Third Prize

Fingerprint Identification System Based on the Nios II Processor

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

Participants: **Linchuan Li, Yao Zhang, Chengdong Ge**

Instructor: **Xiao Kan**

Design Introduction

With the advent of fast-growing digital, information, and network technologies and the desire for a more convenient lifestyle in recent years, users have higher security expectations for electronic systems. Additionally, e-Business, ATM, access control, and intelligent cards all require a safe and easy-to-use identification technology. The traditional identification method, user ID plus password, cannot satisfy users' needs due to various defects such as forgotten passwords, hacker attacks, and theft. Fortunately, identification technology based on biometric characteristics of the human body offers an efficient solution.

This technology uses human physiological features and behaviors to identify a user's ID, and it is more secure, reliable, and human-oriented. Common biometric characteristics used for identification include fingerprint, palm print, iris, face image, voice, handwriting, DNA, etc. Considering the accuracy, durability, convenience, and cost, fingerprint identification technology features a high benefit/cost ratio, security, maturity, and widespread applications. Statistically, fingerprint identification products account for over 90% of the total biometric identification systems in China.

As microelectronics technology advances, programmable logical controllers are becoming more diversified, faster, and more powerful. Today, many FPGA devices support embedded soft-core processors to facilitate the development of FPGA-based hardware. For example, Altera's Nios® II processor, a RISC CPU soft core, features pipelining and a single instruction flow. Designers can embed it in an FPGA and leverage custom logic to build a FPGA-based system. Compared to an embedded hard core, a soft core is more flexible. Additionally, the fast FPGA meets the speed requirements of a fingerprint identification system.

Based on a separated identification system in verification mode, this design authenticates or registers users after they state their ID (i.e., by inputting an ID) and input their fingerprint at a terminal. It also allows a host to manage multiple terminals via the network, and allows an administrator to administrate the system.

Target Users

By integrating other functional service modules, the system can serve as a public service system where the fingerprint becomes the key ID authentication tool. Additionally, the system can be used as an HR management tool or security protection product. The design could be used in the following applications:

- *e-Business*—Credit card consumption, e-buying network
- *Banking*—ATM
- *Enterprises and institutions*—HR management
- *Security administration*—Access control system
- *Qualification*—Examinations

The system's host/slave network mode efficiently implements separated authentication and centralized management, making it suitable for partial authentication. If we improved the communication efficiency and security, it could be extended into a larger system.

Nios II Advantages

Most traditional fingerprint identification technologies depend on a PC or digital signal processor. However, image processing on PCs is expensive, has low speed, and requires a large storage space. Digital signal processors are not flexible enough due to the function and parameter limitations. FPGAs are advantageous for use in fingerprint identification because they feature high processing speed, flexibility, low cost, and embedded system portability. As a high-performance and configurable soft core, the Nios II processor has unique features. Designers can use the C language with short development cycles and portable code. Combined with custom hardware logic, they can conduct complex parallel image processing without losing the advantages of using an FPGA. For our system, it is easy to integrate the relevant components and we can easily adjust the relevant image processing parameters. As a result, the system satisfies various performance indicators and users in different conditions.

Function Description

This section describes the functionality of our design.

Scalable Authentication Network

With a host plus terminals mode and a bus-type local area network (LAN), the system can be centrally managed and extended. By adding terminals into the Terminal Management tab, the host administrator can conveniently add new terminals.

Excellent User Interface

We use an LCD and keyboard to facilitate operation.

Fingerprint Collection

The fingerprint collector collects the user's fingerprint and a driver accesses the serial peripheral interface (SPI) to get data. The program has an automatic finger detection function. Three sets of parameters are allocated to account for the skin moisture level when fingerprints are collected. The

system picks best image of the three. Additionally, the fingerprint collector stays in sleep mode and is only activated during fingerprint collection, minimizing power consumption.

ID Verification

The terminal collector collects fingerprint signals, processes images, and extracts fingerprint minutiae information. After registration, the host acquires the fingerprint and corresponding ID information from the terminal and stores them in a fingerprint database. Upon login, the host returns the corresponding fingerprint information based on the ID, and the slave compares and displays the corresponding login information.

Information Management

The host is a powerful PC that operates a set of management programs, including:

- *User account management*—View or modify registered users.
- *Terminal management*—Add new terminals or modify the terminal priority.
- *Log review*—Review the system access log.
- *Password change*—Change the administrator's login password.

Performance Parameters

Performance parameters include the fingerprint image processing speed and accuracy.

Fingerprint Image Processing Speed

Table 1 shows the time used by the main processes and the total time required for fingerprint image processing before and after hardware acceleration (for the configuration and hardware acceleration principles, see "Design Methodology" on page 288). We use a 256 x 300 8-bit grayscale image as the object that is processed. Using hardware acceleration for image processing greatly improves the processing speed.

Table 1. Image Processing Speed

Operation	Time Required before Hardware Acceleration (Seconds)	Time Required after Hardware Acceleration (Seconds)
Image filtering	36.40	4.77
Ridge thinning	13.54	2.67
Total	54.93	11.57

Fingerprint Identification Accuracy

Because the system's fingerprint image processing and comparison algorithm are designed for a special fingerprint collector, we do not use a universal fingerprint database in the test. Instead, we chose about 40 fingerprints of 10 people at random to test the system's fingerprint identification accuracy.

The statistics show that the system's false accept rate (FAR), i.e., the probability of mistaking non-identical fingerprints as identical fingerprints, is less than 5%. The false reject rate (FRR), i.e., the probability of mistaking identical fingerprints as non-identical fingerprints, is less than 20%.

Identification accuracy is greatly influenced by the skin's cleanliness and moisture, these results are the system's comprehensive fingerprint identification performance and are not from a separate algorithm performance test.

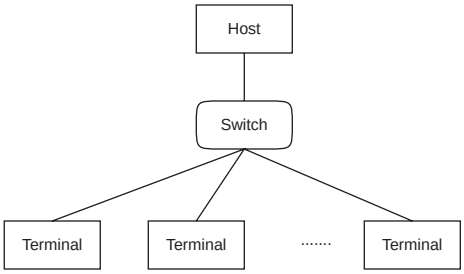
Design Architecture

This section describes the design architecture, including the network topology, modules, hardware, and software design.

Network Topology

Figure 1 shows the network topology. In the system, a switch is the central node that connects the terminals and the host.

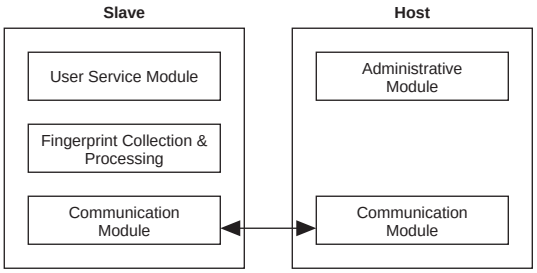
Figure 1. Network Topology



Module Division

Figure 2 shows the modules in the design.

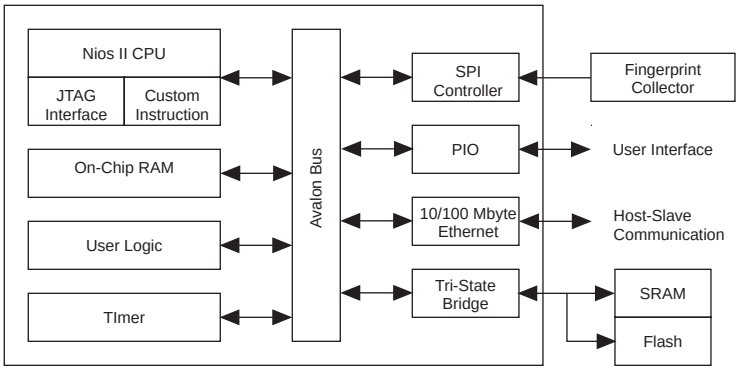
Figure 2. Modules



Hardware Design

Figure 3 shows the hardware design.

Figure 3. Hardware Design



Software Design

Figures 4 through 7 show the flow charts for the system.

Figure 4. Slave Software Flow Chart

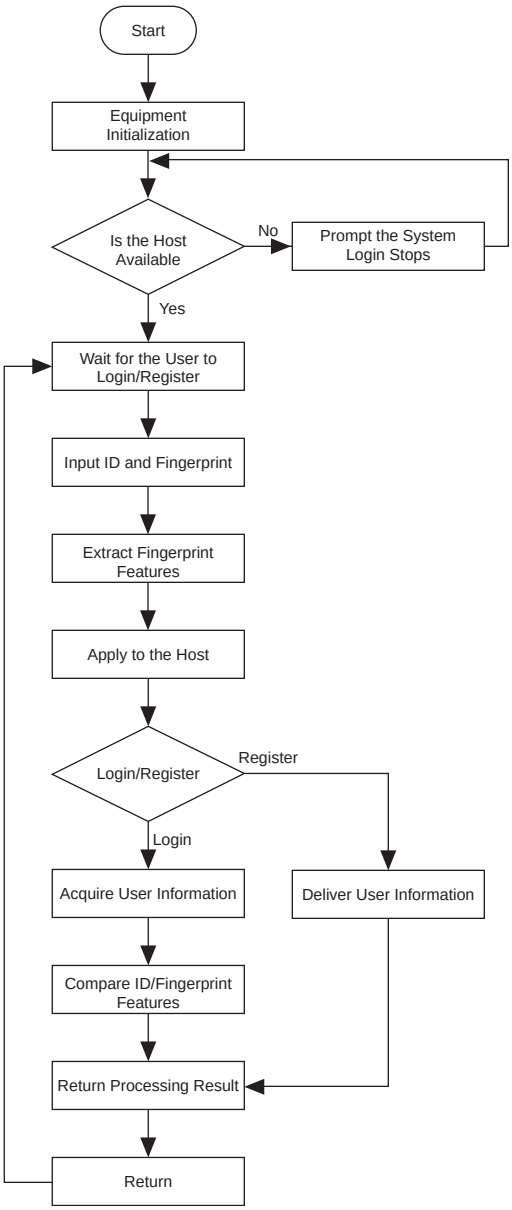


Figure 5. PC Host Flow Chart

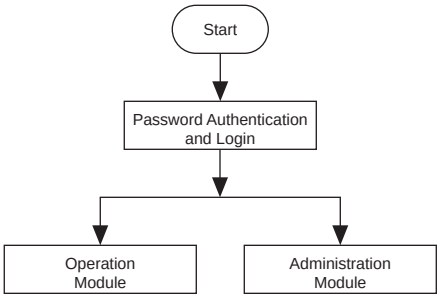


Figure 6. Operation Module Flow Chart

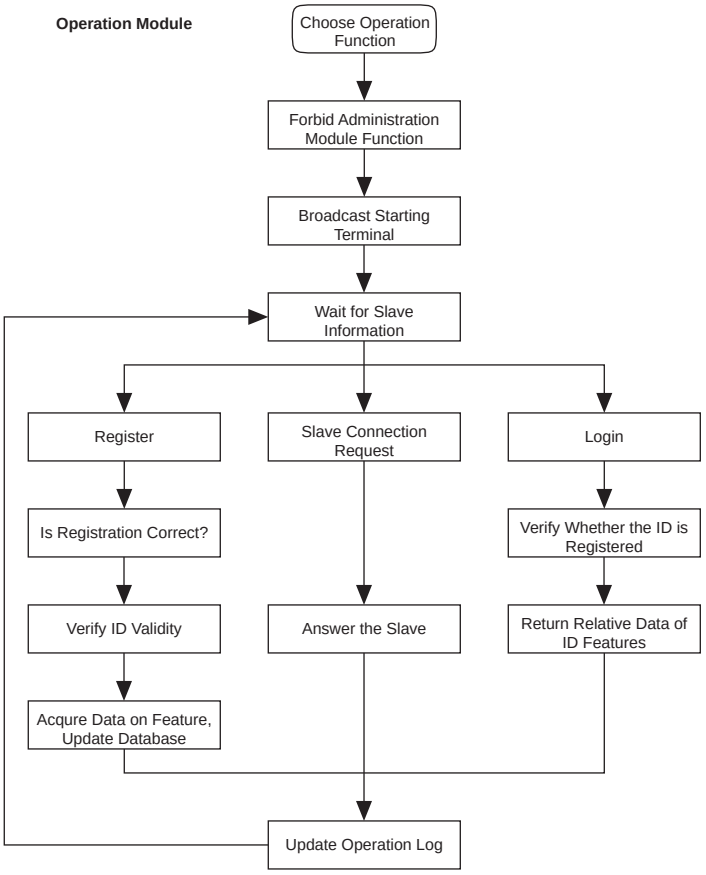
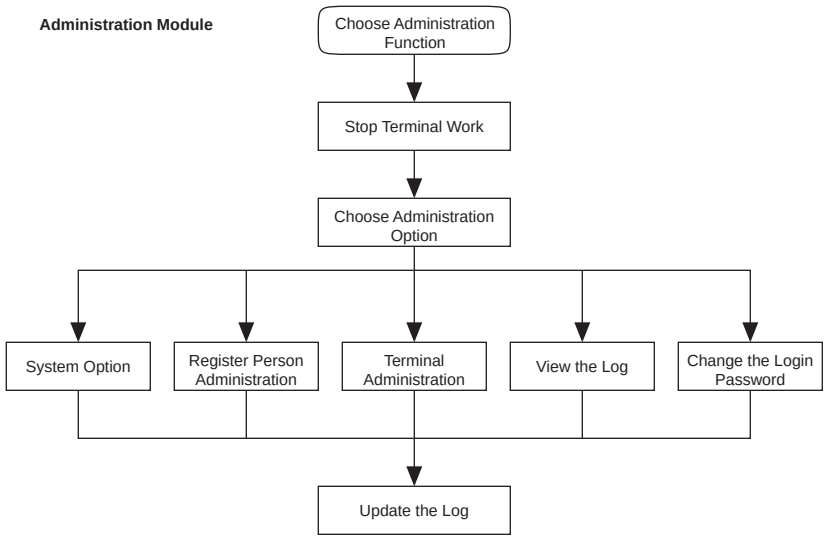


Figure 7. Administration Module Flow Chart



Design Methodology

This section describes our design methodology.

System Development Flow

Most of design's functions are implemented using the Development and Education (DE2) board and a peripheral fingerprint collection circuit. We implemented the system's functional modules, assembled the modules, and debugged the system as described below:

- Referring to examples and test documents provided with the DE2 development board, we implemented the Nios II processor, performed C language programming in the Nios II Integrated Development Environment (IDE), performed on-line program debugging, and loaded the program onto the board.
- Using SOPC Builder, we created access to the peripheral memory, RS-232 serial port, DM9000A network interface, and programmable I/O (PIO) interface on the DE2 board.
- We implemented access to the 4 x 4 keyboard using the expanded PIO interface on the DE2 board.
- We used SPI on the DE2 board to read the fingerprint collection data.
- We used custom peripherals and instruction hardware acceleration on the DE2 board to complete the fingerprint processing algorithm.
- We purchased a fingerprint collection chip, which we used to design and implement the fingerprint collection circuit, automating the finger detection function.
- We developed the administrator program for the PC, and debugged the system.

Hardware

The following sections describe the method we used to create the hardware system.

Nios II Soft Core Configuration

We configured the Nios II processor as described below:

- The Cyclone® II EP2C35 FPGA on the DE2 board is the control chip, and we designed all functions based on it. The hardware represents highly-integrated system-on-a-programmable-chip (SOPC) design ideas and principles.
- We used the 50-MHz Nios II/f fast CPU to support JTAG level 3.
- We used a 12.5-MHz SPI core to transfer fingerprint data.
- We added the DM9000A Ethernet control chip and implemented an Ethernet physical layer.
- We added the JTAG debug module to facilitate on-line system debugging.
- We used a 38,400 bps serial port communication module. With a UART, we can transfer data between the Nios II system and the PC before fully implementing network communication and monitor the fingerprint image processing procedures on the PC.
- SRAM, SDRAM, and flash memory are used to implement the program operation memory, data distribution space, and program writing space, respectively.
- The design has a tri-state bridge connection between memories.
- Two timers implement the system's delay requirements and test the time that the Nios II system requires to process the fingerprint data.
- Keys, nixie tubes, an LED, and a 16 x 2-character LCD interface facilitate the user interface.

Using SOPC Builder, we conveniently built a tailor-made, configurable system that met our requirements.

Making the Fingerprint Collector

We designed the fingerprint collector as described below:

- *Functional design*—We used the fps200 fingerprint collection chip to collect the original fingerprint image data. The circuit has an automatic finger detection function and only instructs the CPU to accept data when a finger is detected. The image data is transferred to the Nios II processor via the SPI.
- *Schematic diagrams*—After studying the fingerprint collection chip data sheet, we designed the schematic diagram to implement the SPI and preserve the microcontroller (MCU) and USB interfaces.
- *PCB schematics*—We used a two-side PCB for the final circuit board.
- *Circuit board welding and debugging*—Our experiments verified the functions of the fingerprint collector, data collection, and data transfer. We adjusted the parameters related to fingerprint collection as needed.

Fingerprint Image Processing Hardware Acceleration

When the Nios II CPU is configured as fast and the fpoint operation instruction is added, the image processing algorithm using C requires 50 seconds (see "Performance Parameters" on page 283 for more details). This speed is acceptable for a real-time processing system.

Two methods can improve the image processing efficiency: implement a digital signal processing (DSP) module on the FPGA or use hardware acceleration. DSP can process images more efficiently without depending on the CPU operation speed, but it is difficult to implement. In contrast, it is easy to use hardware acceleration for image processing. Therefore, we decided to use hardware acceleration, which greatly improved the algorithm performance efficiency.

The fingerprint image processing algorithm has unique features: it requires a large number of repeat operations and identical pixel processing procedures. Therefore, if we improved the performance efficiency of the most fundamental operations, we could improve the performance efficiency as a whole. We divided the fingerprint processing procedures into pattern finding, image filtering, binarization, ridge thinning, minutiae location, etc. (see "Software" on page 290 for more details), and calculated the time required for each procedure. We determined that two procedures should be accelerated: image filtering (36.4 seconds) and ridge thinning (13.5 seconds).

To perform image filtering, we take the data of 52 pixels around the target pixel, multiply the data with corresponding filtering coefficients, and accumulate the results to use as the new value of the target point. In the procedure, the filtering coefficient is taken from a coefficient array with relevant directions. The complete software procedure requires 52 multiplication accumulations. We designed a custom instruction `CI_multi_accumilate`, to complete one multiplication accumulation in three clock cycles. After the acceleration, the image filtering procedure uses only 4.77 seconds.

To perform ridge thinning, we take the data of the target pixel and 15 pixels around the target pixel and compare them with 16 templates to decide whether the target pixel should be removed. After scanning the whole image many times, the fingerprint ridge is thinned into a single pixel width. The complete software procedure requires 16 comparisons. We designed two custom peripherals, `prematch` and `user_delete`, to complete the 16 comparisons in six clock cycles. After acceleration, the ridge thinning procedure uses only 2.67 seconds.

Ethernet Implementation

In the system, a switch is the central node that connects the terminals and the host. Because there is not a large volume of data and no complex routing in the communication between the terminals and the host, the system does not have a physical connection with the public network. However, an embedded real-time operating system (RTOS) must implement a good IP protocol. It is unnecessary for our slave system to use an operating system for dispatching; therefore, we did not use the TCP/IP protocol to build the network. Instead, we used a MAC-to-MAC method for physical addressing, that is, the system directly sends an original data packet that adds source and target MAC addresses as headers.

Software

The software design includes the Nios II program and PC program. The PC program is the fingerprint authentication administrator program that fulfills functions such as responding to login/register requests and system administration.

The Nios II program includes the initialization, fingerprint collection, image processing, host-slave communication, user interface, etc. modules. By organically combining the modules, we created a simple, client fingerprint authentication program as described below:

- *System initialization*—After the system is powered-on, all modules must be initialized. In particular, the network adapter must be initialized to connect the Nios II processor to the PC. Initialization makes the fingerprint collector go into a low-power mode and it does not wake up until a fingerprint needs to be collected.

- *Fingerprint collection*—Based on a driver, the fingerprint collector obtains images through the SPI. The procedure must perform automatic finger detection and skin moisture adaptation.

- *Fingerprint processing*—This module provides fingerprint data processing and comparison. It includes two sub-modules: fingerprint image processing and fingerprint comparison.

- *Host-slave communication*—This module implements communication between the Nios II slave and the PC. Its primary transmitting function is to request commands such as register and log-in requests, fingerprint minutiae information, ID information, etc. Its primary receiving function is to obtain the PC's reply, the Nios II control information sent by the PC, the fingerprint template information that the PC returns to the Nios II processor, etc.

- *User interface module*—This module consists of a 4 x 4 keyboard, 16 x 2 LCD, LED, and nixie tube. The LED indicates the current working status of the Nios II system as well as information such as success, failure, and timeout. The nixie tube displays the input ID information, and the LCD displays real-time system information such as the system status and operation prompt.

The LAN transfers original data packets that use source and target MAC addresses as headers; therefore, we use the winpcap (windows packet capture) protocol on the LAN. winpcap is a free public network access system for Windows platforms, and it gives win32 applications the ability to access the network infrastructure. It provides the following functions:

- Captures the original datagram, including the datagram that hosts send/receive and exchanges them on a shared network.

- Filters special datagrams according to user-defined rules before they are sent to the application.

- Sends the original datagram over the network.

- Collects the statistical information in network communication. Our tests and final design demonstrate that winpcap conveniently receives/sends data packets between the PC and Nios II processor and also satisfies our functional requirements.

Fingerprint Image Processing Module

Figure 8 graphically shows the fingerprint processing procedure (see references [1] and [2] for more details).

Figure 8. Fingerprint Processing Procedure

The procedure includes the following elements:

- *Pattern finding*—This process includes two steps: first, it calculates the direction of single points based on the grayscale value of the points around the target point. For simplification, we divide 180° into 8 directions. Next, it obtains a 5 x 5 block pattern using a statistical method and marks the blocks without clear direction about the background. Pattern finding lays a foundation for the direction-based image filtering.

- *Image filtering*—We designed filtering coefficient templates for 8 directions in advance, took 52 pixels of data around the target pixel, multiplied the data with the corresponding filtering coefficient, and accumulated the result, which is used as the new value of the target point. Filtering enhances the image continuity along ridges and improves the image contrast perpendicular to the ridges to segment neighboring ridges.

- *Binarization*—After filtering, the image ridges are distinct. Therefore, we only need to use fixed threshold binarization, i.e., to segment the image into black and white images by taking a fixed gray value as the standard. After binarization, the image can be completely thinned.

- *Ridge thinning*—We use an OPTA (or parallel thinning) method to corrode the ridges gradually until the ridges are thinned into a single pixel width. Ridge thinning facilitates minutiae location.

- *Locating minutiae*—First, we scanned the pattern to locate the central point of the fingerprint. Then, we located the tip points and the split points in the thinned ridge image. To find the points, we assumed that the tip points are black and surrounded by only one black point and spit points are surrounded by three black points.

Fingerprint Comparison

Through many tests, we improved the central point-based fingerprint comparison method into three steps: coarse matching, coordinate transformation, and precise matching.

- *Coarse matching*—The system makes a coarse match for two fingerprint images by taking their central point as the original point and ±30° as the range for the angle of rotation. Then, we ascertain the most matched angle of rotation and some minutiae pairs.

- *Coordinate transformation*—Based on obtained minutiae pairs, we calculate the precise coordinate translation and rotation parameters of the two images.

- *Precise matching*—Based on the obtained coordinate translation and rotation parameters, we precisely match the minutiae a second time and evaluate the degree of matching. The system gives three points to point pairs matched in type (tip or splitting) and location, and gives two points to those matched in location only. If the total score is more than the threshold value, there is a match. If any step fails, it is not a match.

The central point-based fingerprint comparison method efficiently resists the interference caused by image translation. That is, the combination of the final precise matching with the point-to-point comparison method partially resists interference caused by central point location errors. Considering that the minutiae type (tip or splitting) is usually incorrect, we regard cases where the fingerprints have unmatching types but matched location as matching, improving the comparison accuracy.

Design Features

Our design has the following features:

- *Bus topology between hosts and slaves*—Because organizations or institutions may identify fingerprints in different locations and it is impossible to configure a system at each location, we use hosts and slaves to fulfill the task. Slaves collect and process fingerprints and transfer the processed results to the hosts over the network. The hosts provide management and storage. In this way, hosts and slaves have clear duties to leverage their own advantages.

- *Custom instructions to acceleration of key algorithms in hardware*—Extracting fingerprint minutiae is a complex DSP function, and it greatly slows the system speed if software is used for extraction. Fortunately, Altera's SOPC Builder software allows users to create and deploy their own Nios II system as well as add custom instructions. Therefore, the system uses a hardware description language to implement the minutiae extraction algorithm. We used custom instructions to define the IP algorithm as a special instruction that directly invokes processing, implements hardware acceleration, and greatly improves system speed.

- *Ethernet transmission*—We used Ethernet to transmit requests from the slave to the host for processing. Additionally, we used a competitive terminal access mechanism that provides high efficiency when the system load is light. Ethernet ensures the application of the system and allows terminals to be added conveniently in the future.

- *Easy hardware and software upgrades*—It is impossible for an application to fully meet the requirements of each user in a huge user base. However, Altera's SOPC system design solution

can solve the problem, allowing different users to modify the hardware and software easily based on the original system. They can develop new products and improve competitiveness by utilizing FPGA programmability. In the system, we can put the fingerprint minutiae extraction module at terminals or hosts based on the number of user terminals, balancing the communication traffic and host load. In this way, we dramatically improved the adaptability of the system.

- *Benefits in cost, power consumption, portability, and integration*—The FPGA-based system design integrates processor, peripherals, memory, and I/O interface into a single FPGA, reducing the system's costs, complexity, and power consumption. Additionally, because the Nios II processor is superior to hard cores in terms of cost, the system has higher integration and is more cost-effective.

Conclusion

The two-month contest was a process of SOPC development, learning, and application, as well as learning about Nios II embedded processor design. By using the development and design technology, we now understand the technology more. We learned the following things during the contest:

- *Easy-to-use design environment*—It was a challenge for us to gain basic knowledge about software and hardware development and conduct systematic design, implementation, and verification within two months. However, SOPC Builder and the Quartus® II software provided a visual, fast methodology and the Nios II IDE, which helped us achieve our goal in a short time with a fast learning curve.

- *Software/hardware co-design and verification*—Parallel hardware/software design is a crucial task for embedded system design. During the design, the major challenge is synchronizing and integrating software and hardware design. During the contest, we used SOPC Builder to conduct comparatively independent software and hardware implementation with comprehensive system design and the reasonable division of software and hardware. Independent functional verification shortens the development cycle.

- *Innovative Nios II system hardware acceleration*—Compared with traditional DSP solutions, custom instructions or peripherals can be better integrated into the Nios II system, ensuring a fast processing speed and flexible controls. Compared with an ASIC solution, the Nios II processor is more cost-effective because it can provide the same function using internal FPGA resources without adding other devices.

- *Improving Nios II stability*—The issue may not be apparent for small system designs but can occur during complex system designs. For example, random software operation errors occurred during same hardware deployment. The system handled the fault after we recompiled the wiring. The error was often due to an infrastructure I/O driver. In our opinion, the Avalon® bus is not a fixed connection wire but is a set of connection wires generated in the FPGA after Quartus II compilation, so the random wiring leads to bus instability. We did not verify our guesswork, but the problem was a big challenge.

- *Problem solving*—Although many problems are simple now, they seemed insurmountable barriers when we first met them and did not have enough information. We had to consider and collect various data, dare to practice, be persistent, and consequently find a way out of desperation. We made arduous efforts to discover solutions and tackle problems, but we gained knowledge and the methodologies to solve problems in the process, which will aid us in our future study and life.

References

Lingli, Liu, *Preprocessing and Minutiae Extraction of Fingerprint Image*, Master degree thesis of Hunan University, December 2005.

Chunlei, Li, *Research on Fingerprint Identification Algorithm and FPGA-based Hardware Realization*, Master degree thesis of Shandong University, April 2005.