALTPLL component added in Qsys can output five different clocks at most, but only two outputs are needed in this design. According to the working principle of SDRAM, it is known that the clocks inputting to the SDRAM and the SDRAM controller need to be the same frequency and have a certain phase difference to ensure reading and writing data correctly. In this design, the input to the SDRAM and SDRAM controller is a two-way clock signal with a same frequency of 100MHz and a phase difference of 90°. And clk_100m2 exports for the clock of SDRAM, clk_50m is the system clock. We need to configure the NIOS II processor system, select and configure the peripherals and IPs, then connect the peripheral modules, generate the system, design the top-level module in the

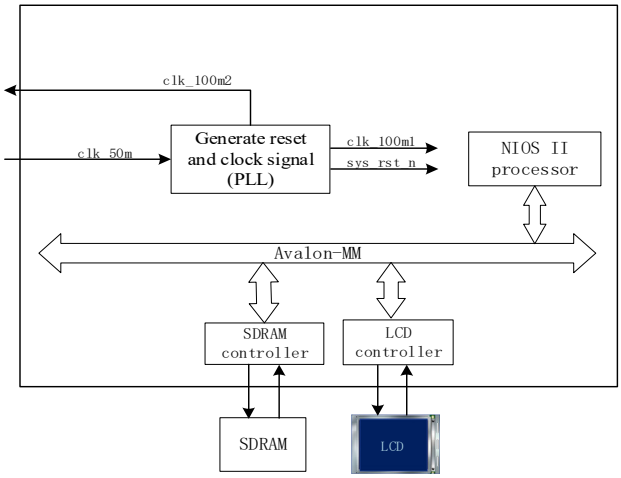

Figure 5. Structure diagram of SOPC system

Quartus II software, assign the pins and fully compile them, and download them into the development board if it is compiled successfully, the model of the FPGA chip in this development board is EP4CE10F17C8.

4.2 Software design

The operating environment of the soft code is Eclipse. Flow chart of software is shown in Figure 6, firstly, we parameterize the instruction code and initialize the LCD; wait until the Busy_Flag is low, and then we can write data into the LCD.

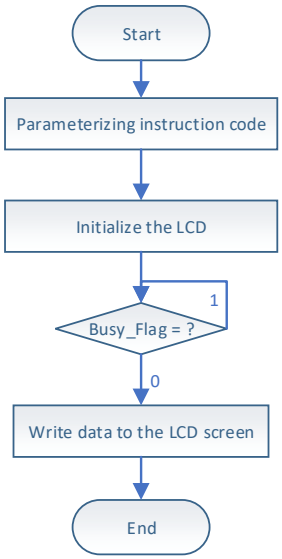

Figure 6. Flow chart of software

5 Experimental verification

We can download the soft code into the FPGA development board for testing if there is no problem after the software code is compiled and debugged. The LCD we use is monochrome screen, so we can't display any colourful image; and then we modify the image in Image2lcd tool, convert it to monochrome image with the format of 320*240, and output corresponding C code. The image is split into data, and the data is written to the

LCD video memory and displays. The experimental result is shown in Figure 7.

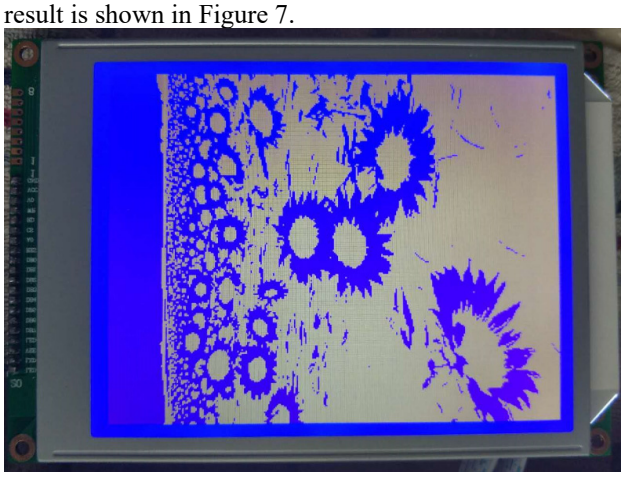

Figure 7. The experimental result

6 Conclusions

In this paper, we discussed the design of IP core for LCD controller. According to the top-down design idea, the function of the IP core is designed, the timing of the IP core is simulated, and the experimental verification of the SOPC system composed of the IP core was carried out. Since the IP core is configurable, it can be easily applied to embedded systems based on the NIOS II, enhancing the portability of the LCD controller. For the IP core is reusable, and the screen we designed is a kind of industrial screen, this design also has great application value. This design is also a good example of LCD controller design. Users can also design IP cores for controllers of other display screens according to the design ideas of this paper.

References

1. Junkai Huang, Cunbo Jiang, Hanmin Ye, Jian Xu, Yan Zhi, "Design and Realization of Embedded LCD Controller", Computer Engineering, pp. 218-220 (2005).
2. Wei Han, Jing Xie, Zhigang Mao, "Design and realization of a multifunctional LCD controller based on FPGA", Information Technology, pp. 58-61 (2008).
3. A. Ghasemazar, M. Goli, A. Afzali-Kusha, “Embedded Complex Floating Point Hardware Acceleration”, 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems, pp. 318-323 (2014).
4. C.K. Lai, K.L. Ho, “To Develop a Parallel Processing System with the FPGA for Multi-Axis Motion Control Platform”, Applied Mechanics and Materials, 284-287, pp. 2396-2401 (2012).
5. Weijia Su, Peng Zhang, "Design and Implementation of TFT-LCD Controller Based on FPGA", Chinese Journal of Liquid Crystals and Displays, **25**, pp.75-78 (2010).
6. A. Swarnalatha, A.P. Shanthi, "Complete hardware evolution based SOPC for evolvable hardware", APPL SOFT COMPUT, **18**, pp. 314-322 (2014).
7. Baojian Ge, "The Research and Implement of Hardware/Software Co-design Platform Based on SOPC", Wuhan University of Science and Technology (2008).
8. Zhiqiang Qi, "LCD Controller and Driver design Based on NIOS II", Harbin institute of Technology (2008).
9. Hu Bing, “Design and Research of SOPC Embedded Digital Frequency Meter Based on FPGA”, Proceedings of the 2011 International Conference on Computing, Information and Control(ICCIC 2011 Part5), pp. 489-495 (2011).
10. Xiangdong Li, Huaiwei Hu, Lujun Hao, Xinfen Yan, “Design of SOPC Custom IP Core Based on FPGA”, Computer and Modernization, pp.47-51 (2010).
11. Armando, Astarloa, Jesús, Lázaro,Unai, Bidarte, Jaime, Jiménez, Aitzol, Zuloaga, "FPGA technology for multi-axis control systems", MECHATRONICS, **19**, pp. 258-268 (2008).
12. Zhirong Fan, Linyan Shi, Xiangsheng Huang, "The design of IP core for AD controller based on FPGA", Electronic Products, pp. 55-57 (2014).
13. Changhong Hou, Huimei Yuan, "The Interface and program design of NIOS II and liquid crystal module based on SOPC", Chinese Journal of Liquid Crystals and Displays, pp. 307-311 (2008).
14. Hongfeng Ma, Jianwu Dang, Hongbin Wan, "Design of General TFT-LCD Controller IP Core Based on SOPC", Modern Electronics Technique, **33**, pp. 77-79 (2010).
15. Xiaohong Hao, "Design of IP core in configurable TFT-LCD controller", Information Technology, **33**, pp. 33-35+39 (2009).
16. Weidong Wang, "Design of IP Core for Configurable LCD Controller of MIPI Interface Based on NIOS II", Proceedings of 2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering (ICMMCCE 2015)", pp. 2658-2662 (2015).