Design and Implementation of Nios II-based LCD Touch Panel Application System

Tong Zhang[1], Wen-Ping Ren[2], Yi-Dian Yin, and Song-Hai Zhang

School of Information Science and Technology, Yunnan University

No.2, North CuiHu Road, Kunming City, Yunnan Province, China

[1]zhangt13@mails.tsinghua.edu.cn

[2]rwp3053@sina.com

Abstract— **This paper explains a system design for developing LCD touch panel applications by means of embedded system approach. Two Avalon-compatible IPs, acting as the LCD-display controller and the touch panel ADC controller are designed respectively for Nios II-centered SOPC. To verify the design, a game software named *Lianliankan* is developed on a SOPC containing the two custom IPs. The result indicates that the design has good usability, and satisfies the demand of application development better.**

***Keywords*— touch panel; Intellectual Property; Nios II; Avalon; system-on-a-programmable-chip (SOPC)**

I. INTRODUCTION

As a good kind of human-machine interaction interface，touch panel, especially those with high-resolution and true-color LCD display are increasingly applied to consumer electronics and industrial equipment, being embedded into various embedded systems.

Currently, developments of LCD touch panel based on FPGA are usually realized in two ways. One is building display buffer model and touching controller via software where embedded OS is involved in most cases, such as Windows CE, Android and Linux. Systems of this kind support various functions, but may be complicated to implement and have requirements for hardware. The other way is using loose HDL to describe each functional module and putting them together in the top module. This approach is easy to realize, but hard to apply to practical application developments.

This design realizes the functioning of LCD touch panel in the form of IP core as a component of SOPC and application functions are implemented via software design. This approach combined with hardware and software development provides higher resource efficiency, and is suitable for specified application or ASIC developments.

II. DESIGN FRAMEWORK

LCD touch panel is a conjunction of inner LCD screen and outer touching screen. The fundamental principle of LCD display part is horizontal/vertical scanning. The part of touch panel is an absolute coordinate sensor. The coordinate of touched point is outputted in the form of analog signal, which is converted to digital by ADC.

The circuit for driving LCD touch panel includes three parts, which are

- LCD display circuit, which deals with pixel signal, horizontal/vertical scanning signal, etc, in parallel form.
- LCD control circuit, which communicates with built-in LCD driver IC and configuring display function.
- ADC controller circuit, which communicates with built-in ADC and transmitting configuration words and digital output, in serial form.

The design and application of above circuit is implemented in the form of IP core built in SOPC. The overview of hardware structure is shown in Fig.1 (Only related parts are shown). The above three parts of circuit are implemented in two IP cores and they communicate with CPU and other components via Avalon bus.

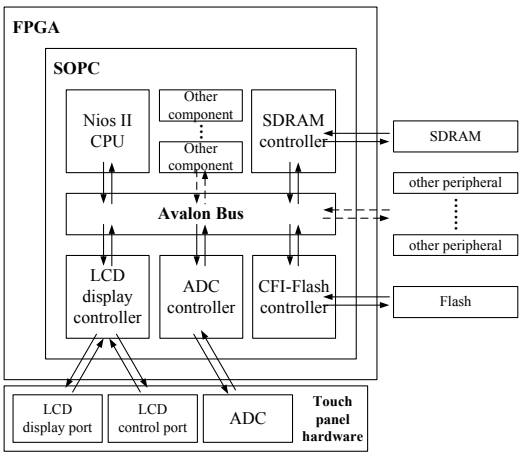

Fig. 1 Overview of hardware structure

The core work is the design of the two IP cores, i.e. the LCD display controller and the ADC controller. IP is short for Intellectual Property. The development of IP core includes two parts. The first is to develop hardware function logic, which is to design hardware behaviors using HDL. The second is the development of driver program, in the form of C language. The entire Nios II application development model is shown in Fig.2, which is an embedded C environment from the view of application program.

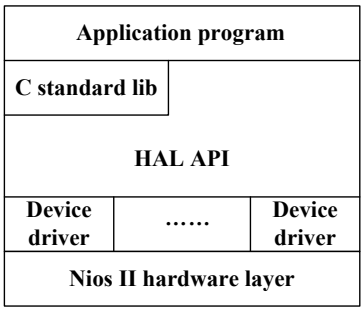

Fig. 2 Application development model

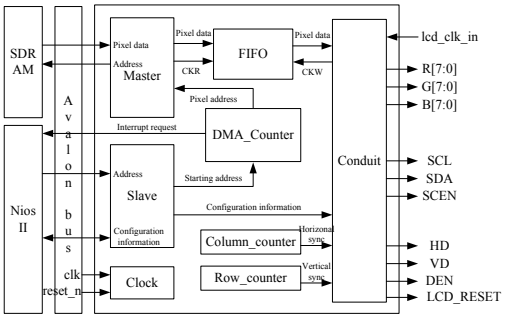

Fig. 3 Overview of LCD display controller IP

III. IP AND DRIVER DESIGN FOR LCD DISPLAY CONTROLLER

According to hardware requirements, this part involves two tasks. The first is to receive pixel information from Avalon bus, create display buffer and output pixel to the correct position on the screen. The second is communicating with LCD driver IC to configure LCD display. For the latter, we only need to leave ports in the IP core and use driver program to transmit configuration words. While the former needs coordination of hardware logic design and driver programming.

A. IP Design

1) Functional Structure

The pixel data for LCD display is stored in SDRAM in the form of pre-build buffer. In order to read and write to this buffer rapidly, the method of direct memory access, or DMA, is used via Avalon master port acquiring data from SDRAM.

As the rate of reading pixel data (Avalon bus frequency, 100MHz) is bigger than LCD pixel clock (33.2MHz), an FIFO is added between Avalon master port and pixel output. As long as FIFO is not full, the master port requests the bus for data and writes data into FIFO. To satisfy this requirement, the master port functions in pipelined mode, which means the port doesn't wait for the end of one transmission and requires next transmission by specifying a new address.

An interrupt signal *irq* is set to configure multi-page buffer. Every time the master address gets to the end of a frame, the *irq* signal goes high and CPU specifies the starting address of next farme in the ISR of display interrupt.

Avalon slave ports are used to transmit configuration information, including driver IC configuration words, starting buffer addresses and so on.

Pixel data from FIFO is outputted after spilt into R, G and B segments. Sync signals including HD, VD and DEN are generated by horizontal/vertical count logic and outputted.

The work clock *clk* and reset signal *reset_n* come from Avalon bus. LCD pixel clock *lcd_clk_in* is inputted form a PLL module in SOPC. The overview of the entire IP is shown in Fig.3.

2) Design of Pixel Output Buffer dcfifo

This part is implemented by instantiating DCFIFO (Dual-clock FIFO) in Megafunction, whose I/O structure is shown in Fig.4.

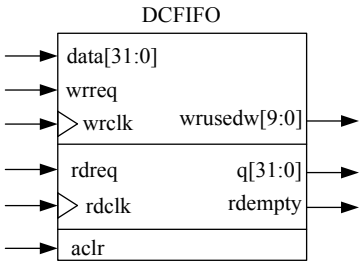

Fig. 4 I/O ports of *dcfifo*

Data source *data* of *dcfifo* is inputted from Avalon master port and the depth of FIFO is 1024 words, which can satisfy the need of more than one line's pixel data (800 words). Data input clock *wrclk* is the Avalon bus clock, while data output clock *rdclk* is the LCD pixel clock *lcd_clk_in*.

When horizontal/vertical sync process is in the period of pixel output, the read request signal *rdreq* of FIFO is set; as long as the master port received available data, the write request signal *wrreq* of FIFO is set. Such design can guarantee that this FIFO is in the "almost full" state all the time.

3) Design of DMA Address counter DMA_COUNTER

This unit acts as the up-counter of DMA address, which is implemented by instantiating LPM_COUNTER in Megafunction, whose I/O structure is shown in Fig.5.

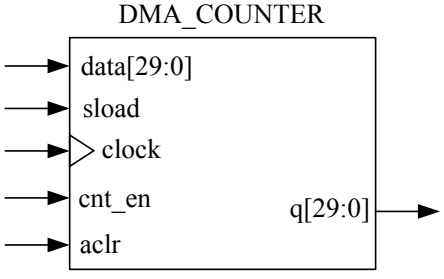

Fig. 5 I/O ports of *DMA_COUNTER*

As data of one pixel occupies two bits in SDRAM address, the output signal *q* of the counter is 2 bits shorter than system address (32), and *q* should be zero-filled in LSB

before delivered to the bus.

The default value *data* of the counter is the starting address of display buffer, and is written in via Avalon master port. Switches between different pages of buffer are realized by writing different values to *data*. *Sload* is the sync preset signal, which is set when DMA address comes to the ending address of a page of buffer.

The key to realizing pipelined transmission is to request the bus for data continuously. Therefore, the counter's enable signal *cnt_en* is set whenever FIFO is not full and data is available on the master port.

4) Logic of Horizontal/Vertical Sync and Pixel Output

Counter *column_counter* and *row_counter* control the transition of horizontal sync signal *HD*, vertical sync signal *VD* and display enable signal *DEN*, which are designed according to LCD's time sequencing.

One pixel on the screen occupies 32bits, whose structure is shown is Fig.6.

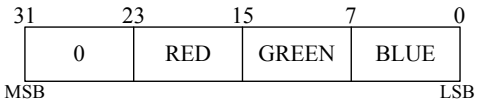

Fig. 6 Storage structure of pixel data

Being synchronized by pixel clock, when DEN is high, output signals R, G and B get the 23th to 15th, 15th to 7th and 7th to 0th bits in a pixel respectively as the color components. If DEN is low, the above signals all go to zero.

5) Operations of Slave Ports and Configuration of LCD Driver IC

Avalon slave ports deal with the access of configuration information stored in a group of registers representing resolution, color depth, DMA starting address, etc. The target register is specified by the slave address.

Particularly, a register *serial_control_reg* is defined to control the output of serial signals *SDA*, *SCL* and *SCEN*. Behaviors of these signals are defined in the driver program.

6) Interrupt Logic

When a sync preset of *DMA_COUNTER* happens, *irq* is set and interrupt request is sent to CPU. At this time, the stating address of next screen's buffer can be specified by program instructions, and the switch between different pages of buffer is realized.

Interrupt enable is controlled by a bit in a register written via a slave port, and interrupt flag can also be reset by clearing another bit in the same register.

B. Driver Program Design

1) Configuration of LCD

We want the driver program to be loaded by Nios II's initialization file *alt_sys_init.c,* so a _INSTANCE macro and a _INIT macro are respectively defined in the driver according to the specs of character-mode device.

In the header file, a structure type named *video_display* is defined as the programming type of LCD display controller, whose members includes the base address of controller, interrupt enable, display buffer pointer, resolution, color depth, etc.

The configuration function *video_display_init* is used to receive display parameters set in the program, initialize new *video_display* variables, register display ISR function, configure multi-page display buffer, export LCD driver IC parameters and return *video_display* pointer to be called by the program.

2) Multi-page Display Buffer and ISR

In some applications, such as video playing, the content on the screen need to be refreshed fully for the next frame. To guarantee the supplying of display data, we need to generate pixel data of several frames in advance so that pixel signals can be outputted continuously. In such applications, multi-page display buffer is used.

The approach in this design to implement multi-page display buffer is that several areas of buffer are set up in the memory and are labeled continuously. In the display process, the current page label is specified according to the reading and writing progress of display content.

When multi-page display buffer is used, the main program calls the function *video_display_buffer_forward* in the driver program to request to write to the next page of buffer in order to buffer the content of next frames. When display interrupt is triggered, the program confirms that next page of buffer is ready and put it out, otherwise the content on the screen doesn't change

3) Configuration of LCD Driver IC

The function *lcd_config* is used to write configuration words into the driver IC. It outputs serial data *SDA* and serial clock *SCL* by shifting out the configuration word bit by bit. The word contains information ranging from resolution, brightness, contrast, to GAMMA correction, etc.

IV. IP AND DRIVER DESIGN FOR ADC CONTROLLER

The process of acquiring coordinates is the process of analog to digital conversion via ADC. The task for this IP is to control the process of configuration, capturing, sampling, converting and outputting in the conversion according to ADC IC's time sequencing.

A. IP Design

According to ADC IC's time sequencing, the workflow of this IP is as shown in Fig.7.

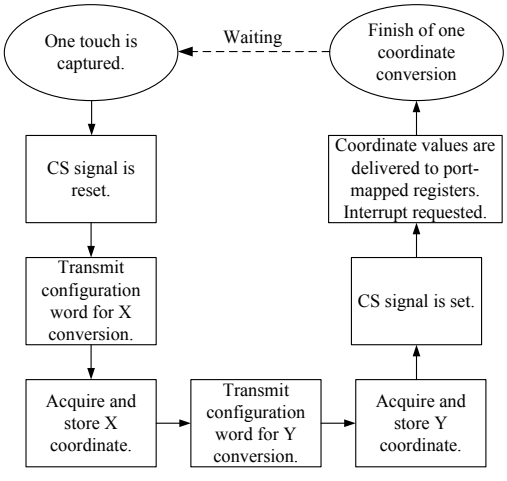

Fig. 7 Flow chart of ADC controller IP

From the view of the system, IP sends out coordinate values and interrupt signals, and receives interrupt reset and interrupt enable instructions. Actually, only one Avalon slave port is enough for this IP.

1) Capturing a Touch

To decide the touching point correctly, the falling edge of *PENIRQ_n* outputted by ADC need to be captured. Two registers are set to record the state of *PENIRQ_n* in the previous clk and the current clk respectively. The flag for a touch captured is that *PENIRQ_n* is high in the previous clk and low in the current clk.

2) Controlling Time Sequencing

Enough *DCLK* cycles are needed for the conversion to be finished. Register *conv_ctrl_cnt* is the counter for *DCLK* number and *spi_ctrl_cnt* is the counter for working progress. The operation of writing which bit in a configuration word or outputting which bit in a digital value is done according to *spi_ctrl_cnt* 's value.

3) Switching between X/Y Conversions

According to ADC hardware specs, different configuration words need to be written into IC before different (X or Y) coordinate conversions.

Switches between X/Y conversion are controlled by register *y_coordinate_config* whose initial state is 0. After X conversion is finished *y_coordinate_config* goes to 1, and returns to 0 after Y conversion finished.

Registers *x_config_reg*/*y_config_reg* record the configuration word needed by X/Y conversion respectively. The final word to write into driver IC is stored in register *ctrl_reg*, which switches between *x_config_reg* and *y_config_reg* according to the state of *y_coordinate_config*.

4) Start and Stop Control of Conversion

Internal signal *eof_trasmission* is the flag of the finish of one conversion, which is set when *y_coordinate_config* returns to 0, meaning that X and Y conversions are both finished.

Internal signal *transmit_en* acts as the conversion enable signal and controls the entry of conversion logic. *Transmit_en* is set when the falling edge of *PENIRQ_n* is captured, and reset when *PENIRQ_n* returns to high and *eof_transmission* is also high.

5) Reading and Writing Serial Data

Internal signal *wr_config_strob* acts as the strobe signal during writing configuration words. When *wr_config_strob* is high, bits in *ctrl_reg* is left shifted out to *DIN* synchronized with *DCLK* and configuration information is sent out.

Internal signal *rd_coord_strob* acts as the strobe signal during outputting digital values. Bits in *DOUT* are left shifted into *x_coordinate* or *y_coordinate* synchronized with *DCLK* according to the state of *y_coordinate_config* and afterwards the converted coordinates are acquired.

The setting and preseting of *wr_config_strob* and *rd_coord_strob* are controlled by *conv_ctrl_cnt* mentioned above and further determined by ADC time sequencing.

6) Interrupt Logic and Operations of Slave Ports

When *eof_trasmission* goes to high and the value of *y_coordinate* is nonzero, a valid touching point is acquired, and meanwhile the interrupt flag *touch_irq* is set.

Via specifying different slave addresses, related registers are read or written, and X/Y coordinates, interrupt flag and interrupt enable are accessed.

B. Driver Program Design

Similar to LCD display controller, _INSTANCE macro and _INIT macro are respectively defined in the driver program of ADC controller for auto-calling in initialization. Particularly, program statements for disabling touch interrupt are added in _INIT macro, avoiding the interrupt is triggered before touch function is started.

Similarly, structure type *touch* is defined to be used by touch function, whose members include the base address of ADC controller, current X/Y coordinates, interrupt flag, etc.

The configuration function *touch_control_init* is used to initialize new variables of *touch* type and determine whether to enable the interrupt or not. When the interrupt is disabled, touch function can work in query mode.

The ISR of touch interrupt is not defined in driver program, which should be defined in the main program according to different applications.

V. IMPLEMENTATION, TEST AND CONCLUSION

Build an SOPC containing the above custom IPs, whose component list is shown in Fig.8. The subsequent tests are based on this platform.

Module	Clock	Base	End	IRQ
cpu_0	pll_sys	0x01081000	0x010817ff	
onchip_memory2	pll_sys	0x01040000	0x0107ffff	
sdram	pll_sys	0x08000000	0x0fffffff	
jtag_uart	pll_sys	0x01082060	0x01082067	0
epcs_flash_controller	pll_sys	0x01081800	0x01081fff	1
timer	pll_sys	0x01082020	0x0108203f	2
cfi_flash	pll_sys	0x00000000	0x007fffff	
tri_state_bridge	pll_sys			
lcd_controller_0	pll_sys	0x01082040	0x0108205f	3
sysid	pll_sys	0x01082068	0x0108206f	
touch_control_0	pll_sys	0x00800020	0x0080003f	4
altpll_0	clk_0	0x00800010	0x0080001f	

Fig. 8 SOPC component list

To check the actual performance of the entire design in practical application, a game named *Lianliankan* is programmed and run on the SOPC above. The whole design of software and hardware is downloaded to DE2-115 board to be tested. A screenshot during test is shown in Fig.9.

Fig. 9 A screenshot of application implementation

The test confirms that, pixels on the LCD screen locate precisely, colors are accurate, and characters display correctly. The touch coordinates are well detected. Interrupts

can be triggered as expected. The touch panel responds well to touches and never a touch is left out or captured repeatedly. Driver programs function well. The logic of *Lianliankan* shows no error and games can be finished successfully.

The result shows that this design comes up to expectation and possesses availability. It can be used by various developments of different levels, containing LCD touch panels.

REFERENCES

[1] Altera Corporation, *AN527: LCD Controller*, www.altera.com/literature/an/an527.pdf, 2008.

[2] Toppoly Optoelectronics Corporation, *LTPS LCD Specification (TD043MTEA1)*, TRDB-LTM CDROM, 2006.

[3] Toppoly Optoelectronics Corporation, *LTPG110 Preliminary Datasheet*, TRDB-LTM CDROM, 2006.

[4] Analog Devices Corporation, *Touch Screen Digitizer AD7843*, TRDB-LTM CDROM, 2004.

[5] Altera Corporation, *Embedded Peripherals IP User Guide*, www.altera.com/literature/ug/ug_embedded_ip.pdf., 2011.

[6] Altera Corporation, *Nios II Software Developer's Handbook*, ver 11.0, www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf, 2011.

[7] Altera Corporation, *Quartus II Handbook*, Version 10.1, www.altera.com/literature/hb/qts/.../quartusii_handbook_10.1.0.pdf, 2010.