First Prize

Nios II Processor-Based Remote Portable Multi-Function Logic Analyzer

Institution: **Huazhong University of Science and Technology**

Participants: **Lian Zeng, Yong Li, and Hong-mei Zhu**

Instructor: **Jie Luo**

Design Introduction

With the rapid development of digital technology, people enjoy the benefits of updated digital products and systems. System design, debug, and test instruments have expanded from simulated digital oscillographs and large logic analyzers to portable multi-function instruments that enable network resource sharing.

Project Background

Two types of logic analyzers are available in the market today. Agilent and Tektronix provide high-end desktop instruments that use Intel Pentium processors, run the Windows XP Professional operating system, and feature powerful functions. For example, the TLA7000 series supports 136 channels, and has a sampling frequency up to 8 GHz, clock speeds up to 800 MHz, and a memory depth of 16 Kbits. Agilent's product series is similar. These desktop products are very expensive, for example, more than 100,000 Yuan. Most engineers will not choose such a high-performance, expensive logic analyzer.

The other available logic analyzer is a simple data collection card (using a USB or PCI interface) based on a PC. When the user runs data analysis software on the PC, the card analyzes logic. The Flyto Electronics Co., Ltd. Beijing L-800 product uses a USB interface, supports 24 channels to collect and input data, and is priced about 9,900 Yuan. Agilent's latest 16900 series product is also based on a PC, runs on the Windows XP Professional multi-threaded, multi-processor architecture, and has 68 input channels. It sells at about 20,000 Yuan. These products satisfy most engineering requirements at a comparatively low price. However, these analyzers are not easy to carry and must be used with a high-performance PC.

Logic analyzers are widely used in industry, business, education, and even military fields. With so many applications, engineers require a portable logic analyzer that supports cooperation and sharing to meet their needs.

The Nios® II-based multi-functional logic analyzer uses FPGA hardware to collect and store data. It uses a Nios II processor for interaction, control, and communication, and also supports a VGA monitor and remote network sharing. This analyzer meets most engineers' requirements, costs less than 2,000 Yuan, is portable, and supports remote sharing.

Design Objectives

This paper discusses our work creating the logic analyzer. With the Altera® Development and Education (DE2) board, Nios II soft-core processor, and system-on-a-programmable-chip (SOPC) design concepts, we used a single FPGA to implement the logic analyzer. With the Nios II processor, this analyzer can run a dual-CPU core, enabling an off-line VGA monitor and a PC as the network monitor. Additionally, the analyzer uses its powerful processing and storage function to analyze varied protocols. It uses a network interface to transmit data, and to implement remote assistant measurement and multiple user collaborative analysis.

The analyzer supports parallel collections, and provides 32-channel signal analysis and storage in the FPGA. The sampling velocity is 100 megasamples per second (MSPS), the storage depth is 1 Kbit, and the data bandwidth is 33 megasamples. The Nios II CPU delivers the processed data to the VGA interface for the LCD display and also delivers it to the PC via the Ethernet interface. The design can save the waveform as a document to ease analysis and data comparisons. The analyzer supports serial protocol analysis, including I^2C, serial protocol 2 (SP2), serial peripheral interface (SPI), and UART.

Functional Description

This section describes the project functionality.

- *Implementation of the signals' parallel collection, real-time analysis, and synchronous storage with the FPGA*—Most domestic logic analyzers use an embedded processor to collect and analyze data. Software serial collection is so slow that the data analysis severely lags behind the data storage. In contrast, the FPGA hardware processing signal enables the synchronization of data collection, analysis, and storage, achieving high-speed collection, real-time analysis, and synchronous storage.

  This design uses an Altera Cyclone® II EP2C35F672C6 device, with the overall clock frequency at about 150 MHz. To ensure that the FPGA runs smoothly, we used a 100-MHz clock to collect the data, analyze the collected data's real-time triggering condition, and save the data into a FIFO buffer with the specified storage depth. If conditions permit, the trigger core notifies the FIFO buffer to read data according to the trigger point. The FIFO buffer then locks the data for the Nios II processor to read. The FPGA's processing mechanism is far superior to the embedded processor's and has the same performance.

- *Blending measurements of varied levels*—Logic analyzers measure logic levels. Varied logic levels account for the differences when defining CMOS high and low levels. For example, the TTL level is usually supplied by 5-V power, with a logic high of 3.5 to 5.0 V and a logic low of less than 0.4 V. The CMOS logic high is about 80% of V_{DD}, logic low is about 20% of V_{DD}, and the midpoint is at 50% of V_{DD}. The CMOS power supply voltage can be less than 5 V, e.g., 3.3 or 1.8 V. We use a distributed amplifier (DA) and a comparator to convert the level, using the DA to alter the comparator's voltage reference and translating various levels into standard TTL.

- *Friendly user input*—Local users can use a touch screen to input triggering conditions instead of using a keyboard and mouse. Using the Nios II processor to control the touch screen ensures the screen's real-time response.

- *VGA color monitor support*—The DE2 development board provides VGA demo code that supports one-bit color (i.e., bottom color and top color). The board uses on-chip RAM as the graphic memory, covering 640 x 480 = 307,200 RAM bits, which is about 65% of the available RAM resources. The channel storage capacity is not sufficient, and one-bit color is too dull to be suitable for viewing.

  Instead, we use SRAM as the graphic memory, with 512 x 8 bits and 8-bit color. To display a frame requires reading SRAM data, the Avalon® bus streaming mode, and direct memory access (DMA) to update the VGA data and color image on the monitor in real time.

- *Ethernet support*—The local color VGA terminal supports an Ethernet interface. A remote PC can act as a terminal via the Ethernet connection. Therefore, the measurements and monitoring are not limited by space, which is very convenient for cooperative work.

- *Synchronous operation and updating multiple terminals*—When multiple terminals work simultaneously, if the logic analyzer operates on one terminal, the other terminals should be synchronously updated as well. If the logic analyzer stops, the user can view the operation settings.

- *Simple serial protocol analysis*—When using the serial protocol, a clock matches data; therefore, the logic analyzer can only show a waveform. Detailed data cannot be read. Our design analyzes four serial protocols—PS2, UART, I^2C, and SPI. Users can pick any of them as the test signal for which to select channels and simple settings for the corresponding signals. Aside from the tested signals, the logic analyzer also displays the data waveform in the protocol. This operation helps users analyze the serial protocol.

- *Real-time voice counting*—When analyzing the protocol, the system uses the DE2 board's audio interface to report the protocol data in real time with sound. Meanwhile the system activates the sound card on the remote terminal to report the protocol voice. The protocol voice counting helps users get the testing result, making the design more cooperative and more universal.

Technical Specifications

Our design has the following technical specifications:

Technical Specifications

Specification	Value
Sampling parameter	The highest frequency is 100 MHz (limited by the component's speed)
Threshold voltage	0 to 5 V
Input voltage	0 to 5 V
Triggering conditions	The four triggering conditions include debugging up edge, debugging down edge, high level, and low level
Input channel	32 routes
Triggering position	Front, middle, and backward
Triggering level	12
Storage depth	1 Kbyte at most (limited by the size of the storage)
Displaying mode	VGA and PC
Clock mode	Internal clock and external clock
External clock velocity	100 Hz to 100 MHz
Internal clock velocity	100 MHz
Operating environment	Windows 2000 and Windows XP
Connection mode	Internet

Design Architecture

This section describes the project architecture.

Implementation Theory

Because of the system's performance parameters and functional requirements, we decided to use the DE2 development board and relevant technology to develop the system. The Nios II processor helps solve complex problems such as limited hardware processor peripheral resources, complex interface configuration, hardware design, and software programs. Using dual CPUs effectively controls and reasonably allocates access to external devices, and helps us meet our functional requirements via the bus protocol communication between the CPUs. Therefore, the system makes full use of the chips, which greatly improves the operational efficiency. Because the analyzer frequently accesses the external storage, we define the external devices so that storage access and data transfer are simple. Using the FPGA's simultaneous operation, the logic analyzer's sampling speed is greatly improved. Because the VGA terminal requires real-time data, we use DMA data transfers. Altera's SOPC solution provides the advantage of multiple processors, FPGA logic control, and data processing.

Design Outline

We used the following general steps when creating our design.

1. Analyze system demand according to the analyzer's technical parameters and functional index.
2. Refer to the demand analysis to define the system modules and overall design structure and clarify key technology.
3. Consult the DE2 development board examples and documents to learn how to connect with external devices.

4. Perform the system hardware design. Draw diagrams of the external expansion circuit, PCB and FPGA hardware logic, and Nios II dual-CPU design.

5. Implement the system software, including the Nios II software and PC interface.

6. Perform system debugging and testing.

Project Plan

The following table shows the project schedule.

Project Schedule

Time	Phase	Specific Tasks
2006-4-14 to 2006-5-09	Demand Analysis	Clarify system performance parameters and functional requirements. Master FPGA/Nios II technology.
2006-5-10 to 2006-6-01	Overall Design	Create the system's overall design and module division.
2006-6-02 to 2006-7-30	Software and Hardware Design	Design the external expansion circuit. Draw the PCB, design FPGA logic section, and Nios II CPU and interface.
2006-8-01 to 2006-8-20	System Testing	Create the design testing scheme, set up a testing plan, and perform the test according to the function and performance index.
2006-8-01 to 2006-8-25	Write Design Report	Write the design report with reference to the design documents and system implementation.

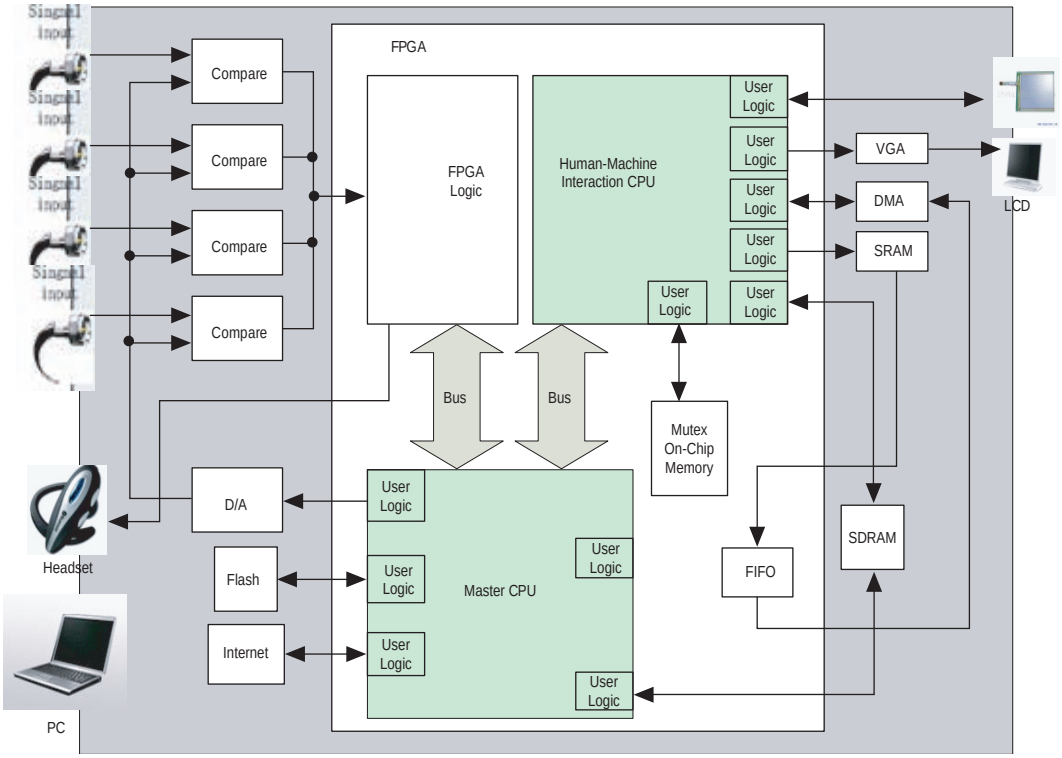
Hardware Design

This section describes the hardware design.

Hardware Architecture

As shown in Figure 1, the setting information is sent to the kernel CPU from the PC via the network, or is sent to the FPGA logic from the touch screen via a human-machine interaction CPU. The external signal is sent to the FPGA logic section through the comparer, whose threshold is set by the kernel CPU. Then, the FPGA logic sends the signal to the kernel CPU for processing. Next, the signal is sent to the PC via the network interface, while the data is written to the public memory (this action notifies the human-machine interaction CPU to read the data). The human-machine interaction CPU writes the data into SRAM, and the SRAM sends the data to the VGA terminal through a FIFO cache controlled via DMA. During protocol testing, the kernel CPU sends the processed signal and opens the FPGA-driven audio driver to report the analyzed data.

Figure 1. Hardware Architecture



Module Description

The hardware has two parts: the logic circuit (including data collection, trigger analysis, data storage, external clock frequency measurement, and voice processing) and the Nios II processor (including the kernel CPU and human-machine interaction CPU, which controls the external devices).

Logic Section

The logic circuit is described in the following sections.

Data Collection Module

There are various types of decision voltages for high- or low-level external logic signals. To make the system adaptive to various signals, the tested signals go through level conversion, which converts various levels into the TTL level to input to the FPGA. We use a DA+ comparer to implement level conversion. We used the MAX516 with the 8-bit DA+ comparer to make the conversion. The user sets up the decision voltage. After converting the data according to the following formula, the Nios II processor delivers the converted data to the DA.

$$N = V_{DAC}/V_{REF} \times 256 \qquad (1\text{-}1)$$

The voltage converted by the DA's output goes to the comparer's comparing side to become the comparer's voltage reference. If the external signal voltage is higher than the voltage reference, it is considered a logic high, which is output at a high level. If the voltage is lower than the voltage reference, it is considered a logic low, which is output at a low level. According to the sampling clock, the FPGA simultaneously collects the 32 route signals output by the comparer.

Trigger Analysis Module

The Nios II processor sends the triggering information to triggering core. Then the FPGA sends the collected signals into the triggering core to make a judgment and outputs a pulse to notify the storage device when it has satisfied all trigger conditions.

The triggering information includes a mask word and trigger conditions. The mask word is made up of 1s and 0s such that a triggering condition is 1 and the absence of a trigger condition is 0. The system exchanges the input signal and the mask word and compares the result. If they are the same, the trigger activates, otherwise the system waits.

There are four trigger conditions: rising edge triggering, falling edge triggering, high-level triggering, and low-level triggering. To determine which condition is occuring, the system needs at least two consecutive clock cycles of data. The trigger conditions on each level have two conditional distinguishing words, and the triggering core uses two comparisons to decide whether to trigger. The decision data of the rising edge triggering is 10, and that of the falling edge triggering is 01, the high level is 11, and low level is 00.

The triggering in this design is scaled into 12 levels. The system judges the trigger condition of the next level after the current trigger requirements are satisfied. For example, the triggering core sends out trigger signals when all conditions are satisfied. Due to uncertain intervals between the triggering on different levels, the trigger point for multi-level triggering is set backwards so that the system sees as many triggering processes as possible.

Data Storage Module

Collected data is stored into the device's FIFO buffer, which is a fixed size. If the amount of data in the FIFO buffer is more than the depth, the data entering the FIFO buffer first is read immediately so that the data stored in the FIFO buffer is the size of the storage depth. If the triggering core sends out a pulse satisfying the trigger condition, the FIFO buffer rewrites data as required by the position of the trigger point. Then, it stops storing data and notifies the kernel CPU that it is reading the data.

Measure External Clock Frequency

This design uses equal precision frequency measurement, which enables a broad clock frequency scope from 100 MHz (high frequency) to 1 KHz (low frequency). If the time on the actual gate is integer multiples of the tested signal frequency, this method eliminates the ±1 word error produced when the tested signal is counted, and ensures accurate frequency. Figure 2 shows how this operation works. Figure 3 shows the timing simulation.

Figure 2. Equal Precision Frequency Measurement Operation

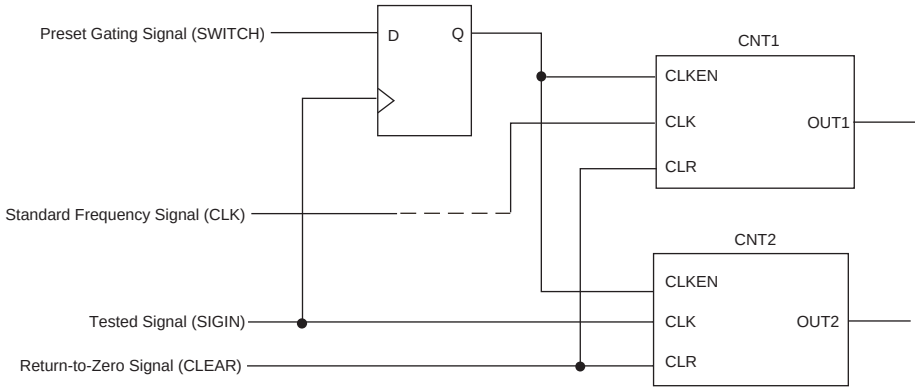
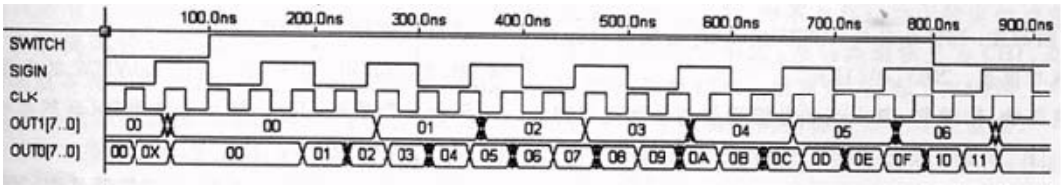


Figure 3. Equal Precision Frequency Measurement Timing Simulation



After the gate starting signal is given (the rising edge of the `SWITCH` preset gating signal), the `CNT1` and `CNT2` counters begin after the rising edge of the tested signal `SIGIN`. When the preset gate shutting signal (falling edge of the `SWITCH` gating signal) is given, the counter stops counting after the rising edge of the `SIGIN` tested signal, completing the measurement. At this time, the `CNT1` and `CNT2` values (`OUT1` and `OUT2`, respectively) can be read. We divide the `SIGIN` tested signal's value, `OUT2`, by the standard `CLK` signal's value, `OUT1`, and then multiply by the standard `CLK` signal's frequency to obtain the `SIGIN` tested signal's frequency.

Voice Module

The system delivers a pre-recorded voice file (**.wav**) to the audio digital-to-analog converter (DAC) for display, according to the protocol analysis data. The voice module implements the DAC configuration, audio data transfer, and the audio data cache. The audio DAC uses the WM8731 device, and is configured via the I^2C bus. Due to fixed configuration data, the DAC configures the WM8731 device with a hardware logic simulation of the I^2C timing.

The configured audio's sampling rate is 8 K, 16 bits. The Nios II processor can transfer the audio **.wav** file faster than the sampling rate. Therefore, we added a data cache FIFO buffer to match the speed of the DAC and the processor.

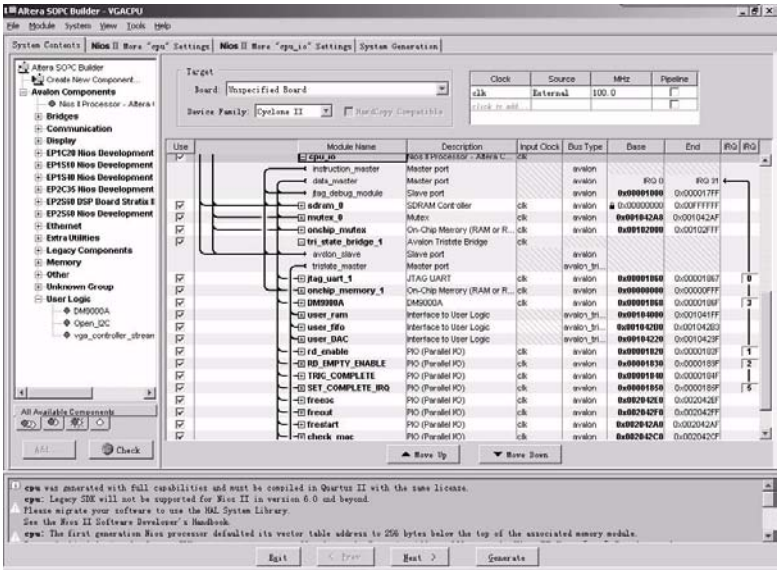
CPUs

Our system has a variety of complex processing. The VGA module displays a lot of data in real time, uses the Avalon bus's streaming mode, and uses frequent DMA operations. Additionally, the collected data is complex and the network interface interruption operation adds to the system complexity. Therefore, we decided to implement the system with two CPUs: the kernel CPU and human-machine interaction CPU. The kernel CPU processes the collected data and network data, and the human-machine interaction CPU displays the VGA data and controls the touch screen.

Kernel CPU

Figure 4 shows the kernel CPU SOPC Builder settings.

Figure 4. Kernel CPU SOPC Builder Settings



The kernel CPU is composed of the bottom data-exchanging interface, the communication interface between the network interface and PC, and other parts, as described in the following sections.

The bottom data exchanging interface is divided into the following sections:

- *Nios II interface for sending setting information*—This interface to the user logic defines **user_ram**, and delivers all setting information to the triggering core and FIFO buffer. This interface has 7 address signals, 32 data signals, and one write signal, which can deliver 64-bit data.

- *Read interface for storing FIFO data*—After saving the required data, the FIFO buffer produces the interrupt signal `rd_enable`, which notifies the kernel CPU to read the data. When the system finishes reading all data in the FIFO buffer, the FIFO buffer issues the interrupt signal `RD_EMPTY_ENABLE`, which notifies the kernel CPU to stop reading. The data read interface `user_fifo` connects the Nios II processor and the logic circuit. It has 32 data signals and one read signal, and delivers the data from the FIFO buffer to the Nios II processor.

- *Output interface of the sampling circuit decision voltage*—The sampling circuit decision voltage's output user interface DAC is defined by the user logic interface. The external circuit is a bus interface. Figure 5 shows the design.

  The five address signals are the chip select (`CS`) of the 32-route DAC, and the 8-bit DAC uses 8 data signals. The write signal is a valid low level. Figure 6 shows the MAX516 timing.

  If `CS` is low, the system locks and saves the data on the data signals when the write signal is on the falling edge. The scope of t_{AS} is 25 to 50 ns, the scope of t_{WR} is 50 to 120 ns, and the scope of t_{DS} is 20 to 50 ns. The time required to establish the signal is 30 ns, to wait is 80 ns, and to remain is 20 ns.

Figure 5. Voltage Module Interface and Timing

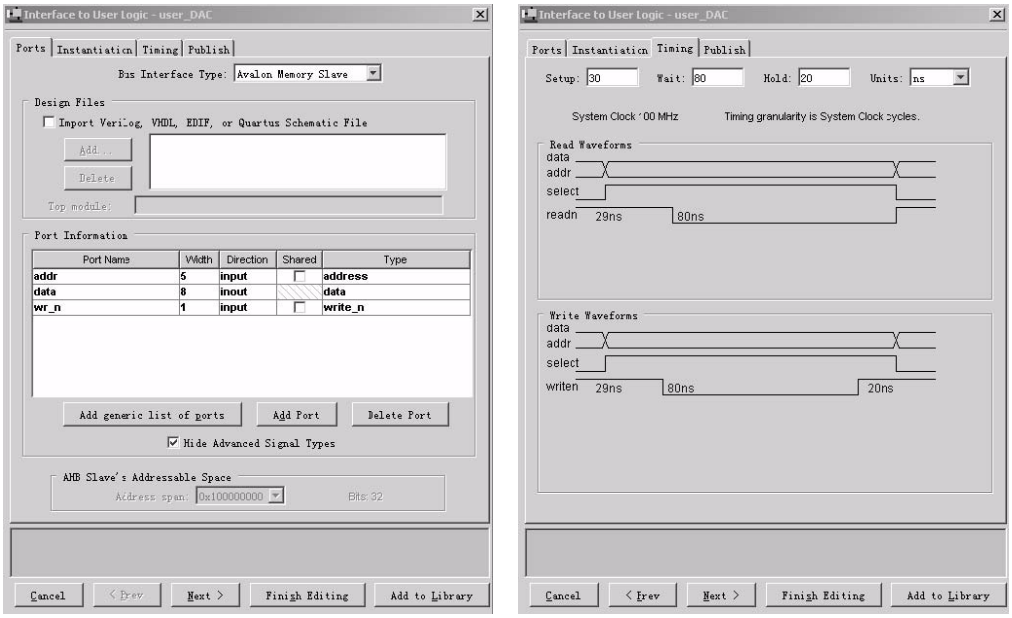
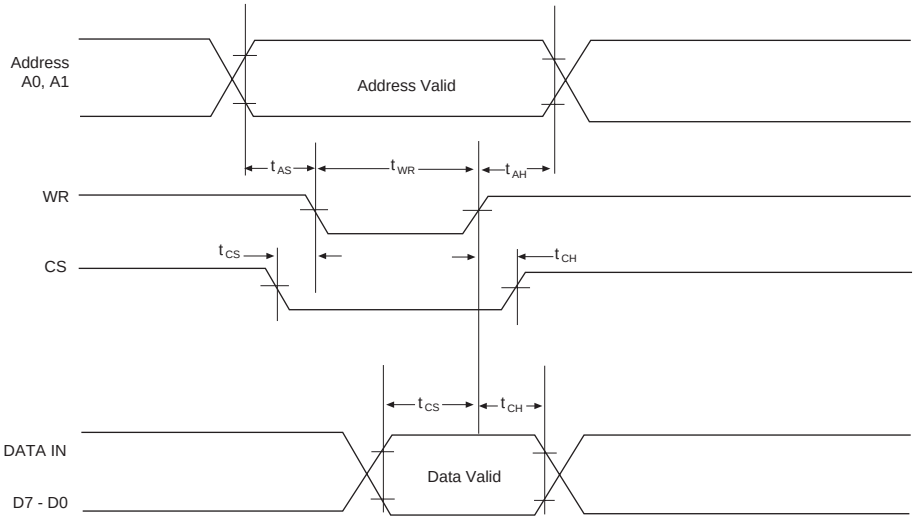


Figure 6. MAX516 Timing



- *External frequency read interface*—All external frequency read interfaces are composed of a programmable I/O (PIO), in which `frestart` is the equal precision frequency measurement module start signal that initiates the module. `freou` is frequency data, which is valid only when `freoec` is low. Therefore, it can be read only when `freoec` is low.

- *Audio data interface*—The audio data interface is a group of bus interfaces that defines `wr_audio` and delivers the required audio data to the voice module for playing. This interface consists of 16 data signals and one write signal.

The DE2 development board integrates the DM9000A Ethernet interface control chips, which connect to the RJ-45 external port and link to the LAN via a 100B-TX network cable. Integrating 16 Kbytes of

SRAM in the DM9000A chips ensures a data reception and delivery cache, and decreases the packet loss rate. Designers can choose from a wide range of chip performance parameters, for example, transmission velocity of 10 or 100 Mbytes, a power supply of 3.3 or 5 V, an interface of 10 M-BaseT in UDP3, 4, 5 or 100 M-BaseTX in UDP5. Considering the network data transmission velocity and size, and system compatibility, our design uses 100 megabits per second (Mbps) transmission velocity, 3.3 V, and 100 M-BaseTX in UDP5.

Figure 7 shows the read timing. The `CS#` and `CMD` signals are the chip select and control signals. `IOW` is a read signal, `SD` is a system data signal, and `IO16` is a 16-bit transmit control signal. The read signal is low after the chip select is effective. The data signal is effective after the T2 time period. At T3, `IO16` is a valid low level.

Figure 7. DM9000A Read Timing

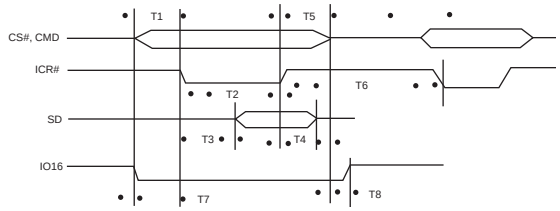
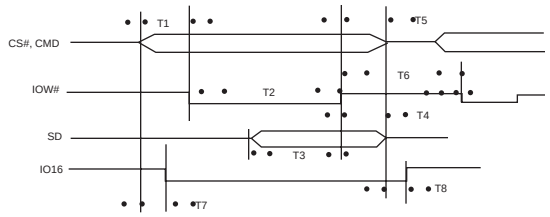


Figure 8 shows the write timing. The timing signal names are the same as in Figure 7, and the write timing is almost the same as the read timing.

Figure 8. DM9000A Write Timing

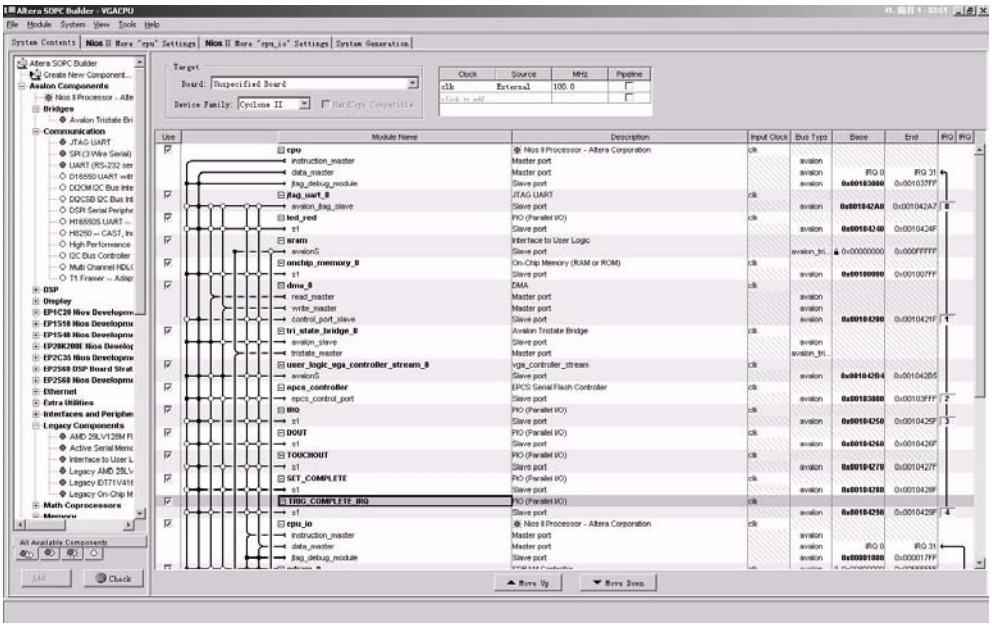


We use the **jtag_uart** debug interface to debug the program. The following data and configuration is shared by the CPUs: SDRAM, flash, EPCS controller, volatile and non-volatile storage data, and programs.

Human-Machine Interaction CPU

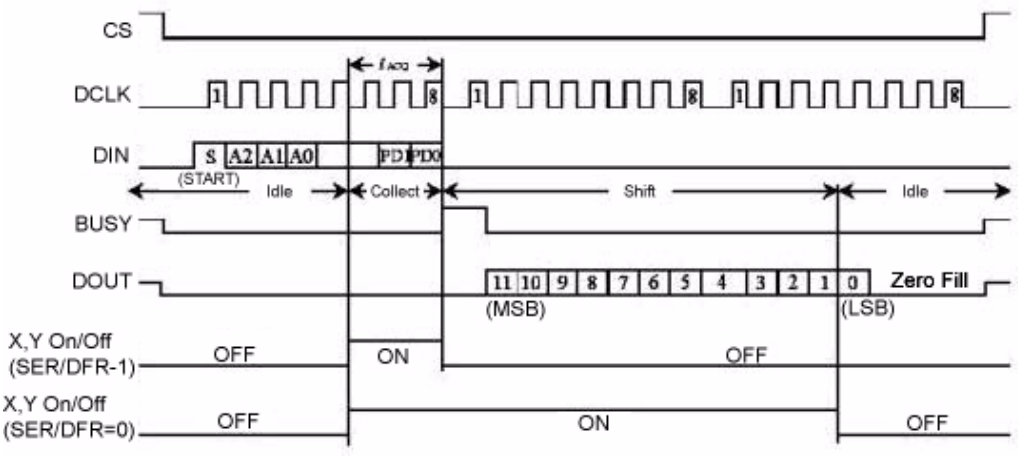
Figure 9 shows the human-machine interaction CPU.

Figure 9. Human-Machine Interaction CPU SOPC Builder Settings



The human-machine interaction CPU consists of the touch screen input interface, VGA monitor output interface, and other parts. The touch screen controller uses the TI ADS7846 device to implement the following microcontroller signals: IRQ, DOUT, DIN, DCLK, CS, and BUSY. In Figure 9, IRQ, DOUT, and TOUCHOUT (DINL, DCLK, and CS) use the PIO interface to simulate the ADS7846 signal. See Figure 10 for the simulation sequence.

Figure 10. Touch Screen Sequence



When the touch screen changes its shape (i.e., the user clicks the touch screen), the ADS7846 device generates an interrupt signal through IRQ. After receiving the interrupt signal, the Nios II processor generates CS, sends the control word to the ADS7846 device, waits for the busy signal, and receives the touch screen coordinate signal, which finishes one read sequence.

In practice, two coordinate signals are read to prevent false triggering. When the difference between the two signals is less than 20, the system assumes the trigger was correct. Otherwise, it considers it to be a false trigger.

The VGA monitor has 640 x 480 resolution and scans 60 frames per second. The pixel clock is 15.175 MHz, so the scan period of each point is about 40 ns. If every point is generated by the Nios II software scan, the 40-ns interval means that a Nios II CPU operating at less than 50 MHz can only execute two instructions.

According to VGA sequential scanning principles, the next scan point is predictable. So the system puts the pixel information to be sent in a line and sends it in an orderly fashion. Using the Avalon peripheral streaming mode, we implement the VGA controller with one Avalon stream. We used the DMA controller to build a DMA channel between the VGA controller with the streaming mode and SRAM. We use the hardware to read the pixel information automatically. Therefore, the Nios II CPU updates VGA images by operating the corresponding blocks in SRAM.

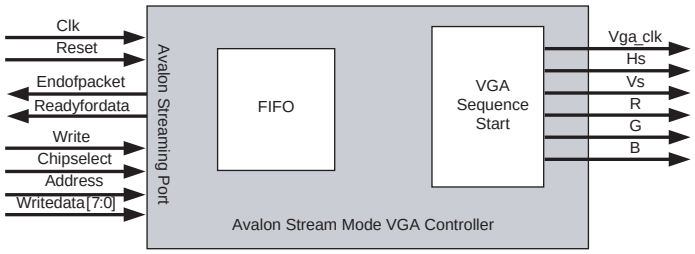
As shown previously in Figure 10, the `dma_0`, `tri_state_bridge_0`, `sram`, and `user_logic_vga_controller_stream_0` signals form the VGA monitor. The SRAM connects to the Avalon bus with custom logic. The DMA read master connects to the tri-state bus (read data in SRAM); the DMA write master connects to the VGA custom logic.

As shown in Figure 11, the VGA control core with the Avalon streaming mode consists of three parts: a VGA timing generator, the FIFO buffer, and the Avalon streaming port. The VGA sequence consists of clock, horizontal and vertical synchronization (hs, vs), and color (R, G, B) signals. It complies with the VGA standard, i.e., the 640 x 480 pixels at 60 Hz. Specifically:

- *Clock frequency*—25.175 Hz (pixel output frequency)
- *Line frequency*—31,469 Hz
- *Field frequency*—59.94 Hz (image refresh frequency per second)

The clock frequency is obtained by the 50-MHz phase-locked loops (PLLs). The line and field frequencies are obtained by frequency division of the VGA standard.

Figure 11. Avalon Streaming Mode



The FIFO buffer is mainly used as a buffer; read and write clocks are asynchronous. The Avalon streaming port generates bus interfaces and communicates with the FIFO buffer.

The **jtag_uart** debug interface is used to debug the program. Data and configuration shared by the dual CPUs are: SDRAM, flash, and the EPC controller, shared dual CPU cores, volatile and non-volatile memories, storage data, and programs.

Communication Between Dual CPUs

- *Communication data*—Users use the touch screen to enter trigger levels, trigger conditions, storage depth, names of channels, etc., that are sent to the kernel CPU. The kernel CPU judges

whether they meet trigger conditions and transmits the waveform data of the qualified channels to the human-machine interaction CPU. The conditions and names are also sent back to the human-machine interaction CPU (a remote PC and the local VGA display must be synchronized).

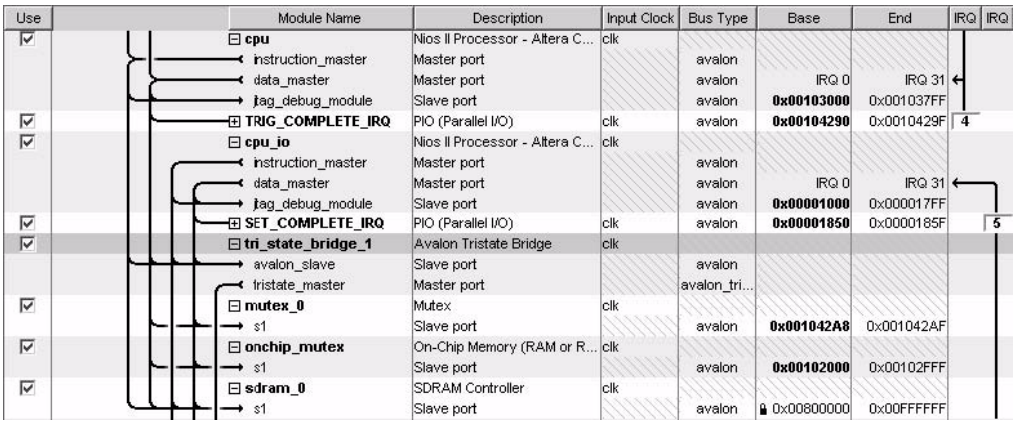
- *Communication interface*—When the user enter conditions via the human-machine interaction interface, the kernel CPU is notified. After setting conditions, the human-machine interaction CPU notifies the kernel CPU via a PIO interrupt. Likewise, when trigger conditions are met, the man-machine CPU needs a real-time response. The system displays the qualified waveform on the VGA terminal. The kernel CPU notifies the human-machine interaction CPU via a PIO interrupt. As shown in Figure 12, the interrupts are `SET_COMPLETE_IRQ` and `TRIG_COMPLETE_IRQ`.

- *Communication modes*—The trigger level, trigger conditions, storage depth, and channel names require two-way communication. The data volume is not very large, so the chip's `mutex_ram` can be used as shared memory to implement two-way communication. Both CPUs should read and write to the RAM, but not synchronously.

The waveform, bus, and protocol data only need to be transmitted from the kernel CPU to the human-machine interaction CPU. The data volume is large, so the system uses SDRAM as a shared memory. The kernel CPU writes data, and the man-machine CPU reads data, asynchronously.

Figure 12 shows hows the two CPUs communicate. `CPU` is the kernel CPU, `CPU_IO` is the man-machine CPU, `SET_COMPLETE_IRQ` and `TRIG_COMPLETE_IRQ` are the interrupt PIO, and `onchip_mutex` and `sdram_0` are shared communication memory.

Figure 12. SOPC Builder Setting for Dual-Core Communication

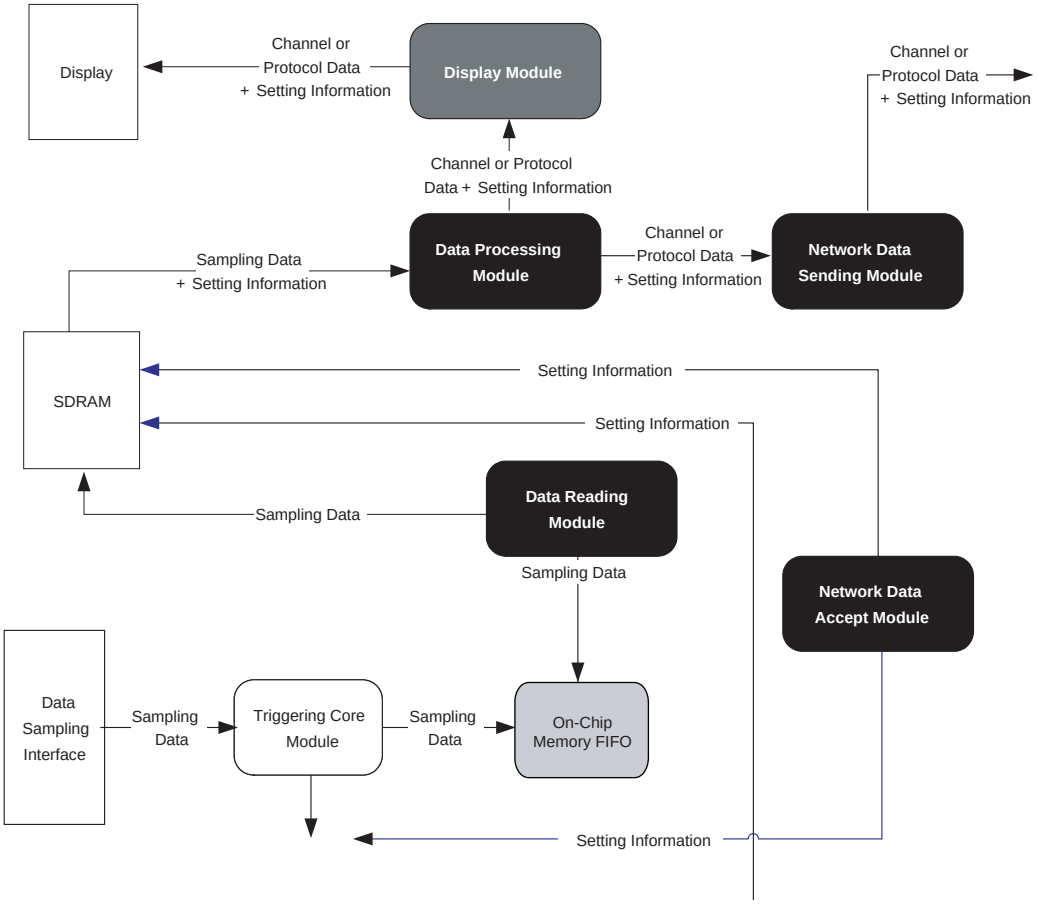


The human-machine interaction CPU is specially designed for the peripheral. The touch screen input implements I/O control, while the VGA monitor is implemented via the DMA streaming mode, which is independent of the Nios II I/O subsystem. We implement the monitor by sharing memories with a lot of data communication with the kernel CPU, according to Altera-recommended design principles.

Software Design

The software design has three parts: kernel CPU data processing, human-machine interaction CPU interface design, and the PC interface design. Figure 13 shows the software flow. The white boxes represent external interfaces, the gray boxes are the human-machine interaction CPU modules, the black boxes are the kernel CPU modules, and the light gray boxes are the FPGA logic.

Figure 13. System Software Flow

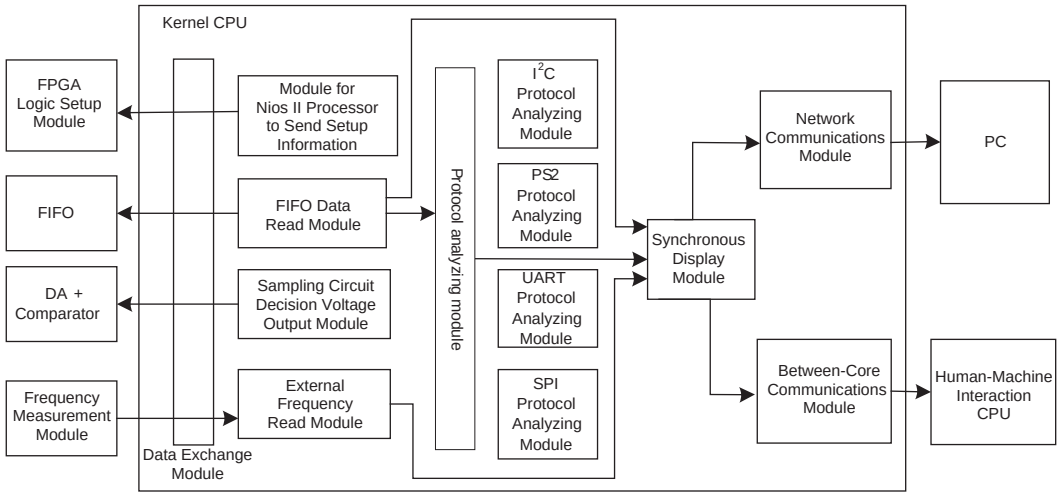


As shown in Figure 13, setting information is entered and stored in SRAM through the PC or touch screen. It enters the trigger core via the trigger core settings module. After entering the trigger core in real time, sampling data is sent to the FIFO buffer. If the sampling data meets the requirements, the data read module reads and stores the data in SDRAM. The setting information and sampling data in SDRAM sends the data to the PC via the network interface, after passing the data transformation module, and then sends it to the human-machine interaction CPU display module.

Kernel CPU

The kernel CPU software design consists of four modules: the data exchange module, protocol analyzing module, network communications module, display and synchronization refresh module, and inter-core communications module. See Figure 14.

Figure 14. Kernel CPU Modules



Data Exchange Module

The kernel CPU and data exchange modules are divided into a Nios II module that sends setting information, a FIFO data read module, an output module of the sampling circuit decision voltage, and an external frequency read module.

- *Nios II module that sends setting information*—Two display terminals can set the logic analyzer. If the PC sends out the start signal first, it sends, via the network interface, the settings package to the kernel CPU, which stores all conditions in the shared memory. If the human-machine interaction CPU sends out the start signal, it stores setting conditions in the shared memory and sends an interrupt signal to the kernel CPU. The kernel CPU reads the setting conditions from the shared memory and writes setting conditions to the interface address for sending settings information and sends them to the trigger core and the FIFO buffer.

- *FIFO data read module*—After receiving the data, the FIFO buffer tells the kernel CPU to read them using interrupts. By reading the FIFO read interface address, the kernel CPU receives the data in the 32-channel FIFO buffer and stores them. When all data is read, the FIFO buffer tells the kernel CPU to stop reading using inerrupts, and puts all data in channels.

- *Output module of the sampling circuit decision voltage*—When the two display terminals set the logic analyzer, the kernel CPU receives 32 bits of data from channel 1 through channel 32. The kernel CPU first judges whether the datum is 0; if it is 0, the channel is not selected and requires no settings. The CPU changes all other values in the corresponding DAC input and writes to the MAX516 device by writing the output address of the sampling circuit decision voltage.

- *External frequency read module*—The module is active only when the external clock is set. When data processing is finished, the system examines the PIO. The `freoec` signal is used to give the frequency measurement module enough time to finish the measurement. When `freoec` is low, read its value.

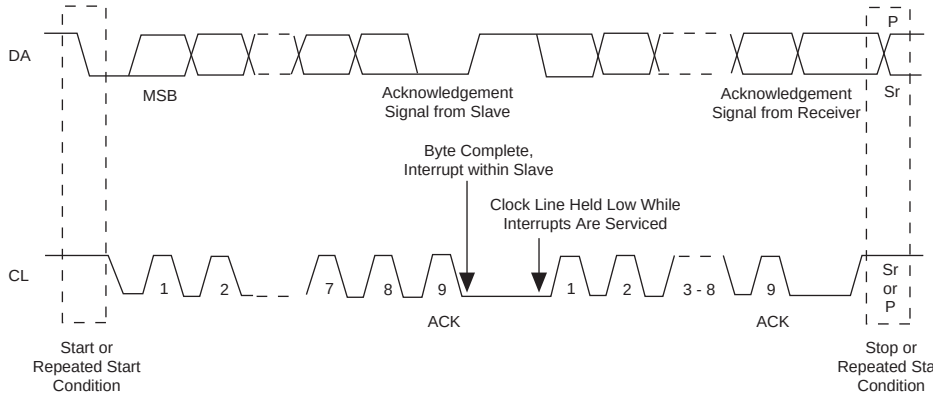
Protocol Analyzer Module

During protocol analysis, the logic analyzer automatically sets conditions that suit each protocol analysis. If qualified signals appear, the kernel CPU reads the data to analyze the software after the trigger core is triggered.

I²C Protocol

Figure 15 shows the I²C protocol.

Figure 15. I²C Protocol



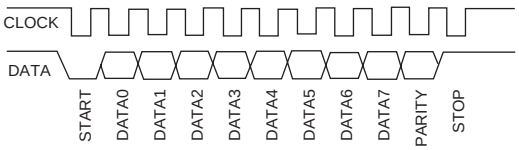
When SCL is high, the falling edge appears on SDA and data transmission begins. The 8-bit data sent to SDA is unchanged when SCL is high but changes at the low level. One response bit is followed after one byte is transmitted. Bytes sent in each transmission are not limited; several start signals are also allowed. The SCL signal generated after each transmission is high. A rising edge of the stop signal appears on SDA.

In the I²C protocol, the system exchanges the data and channel mask words. It leaves data on the SCL and SDA channels and sets all other bits to zero. Then, it checks the masked data start bits. When a start bit appears, it records the position of the datum and begins data detection. After detection, it records the detected data and its position. When the data is finished it completes detection. Finally, the system, examines end bits successively. If no end bit appears, it examines start bits, detects data, and records data and its position. The system repeats the process until all data is examined.

PS2 Protocol

Figure 16 shows the PS2 protocol.

Figure 16. PS2 Protocol



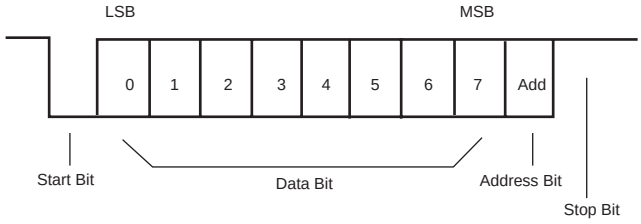
The maximum clock frequency is 33 kHz. All data are arranged in bytes. Each byte is one frame that contains 11 to 12 bits. The first bit is the start bit, which is always low. The second to ninth bits are 8 data bits; low levels are first and high levels follow. The tenth bit is the parity bit, which is odd. The eleventh bit is the stop bit, which is always high. The twelfth bit is the acknowledge bit, which only appears in the communication from the host to equipment. The data is locked on the falling edge of the clock signal.

In the PS2 protocol, the system exchanges the data and channel mask words. Data is left on the clock and data channels and all other bits are set to zero. The masked data examines the low level of the data channel in turn. When the data channel appears low, the system records the position and begins data detection. It records the detected data and checks whether the parity bit is correct. If the parity bit is correct and the end signal is detected, it records the detected datum and its position, and finishes detection. Finally, it detects the start bit and data, record data, and their positions. The system repeats the process until all data is examined.

UART Protocol

Figure 17 shows the UART protocol.

Figure 17. UART Protocol



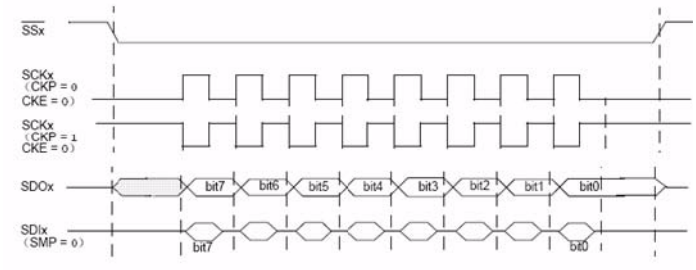
The baud rate is changeable and the number of data bits is not set. The low bit is first. The user can select the parity method and number of stop bits.

In the UART protocol, the system exchanges the data and channel mask words. Data is left on the transmit (TXd) or receive (RXd) channels and all other bits are set to zero. When the serial input signal has a level ranging from low to high, transmission begins. The system records the position of the datum. The baud rate can be set, so the data can be detected according to the relationship between the sampling frequency and set frequency and digits of data. The system verifies according to the data verification method. It confirms the position when the data ends according to the number of stop bits. It then records the detected data and positions when they end. The system repeats the process until all data is examined.

SPI Protocol

Figure 18 shows the SPI protocol.

Figure 18. SPI Protocol



Transmission begins when CS falls. During transmission, SCK generates 8 clock cycles as synchronous clocks. For the data transmission of 8 clocks between SDI and SDO, the high bit comes first and the low bit follows. Due to the difference of peripheral chips, some data is valid on the rising edge of SCK; others are valid on the falling edge. The maximum clock frequency can reach as high as 1.05 Mbps.

In the SPI protocol, the system exchanges the data and channel mask words. Data is left on the SCL, SDI, SDO, and CS channels and all other bits are set to zero. The masked data examines the low level of the CS channel. When a low appears on the CS channel, the system records the position of the datum and begins data detection. After 8 data are examined, it records them. When high appears on the CS channel, one byte transmission ends and the system records the position. The system repeats the process until all data is examined.

Network Communication Module

This section describes the network communication module.

Communication Theory Introduction

In the system, the interface transfers data between the DE2 development board and a PC. The DE2 development board provides an Ethernet RJ-45 interface, which connects the switch or hub within the LAN with twisted pair (TP) wire. In this case, all PCs (wired or wireless) connected on the LAN can achieve easy, fast, reliable communication with the DE2 development board.

Design Standard Used

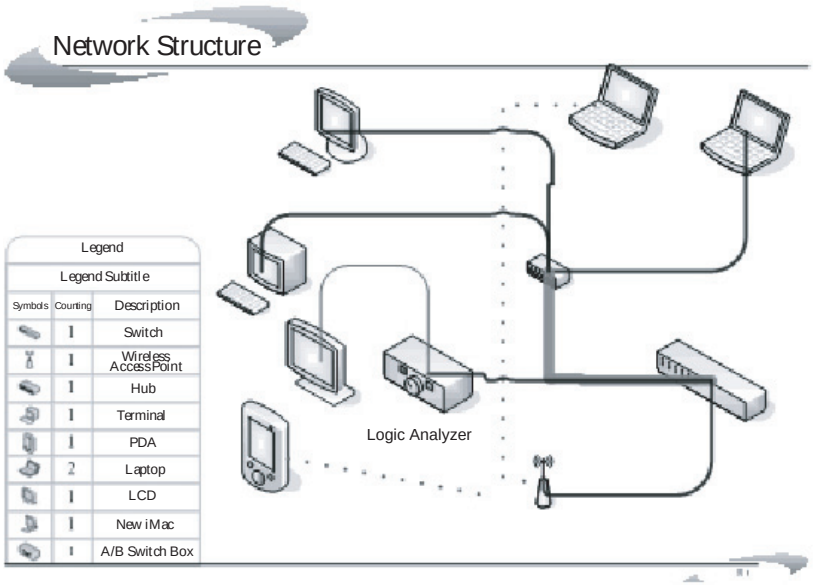
To be universal, our system uses the 10/100 Mbps Ethernet control transfer protocol. Fast Ethernet is the new standard set by the IEEE802.3 LAN committee. It is the extension of the 10 Mbps Ethernet standard, but it can transfer and receive data at 100 Mbps speed, with a transfer media of TP or fiber optic cable. Fast Ethernet is compatible with 10 Mbps Ethernet environments, and can be upgraded directly without wasting the existing hardware and software resources of a company,

Fast Ethernet supports the carrier sense multiple access/collision detection (CSMA/CD) Ethernet protocol. Its working principle is that it will detect whether the line is free when a site wants to transfer the data packet. If the line is free, it transfers immediately; if the line is busy, it detects whether the line is free again after a random time and sends the data packet until the line detected is free. If two or more sites transfer on the free fiber optic cable simultaneously, a conflict occurs. In this case, all conflicting sites stop transferring data and repeat the above procedure after a random time. This operation guarantees LAN communication reliability and stability.

Network Design

Based on the existing LAN in the office and lab, our system establishes a communication network with the DE2 development board as the central node and other display terminals as sub-nodes. All PC terminals connected on the LAN could implement the logic analyzing function after installing the logic analyzer PC interface software. Multiple labs or research teams within a LAN could share a logic analyzer, which saves research equipment costs and improves the logic analyzer efficiency. When a PC is analyzing logic, other terminals could also establish a communication connection with the DE2 board and observe the output result, i.e., the analyzed result can be displayed on different terminals. See Figure 19.

Figure 19. Network Structure

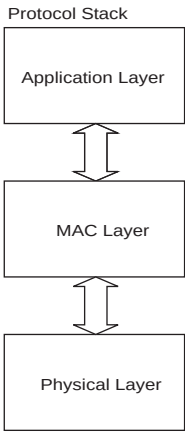


Protocol Design

The protocol stack design has the following layers (see Figure 20):

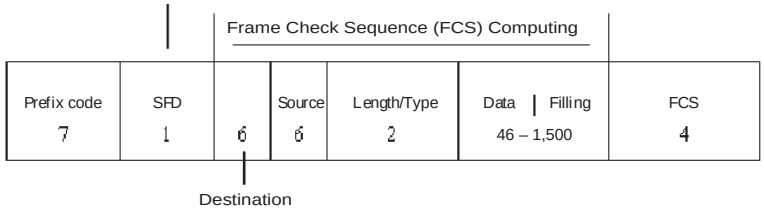
- Physical layer
- Medium access control (MAC) layer. It transfers the MAC frame, which adopts the 802.3 Ethernet standard to implement reliable end-to-end communication.
- Application layer. It encapsulates the application layer for data to be transferred and calls the service on the MAC layer to transfer the data.

Figure 20. Communication Protocol Stack



The 802.3 Ethernet frame structure begins with a 7-byte prefix code. The content of each byte is 10101010. The subsequent byte's content is 10101011, indicating the start of the frame. The destination address and the source address follow. Although the standard permits 2- and 6-byte addresses, the 10 Mbps baseband network standard only allows 6-byte addresses. (If the top digit of the destination address is 0, it indicates a general address; a 1 indicates multicast address. If all digits of the destination address are 1, it indicates the broadcast address). Following is the 2-byte length field (whose value is 0 to 1,500) and the data section. If the frame's data section is shorter than 46 bytes, a filling field is used to meet the shortest length required. The last field is the checksum, whose checking algorithm is a cyclic redundancy code (CRC). See Figure 21.

Figure 21. Frame Structure

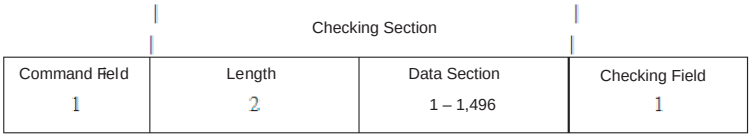


The frame format design on the application layer is (see Figure 22):

- *Command field*—This field is 1 byte long and counted from 0x00 to distinguish different data types.
- *Length field*—The length field size is the length of the data section (in bytes), which makes it easy for the receiver to analyze the data section.

- *Data section*—The maximum data section length is 1,496 bytes, according to the frame size of the Ethernet MAC frame.
- *Check field*—The check algorithm uses the simple sum check method to guarantee the data transfer accuracy.

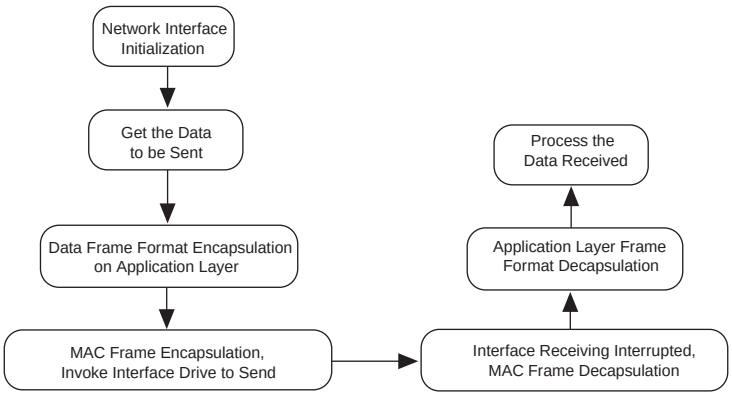
Figure 22. Application Layer Frame Format



Process Description

Figure 23 shows a single communication process.

Figure 23. Single Communication Process



Display and Synchronization Refresh Module

The system supports multiple display terminals, therefore, all display terminals should be synchronized when the logic analyzer operates. Users can run the system by themselves, so we added an arbitrator to the dominant CPU. When the PC section sends the setting information, the arbitrator closes the set completed interrupt in the human-machine interaction CPU, which ensures that all settings arrive at the FPGA logic correctly. When the human-machine interaction CPU issues the set completed interrupt signal, the dominant CPU reads the setting information in the public memory after closing the interface interrupt. When data processing finishes, the dominant CPU sends the setting information and data to PCs via the network. It then tells the human-machine interaction CPU to read the setting information and data with the process completed interrupt, which guarantees synchronization when the logic analyzer operates.

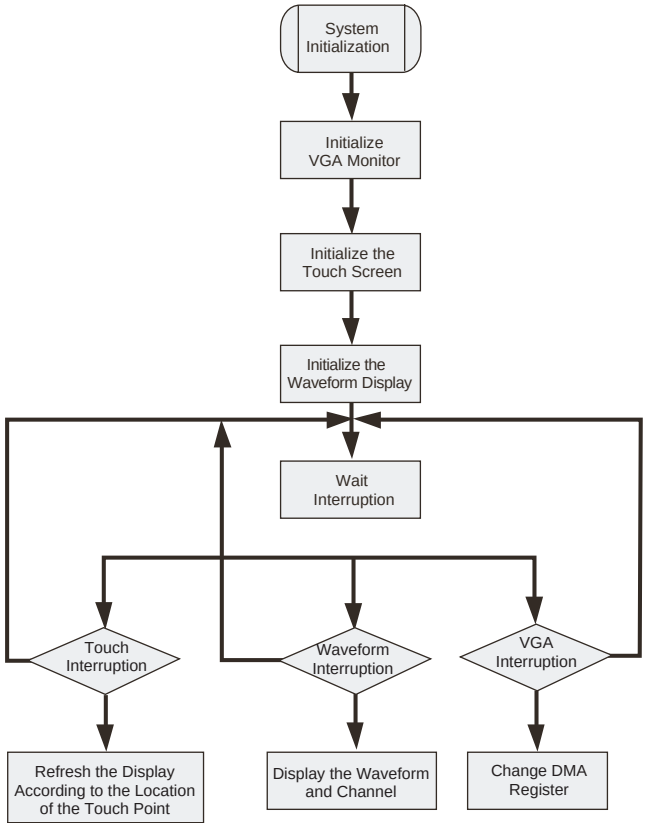
Inter-Kernel Communication Module

In the end, the dominant CPU returns the data to the two display terminals according to the origin of the setting packet. If the setting packet is sent by the PC, the dominant CPU will only send the data packet of the selected channel to the PC, and inform the human-machine interaction CPU to read back all setting information from the public memory and read data from SDRAM. If the setting information is sent by the human-machine interaction CPU, the dominant CPU sends all setting information to the PC as well as the selected data packet. Then, it informs the human-machine interaction CPU to read back all setting information from the public memory and read data from SDRAM.

Human-Machine Interaction CPU

The key function of the human-machine interaction CPU is real-time detection of the user touch screen input, accurate and in-time display on the monitor, and dominant CPU communication. Figure 24 shows the software design flow of the Nios II human-machine interaction CPU.

Figure 24. Software Flow of Human-Machine Interaction CPU



First, the CPU initializes the system and the interrupts, which includes touch screen, DMA, and trigger requirement interrupts, etc. Next, it waits for the user to input information using the touch screen. If there is touch screen input, the CPU transfers the corresponding display concurrent control word to the dominant CPU and waits for the trigger requirements to be met. When the trigger requirements are met, the trigger waveform is sent to the VGA monitor. Additional details are described below.

- *Real-time touch screen input*—The touch screen input uses an interrupt trigger. The control word is sent to the touch screen controller after it receives the touch screen interrupt. Then, the user touch coordinate is read until the busy status ends. In this case, the CPU receives the user input in time to make a real-time response.

- *VGA display module*—The low-level VGA driver is established with the Avalon bus streaming mode hardware. The frame refresh is implemented via a DMA interrupt, and the data refresh is implemented by writing to the corresponding area of the SRAM memory. The upper-level graphics display calls the graphic function directly to display all types of graphics and waveforms.

- *Multiple-core communication*—After the user has set the trigger channel, trigger requirements, and depth via the touch screen, the human-machine interaction CPU tells the dominant CPU that the setting is complete. The information is transferred to the dominant CPU via the PIO trigger

interrupt. The human-machine interaction CPU waits until the trigger requirements are met, at which point, it triggers a human-machine interaction CPU PIO interrupt. The human-machine interaction CPU reads the trigger data and sends the waveform to the VGA monitor.

PC Interface

This section describes the PC interface.

Platform Introduction and Design Idea

We developed the PC interface with the Microsoft Visual C++ software version 6.0. This software is commonly used for developing desktop application software, and reduced our development time and improved efficiency. The Microsoft Foundation Class (MFC) framework supports fast GUI development. We could optimize the program interface and enhance the user experience with its strong operability and interaction performance.

The interface is analyzed, designed, and developed using object-oriented concepts. Based on the model control view (MCV) design mode, the whole system uses the Windows message mechanism, which separates data from the interface, ensuring real-time user operation and data reliability.

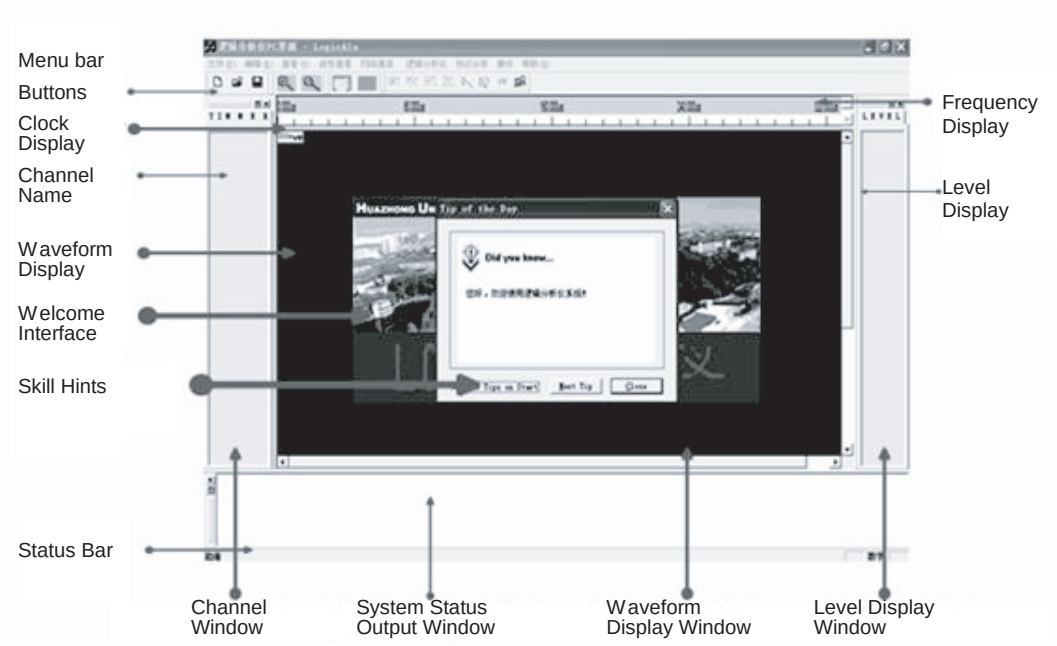
The PC interface communication drive uses the mature Winpcap software system to send and receive data. The free, public Winpcap software system is used for the direct network programming in Windows systems. It provides the following capabilities:

- Captures the original data packet, whether the data packet is sent to the local PC or is a switching packet between other devices.
- Filters the user-defined rules before the data packet is sent to the application.
- Sends the original data packet to the network.
- Generates network traffic statistics.
- Combines with the design of the whole communication section to implement communication between the DE2 development board and PCs in the LAN.

Interface Introduction

Figure 25 shows the main PC interface.

Figure 25. Main PC Interface



System Function Test

This section describes the functional tests we performed on our system.

Testing Scheme

We used a signal sent from a 51 single-chip microcomputer, Lingyang's single-chip microcomputer, and a Philip's ARM as the test object.

Logic Signal Test

We performed the following logic signal tests on our design.

- *Bus test*—We used the 51 single-chip microcomputer's address and data bus to measure address reads and writes. For the test, we drew one 51 testing board, enable 16 address signals, 8 data signals, 1 read signal, and 1 write signal, and connected some of them to the logic analyzer test pins. Then, we set the 16 address signals as 1 bus and the 8 data signals as 1 bus with separate read/write signals at the PC end. The storage depth is set to 512, and we use a single-level trigger, which requires a falling edge to write the signal. The trigger location is at the front. We chose a single trigger and, after reading the data, displayed the tested signal in bus mode on the PC.

- *Depth of location test*—We used the same location as in the bus test and varied the storage depth (we used 32, 64, 128, 256, and 1,024). We repeated the test procedure and displayed the resulting waveform on the PC.

- *Location of trigger test*—We set the storage depth to 512 and changed the trigger location from the middle to the back. We repeated the test procedure and displayed the resulting waveform on the PC.

- *Multi-level trigger test*—We generated several phase shift signals using the 51 single-chip microcomputer's I/O and P1 interfaces. We connected these signals to the logic analyzer test pin.

Then, we set 3 trigger levels on the LCD using the touch screen. The corresponding trigger requirements were:

- *First level*—`signal1` rising edge
- *Second level*—`signal2` rising edge
- *Third level*—`signal3` rising edge

We used a 1-Kbyte storage depth and the external clock as the trigger clock. We selected a single trigger, and displayed the phase shift signal that meets the requirements set on the LCD after data is read.

- *External clock test*—In this test, the 51 single-chip microcomputer, which is connected to the external clock pin of the logic analyzer, generates a clock signal. We chose the external clock as the trigger clock and displayed the waveform on the LCD.

Protocol Test

We used the following protocol tests:

- *I²C protocol analysis test*—We used Lingyang's single-chip microcomputer to generate a set of I²C signals, which we connected to the logic analyzer test pins, with the keyboard of an I²C interface. Next, we set the channel number for the `SCK` and `SDA` signals of the I²C connector on the PC. After running the test, we displayed the waveform and analyzed the protocol on the PC. To determine whether the test result was correct, we verified the results with the I²C interface chip ZLG7290.

- *PS2 protocol analysis test*—We used a PS2 keyboard as the test object. We fetched the PS2 interface clock and data signals, and connected them to the logic analyzer test pin. We set the channel number for the PS2 clock and data signals using the LCD touch screen. After running the test, we displayed the waveform and analyzed the protocol on the LCD. We determined whether the test result was correct by checking the PS2 key code table.

- *UART analysis test*—We used the serial port on the PC as the test object and sent a signal with the serial port tuning assistant. We fetched the serial port RXd signal, and connected it to the logic analyzer test pin. Next, we set the channel number for the RXd signal and the serial port type using the LCD touch screen. After running the test, we displayed the waveform and analyzed the protocol on the LCD. We determined whether the test result was correct by comparing the value sent by the serial port tuning assistant and the value generated by the logic analyzer.

- *SPI analysis test*—We used an SPI signal, generated by the Philip's single-chip microcomputer SPI interface, to illuminate the digital LED. We fetched the `SC`, `SDI`, `SDO`, and `SCK` SPI interface signals and connected them to the logic analyzer test pin. Next, we set the channel number for the `SC`, `SDI`, `SDO`, and `SCK` signals, and set the clock rate and signal lock-and-save mode on the PC. After testing, we displayed the waveform and analyzed the protocol on the PC. We determined whether the test result was correct by checking the segment code table of the 7-segment digital LED.

Synchronization Display Test

For this test, we connected the LCD and PC to the logic analyzer and performed the protocol tests. Both terminals displayed the results and waveforms.

Idiographic Test

This section descibes the idiographic tests. We use an oscillograph, a multi-meter, concentrators, and a PC to perform the tests.

Logic Signal Test

Our testing results showed that our design functioned correctly, for all storage depths, triggering point positions, and sampling clocks. Figures 26 and 27 are the bus signal testing waveforms, which tested the 51 single-chip microcomputer's address and data bus.

Figure 26. Bus Test Signal (PC Display)

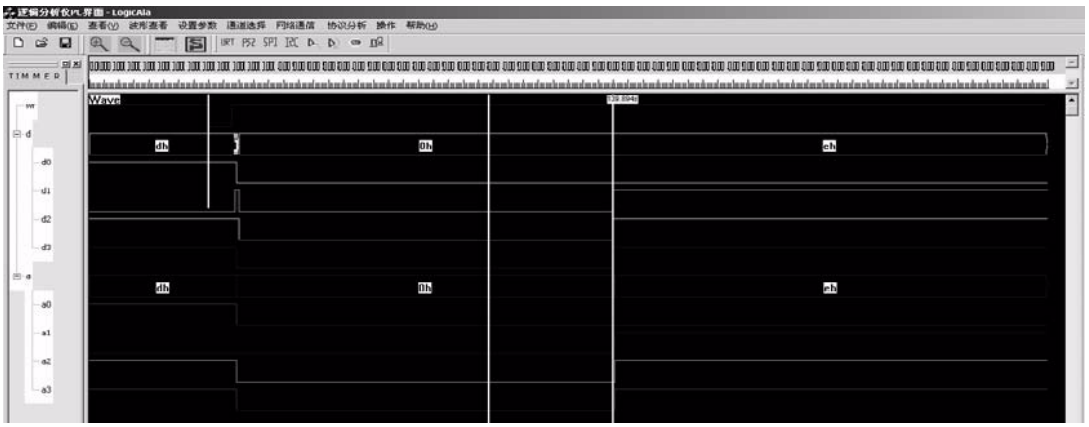
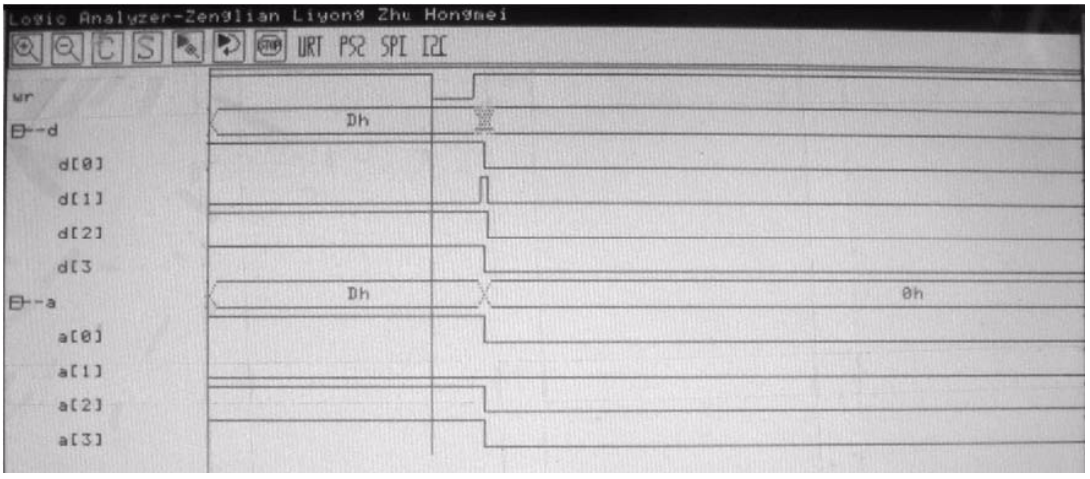


Figure 27. Bus Test Signal (VGA Color Monitor)



Figures 28 and 29 are the waveforms for multi-level triggering signal testing. We used four-level triggering, conditioned by 1-1-0-1 serial of the CH 24 signal. The external clock was the sampling clock.

Figure 28. Shifted Test Signal (PC Display)

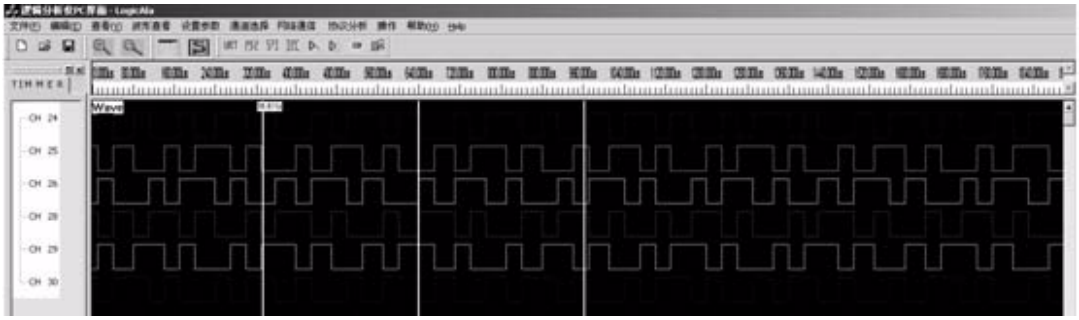
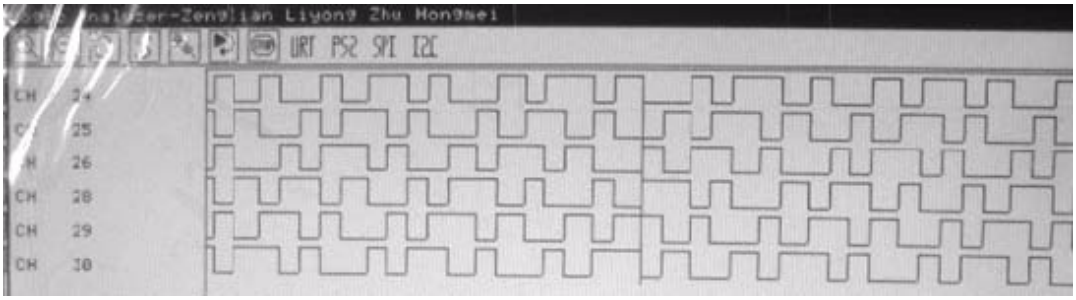


Figure 29. Shifted Test Signal (VGA Color Monitor)



Protocol Test

The test was successful. Figure 30 shows the I²C waveform, which was displayed on the PC screen. Figure 31 is the synchronously displayed waveform on the LCD. We pressed the 6 key on the I²C keyboard. The program lists the key codes according to I²C keyboard's ZLG7290 chip information. The key code is 70h-1h-71h-33h-0h-ffh, which corresponds to 6.

Figure 30. I²C Protocol Test Signal (PC Display)

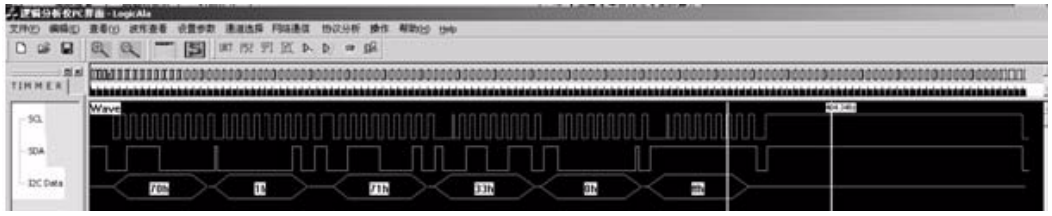
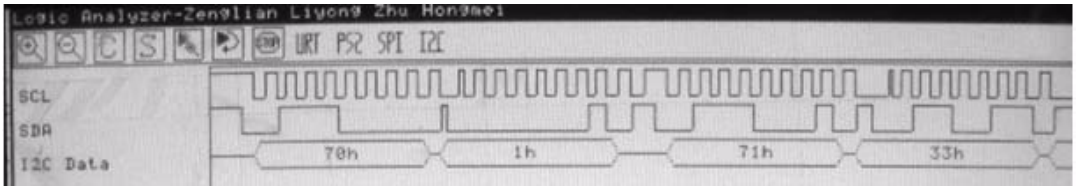


Figure 31. I²C Protocol Test Signal (VGA Color Monitor)



The test was successful. Figure 32 is the PS2 waveform displayed on the PC screen and Figure 33 is the waveform synchronously displayed on the LCD. We pressed the Enter key on the PS2 keyboard. 5Ah is displayed, which corresponds to Enter.

Figure 32. PS2 Protocol Test Signal (PC Display)

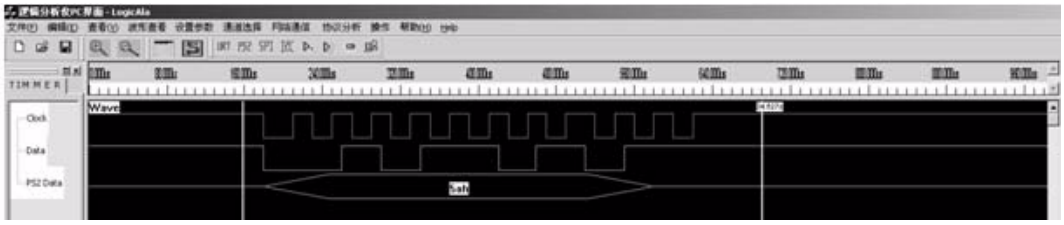
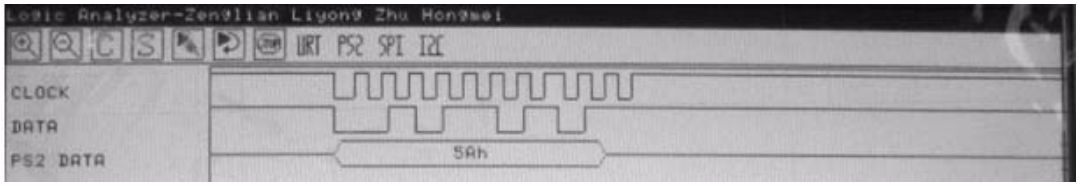


Figure 33. PS2 Protocol Test Signal (VGA Color Monitor)



The test was successful. Figure 34 is the serial waveform displayed on the PC screen and Figure 35 is the waveform synchronously displayed on the LCD. The serial tuning assistant delivered the data 1-2-3-4-5-6-7-8-9-0. The serial port actually transfers the data's ASCII code. 31h-32h-33h-34h-35h-36h-37h-38h-39h-30h was displayed, which corresponds to the data sent.

Figure 34. Serial Protocol Test Signal (PC Display)

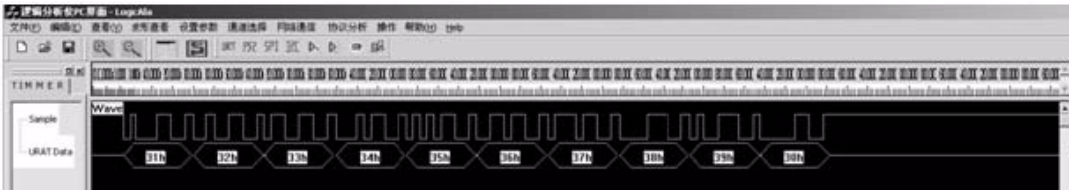
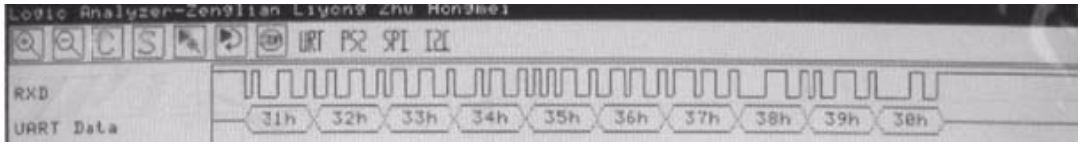


Figure 35. Serial Protocol Test Signal (VGA Color Monitor)



This test was successful. Figure 36 is the SPI waveform displayed on the PC screen and Figure 37 is the waveform synchronously displayed on the LCD. The Philips ARM device sends a one-way `SDO` signal and lights a numeral tube via the SPI interface. The `SDI` signal obtained the segmented code 3, which corresponds to 4Fh as shown.

Figure 36. SPI Protocol Test Signal (PC Display)

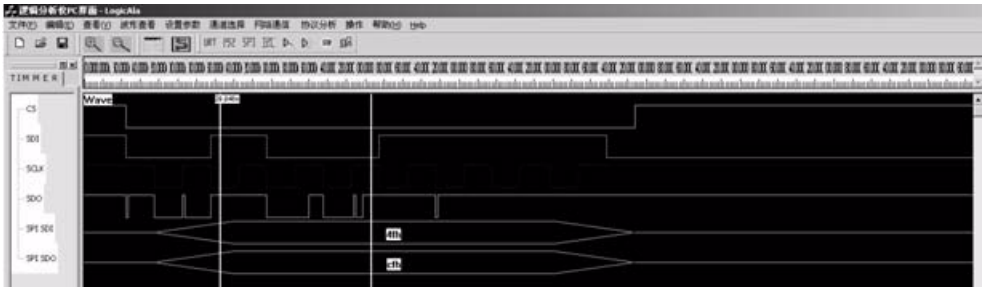
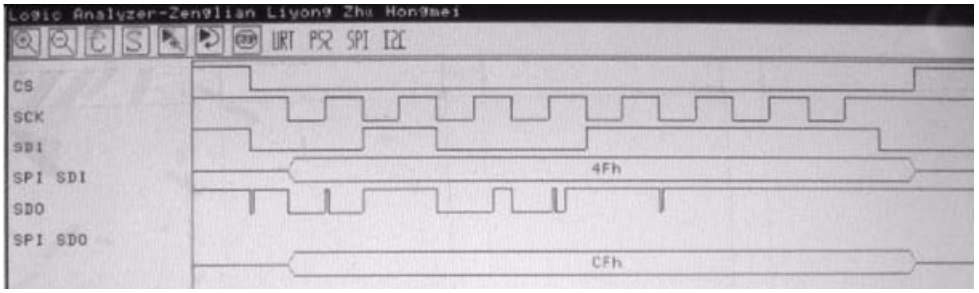


Figure 37. SPI Protocol Test Signal (VGA Color Monitor)



Testing Result and Functions Achieved

We drew the following conclusions from our testing:

- This design implements the following logic analyzer functions:
  - Variable storage depth
  - Variable trigger point position
  - Internal and external trial planting modes
  - Multi-level triggering
  - Bus measurement and display
- The design has protocol analysis capability, and can analyze I^2C, PS2, UART, and SPI signals. If the storage depth permits, the system can analyze all signals for these four protocols.
- The design supports a remote display. The PC terminal correctly analyzes and displays data when connected to the logic analyzer and the concentrator.
- Because of synchronous updates and an independent setup, the two display terminals do not interfere with each other.

Design Features

Our design has the following features:

- *Low cost and high performance*—The single chip logic analyzer is small, easy-to-carry, and cost effective, which satisfies the needs of most engineering technicians. Compared to the expensive logic analyzers currently on the market, this system provides excellent performance at a lower cost.
- *Portability*—The single-chip FPGA and Nios II processor can implement data collection, analysis, storage, control, and transfer, which allows the logic analyzer to migrate from large a desktop to small handsets. Additionally, it provides portable terminals suitable for working outdoors.
- *Remote display and multi-user collaboration*—Ethernet technology gives the logic analyzer remote control and display capability. Data transfer rates in high-speed LANs improves the system's real-time performance. Using a network, many users can share the same device for cooperative tuning and analysis.
- *User input supported via touch screen*—The system uses a touch screen, instead of a mouse and keyboard, to receive the user's input. The portable terminal greatly simplifies the logic analyzer's operation.
- *Real-time voice accounting*—The system broadcasts the protocol data while performing its analysis, which helps users cooperate when running the machine and sharing the resources.
- *Interface variety*—The VGA and network interface can both control the logic analyzer. Therefore, the system provides portable and remote terminals that can work for various situations.
- *Comparison and analysis of the logic analyzer*—Traditionally, the logic analyzer can only analyze timing and state. This system, using the processing capability of the Nios II processor and the FPGA hardware's flexibility, can analyze timing and state as well as varied logic.
- *System upgrades*—Combining the Nios II processor and FPGA allows this system to improve hardware functionality and implement software updates.
- *Embedded processor technology*—The system uses the FPGA to collect and analyze data and process it in parallel. The Nios II processor reads and writes to the external peripherals, which speeds processing and ensures real-time operation. Custom external peripherals, connected with SOPC Builder, increase the design's flexibility.

Conclusion

By participating in Altera's SOPC embedded processor competition, we obtained basic knowledge of FPGAs, SOPC concepts, and the Nios II processor, which changed the way we thought about FPGAs. SOPC concepts improve upon the FPGA functions (e.g., parallel operation, high speed, logic, simple timing, and complex operations) by integrating an external control unit (such as a single-chip microcomputer). The designer can easily add or remove external peripherals and interfaces from the Nios II processor, making a seamless interface between processor and hardware logic. We can abandon the traditional design concept that design alteration causes hardware modification, streamlining the design process. The Nios II processor decreases problems with hardware threading and ensures system stability.

SOPC concepts and the Nios II processor have totally overturned the traditional embedded system. Instead of passively adapting to the hardware processor, we can customize both hardware and software. The Nios II processor improves upon the embedded system's hardware circuit with its simplicity, efficiency, and understandability. Collaborative development of software and hardware enables the

synchronization of FPGA logic development and Nios II program development in a single FPGA, which greatly increases design efficiency.

The logic resources and external peripheral resources, as well as intellectual property (IP) cores provided with Altera's latest DE2 development board is very helpful for function expansion. Using the development board, FPGA, Nios II processor's collaborative development, and the dual CPUs, our system has better performance.

We were able to implement dual-CPUs and remote network control for the competition, but it is a pity we did not transplant the embedded operating system into the Nios II processor as well. Performing this task would have been difficult, because network communication would be implemented with the MAC protocol. However, implementing the embedded operating system in the Nios II processor would facilitate networking, documentation, and GUI design. We plan to work on this aspect of the design in the future.

SOPC design concepts, the Nios II processor development method, Quartus II development tool, SOPC Builder, and Nios II Integrated Development Environment (IDE) provide us with the system design. Specifically, we can embed a real-time operating system in the Nios II processor to strengthen theCPU's function. We think that SOPC, a programmable system-on-chip, will have a rosy future and be more widely used in people's lives.

Acknowledgments

We would like to express our gratitude to the Altera 2006 Undergraduate Electronic Design Competition SOPC organization committee in the Hubei province, and Altera (China). Without your efforts, we would not have the opportunity to learn new technology, expand our horizons, and inspire our sense of innovation. We also wish to thank Huazhong University of Science and Technology and its Electric Engineering and Electronic Innovation Center for the support, environment, and instruction they provide.

References

Peng Chenglian. *Challenge SOC-Nios Based SOPC Design and Practice*. Beijing, Tsinghua University Press (2004).

Zeng Fanta. *Survey of EDA Engineering*. Tsinghua University Press (2002).

Chu Zhenyong. *FPGA Design and Application*. XiDian University Press (2002).

Wang Jianxiao, Wei Jianguo. *SOPC Preliminary Design and Practice*. XiDian University Press (2006).

Pan Song, Huang Jiye, Zeng Yu. *Practical Course of SOPC Technology.* Tsinghua University Press (2005).

Ren Aifeng, Chu Xiuqin, Chang Cun, Sun Xiaozi. *FPGA Based Embedded System Design*. XiDian University Press (2004).

Kairus J., Forsten J., Tommiska M., and Skytta J. *Bridging the gap between future software and hardware engineers: a case study using the Nios softcore processor Signal Process*. Labratory, Helsinki University of Technology, Espoo, 33rd Annual Finland Frontiers in Education, (2003).

Du Huimin. *Verilog Based FPGA Designing Basis*. XiDian University Press (2006).

Luthra M. Sumit Gupta Nikil Dutt Rajesh Gupta Nicolau A. *Interface synthesis using memory mapping for an FPGA platform.* USA Computer Design (2003).

Wang Jianxiao. *SOPC Preliminary Design and Practice*. XiDian University Press (2006).

Member of China Computer Federation Microcomputer Society. *The 5th National Embedded System Academic Exchange Symposium in 2004*. China Computer Federation Microcomputer Society (2006).

Ben Atitallah, A. Kadionik, P. GHozzi, and F. Nouel P. Masmoudi N. *Marchegay Ph. Hardware platform design for real-time video applications*. Tunisia Microelectronics, ICM 2004 Proceedings (2004).

Altera Nios II and other product documentation.