Second Prize

Digital Watermark-Based Trademark Checker

Institution: **Institute of Information Science, Beijing JiaoTong University**

Participants: **Sheng-Kai Song, Wei-Ming Li, and Li Song**

Instructor: **Xiao-Ming Ding**

Design Introduction

A digital watermark adds undetectable identification information to a digital product that can be verified with certain algorithms. Using watermarking to prevent counterfeit printing is of great significance for social development. Digital watermark-based anti-counterfeit technology is different from traditional anti-counterfeit techniques and has its own features: it is imperceptible, unambiguous, robust, secure, advanced, and universal.

This design leverages Altera® FPGA and Nios® II resources, implementing a digital watermark trademark checking algorithm on the Nios II platform and using hardware to implement the watermark checking module. The system has a reusable intellectual property (IP) core, which protects the algorithm IP and facilitates a new trademark anti-counterfeiting method.

The algorithm-protected object is the printed trademark image, and all such images can be added with the anti-counterfeit watermark during printing. When the checker reads the trademark through the interface between the checker and the image collector during checking, the watermark checking algorithm embedded in the checker can check the identification for the trademark.

The checking object is the printed trademark image and it identifies whether a commodity infringes the IP. The checker has an extensive user base, including supermarkets, wholesalers, and retailers who offer customers products with image trademarks. It provides an objective, convenient, quick method to check the commodity identification, which protects the IP and improves customer confidence in commodities. Additionally, with the proliferation of 3G mobile phones, this algorithm's IP core can be applied to the camera's mobile phone to implement remote on-line trademark checking for ordinary users.

The Nios II processor and FPGA achieve design portability and allow designers to integrate peripheral circuits and processors in the same chip, reducing system volume and scale and implementing the on-

chip system. Additionally, the PC-developed algorithm and C language program can be rapidly migrated to the Nios II processor to shorten the system development cycle.

With this solution, we can achieve algorithm/class system development using the Simulink software, DSP Builder, and IP cores. We can design important modules—or even the whole digital signal processing (DSP) system—as custom instructions with DSP Builder, according to the requirements of the DSP algorithm. Then, we can integrate the instructions into the Nios II processor using SOPC Builder and the Quartus® II software. The design performs the DSP algorithm through software by invoking the custom instructions. This process facilitates development and replaces the high-end DSP device that is used in traditional image processing.

Function Description

If the trademark image is printed or scanned (videography) once, the watermark algorithm is robust. However, if the resulting image is printed and scanned again, the algorithm is frail.

Figures 1 and 2 show the transmission process required to set the watermark image for fabrication and non-fabrication. After the transmission process, the watermark image becomes the trademark to be checked (using a camera to collect the image). The anti-counterfeit function operates as follows: if the trademark image is transmitted as shown in Figure 1, the checker indicates that the watermark exists; if the image is transmitted as shown in Figure 2, the checker indicates that the watermark does not exist.

Figure 1. Non-Fabrication Transmission Process

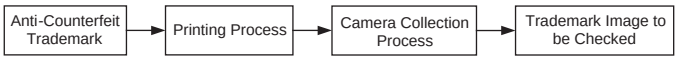
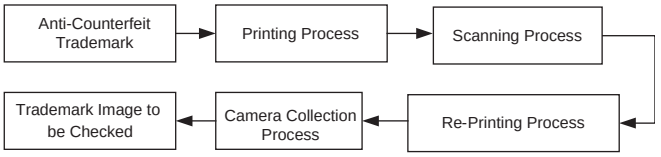


Figure 2. Fabrication Transmission Process



The design rapidly checks the protected trademark on- and off-line in the on-chip watermark checker. It leverages hardware to configure the reusable IP core and implement the patent algorithm for trademark image anti-counterfeit printing.

The design consists of a collection module, image display module, image preprocessing module, and watermark checking module. The Nios II CPU implements the system control function and the FPGA performs high-speed data processing. The design contains the following modules:

- *Nios II soft-core CPU*—Implements the required system management logic. Dispatches the relevant algorithm and controls the whole data flow from image collection and preprocessing to watermark checking. Manages devices, users, and remote interaction.
- *Image collection module*—Integrates the camera configuration module with the system bus, and becomes the self-defined peripheral that flexibly controls the camera. Collects the specific trademark image in memory through the on-chip control logic. Stores the image data in a block so that the system can transmit image data using direct memory access (DMA) mode.
- *Image display module*—Saves the collected, preprocessed image in SRAM. The module's read/write image data generates the VGA signal control ports, such as the horizontal and vertical sync signals that the monitor displays.

- *Image preprocessing module*—Performs geometric correction for the collected trademark image to be checked according to the rotation, horizontal move, and zoom control operations. Sends the parts that have large calculations and long processing time to the hardware module, which accelerates preprocessing.

- *Trademark checking module*—Has three parts, which are implemented in a combination of hardware and software:

  - *Software*—Operates the watermark checking algorithm in the Nios II CPU with software.

  - *Software and hardware*—Implements the main algorithm operation with a custom instruction and peripheral to accelerate operation.

  - *Hardware*—Implements the algorithm with a hardware description language and integrates it as an independent processing module. Saves the preprocessed trademark image into SRAM. That module automatically interacts with the SRAM, conducts corresponding operations, and returns the result after the operation finishes.

The design also has an LCD screen, LED light, and user keys to support user interaction.

Performance Parameters

For this system, accuracy and identification speed are the most important performance parameters. Therefore, we focused on these areas. Using the Altera design platform, we were able to speed up the design without lowering the design's complexity. A single identification, including complex preprocessing, watermark checking, accelerated C-to-hardware acceleration (C2H) preprocessing, and hardware watermark checking, takes about three seconds.

Based on the original limited print samples, we checked the trademark images of printing and reprinting. According to the experiment data, the threshold gives a 95% identification rate.

Watermark Checking Module Performance Analysis

The operation time when implementing the module with software alone is about 20 seconds.

The operation time when implementing the design using a custom instruction and peripherals for the key algorithm and a combination of software and hardware is about 7 seconds.

The operation time when implementing the design with hardware cannot exceed 20 million clock cycles. Therefore, at 50 million clock cycles, the operation is limited to 0.5 seconds.

C2H Compiler

The Nios II C2H Compiler can automatically integrate the high-performance C program into the hardware accelerator, which is then integrated into the FPGA-based Nios II subsystem. The C2H Compiler supports standard ANSI C code, accelerates multiple application programs, and improves operational efficiency, including access to local and external memory and peripherals. We can use the Quartus® II SOPC Builder to generate a broadband Avalon® interconnected architecture, which can process the external memory and peripherals, such as pointer dispersal and array access.

The Nios II C2H Compiler accelerates implementation of memory interfaces, and generates hardware accelerator logic and the correct Avalon host and slave interface to match the memory delay. It shares the data computing and memory access functions with the Nios II processor, and lets the processor perform other tasks. Because the Avalon architecture does not limit the number of hosts and slaves in a system, the Nios II C2H Compiler can generate multiple hardware accelerators according to the target code's transfer requirements. The C2H Compiler helps embedded system developers improve design efficiency.

In our system, the image preprocessing function is implemented in software. Because we have high-speed identification requirements and the C software code takes a long time to perform the task, we optimized the image preprocessing module with the Nios II C2H Compiler to accelerate processing.

We tested the implementation speed. Image preprocessing and integrating all the software requires 47 seconds. If we change the code into fixed point and optimize it using the C2H Compiler, the time is reduced to 2 seconds, a speed increase of more than 20 times.

The design uses extra logic resources, however: 65% instead of the original 20%.

Nios II Processor

The Nios II processor's excellent performance facilitated our design. We chose the Nios II/f CPU because we have high-speed processing requirements. We combined the processor, peripherals, memory, and I/O interface with the Nios II processor and FPGA design.

Because the Nios II processor is configurable, we can modify the system performance requirements at any time. Furthermore, we were able to improve the module performance with Nios II custom instructions.

Design Architecture

The hardware is composed of the watermark trademark image collection system, watermark extracting and identification system, and display system. Figures 3 and 4 show the general system and modules.

Figure 3. Data Flow

Figure 4. Hardware Structure

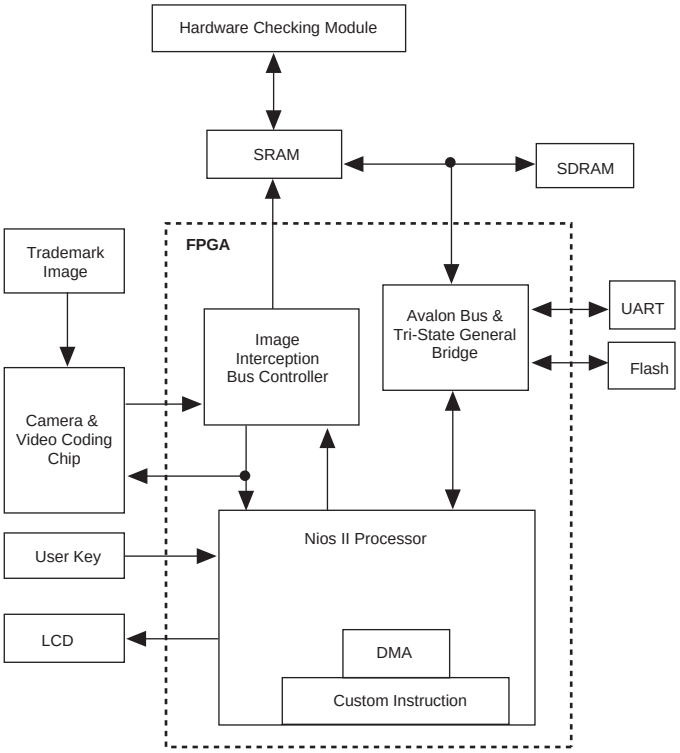
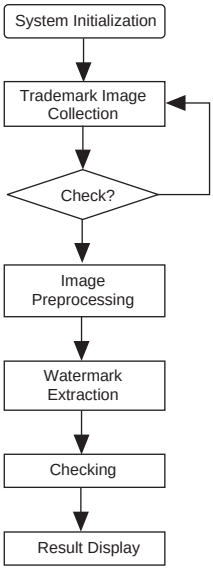


Figure 5 shows the device algorithm software flow.

Figure 5. Software Flow



Design Methodology

This section describes our design methodology.

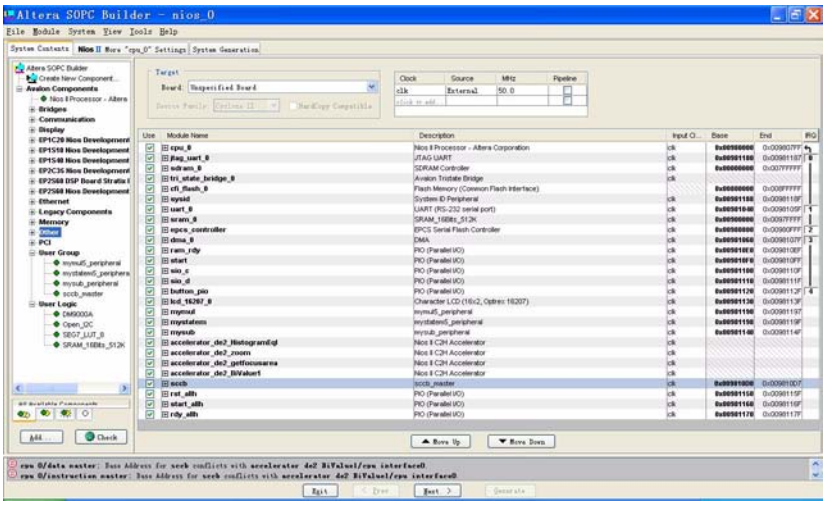
Design Environment

We used the Altera Stratix® development board for the initial code debugging and then used the Development and Education (DE2) development platform for final implementation. The DE2 board has a variety of integrated peripheral interfaces, which were convenient to use in our design.

Software and Hardware Design

Figure 6 shows the SOPC Builder configuration.

Figure 6. SOPC Builder Configuration



Modules and Implementation

This section describes the modules in our design and how we implemented them.

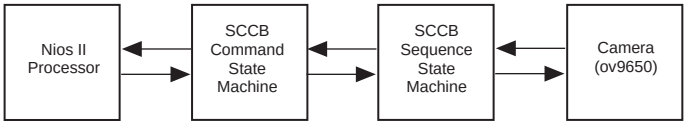
Trademark Image Collection Module

Trademark image collection is an important part of the front-end system because the collection quality can directly impact identification. Because we need high image quality, we use a mobile camera with 1.3 million pixels and the ov9650 chip sensor to allow the system to be more portable in the future.

The camera is controlled using the serial camera control bus (SCCB). To collect the image data, the system controls the SCCB to configure the ov9650 CMOS image sensor chip parameters. The SCCB provides simple control measures. Through the SCCB interface, we can modify the ov9650 CMOS image sensor chip register values, which allows us to control the device.

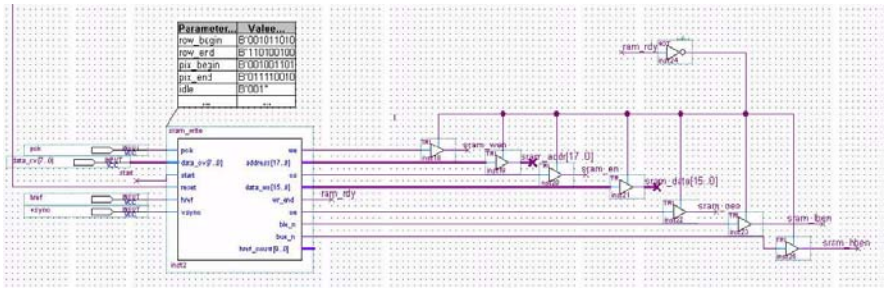
The SCCB is bidirectional (two-slave or multi-slave with a serial interface standard) and has a bus arbitration system that can perform short-distance and irregular data communication between the components. Because the system uses only one CMOS image sensor chip, it uses a two-slave SCCB mechanism. To keep the system small, use the FPGA logic resources effectively, and improve the hardware system integration, we designed SCCB control core in the FPGA. The SCCB control module is integrated into the system as a user-defined peripheral, and it can configure the camera's operations. Figure 7 shows the SCCB controller structure.

Figure 7. SCCB Controller Structure



We write and store the camera's image in the IP core's SRAM, and then integrate it into the system. The system performs a variety of image collection activities. It captures the image data required by the system (the collected image is 660 x 660 in YUV format; we remove the Y data to save storage space) and stores the data into corresponding SRAM. Figure 8 shows the SCCB module's symbol and connections.

Figure 8. SCCB Block Diagram



VGA Display Module

VGA display module shows the original trademark images we collected and the processed images so that we can observe the effects of the front-end image collection and image preprocessing. The VGA module reads the image data from the SRAM repeatedly. The module sends the data to the VGA chip with a module-generated control signal. An image can then be displayed on the LCD.

The DE2 development board uses the ADV7123 device to transmit a 10-bit digital-to-analog (D/A) video signal, which is connected to the 15-pin interface, as the VGA output. Figure 9 shows the VGA waveform. The vertical and horizontal time cycles are in four zones: H-synchronization (a), descend h (b), ascend (d), and display (c).

Figure 9. VGA Signal Waveform

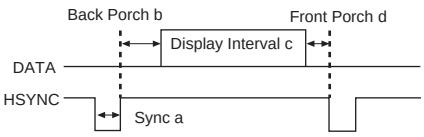


Figure 10 shows the VGA sequence standards.

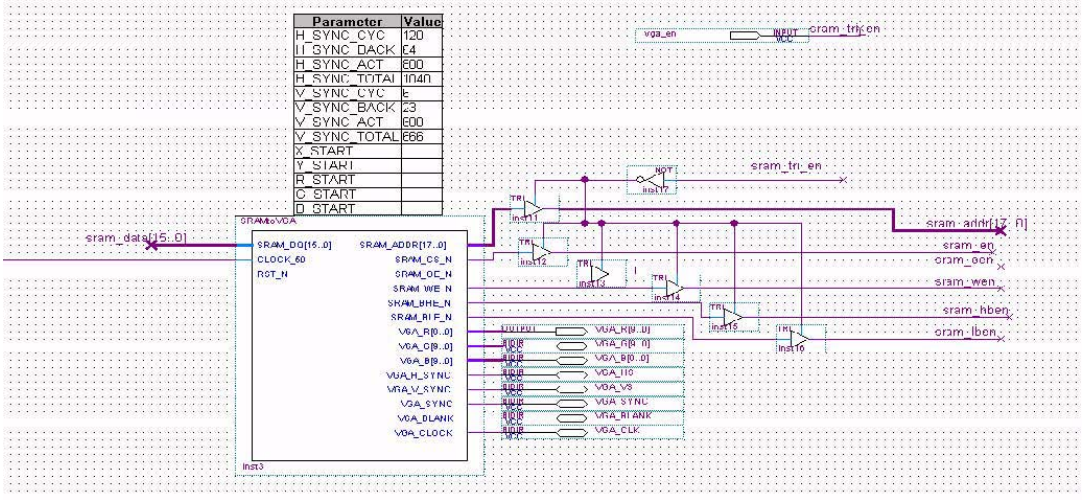
Figure 10. VGA Sequence Standards

VGA mode		Vertical Timing Spec			
Configuration	Resolution (HxV)	a(lines)	b(lines)	c(lines)	d(lines)
VGA(60Hz)	640x480	2	33	480	10
VGA(85Hz)	640x480	3	25	480	1
SVGA(60Hz)	800x600	4	23	600	1
SVGA(75Hz)	800x600	3	21	600	1
SVGA(85Hz)	800x600	3	27	600	1
XGA(60Hz)	1024x768	6	29	768	3
XGA(70Hz)	1024x768	6	29	768	3
XGA(85Hz)	1024x768	3	36	768	1
1280x1024 (60Hz)	1280x1024	3	38	1024	1

VGA mode		Horizontal Timing Spec				
Configuration	Resolution (HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(Mhz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25 (640/c)
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36 (640/c)
SVGA(60Hz)	800x600	3.2	2.2	20	1	40 (800/c)
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49 (800/c)
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56 (800/c)
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65 (1024/c)
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75 (1024/c)
XGA(85Hz)	1024x768	1.0	2.2	10.8	0.5	95 (1024/c)
1280x1024 (60Hz)	1280x1024	1.0	2.3	11.9	0.4	108 (1280/c)

The images we used in our system are 660 x 660 and 600 x 600, so we chose the 800 x 600 mode. Figure 11 shows the module integrated into the system.

Figure 11. VGA Module in the System



Trademark Image Preprocessing Module

Due to illumination differences, shooting distances, and trademark position, the collected images may contain noise and geometrical changes caused by light, size, and angles. Therefore, preprocessing is very important and provides the basis for future authentication. Our preprocessing module removes illumination differences and extracts the standard trademark image. The module has the following functionality:

- *Image balance*—To remove illumination differences, we balance the images by applying the same grayscale standard to them. Because the images we selected are 600 x 600, the data is comparatively large. Therefore, we used the C2H Compiler to optimize the image balancing.

- *Trademark image extraction*—We position and extract the central trademark image. First, we perform a binary process on the balanced image and separate the image from background. Then we extract the image with a geometric operation. The processing is fairly complex due to the large image size, slow operation speed, and high accuracy. Therefore, we used the C2H Compiler to optimize some functions that have large operation volume.

- *Trademark image standardization*—To match the authentication interface, the extracted image is standardized and zoomed to 600 x 600. This part is also optimized with the C2H Compiler.

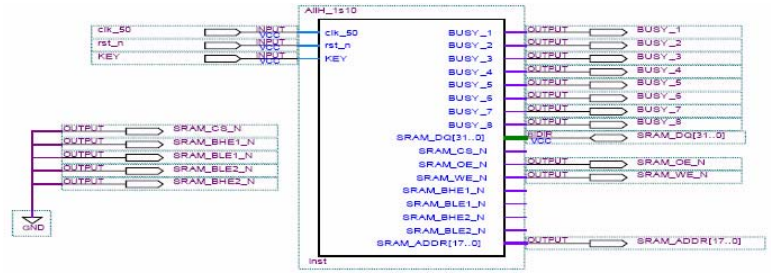
  Figure 12 compares the original image and the standardized one. Some small deviations may exist in the original image that was collected on site.

Figure 12. Original vs. Standardized Image

Watermark Checking Module

We used the lab's watermark patent algorithm in the watermark embedded algorithm and checking algorithm. With a hardware module implementing the whole process, we greatly improved the system speed. During processing, the image to be processed is placed into SRAM (we used 1-Mbyte SRAM). The module automatically interacts with SRAM (read/write) and shows the results after processing. We implemented the module in Verilog HDL. Figure 13 shows the watermark checking module symbol.

Figure 13. Watermark Checking Module Symbol



Applying System-on-a-Programmable Chip (SOPC) Concepts

Altera introduced SOPC technology and its related development platform, the Quartus II software. SOPC is the FPGA version of system-on-chip (SOC). Compared to ASIC SOC, SOPC has many unique features. Our design uses SOPC concepts in the following ways:

- *Modularized system design*—At the beginning of the system design, we partitioned the system into image collection, preprocessing, and processing. The system is divided and simplified, which makes it easier to implement. According to the module interfaces, we can accurately evaluate the design's application scope and future development at the initial design stage. We can perform market exploration and product research and development simultaneously in the practical product design, which shortens the time-to-market and accelerates enterprise development.

- *System integration*—An embedded system shows its features with its size, power consumption, and integrity. Except for the expanded 1-Mbyte SRAM front-end collection module, we could implement all system functions on the development board. It is very difficult to implement such a highly integrated design without lowering the design target or using a different FPGA.

- *Various modes*—We can diversify the implementation. For example, the front-end module uses an IP core, preprocessing is implemented with software, and key steps are fulfilled using the C2H Compiler to transfer operations to hardware. The trademark checking module uses hardware, software, or hardware/software with custom peripherals and instructions. Using SOPC and the excellent design tools enabled us to use these various modes.

- *Final system can be upgraded*—The design can be flexibly configured and updated during the design process.

Design Features

Our digital watermark technology-based trademark is different from traditional trademarks, and has many real applications. Implementing the watermark checking in hardware and producing a portable device effectively promotes watermark technology for anti-counterfeit areas. Currently, watermark checking devices do not exist. Our system has the following features:

- The IP core used for the hardware-based watermark checking algorithm significantly facilitates algorithm protection and volume applications. Additionally, the speed is high compared to the hardware/software-based or software-based watermark checking algorithms.

- The hardware/software-based watermark checking algorithm uses custom instructions and peripherals, improving its operation speed and making the system integrated and modularized. With the existing IP core, it can implement the image processing algorithm in the FPGA.

- The system can perform both offline trademark checking using an independent database and online checking using a central database. The independent operation uses integrated Nios II on-chip functions, including data collection, processing, display, and storage. Additionally, the system can use the Nios II extension interface to add a GPRS module that communicates with a remote server and checks the trademark online.

- The Nios II C2H Compiler can automatically transfer the high-performance C program into the hardware accelerator and integrate it with the FPGA-based Nios II subsystem, which enhances the system's operation speed.

- The FPGA-based system design integrates the processor, peripherals, memory, and I/O interface in a single FPGA, lowering costs, complexity, and power consumption. The Nios II processor provides a series of CPUs, so the user can create a perfect solution that meets their system

requirements. The processor provides reasonable performance, saves development costs, and increases the product's competitiveness.

- Our design implements an SOPC system, a real-time, portable checking device featuring one or multiple Nios II processor(s) as well as a hardware accelerator, custom instruction, and custom configurable peripherals.

Conclusion

We learned a lot using the Nios II processor during the competition. For example, we learned to:

- *Develop a design specification*—We started by writing a design document, including the design idea and implementation procedures, which helped all the team members understand the scope of work. The design document provided general guidance and facilitated coordination during the design process. We also made the code readable and portable, which helped us understand each module's operation and perform testing at each stage.

- *Fully use the design software*—In addition to Nios II-based SOPC design and FPGAs, Altera provides good design software: the Quartus II and SOPC Builder software. The new Nios II C2H Compiler effectively enhances the embedded software's performance, and helps developers improve efficiency and implement a successful design. The tool can automatically transfer high-performance C programs into a hardware accelerator for integration with the Nios II subsystem on the FPGA, which shortens the development cycle from several weeks to several minutes. Compared to other implementations, the Nios II C2H Compiler increases the system's performance by 10 to 45 times, and only uses about 0.7 to 2.0 times more logic resources. With Altera design software, we can produce an excellent system in a short time.

- *Leverage resources, software, and hardware*—There are many implementation methods. We needed to leverage the existing resources, software, and hardware to implement the design. We optimized the complex software algorithm using the C2H Compiler. Altera SOPC design helped us simplify the design, allowing us to focus on the system analysis and embedded design instead of the module design.

- *Cooperate and use teamwork*—Guided by the design document, we could cooperate, communicate, and improve our team spirit, which was the key to success.