Third Prize

Nios II Soft Core-Based Double-Layer Digital Watermark Technology Implementation System

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

Participants: **Lian Jiezhen and Ye Qingfeng**

Instructor: **Liang Xiaowen and Li Yuhu**

Design Introduction

With increasingly powerful functions available from digital imaging software, the reliability and authentication features of digital images are gradually decreasing. Therefore, reliability and authentication are important for a wide range of digital image applications. Digital watermarking provides digital copyright protection, ensures integrity, and assists during digital transmission. The technology guarantees the reliability and authentication of digital images. We can expect to see more usage of digital watermarking in the future.

Today, digital watermarking algorithms are generally realized in software. However, such algorithms have low execution efficiency and suffer from not being robust enough to withstand hacking. Watermarking technology implemented in hardware can overcome these weaknesses. Additionally, hardware-based watermarking offers high reliability, imperviousness to hacking, and high-execution speed. Thus, hardware-based watermarking can be deployed safely in applications that need high reliability and speed. Given today's advances in FPGA technology, we can implement hardware-based watermarking technology quite efficiently. In particular, with its outstanding performance and numerous design advantages, the Altera® Nios® II embedded soft-core processor is the first choice to implement a hardware-based watermarking system.

Application Scope

Our project is designed to monitor devices in public places, aid in forensics, provide traffic-offense monitoring identification of medical images, and important news pictures.

Target Users

Our design targets users with special requirements regarding the credibility of digital images, such as government departments, traffic offense governing agencies, courts, hospitals, and the media.

Why We Chose the Nios II Processor for Our Design

We chose the Nios II processor for the following reasons:

- Being completely customizable, the Altera Nios II processor has high performance, and supports flexible product development designs at low cost.
- The Nios II processor's customizable instruction set can accommodate complicated arithmetic operations and can accelerate algorithm processing, which provides faster execution than implementing these operations in software.
- A configurable design enhances system performance. Implementation of a watermarking algorithm requires a custom instruction set, DCT/IDCT hardware accelerator, and so on. The configurable design approach enabled us to achieve our performance goals in a short time.
- The MicroC/OS-II and real-time operating system (RTOS) in the Nios II integrated development environment (IDE) are very user friendly.
- Based on the Nios II processor, we can enhance system performance by adjusting the Avalon® switch fabric. This technology supports multiple parallel data channels to achieve high throughput in watermarking applications.
- Implementing the processor, peripherals, memory, and I/O interface in a single FPGA can reduce the total system cost.
- Rapid system implementation. We could go from the original concept design to system realization in a short time because of using the Nios II processor. Additionally, we could easily upgrade the hardware and software on site. The flexibility allows us to design products in line with the latest specifications and equipped with new features.
- Unmatched levels of flexibility. The Nios II processor features complete customizability and reconfiguration. For example, it supports three processor cores, peripherals, the Avalon switch fabric, custom instructions, and hardware acceleration. All of these functions can be implemented using commonly available Altera FPGAs.
- Powerful combination of Altera intellectual property (IP) functions and FPGAs. Using IP optimized for the FPGA architecture, we can redesign standard functions easily, rapidly customize hardware peripherals, focus on design partitioning, and improve our design knowledge.
- Integrated development kits. The Quartus® II software, SOPC Builder, ModelSim®-Altera software, and SignalTap® II embedded logic analyzer provide a complete set of test and debug tools for hardware design. With the Nios II IDE, it is possible to simplify software design and all software development tasks, such as program editing and debugging.

Function Description

While designing the system for this project, we gave consideration to whether the implementation was feasible and practical. The two-layer digital watermarking algorithm features IPR protection, authentication, creation identification, and other functions.

Function 1

This design implements both IPR protection and creation functions, and delivers the following advantages:

- Adds watermarking to the image without affecting the nature of the original image.
- Provides unique watermarking that is difficult to counterfeit.
- Allows you to generate a large number of watermarks.
- Supports authentication and watermarking identification without the original image.
- The watermarking can resist the JPEG compression algorithm with a high compression ratio.
- The watermarking offers strong resistance to ordinary geometrical transforms, such as low-pass filtering, sharpening, and scaling.
- Several watermark effects can be applied simultaneously, thereby protecting the intellectual property rights (IPRs) of the owner and purchaser.
- The algorithm is secure from strong attacks, such as collusion attacks and extra watermarking, to authenticate images effectively. Where images have been modified, we can show the part that was modified, which helps to distinguish common operations, such as JPEG compression, from deliberate image modification.

The image processing algorithm should be the first design objective when creating digital image watermarking technology in hardware. We designed a two-layer digital watermarking algorithm for this design contest. In our algorithm, we selected the secure Gaussian random sequence as the watermarking signal on the first layer. Additionally, to make the watermarking more secure, the watermark data is generated by a unique key that generates a unique watermark. The second layer watermark is based on a look-up table algorithm for the authentication of the image's integrity and reliability. Finally, we simulated the algorithm with the MATLAB software to verify whether the algorithm correctly realized the functions required for two-layer watermarking.

Function 2

We used the Nios II development board to implement the watermarking algorithm. A PC performs real-time tasks for proper operation of the algorithm. In practical applications, the detection algorithm cannot be implemented on a development board. Instead, it is more likely to be implemented on a PC. Therefore, we needed to design the host system and a user-friendly graphical user interface that features all control and detection functions. We designed the GUI to be very user friendly and to enable users to intuitively see and operate functions of the two-layer digital watermarking system.

To implement the watermarking algorithm, we needed to create the communication system between the Nios II development board and the host system. Because the USB interface did not work well for us in practice, we used a JTAG UART serial port instead. Communication was based on the host-based file system module in Nios II version 5.0 and operational instructions for files from the C-language library.

Because we needed to simulate the algorithm in early stages in the MATLAB environment, and because the detection algorithm involves complex mathematical computations, we decided to use the MATLAB software to design the host control and detection system as well as the GUI.

In our design, you can display the original image, modified watermarking image, added watermarking, and detected watermarking simultaneously, to facilitate easy comparison on the same display. At the same time, using multiple buttons, users can easily perform operations such as AuthWatDect, OriImgDect, Start, Add, and so on. We designed these buttons to have corresponding functions on the host system. For example, pressing the Add button to add images first invokes the corresponding function on the host system to generate a 128-bit key. Pressing the Start button notifies the Nios II system to start the watermark computation.

The host system mainly detects authentic watermarking and its integrity on a PC. Pressing the AuthWatDect button starts detection of authentic watermarking. The system obtains threshold values, after calculating the relevant peaks, to judge whether a watermark has been added and the type of watermark. Pressing the OriImgDect button starts the integrity detection based on data values from a look-up table. If the image was not modified, the detection image in the look-up table is white. If the image has been modified, the modified part in the detection image is black.

Function 3

A core function of the system is to create the two-layer digital watermarking algorithm with the Nios II soft core processor. The system comprises two functions, which are implemented as follows:

- The digital cameras supporting external monitors are connected with the right interface to allow high-speed data transfer. This process ensures a reliable, practical watermarking implementation. We used the JTAG UART for data downloading and appropriate switches to ensure communication with mainstream digital cameras and hosts at high speed.

- Extend the functions of a digital watermarking system with peripheral components. During the insertion of a watermark, the text “watermarking…” is displayed on the LCD. When a watermark is successfully added, the text “it is great” appears, making the system more humanistic.

Performance Parameters

This section describes the system’s performance parameters.

Fixed-Point Arithmetic

Because of the real-time requirement of the application, we relied on Nios II fixed-point processing, and implemented the arithmetic to handle fixed-point processing to determine the suitable value arrangement, ensuring computing efficiency and accuracy. In the design’s arithmetic calculations, the RGB signals in the original image pixel are set to 8-bit data. To cater to negative value components that might be produced during the YCbCr conversion to color space, we set the Y, Cb, and Cr as 9-bit (-256 to +255) data. Similarly, the two-dimensional DCT conversion results fall in the range of -2048 to 2047, and the quantified value by the quantizer falls within the range of -128 to 127. The LUT design is set to 256 digits.

Optimization of Division

Because the Nios II processor has no special arithmetic instructions, division operations are performed by successive subtraction, which consumes a lot of computing resources. Although we encounter fewer division operations in arithmetic computation, e.g., in normalized operation of partial weight factors, the division operation still has a certain influence on performance. Therefore, we tried to compute these

operations in an optimized way to approximate values through binary shifting and module arithmetic. Because, shift arithmetic can be performed with other operations, we can improve the computing speed while ensuring accuracy. This method also helps conserve system resources.

Using Parallel Processing with the FPGA & Nios II Processor

A lot of multiply-accumulate (MAC) operations are applied during discrete cosine transform (DCT) conversion and during the addition of Gauss serial watermarking with copyright protection. Using the Nios II processor, we can adjust system performance using the Avalon switch fabric, which supports several parallel data channels and helps to implement image watermark processing.

Memory Management

The Nios II processor offers unprecedented flexibility through its customizable and reconfigurable architecture. It features a powerful combination of optimized IP for the FPGA system architecture, which allows us to add SDRAM, SRAM, and flash using the SOPC Builder tool. We use these tools to implement a large scale, real time system with abundant memory resources.

Communication Between the Development Board & Host PC

The JTAG UART interface must only be used for communication because the UP3 board needs to use a special USB-Blaster™ data cable for downloading. Because the UART interface has a slow communication speed, we adopted several measures to avoid a time-penalty for using the Nios II processor for image processing, and to utilize CPU resources more efficiently. For example, the image that needs to be processed is read in eight lines of data each time and sent back to the host PC after processing. Then, we read the next eight lines of data, and repeat the previous steps. The host PC and Nios II processor are linked via the JTAG UART interface, which greatly influences the interactive speed between them, and limits the whole image processing speed.

A new function provided in the Nios II processor version 5.0 also facilitates file output. Selecting the software component in the **syslib** engineering attribute, and then choosing **Add this software component** in the Altera host-based file system, you can read/write host PC files in debug mode.

To implement the watermarking algorithm arithmetic, we integrated the first and second layer watermarking. Figure 1 shows the watermark insertion arithmetic block diagram. Figure 2 explains the watermark extraction arithmetic block diagram. Our performance tests showed that combining two-layer watermarking has no influence on an individual layer's performance. However, this combination allows you to determine the sequence of multiple watermarking without any interference, in case they were added to an image.

Figure 1. Arithmetic Insertion of Double-Layer Digital Watermarking

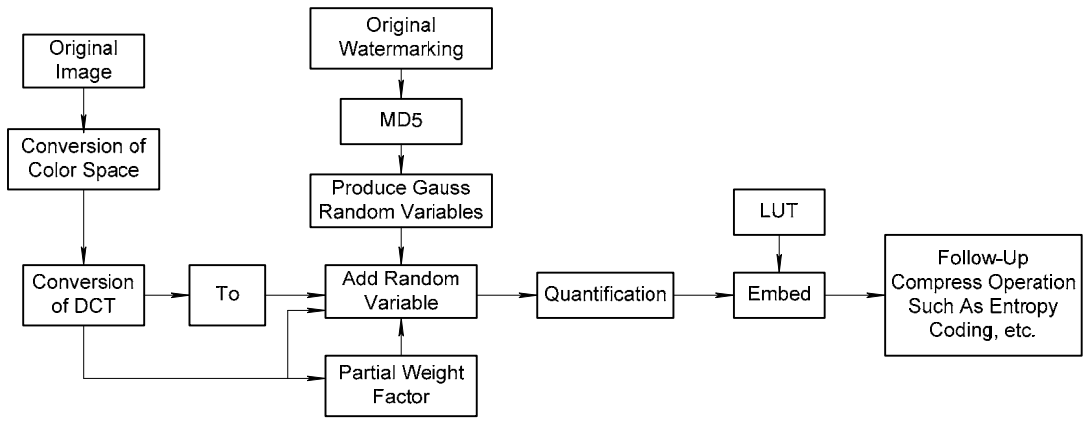
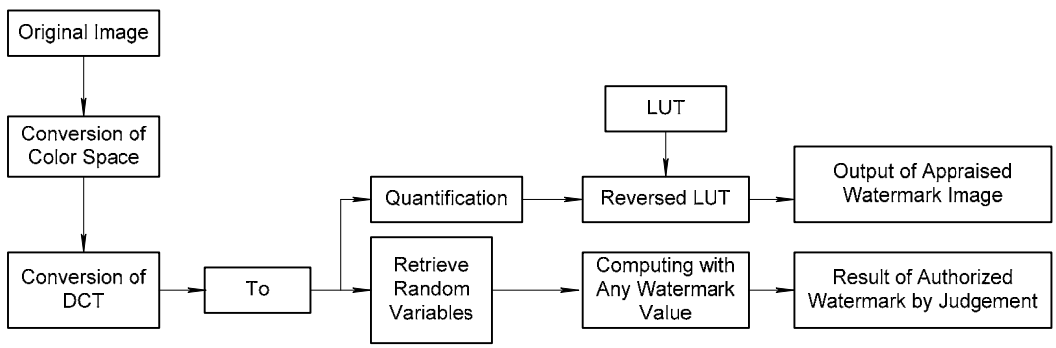
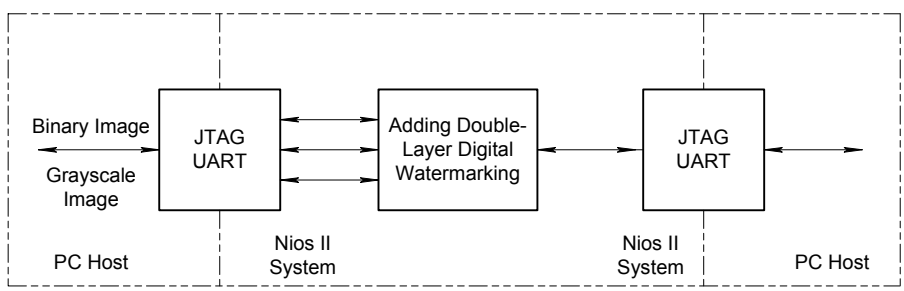


Figure 2. Arithmetic Extraction of Double-Layer Digital Watermarking



In Figure 3, the host system sends images that need to be watermarked to the Nios II processor via a USB-Blaster communication interface. After applying double-layer digital watermarking, images are sent back to the host. Normal operations are handled by the host PC in real time. Arithmetic testing cannot be performed in the Nios II system, but can be done in the host PC. Therefore, we designed an easy-to-use host system that features all control and testing functions.

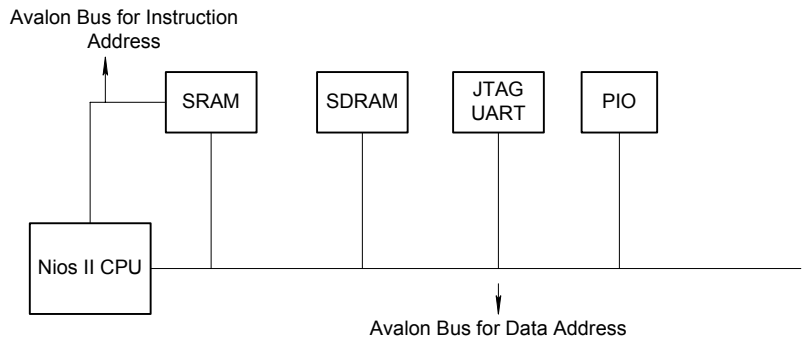
Figure 3. Digital Watermarking Flow Chart



In Figure 4, external SDRAM has more memory space in the Nios II hardware system, and the SRAM can be used for faster storage and access. Therefore, we employed multi-master bus technology to

install SRAM to handle faster data requiring rapid storage, process some commands, and variables of operating system. On the same bus, we can use the peripheral 8-Mbit SDRAM as data cache.

Figure 4. Nios II Hardware System



Design Methodology

Our design methodology involved the following steps:

1. Design the arithmetic algorithm and improve simulation to enable the Nios II processor to execute parallel processing fully.

2. Propose a system hardware block diagram and simplify the design flow by deploying powerful IP cores for easy design configuration.

    a) This task involved communication between modules that have different interfaces. Using Altera's system-on-a-programmable-chip (SOPC) design technology, it was easy to create the control interface for these modules, avoid extra peripheral circuits, and reduce overall system design costs.

    b) We could add the IP cores of SRAM, SDRAM, and flash memories directly. However, this method can lead to bus collusion. Therefore, we designed a bus-multiplexing module in HDL using the Quartus II software. Based on this design, three signals ensured a non-collision operation, which highlights the powerful flexibility and control made possible through the user-defined configuration of the Nios II processor. It also illustrates the tremendous advantage of the Nios II soft core, and highlights the fact that the combination of the Nios II processor and an FPGA meets the future market demands perfectly.

    c) We chose the CPU based on available hardware resources. Considering the actual situation of the EP1C6, we used it as the standard CPU for the design, which was sufficient to meet our needs. This selection illustrates the advantage of using an embedded CPU like the Nios II processor. When you choose a CPU in a hard-core development board like the ARM MCU, it cannot be changed. Therefore, this design approach does not provide design configuration flexibility, leads to low design re-use, and has limited application usage when compared with the Nios II processor.

    d) Because we could not use the USB interface for high-speed data transmission, we chose JTAG UART serial communication IP core to communicate with the host PC. We selected the LCD IP core to display prompt information on the LCD. Using the IP cores can save significant time and cost when implementing peripheral circuits and interface protocols. Additionally, components can be easily added or removed, and component parameters can

be changed. By using the Nios II processor, we can fully accomplish the configuration of an embedded system from CPU to peripheral circuits, as planned.

e) Because the current SOPC Builder library does not have a built-in way to target the flash parameters of the UP3 board, we needed to use a separate Flash Programmer User Guide, which was provided in the installation directory of the Nios II processor, to program the flash memory. After debugging and compilation, we successfully programmed the flash memory to perform the whole FPGA data configuration.

3. Employ the Nios II IDE for software development.

   a) On a stand-alone PC, we used a C program to compile and debug software.

   b) We built new modules in the IDE, finished control of the LCD display by programming a module available in the LCD template. We also needed to debug the main program in IDE mode. First, we needed to burn the hardware into the FPGA, then we programmed the peripherals. During debugging, we thought that utilization of the Nios II software was not sufficient. "Run as Hardware" needed to be recompiled before being written in hardware language for each iteration, so the debugging process was very slow.

   c) For the read/write of the host-PC file system, we used the Host Based File System in the Nios II processor version 5.0 in the Debug As Hardware mode. When we tried the Nios software version 1.1, it failed. We think it would be useful if the Host Based File System could be performed in a "Run As" mode condition.

   d) The program can be run under SDRAM and SRAM, separately. To conserve data in case of a power failure, we can burn the program into flash memory by using a flash programmer. Then, we reset the address of the CPU to flash memory, and we can automatically configure the FPGA after power up. Then, the program and files can be read into SRAM and SDRAM easily.

4. We designed a host system with test functions and a user friendly GUI, which intuitively shows the effect after Nios II processing.

In this design, the algorithm arithmetic of adding watermarking is performed on the Nios II development board. Normally we do this addition using a host PC in real time. In a practical application, the test arithmetic cannot be performed on the development board, but we can do it on a PC. Therefore, we needed to design a host system with total control and testing functions, and a user-friendly graphical user interface (GUI). Using this GUI, users can intuitively operate and implement the double-layer digital watermarking system.

Because the pre-simulation of arithmetic calculations was performed in the MATLAB software, and the testing arithmetic involves many complicated mathematical operations, we designed the host control, testing, and GUI using the MATLAB software.

The host system is mainly designed for testing watermark copyrighting and integrity. If you need to test watermark copyright, click **AuthWatDect**. Following this command, the threshold is obtained after computation of relevant peaks and judging whether watermarking has been added; if so, the program determines which watermarking was added. If you need to test the watermark integrity, click **OriImgDect** to test the LUT; if the image is not modified, the testing diagram of the LUT is white, otherwise, black spots appear on the modified parts.

Design Features

To perform two-layer digital watermarking, we designed the system using a local restructuring concept to come up with a strong anti-interference capability for the watermarking function. This process also made it possible to inspect the watermark effectively without the original document. Additionally, the embedded basis of two-layer watermarking algorithm allows us to distinguish, reliably, if images have been modified, and when modified, those areas are marked. We can deploy the watermarking algorithm in a digital image application and have it processed in the frequency domain. The algorithm uses a visual masking model for extra robustness. The two-layer watermarking algorithm also features the IP protection and production authentication functions.

The Nios II processor aids in algorithm implementation in the following aspects:

- To evaluate the execution of algorithm, we need to look at the DCT transformation, random number generation, and local weighting factor calculations in the visual model. Thanks to the powerful processing capability of the Nios II CPU, we can add user-defined instructions and a hardware accelerator (such as a DCT/IDCT hardware accelerator). Additionally, the Nios II processor's broadband converting architecture supports multiple parallel data channels to speed up overall processing.

- A programmable system is necessary to generate the algorithm according to a hardware-based digital watermarking system. Therefore, we naturally thought of using an FPGA in our design. During the design stage, we faced the problems of development costs and technical difficulties. We were able to overcome these problems because the Nios II embedded soft core delivers better price/performance and lowers technical hurdles in development. Using the Nios processor we were able to implement the processor, peripherals, memory, and I/O interface on a single FPGA, which helped us to reduce the total system cost. Additionally, Altera's comprehensive developing tools for the Nios II processor and GUI provided many optimized IP cores, which reduced the software development cost. Therefore, we were able to pay more attention to the design details and completed the digital watermarking system.

- The digital watermarking technology we chose to design is a hot IT technology at present, and will have a wide application in the future. Altera's SOPC design approach allowed us to keep the technology lead in digital watermarking, but also can ensure an early entry into the market with hardware-based products. In this way, we can reap benefits quickly.

  The Nios II soft core processor offers flexibility, which helps make choices between multiple system setup combinations to achieve optimum performance, features, and cost goals. Using the Nios II processor in a design allows us to put our products into market faster and prolong the product lifecycle. According to the varying requirements of our users, the Nios II processor can be implemented using on-site hardware and we can add software upgrades easily. In this way, we can make the product comply with the latest specifications, have the latest features, avoid processor obsolescence, and hold off competition.

- The two-layer digital watermarking technology software should be created on different Nios II hardware systems to provide a strong migration capability. The design contest required us to create the design on the UP3 board. However, we believe the software part of the two-layer watermarking algorithm can be migrated easily to other FPGA development boards.

  Altera has embedded MicroC/OS-II in the Nios IDE, which let us set up RTOS for Nios II processor applications quickly.

  The MicroC/OS-II RTOS provides portable, modifiable, reducible, real-time multi-task functions. Therefore, a MicroC/OS-II based program can be migrated to other Nios II hardware system easily without worrying about the low-level hardware. Additionally, the MicroC/OS-II program supports all HAL services, and you can use it to invoke the API function in HAL. The RTOS can run on different processors to provide the same API interface for users.

  Because our algorithm is quite complicated, the multi-tasking operation of MicroC/OS-II RTOS fully exerts CPU utilization and modularizes the application. When designing complicated applications, designers often adopt a hierarchical model to make it easy to design and maintain the software modules. Additionally, the MicroC/OS-II RTOS allows you to split the application into several tasks, further simplifying the design of the system.

- We designed the Nios II flow diagram after the overall design and validation of the algorithm. The key problem was how to implement the design with the Nios II processor better and faster. The technical support and detailed help files of the Nios II processor enabled us to have a quick start and rapidly master the programmable design skills. We believe that our two-layer watermarking algorithm is an optimized implementation.

Conclusion

This contest helped us better understand the Nios II processor. We think the Nios II processor will be widely used in the future because of its new design methods and modes, especially the SOPC concept, which is described as follows:

- The Nios II processor complies with the developing trend of industrial technology—*software-like hardware* design. Using the Nios II processor can reduce the developer and material costs, thereby improving competitiveness. Additionally, the software-like hardware design makes simulation to hardware easy, reducing hardware design errors.

- The Nios II processor enhances system robustness, which is an advantage of a single chip solution. The Nios II soft core and its development platform help the developer to build most of the modules flexibly. Corresponding drivers can be developed for most of the peripherals used, minimizing design errors.

- The Nios II processor helps users to protect their intellectual property. Primarily, we can prevent reverse engineering when we use the Nios II processor in our designs.

- The Nios II processor is significant, because it exploits a new, development space for FPGAs implementing SOPC-based designs. The UP3 board we used uses a Cyclone™ FPGA, and provides excellent support with a great number of logic units. Additionally, the Cyclone devices's low cost and its configurability will push DSP users to use the combination of FPGA and the Nios II processor for system designs to achieve better performance.

The success of this Nios II design development was possible due to:

- The Nios II processor's excellent hardware scheme, which made it possible to complete the design with merely a few external hardware parts and allowed us to utilize the development board's resources fully.

- Judicious usage of the Nios II processor's flexibility helped us to implement the synchronization of various hardware modules and design software, which sped up the design significantly.

The experience we had from the design is described as follows:

- When designing with conventional (hard core) processors, the designer needs to pre-plan how to generate address decode and control signals. Once errors occur during validation, it is very difficult and complicated to make the necessary modifications. The Nios II processor eases this situation. You just need to connect the corresponding function module with the FPGA, paying attention to the interface and timing. Additionally, SOPC Builder provides a lot of parameterized IP and hides a great deal of initialization details from the designer, such as the UART baud rate, flash timing, and address line, making it easy to develop the system. The tool also provides complete C language header files and hides hardware details, which simplifies software development.

- Abundant peripherals and easy integration. SOPC Builder helps designers design with SOC and IP. Besides providing many IP cores based on Avalon bus, the tool also supports an open IP integration environment. The users can easily integrate their IP with SOPC Builder, which protects their IP and promotes design reuse.

- The Nios II soft core processor still needs further improvement, however. It is affected by system design complexity, the FPGA, and some other factors when the CPU wants to run with a stable frequency. Sometimes the invoked phase-locked loop (PLL) module is not stable enough in the actual debugging process. The hardware prompt "leaving processor paused" occurs occasionally, which may be due to variance of the actual frequency from the PLL and that of the marked frequency. In this case, you could power off to restart the board, or reprogram the hardware a few times.

- The tool version and license need to be handled carefully when using the Quartus II software and Nios II IDE. You need to upgrade if there is a new version of the tool. There is no problem with hardware settings and hardware during earlier debugging efforts. However, an error always occurs in the SOPC Builder tool when reading and writing host system files. We wanted to realize it by means of a JTAG UART. We used version 1.1 of the tool previously, which does not contain the software component of host based file system. We often failed at the very beginning when we wanted to operate the host file with the standard C library function, using the example of character device operation using the string character. Then we upgraded to Nios II version 5.0, which adds the "filling the file on the pc" module. Its drawback is that it works only in debug mode. We hope that Altera can address this problem in future versions. In addition, it is better to upgrade the Quartus II tool to the latest version, 5.0, to avoid licensing problems.

- When Altera introduced the Nios II soft core processor, abundant documentation was provided on the Altera web site, ranging from hardware operation and software guides to a many software routines. Altera also made timely updates. Although there is a big difference between the Nios II and Nios processors, we solved the general problems by referring to the detailed technical documents. However, in our opinion, the documentation is not completely reliable sometimes. For instance, we could have failed if we had followed the guide when we made a flash programmer with target board. Finally, we implemented it by adding asmi and some other modules. If more Nios II guidebooks are available in the Chinese language, using the Nios II processor would be much easier.