First Prize

MRI Spinal Segmentation Based on the Nios II Processor

Institution: **Information Science Institute, College of Computer and Information Technology, Beijing Jiaotong University**

Participants: **Weiming Li, Ruiqiong Shi, Bo Li**

Instructor: **Xiaoming Ding**

Design Introduction

As fast-developing medical imaging technology promotes modern medicine, computed tomography (CT), magnetic resonance imaging (MRI), and positron emission tomography (PET) are becoming widely used in clinical diagnosis and analysis. These systems have developed from tools for non-invasive examination and anatomical structure visualization to operation planning and simulation, operation navigation, and radiotherapeutic planning and focus change tracking, as well as segmenting the anatomical structure from medical images and forming shapes.

Magnetic resonance (MR) spinal image segmentation research plays a critical role in medical imaging-related computer-aided recognition and neuropathology clinical research. If the acantha cannot be segmented and recognized accurately and clearly, computer technology will provide little to medical clinical research. Additionally, it is not enough to complete this task by hand. Traditional acantha fracture assessment involves manually marking the acantha with six points (on the four corners and the centers of the top and bottom edge), and then measuring the height of the acantha on the front, middle, and back, which is a time-consuming task. According to some documents, using a mouse to locate a single patient's vertebra takes more than 15 minutes, and locating the entire acantha takes an extremely long time. In these circumstances, there is an urgent need of a clinical method that automatically segments the MR spinal image. Locating vertebra and intervertebral disks automatically via computer is important for diagnosis during clinical treatment.

Our design implements a feasible, robust algorithm running on the Nios® II processor platform. The design locates intervertebral disks automatically and quantitatively marks acantha nuclear magnetic resonance (NMR) sagittal views using an MRI spine segmentation algorithm in our lab. Fully using the

Altera® FPGA and Nios II resources makes the system small and portable. This algorithm improves the automatic assessment of vertebra fractures derived from osteoporosis, and enables quantitative analysis of the intervertebral disk to facilitate image alignment with other imaging (e.g., CT scans) and image-guided vertebra operation.

Application Scope and Targeted Users

This design is applicable to medical institutions possessing NMR instruments. Because the Nios II design is easy to use, it is also applicable to the average person, who can learn the patient's situation and remind him or her about acantha diseases in daily life. Additionally, the design's network transmission function facilitates telemedicine.

Nios II Design Advantages

The Nios II processor provides several advantages for our design, including:

- *Innovative system-on-a-programmable-chip (SOPC) design concept*—We can adjust the Nios II soft-core system's performance according to the application to satisfy specific user demands. Compared to a fixed processor, the Nios II soft core system achieves higher performance at lower clock rates. The abundant intellectual property (IP) core library helps users create designs and effectively improve the system's computation capability. Moreover, its user logic functions and custom instructions highlight Nios II technologies, allowing optimized, accelerated computation, increasing processing speeds, and facilitating algorithm commercialization.

- *Development environment*—The Nios II Integrated Development Environment (IDE) integrates the µC/OS-II real-time operating system (RTOS), which has been ported for the Nios II processor. We can use the operating system (OS) directly for functional design and system expansion.

- *DSP Builder*—DSP Builder provides a variety of functional modules and IP cores. With DSP Builder, we can perform algorithm-level system development in the Simulink software. Then, we can design the algorithm as a custom user instruction, and integrate it into the Nios II system using SOPC Builder and the Quartus® II software. We access the DSP algorithm by activating the custom instruction via software.

- *C-to-Hardware Acceleration (C2H) Compiler support*—The Nios II C2H Compiler can automatically convert a C language program that requires high performance into a hardware accelerator, and integrate it into the FPGA-based Nios II subsystem. In this way, the program shares the Nios II processor data computation and memory access and helps the processor perform other tasks better. Because the Avalon® interconnection architecture has no limit on the number of hosts and slaves in a system, the Nios II C2H Compiler can generate multiple memory self-governing hardware accelerators according to the target code conversion requirement. This process helps embedded system developers improve efficiency and implement successful designs.

Function Description

MRI spinal segmentation research is an integral step to providing qualitative and quantitative analysis of body tissue with MRI devices. The introduction of computer-aided MRI spinal segmentation technology will make clinical diagnosis more accurate and timely, lower medical expenses, and alleviate doctors' pressure. Therefore, this technology has a bright future.

The design's hardware module consists of the Development and Education (DE1) development platform, an MRI LCD panel, and the network interface panel for the MR device and PC. The system's functional modules include the MRI image preprocessing module, spinal cord extraction module, acantha detection segmentation module, LCD image display and human-machine interaction module, MRI image data access module, and the network transmission module.

- *MRI image preprocessing, spinal cord extraction, and acantha detection segmentation modules*—These modules are part of the design's algorithm and core functionality. Because of the Nios II

processor's computational capabilities, we implemented most algorithms using C programs. The C2H Compiler accelerates the most time-consuming part of the algorithm to shorten the development cycle.

- *LCD image display and human-machine interaction module*—We implemented the image display and human-machine interaction using an LCD display and mouse, which are connected to the Nios II processor as an Avalon slave using a compiled IP core. In the design, we used µC/OS-II in the IDE to deploy the system's task management and algorithms. We adopted the µC/GUI corresponding to the TCB8000C LCD controller chip to enable the system's human-machine interaction.

- *MRI image data accessing module*—Generally the images obtained directly from the imaging device comply with the standard medical image format, i.e., dicom format. Images in this format are not commonly used by ordinary users, and must be read using special software or medical instruments. To allow the images to be read conveniently in most circumstances, we convert them from dicom to bitmap (bmp) before the processing. The images downloaded from the network to the hardware platform's storage devices are in bmp format, and are uploaded to the PC after processing. The images require a large storage space. Therefore, we use a secure digital (SD) card to store the data and migrate the µC/FS file system corresponding to the card to enhance the system's expandability and data management capability.

- *Network transmission module*—We implemented data interaction using an Ethernet interface. The Ethernet interface allows the system to obtain images from the MR device. We developed the network interface panel using the extensive DE1 interface and used the DM9000A chip as the core. Using Ethernet makes the system more scalable.

With the Nios II processor features and SOPC design concepts, our design became a process of optimizing an algorithm, improving the design, and accelerating system operation. We implemented the design in two steps:

- Established the Nios II system to run the algorithm in software and implement the fundamental system functions.

- Accelerated the algorithm operation using a custom instruction and peripherals.

Performance Parameters

Spinal NMR imaging plays a critical role in diagnosing spinal diseases. For example, it is more effective than other imaging methods in describing degenerating intervertebral disks and can assess the curative effect of acantha operation. Analyzing and processing spinal NMR images will make the diagnosis more accurate, and save time and cost. Therefore, for this system, the most important thing is to segment and recognize the acantha accurately and clearly from the image. Additionally, we need to accelerate the calculation speed by optimizing code, employing C2H tools, and creating custom user peripherals.

Figure 1 shows the system's resource usage after compilation.

Figure 1. Resources Used

Flow Status	Successful - Sat Oct 06 15:37:37 2007
Quartus II Version	6.1 Build 201 11/27/2006 SJ Full Version
Revision Name	DE1_SD_Card_Audio
Top-level Entity Name	DE1_SD_Card_Audio
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	No
Total logic elements	8,593 / 18,752 (46 %)
Total combinational functions	7,603 / 18,752 (41 %)
Dedicated logic registers	4,769 / 18,752 (25 %)
Total registers	4886
Total pins	273 / 315 (87 %)
Total virtual pins	0
Total memory bits	79,360 / 239,616 (33 %)
Embedded Multiplier 9-bit elements	28 / 52 (54 %)
Total PLLs	2 / 4 (50 %)

After code optimization, the time required for all algorithm steps to operate is:

- Image preprocessing: 4,128.77 ms
- Spinal cord extraction: 24,208.14 ms
- Disk detection: 6.03 ms
- Image storage on the SD card: 15,881.59 ms
- Image reading from the SD card: 5,730.02 ms

The algorithm segmentation accuracy can be seen in the final processing image that is generated after the system operation. It proves that the system can mark the acantha by the number on the platform precisely. The medical instrument performance can be verified through long-term clinical experiments, during which we can improve our algorithm.

Figure 2 shows the processing effect of a design. Figure 3 shows the image that the system transmits to the computer via the network.

Figure 2. Design Processing

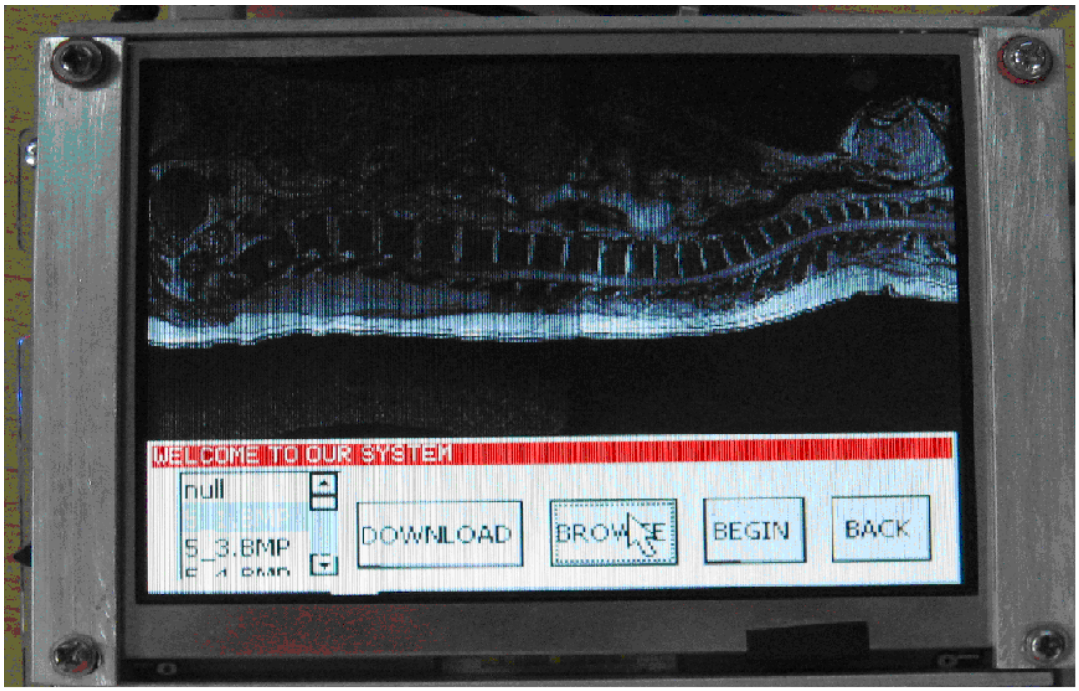
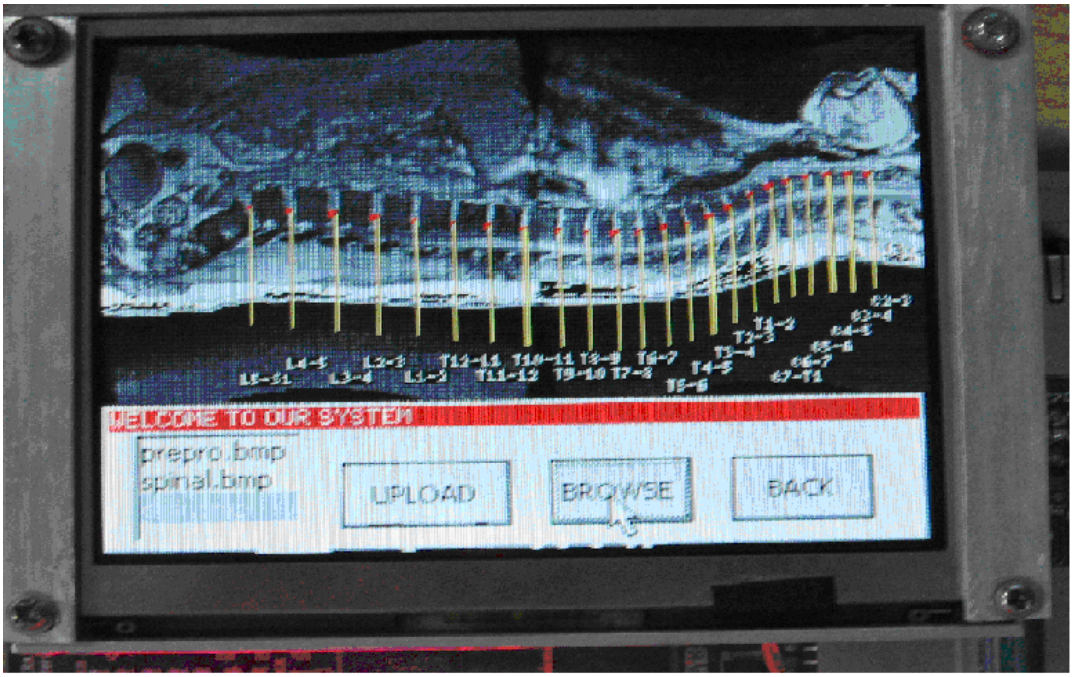


Figure 3. Image Transmitted to Computer



Design Architecture

Figure 4 shows the hardware block diagram of the MRI spinal segmentation system.

Figure 4. MRI Spinal Segmentation System Block Diagram

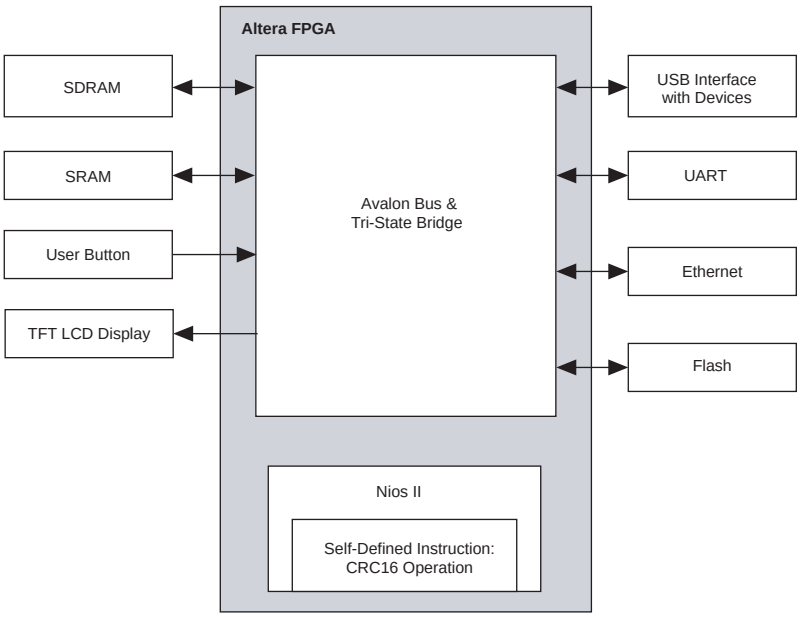
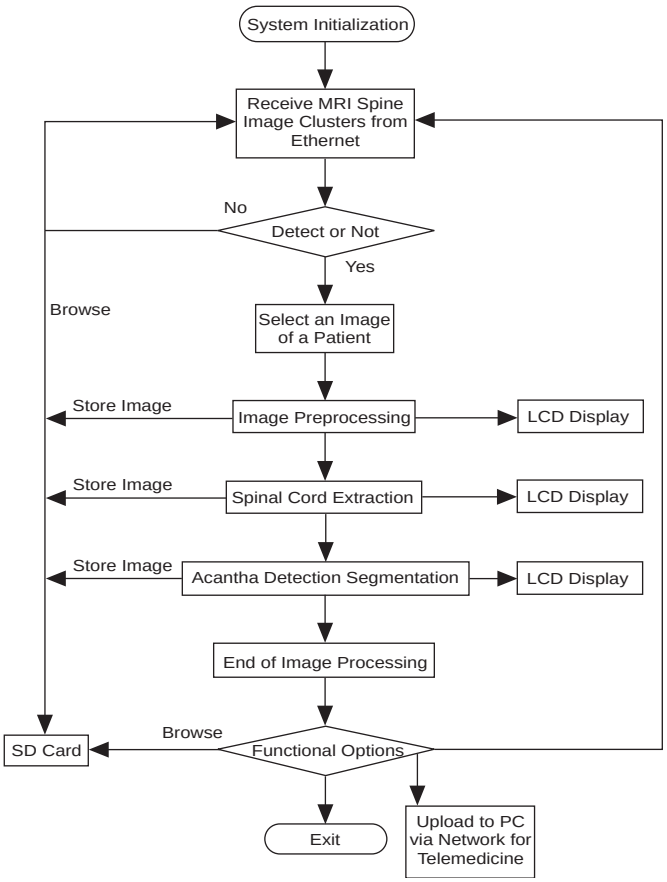


Figure 5 shows the software flow chart when the system initialization receives MRI spine image clusters from the Ethernet.

Figure 5. Software Flow Chart



Design Methodology

This section describes the system hardware and implementation method.

System Hardware

Figure 6 shows the design in SOPC Builder.

Figure 6. SOPC Builder

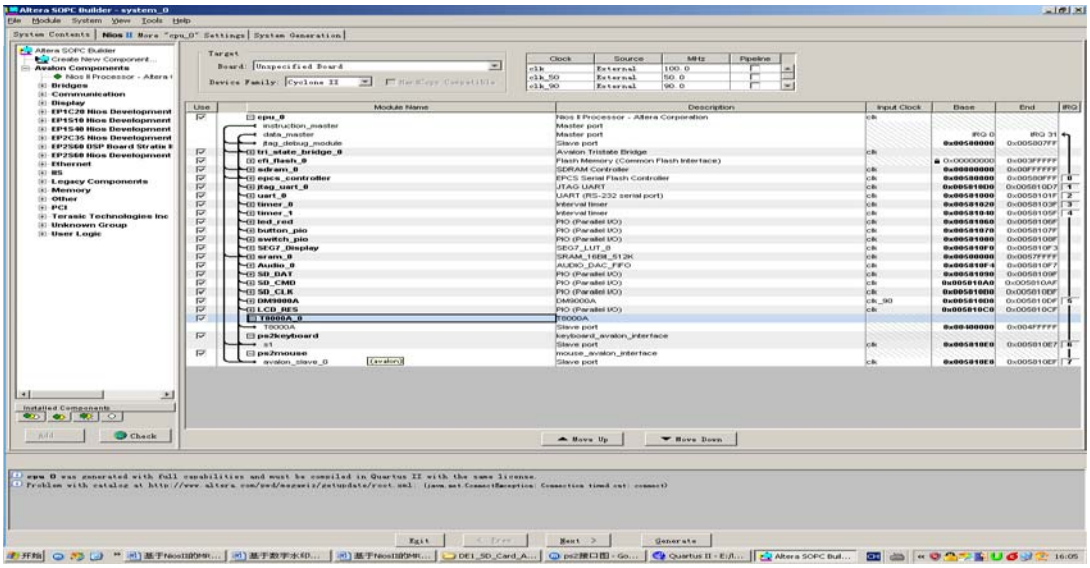
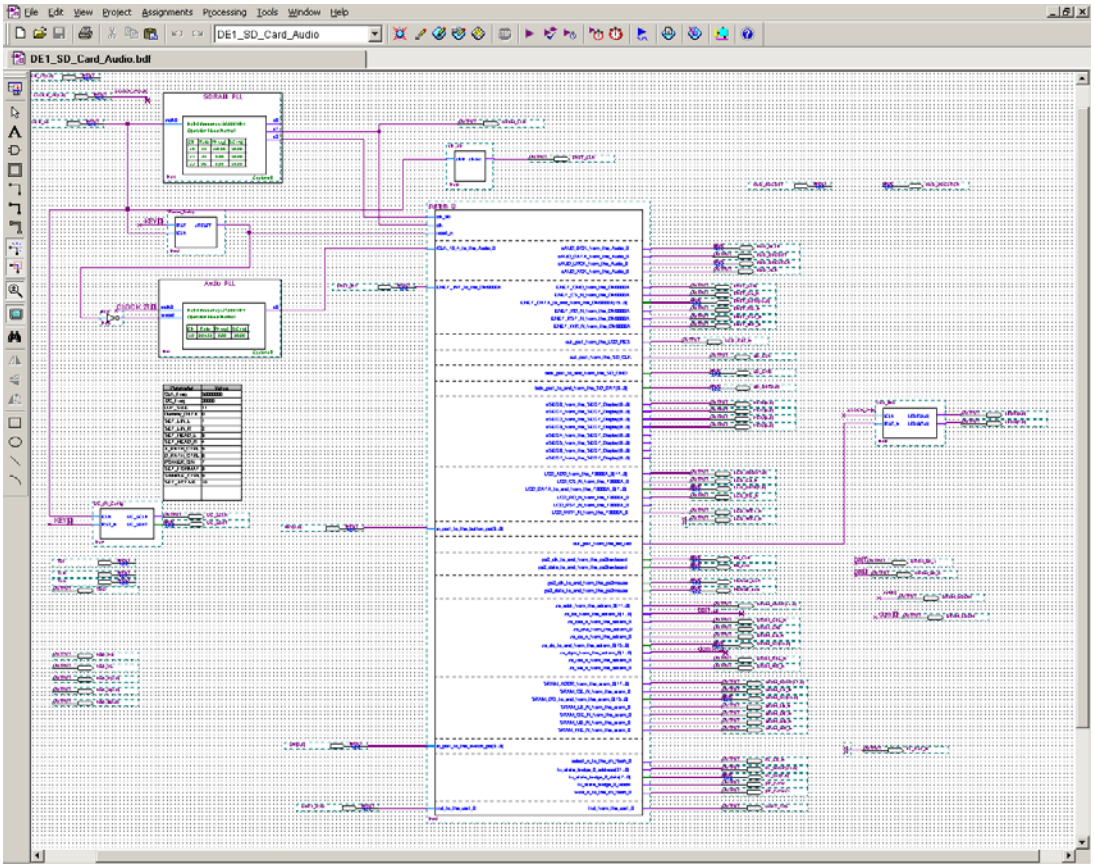


Figure 7 shows the Quartus II schematic diagram.

Figure 7. Quartus II Schematic Diagram



Implementation Method

This section describes the implementation for the various design modules.

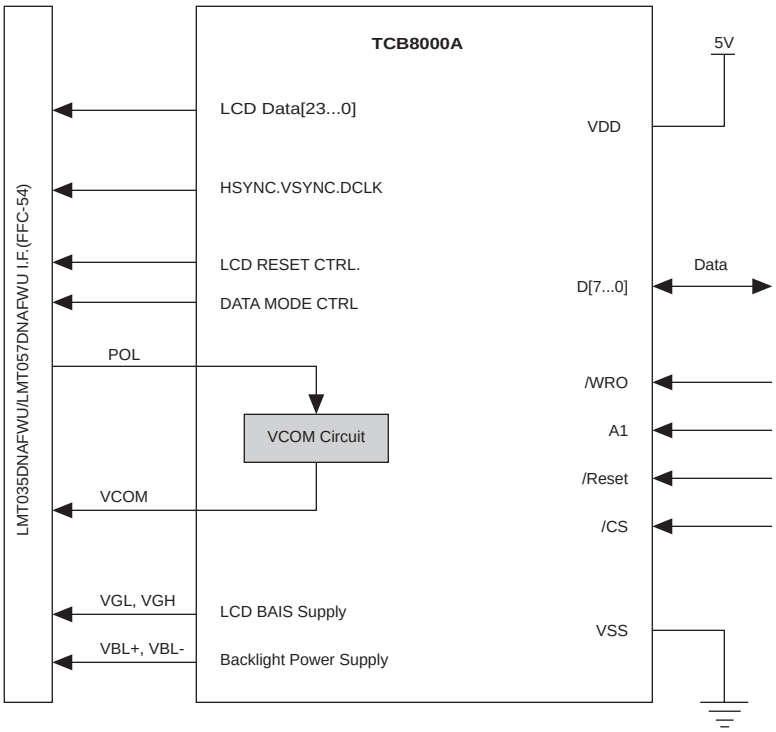
LCD Image Display and Human-Machine Interaction Module

The following sections describe the LCD and mouse hardware implementation as well as migration using µC/GUI.

LCD and Mouse Hardware

We used the TCB8000C device as the LCD controller chip to control a 5.7-inch TFT65000 LCD display. Figure 8 shows the controller and microcontroller (MCU) interfaces. The controller is connected to the Avalon bus as a slave, and the Nios II processor directly accesses the LCD controller. Therefore, the driver can be compiled easily and driving the LCD after GUI migration is simplified.

Figure 8. LCD Controller



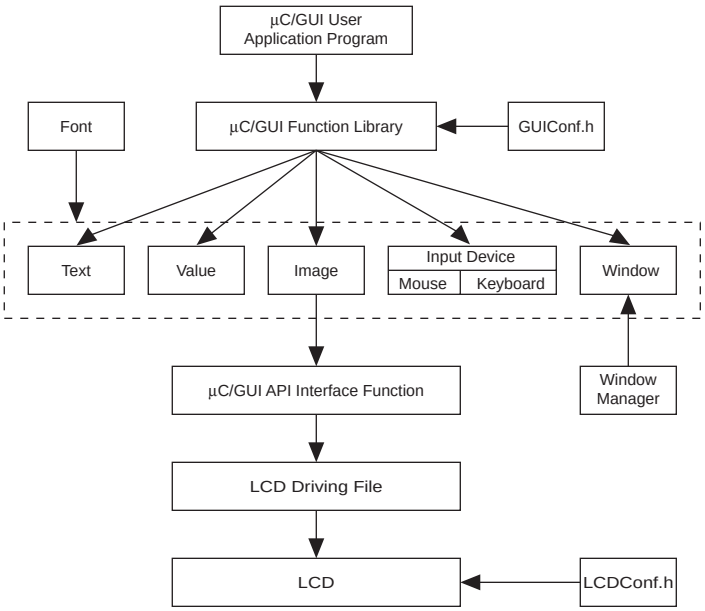
The system employs a PS/2 interface mouse, which has a two-way synchronous serial protocol, i.e., a pulse on the clock line is followed by one-byte data to the data line. For connection with the Nios II processor, we used Verilog HDL to compile the Avalon slave implementing the PS/2 mouse transmission protocol. The Verilog HDL programs include:

- **mouse_avalon_interface.v** is the Avalon bus slave interface.
- **mouse_register_file.v** and **mouse_avalon_interface.v** convert the data and transmit the PS/2 mouse protocol, compile the corresponding driver in the GUI, and implement the GUI mouse operation.

µC/GUI Migration

µC/GUI is an excellent graphic software for embedded systems. It features open source code, portability, clipping, stability, and reliability. µC/GUI provides rich interface elements, such as buttons, edit boxes, sliders, etc., and also supports an efficient window deployment mechanism for providing connectivity between interfaces and applications. The human-machine interaction interface of the small multi-functional digital photo frame system is developed using this tool. Figure 9 shows the µC/GUI software system structure.

Figure 9. µC/GUI Software System Structure



The µC/GUI functions library provides a GUI interface for user programs, including text, value, 2-dimensional (2-D) view, input device, and various window objects. The input device can be a keyboard, mouse, or touch screen. The 2-D view contains pictures, beelines, polygons, circles, ellipses, arcs, etc. Window objects include buttons, edit boxes, progress bars, check boxes, etc.

The user can configure the µC/GUI functions library with the **GUIConf.h** file, including whether to use a memory or window management device, whether to support an OS or touch screen, selecting the dynamic memory size, etc. Additionally, the **GUIConf.h** file defines various hardware-related attributes, such as LCD size, color, and interface functions. The LCD driver interprets µC/GUI functions into the liquid crystal interface function defined in the **GUIConf.h** file, and is not dependent on the hardware connection. Using a driver, the µC/GUI-LCD hardware interface converts the hardware interface function into an LCD read/write function defined in the **GUIConf.h** file.

Migrating µC/GUI involves:

- *Migrating the configuration file*—First we migrated the **GUIConf.h** file and then the **LCDConf.h** file. We configured the relevant configuration file parameters according to the digital album system's display module requirements.

- *Migrating the LCD driver*—µC/GUI provides drivers for different LCD controllers. For example, the KS0713, SED1335, and T6963 controllers have corresponding LCD drivers. However, our system's display module is the TOPWAY TCB8000A LCD controller with a TFT 65,000 color LCD screen, and µC/GUI does not provide a driver for it. Furthermore, unlike the LCD controllers for which µC/GUI provides drivers, the TCB8000A controller has an independent screen control instruction system. These factors made it difficult to migrate the controller for µC/GUI.

During migration, we first used the TCB8000A instruction system to let µC/GUI provide the most application program interface (API) functions for the upper application function. Then, for API functions that the TCB8000A controller does not support in hardware, we repaired the drivers using software. Last, by testing a large quantity of controls, we adjusted the TCB8000A display instruction parameters used in the LCD drivers, optimizing the LCD driver performance and seamlessly migrating µC/GUI for the TCB8000A controller.

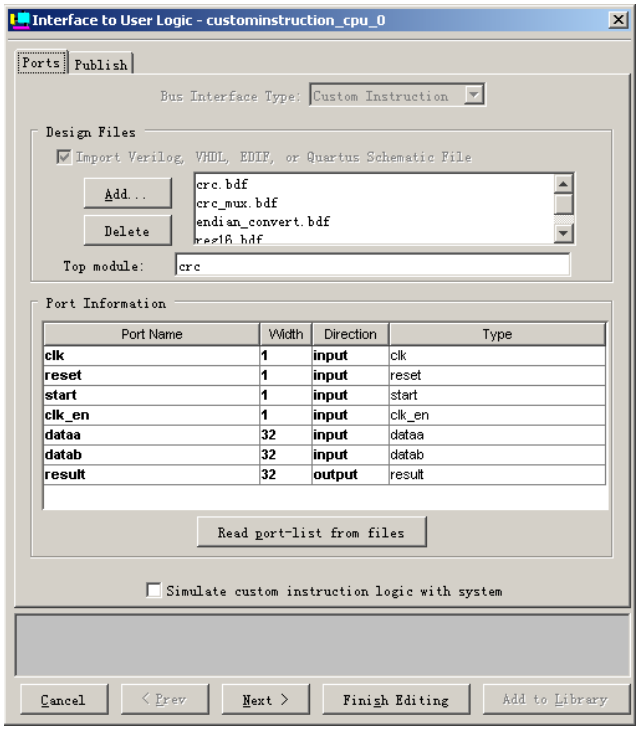
MRI Image Data Access Module

SD card technology was co-developed by Panasonic, Toshiba, and SANDISK. As a totally open standard (or system), SD cards are used in MP3 players, digital video cameras, digital cameras, e-books, audio visual (AV) appliances, etc., and particularly in ultra-slim digital cameras. SD cards are the same as multimedia cards (MMC) in shape, but are a little thicker than MMCs and have more capacity. Additionally, SD cards are compatible with MMC interface specifications. The nine-pin SD card can change the transmission mode from serial to parallel, improving the transmission speed. It reads/writes faster than MMCs in a more secure mode. To make the system more applicable and compatible, we decide to use an SD card as the primary storage media to store photos, music, material, etc.

µC/OS-II is contained in the Nios II software IDE, and users can apply it easily in their own software engineering. To facilitate concurrent task processing and CPU sharing, we used µC/OS-II with a file system. At the beginning, we selected the zlg file system, but its speed was unsatisfactory during testing. Reading/writing 1 Mbyte data into the SD card required 37 and 57 seconds, respectively. When we analyzed the situation we found that the DE1 board's one-line SD card (i.e., it only has one data line) reading functionality is significantly limited in speed. We added data lines, changing the read/write mode to four lines, which makes it more compliant with the system demands.

Testing showed that the speeds were improved to 17 and 27 seconds, respectively. However, these improvements were not quadruple the original speed as we expected. After performing an Internet search, we found that zlg/fs was low performance and consumed too much time. Therefore, we decided to use Micrium's µC/FS, which has excellent compatibility with µC/OS-II and high performance. After several weeks, we successfully migrated µC/FS version 1.34 to the DE2 platform to establish the file system for the four-line SD card. According to our comparison test, the read/write speeds were greatly increased, requiring only 3.6 and 11 seconds, respectively for 1 Mbyte of data. Therefore, this implementation essentially satisfied our speed requirement for accessing data files. Writing required more time because when data is written into the SD card, each block calculated a 16-bit cyclic redundancy code (CRC), which takes some delivery time. To improve write speeds, we accelerated the CRC16 computation with a Nios II custom instruction. With this method, we reduced the time it took to write an MR image (about 1.5 Mbytes) to the SD card from 21.7 to 15.8 seconds. With a higher speed SD card, the write times will be even faster. Figure 10 shows the custom instruction in SOPC Builder.

Figure 10. CRC 16 Custom Instruction in SOPC Builder



We used the following signals:

- CLK—Host to card clock signal.
- CMD—Bidirectional command/response signal.
- DAT0 through DAT3—4 bidirectional data signals.
- VDD, VSS1, and VSS2—Power and ground signals.

Figure 11 shows the data packet formats for one-line and four-line SD card read/writes. Figures 12 and 13 show the single block read and write, respectively.

Figure 11. Data Packet Format

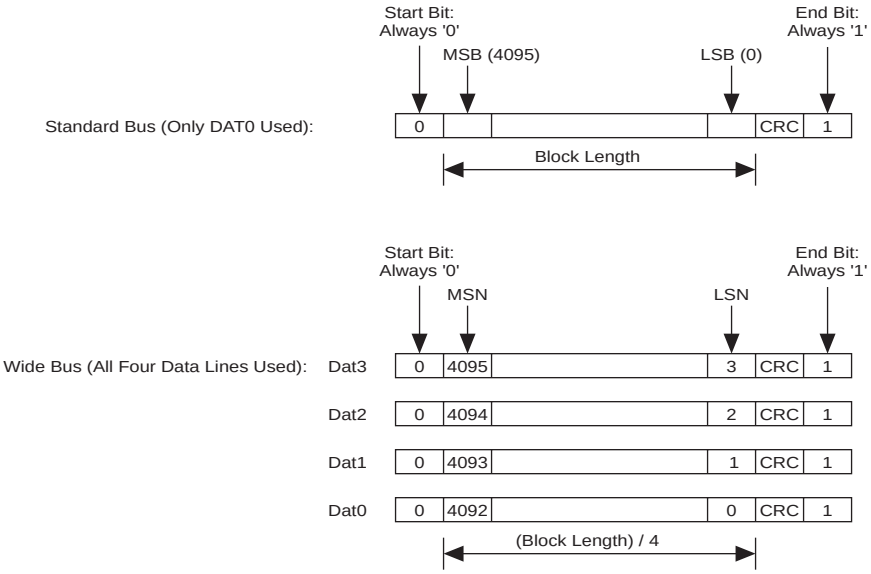


Table 1. Timing Diagram Symbols

Abbreviation	Definition
S	Start bit (= 0)
T	Transmitter bit (host = 1, card = 0)
P	Single-cycle pull-up (=1)
E	End bit (= 1)
Z	High impedance state (-> - 1)
D	Data bits
X	Don't care data bits (from card)
+	Repetition
CRC	Cyclic redundancy check bits (7 bits)
	Card active
	Host active

Figure 12. Single Block Read Timing

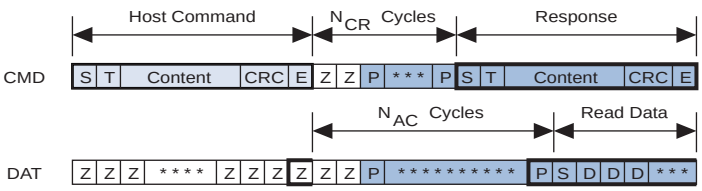
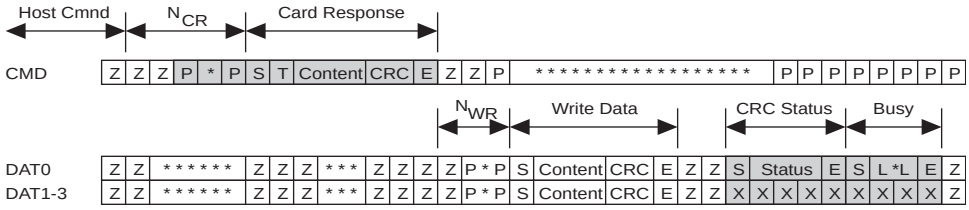


Figure 13. Single Block Write Timing



Network Interface

Because the DE1board does not have a network interface, we designed a network interface board that connects to the DE1 interface to enhance the system's scalability and add network capabilities. Because the board's I/O interface is limited and we also needed to connect the LCD display through I/O, we used a DM9000A network chip that has fewer I/O. Additionally, the DE1 board transmits data at a fixed time while the PC receives data upon inquiry. To use the network, a µC/OS-II driver implements data transmission and data is received with interrupts on the second network layer.

Image Processing Algorithm Implementation

Typically, bodily acantha contain 24 pre-sacral acanthae, including 7 cervical acanthae, 12 thoracic acanthae, and 5 lumbar acanthae. Therefore, in our acantha sagittal views, 23 intervertebral disks are visible in typical images, specifically, C2-3, C3-4, C4-5, C5-6, C6-7, C7-T1, T1-2, T2-3, T3-4, T4-5, T5-6, T6-7, T7-8, T8-9, T9-10, T10-11, T11-12, T12-L1, L1-2, L2-3, L3-4, L4-5, and L5-S1.

The image processing algorithm quantitatively marks the visible intervertebral disks, as well as predicting and marking unclear intervertebral disks. We implemented the algorithm using a C program compiled under µC/OS-II and using µC/GUI and µC/FS, we implemented the MRI acantha image segmentation system with good human-machine interaction and facilitated the system operation.

The algorithm has three steps:

- Image preprocessing
- Spinal cord extraction
- Disk detection

Image Preprocessing

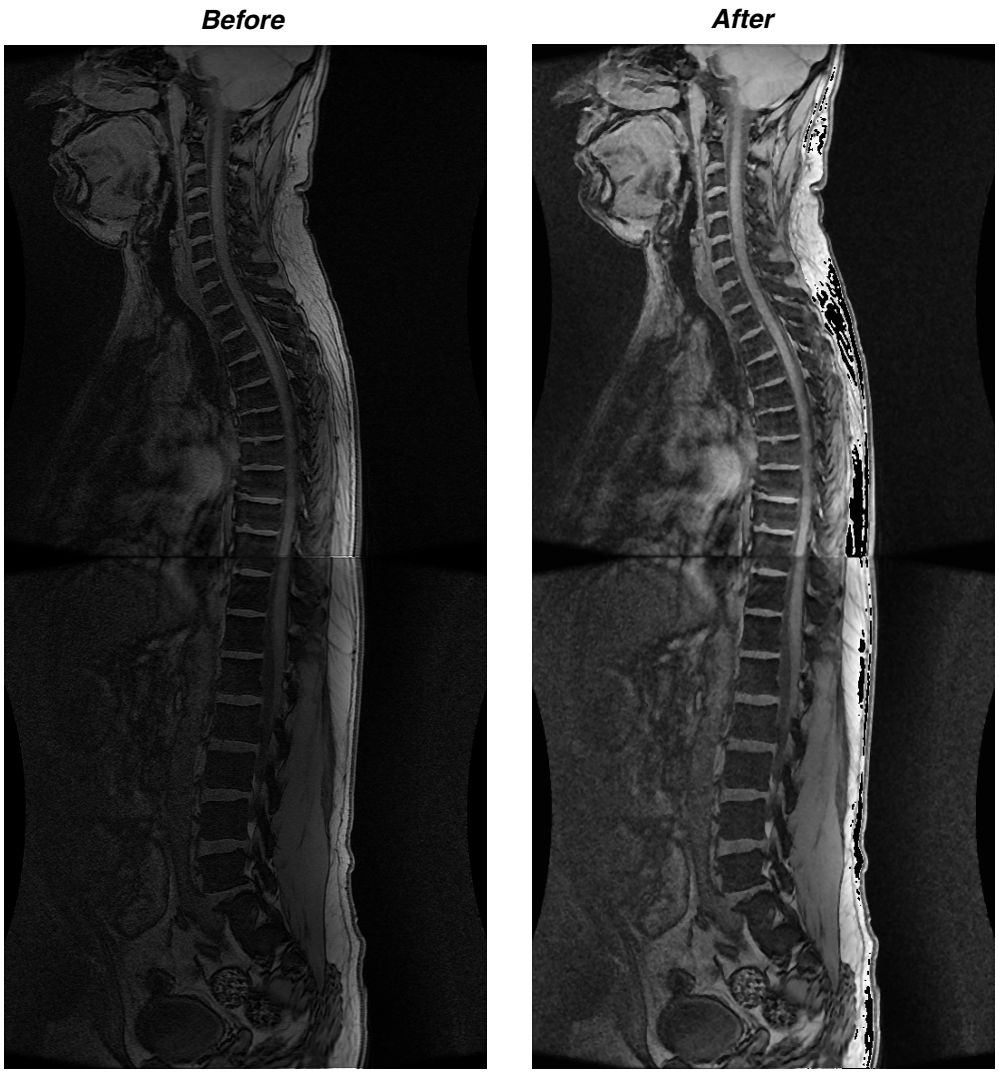
Because the original NMR images have low contrast with unclear visual effect, we first improved the image quality and the visibility of the intervertebral disks. Figure 14 shows the contrast before and after preprocessing.

The image median filtering is time-consuming, but we were able to speed it up using the C2H Compiler to accelerate the algorithm. Table 2 compares the image processing time requirements.

Table 2. Image Processing Time Requirements

Requirement	Before C2H Acceleration	After C2H Acceleration
Time for median filtering	10,077.35 ms	2,811.41 ms
Total time for image preprocessing	11,394.71 ms	4,128.77 ms
Logic resources used	5,446/18,752 (29%)	8,593/18,752 (46%)

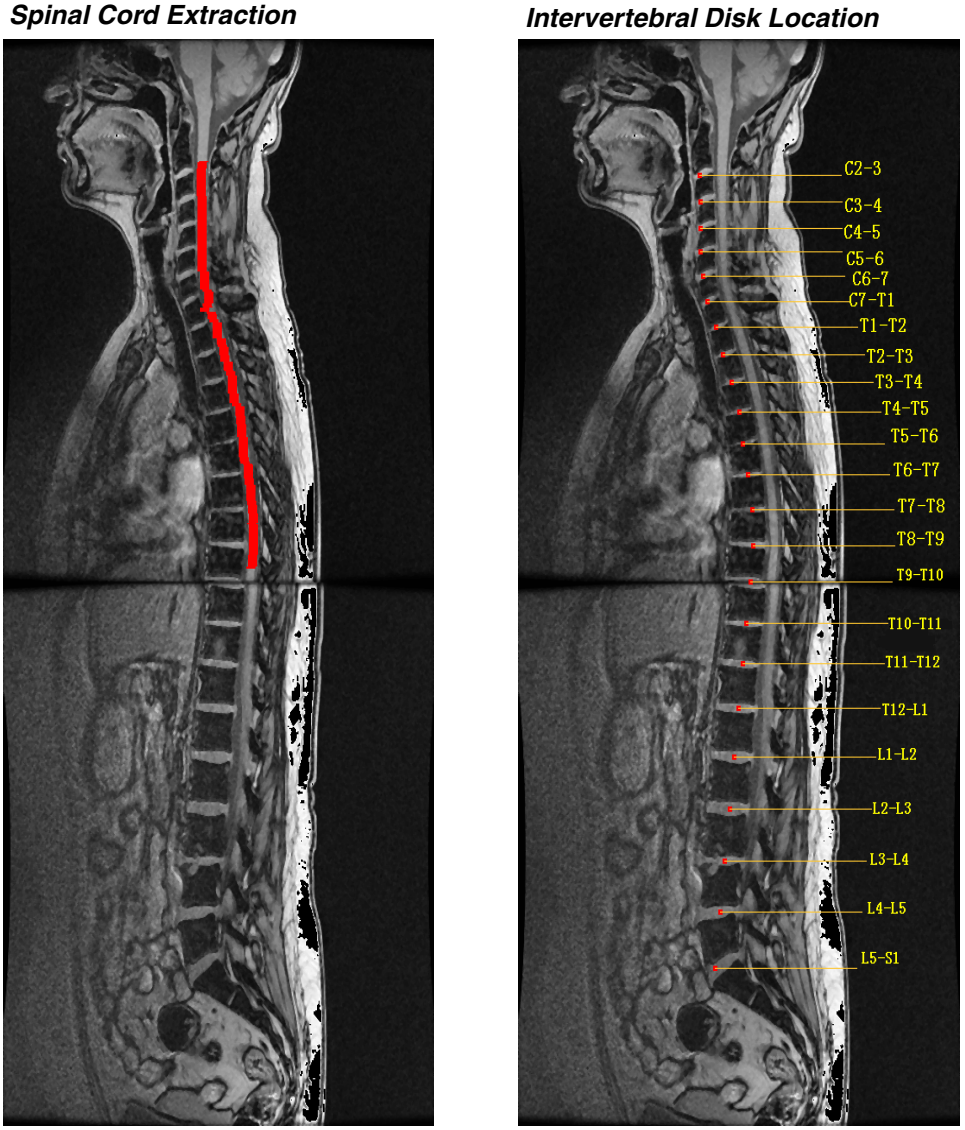
Figure 14. Sample Image Before and After Preprocessing



Spinal Cord Extraction

The spinal cord is conspicuous in the spinal NMR image, and provides directional information for the location of the intervertebral disks. We can extract the spinal cord using fuzzy match in statistical mode recognition. In our design, we only extract the spinal cord for the upper body because this curve is relatively complicated. We can then predict the lower body curve accordingly. The red line in the left image in Figure 15 is the spinal cord extracted from the upper body. Compared with other parts of the algorithm, this part is time-consuming. Given the current algorithm structure, we cannot use C2H acceleration. We expect to further optimize the algorithm, so we did not use other methods to accelerate the algorithm. In the future we will improve the system functions in terms of the algorithm structure and acceleration.

Figure 15. Spinal Cord and Disk Location



Disk Detection

In the right image in Figure 15, the red points represent the located disks, which the system marks with yellow lines.

SOPC Concepts Used in the Design

Using SOPC concepts, we completed the design successfully. In the system design, SOPC concepts are embodied in the following aspects:

- *System reconfigurability*—As a soft-core processor, the Nios II processor can be clipped. Therefore, the system we design has huge potential for scalability. For example, due to time limitations, we used C programs (with limited speed) to implement some algorithms that are actually suitable for implementation in the FPGA hardware. But, we can upgrade the system later without changing the hardware platform. Additionally, because of resolution limitations, we can

compile different LCD controllers for use with other LCD displays. These types of changes are advantages of the system reconfigurability provided by SOPC-based designs.

- *Modular system design*—System design is a process of labor division and teamwork. A typical embedded processor platform is designed according to the specified processor, and software debugging can only be conducted upon completion of the hardware. But designing with the Nios II processor is different: as long as we use an FPGA the supports the Nios II processor, we can later debug the program on other FPGA-based platforms without any differences. This method allows the designer to conduct the hardware and software design synchronously. In actual designs, designers can initiate market expansion and product research and development simultaneously, shortening the product's time-to-market and bringing significant benefits.

- *Diversified implementation modes*—In a SOPC-based system design based, there are various implementation modes. For example, to accelerate an algorithm, you can use a custom instruction, custom peripheral, or C2H acceleration and compare them to find the best method.

Design Features

The Nios II-based MRI spinal image segmentation system features fast computation, small size, and simple operation. It can be integrated easily with the original MRI devices to form a new system, and facilitate consultation using the segmented images.

- With the introduction of Nios II-based µC/OS-II, this operating system has been widely applied in many fields worldwide, such as mobile phones, routers, hubs, aerocrafts, medical instruments, etc. µC/OS-II is suited to a small control system, and features high efficiency, small size, excellent real-time performance, good scalability, etc. The operating system has been integrated into the Nios II IDE, avoiding the need for additional migration. Using the system we found that the Nios II-based µC/OS-II and tasks are very stable. We completed all system software development using the Nios II-integrated µC/OS-II RTOS.

- With µC/GUI, the system gains good human-machine interaction. All system functions can be accessed using a mouse. This feature is very helpful for promoting the system.

- In the system design, a large quantity of MRI images are stored in a high-capacity SD card. Taking advantage of the Nios II processor, we easily added in the four-line SD card controller to improve its read speed. We also migrated the µC/FS file system for the SD card to facilitate the file access operations.

- The µC/OS-II-based network interface enables strong scalability. Additionally, network support allows image updating and supports telemedicine.

- The Nios II processor includes several CPUs, and users can create a perfect solution by using processor, peripherals, storage, and I/O interfaces according to the system demands. In this way, designers build a reasonable performance combination and save system development cost, enhancing cost competitiveness.

- The Nios II C2H Compiler can automatically convert a C language program that requires high performance into a hardware accelerator and integrate it into the FPGA-based Nios II subsystem, which improves system operation speeds.

Conclusion

We completed our MRI spinal segmentation system based on the Nios II processor design for this contest, and implemented the system functions as scheduled. However, we still need to improve some areas. During the contest, we learned many new things about the Nios II processor and it was the first time we tried to implement an algorithm in the Nios II processor for medical imaging. Our experience demonstrated the Nios II processor's strong processing capability and reliable operation. Additionally,

we became experienced with using μC/OS-II, μC/GUI, and μC/FS with the Nios II processor and solved many debugging problems with teamwork.

Nios II system performance can be adjusted according to the application requirements to satisfy specific user demands, which gives Nios II systems strong advantages over fixed processors. User logic function and custom instructions are highlights of the Nios II processor, and provide a variety of methods to implement systems. SOPC concepts bring creative design thought and enlighten students' creativity.

Thanks to Altera for providing us a good opportunity to combine theory with practice. The contest verified the feasibility of our algorithm in hardware and promoted our theoretical research.