

Implementation of Music Broadcast System Using Altera DE2 Boards and Qt

Cheng-Long Zhao, Hui-Bin Shi, Qiao-Zhi Sun, and De-Chun Kong

Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

zhaochenglong08@126.com

hshi@nuaa.edu.cn

504914613@qq.com

263860242@qq.com

Abstract— This paper will introduce a music broadcast system based on schedule using Altera DE2 boards and Qt technology. The system has Client/Server architecture. Qt technology is used to set up a server in Ubuntu operation system. It will manage all online DE2 boards and send commands to them to play music stored on SD card inserted into the board.

Keywords— FPGA; Scheduled Music Broadcast; Qt; Nios II

I. INTRODUCTION

What's the traditional use and implementation of broadcast system? How to implement a school on time bell system? Have you ever thought about these questions?

Broadcast system has many implementation methods which include: traditional transmission with power broadcast system, digital audio broadcasting (DAB), radio broadcast system and so on.

With the development of Internet, IP broadcasting is becoming another kind of broadcasting system. About IP broadcasting you can see in [7]. What we have done is also a broadcast using Internet. But we want to do it in a different way. We use FPGA as the terminal and use UDP (User Datagram Protocol) to communicate between terminals. Our work now is focused on the manageable schedule play which can be used in school or university, used for daily fixed audio playback. Due to joining the network, it can be extended easily according to need. All the play songs are stored in SD card (Secure Digital Memory Card).

We can also see a no network play system which is very similar to our system. It was listed in [8].

II. SYSTEM ARCHITECTURE

The system architecture is show in Fig.1.

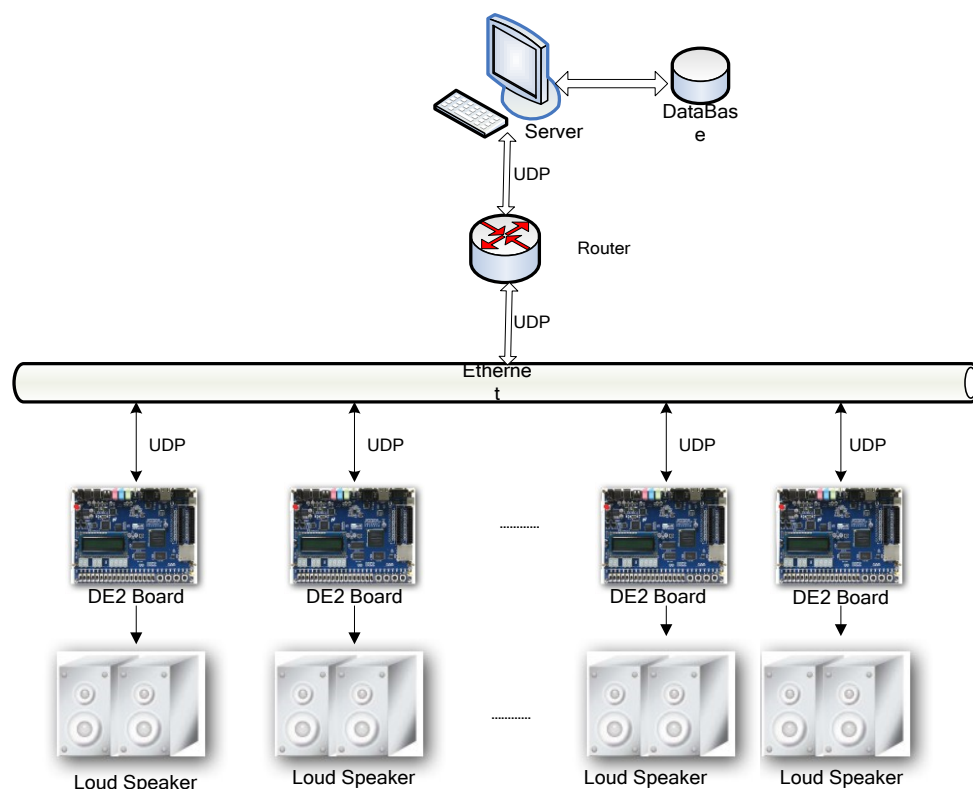


Fig.1 System architecture

As shown in Fig.1, our system uses Client/Server model. On the server side, we use Qt technology to build a management program, while on the client side we use DE2 development board. The communication between server and client uses UDP protocol. The DE2 board we use has a lot of resources as shown in Fig.2.

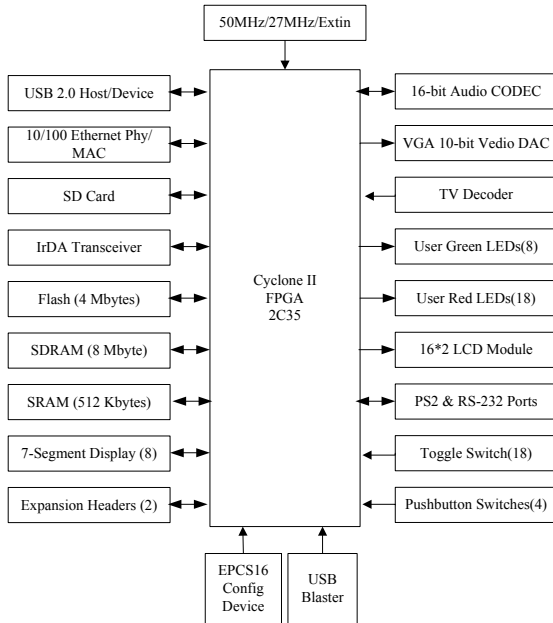


Fig.2 Block diagram of the DE2 board

As shown in Fig.2, the DE2 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects. The following hardwares are used in our system:

- Altera Cyclone II 2C35 FPGA device
- Altera Serial Configuration device-EPCS16
- USB Blaster for programming
- 8-Mbyte SDRAM
- SD Card socket
- 4 pushbutton switches
- 18 red user LEDs
- 9 green user LEDs
- 50-MHz oscillator and 27-MHz for clock sources
- 24-bit CD-quality audio CODEC(WM8731) with line-in, line-out, and microphone-in jacks
- 10/100 Ethernet Controller(DM9000) with a connector
- 16*2 LCD Module

The key modules for our application are Wolfson WM8731 24-bit sigma-delta audio CODEC, which is used for wav file codec and sends out signals to loud-speaker. The SD card socket provides 4-bit SD mode for SD card access and storage of audio files of wav type. The communication device DM9000A, supports full-duplex operation at 10Mb/s and 100Mb/s, with auto-MDIX (Medium Dependent Interface cross-over).

We use a Nios II soft core to control all the modules mentioned above.

Firstly, we give a brief description of the working process of the system. To begin the working process we first enter a music library on the server part. We make sure the library is in sync with the SD card's music file list on DE2 board. Next, we formulate a week music play schedule to indicate the university class starting and ending respectively, which will regularly send broadcast commands to the DE2 development board to play. Upon receiving the command, the board will check to see if the music file name within the command exists in the SD card's music list. If it exists, the board will play the song.

In order to let the manager software know the current terminal online, we use the heartbeat packets sent from DE2 board to inform software on PC. If the software detects that there is at least one terminal online, it will enable the broadcast.

III. MANAGE SOFTWARE DESIGN

In order to build managing software on PC, we choose a cross-platform technology called Qt.

Qt is a cross-platform application and UI framework for developers using C++ or UML (Unified Modeling Language) , a CSS and JavaScript like language. Qt Creator is the supporting QT IDE. Qt, Qt Quick and the supporting tools are developed as an open source project governed by an inclusive meritocratic model. Qt can be used under open source (LGPL v2.1) or commercial terms. More information about Qt can be seen at [1].

Before we introduce the design of our managing software, we firstly introduce some concepts in Qt which have been used in our design. All the information can be found in the Qt develop suite's Qt assistant software.

The first thing we need to introduce is Qt's support for the database. The QSql module uses driver plugins to communicate with the different database APIs. Since Qt's SQL module API is database-independent, all database-specific code is contained within these drivers. Several drivers are supplied with Qt and other drivers can be added. The driver source code is supplied and can be used as a module for writing your own drivers. Since we don't have many data to store, we choose a small and cross-platform database called SQLite. More information about this database can be found at [2].

The next thing is Qt's event driven module. Qt uses signals and slots to communicate between objects. The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks. A signal is emitted when a particular event occurs, for example when we push a button on the widget or we closed it. A slot is a function that is called in response to a particular signal. Qt's widgets have many pre-defined slots, but it is common practice to subclass widgets and add your own slots so that you can handle the signals that you are

interested in. The more things about this topic you have to look up the assistant software with topic Signals & Slots.

The last thing is Qt’s model/view architecture. Model/view architecture is evolved from MVC (model-view-controller). If the view and the controller objects are combined, the result is the model/view architecture. This still separates the way that data is stored from the way that it is presented to the user, but it provides a simpler framework based on same principles. This separation makes it possible to display the same data in several different views, and to implement new types of views, without changing the underlying data structures. The model communicates with a source of data, providing an interface for the other components in the architecture. The nature of the communication depends on the type of data source, and the way the model is implemented.

The view obtains model indexes from the model; these are references to items of data. By supplying model indexes to the model, the view can retrieve items of data from the data source.

In standard views, a delegate renders the items of data. When an item is edited, the delegate communicates with the model directly using model indexes. The work of this architecture can be seen in Fig.3.

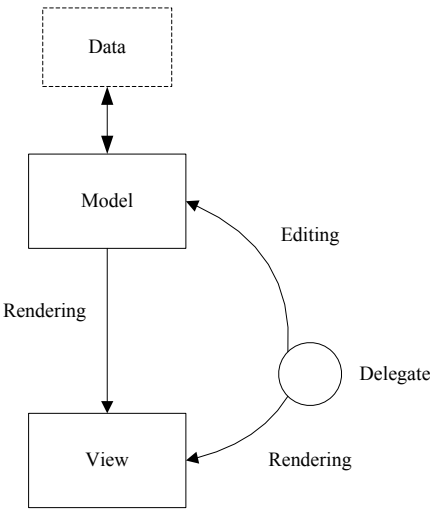


Fig.3 model/view architecture

The manager software we designed has a clear main interface. It is shown in Fig.4.

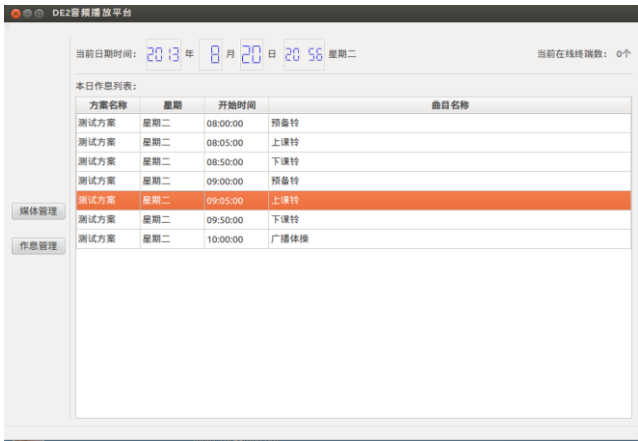


Fig.4 Main interface of manage software

As shown in Fig.4, in the main interface, we can see the information such as date, time, day of the week and the schedule of this day. Also we can see the now online terminal number of this system. On the left of the main frame there are two buttons. The top one is to show the media manager interface. The one below is to show the schedule manager interface.

When we push the top button we can see the media manager interface in Fig.5.

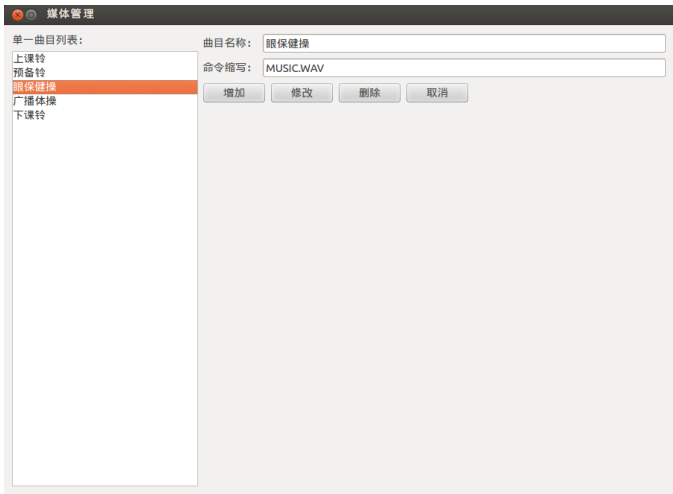


Fig. 5 Media manage interface

As shown in figure 5, we can see an edited media list on the left part of the interface. On the right is edit partition to add, remove and change information about the media. It is important to note that the audio command is upper case string of the song name in the SD card in the terminal which will send to the terminal on time.

When we push the bellow button on the main interface, we can see the routine management interface in Fig.6.



Fig.6 Routine management interface

As shown in Fig.6, we can specify a scheme, this scheme includes a week schedule. Each day of the week has several items to do which can be edited by the below part of this interface. When we add an item we can choose the start time of this item and the play song. The choosing song widget is shown in Fig.7.

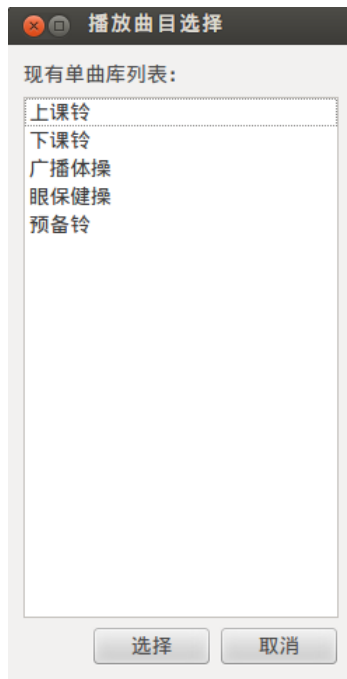


Fig.7 Song Select Interface

As shown in Fig.7, we just need to select one play song for the editing item. In order to store the data of this system, we designed some table in database to store these. The database relationship can be seen in Fig.8.

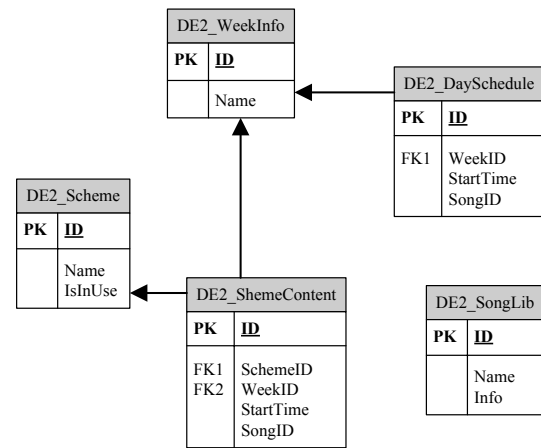


Fig.8 Database relationship of the system

We can see in Fig.8, we don't have a very complicated database structure since we don't have very complicate data to store. In the next part of our paper, we will introduce our hardware design.

IV. HARDWARE DESIGN

Many commercial media/audio players use a large external storage device, such as an SD card or CF card to store music or video files. Such players may also include high-quality DAC (Digital-to-Analog Converter) devices so that good audio quality is produced. The DE2 board provides the hardware and software needed for SD card access and professional audio performance so that it is possible to design advanced multimedia products using the DE2 board. We can use the Nios II processor to send and receive Ethernet packets using the DM9000A Ethernet PHY/MA Controller. Since the packet and command we want to communicate between DE2 board terminal and PC software is very short and we must broadcast our message to all the terminals online, so we choose UDP as our transmit protocol.

The first thing we need to do for hardware design is to set up the system we need in SOPC (System on a programmable Chip). The version of Quartus II we used is version 12.0. In order to better control the audio playback and network communication, we choose the Altera soft core of Nios II as our control chip and drivers of the DM9000A and WM8731 are as slave peripherals articulated on the Avalon bus. The hardwares we used have been listed in section II.

In our system we show how to implement an on time SD card music player by using Ethernet communication on DE2 board, in which the music files are stored in an SD card and the board can play the music files via its CD-quality audio DAC circuits. We use the Nios II processor to read the music data stored in the SD card and use the Wolfson WM8731 audio codec to play the music. The Ethernet device is DM9000A. The structure of this system is shown in Fig.9.

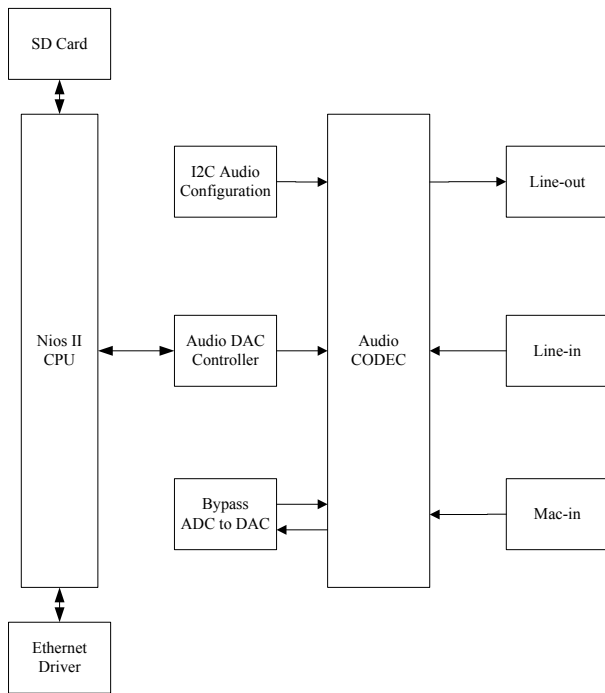


Fig.9 Hardware system architecture

As shown in Fig.9, we use Nios II processor to process reading from SD card, send data to audio codec and communicate with PC program.

The remaining of the paper is organized as follows. We will first introduce the soft main flow chart of hardware system on Nios II. Then we will briefly describe the implementation of the key module of this system like DM9000A, WM8731 and SD card.

A. The main function process

The flow chart about main function of embedded software is shown in Fig.10.

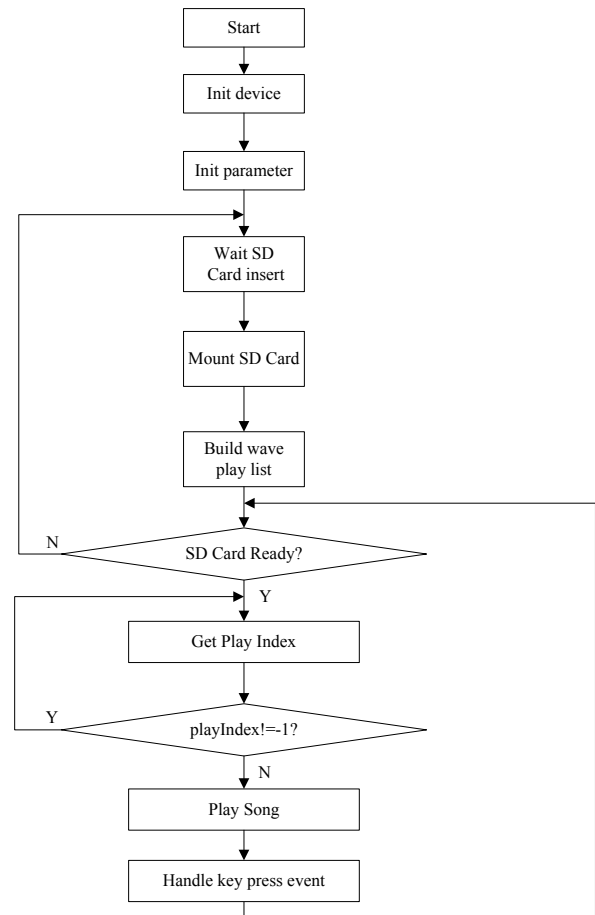


Fig.10 Main function process

When DE2 board is power on, it will first init peripheral equipment such as LCD display, DM9000A. Then the volume of the output for WM8731 is set to initial number. After doing these, it will try to wait SD card insert into the socket on the DE2 board and mount it if the file system is supported. If so, the next thing for DE2 to do is to build up play list. All of the above work is initializing work required. If there is an interrupt on DM9000A, we will get a play song name string and compare it with the play list to find an index. If we found one, the DE2 terminal will start to play.

B. DM9000A and Ethernet Communication

We already have mentioned DM9000A several times before we start this topic. In order to transmit command, the system must contain network interface module which is designed as IP-core in SOPC Builder. This module needs to achieve function of hardware interface design and hardware abstraction layer driving design. All we need is supported by Terasic Company and DE2 board since it has DM9000A device. The DM9000A is a fully integrated powerful and cost-effective fast Ethernet MAC controller with a general processor interface, an EEPROM interface, 10/100PHY and an 4KB SRAM[3].

The system adopts UDP protocol to send command string. UDP is a simple, datagram-oriented, transport layer protocol. UDP provides no reliability: it sends the datagrams that the application writes to the IP layer, there is no guarantee that they ever reach their destination. So in our further design we will add up an answer string to server. The process of commands transmission of this system can be described as follows:

- 1) *Initialize network card and other peripheral equipments.*
- 2) *Arp communication:* ARP protocol is address resolution protocol which resolves IP address of the target host to Ethernet(MAC) address, in order to ensure communication. So at the beginning of first communication, we must know IP address of the target, then send ARP message to local the MAC.
- 3) *UDP communication:* All the UDP packet we send is encapsulated by our self. It must encapsulate data according to Fig.11.

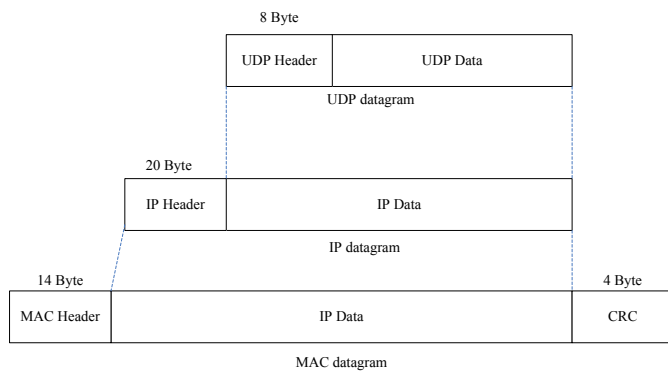


Fig.11 UDP encapsulation format

As shown in Fig.11, the first part of MAC header in the network interface layer, total 14 byte; the second is part o IP header in network layer, total 20 bytes; following is part of UDP header in transport layer; the last part is sending data.

In order to let server know there is terminal online, we init a timer to send const string to server which we called heartbeat package. The format of this packet is “@@terminal_id@HeartBeat”. The terminal id is a serial number string like “00001” which gives a unique string to DE2 board. We set the timer interval of 1s, that is to say, every second, the server will receive a heartbeat packet.

When server receives these heartbeat packets it will check if the serial number of terminal has been added to online terminals list. If so, server will set the out of time of this terminal to zero, otherwise, it will add the time out number until it reach the upper bound and will be remove from the online list.

In order to check whether UDP packet has been send to PC or not, we used a program in Windows 7 called WireShark before we start combine our system to Ubuntu. WireShark is a

capture package tool and easy to use. One of picture about a non-meaning string we send to PC in our design is show in Fig.12.

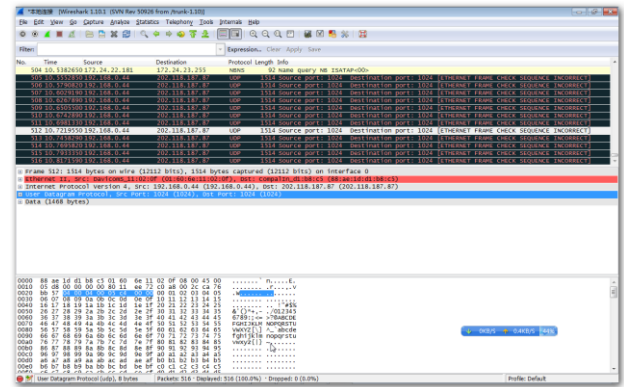


Fig.12 UDP Send to PC

C. Audio Play Module

I²C bus is a kind of two-wire serial bus, developed by PHILIPS, used to connect micro-controller and peripherals. There are two signal lines in I²C serial bus. One is a two-way data line (SDA), the other is the clock time line (SCL). The serial data lines of slave devices must connect to data line (SDA) when it links to I²C bus. The clock lines of devices connected to I²C bus's clock line SCL, which can achieve serial data transmission between master and slave devices easily [4].

In the design, I²C controller realized on FPGA could control the WM8731 and transmit data. The working modes of WM8731 include the master and slave modes. When working in slave mode, WM8731 response the data coming from the audio data interface—convert the digital audio signal into analog audio through the D/A, and then play the sound signal. There were four modes in WM8731 audio-data transmission: left-justified, right-justified, I2S and DSP modes. In this design left-justified mode was chosen to realize the audio-data transmission between FPGAs and WM8731.

The audio codec is configured in the slave mode, where external circuitry must provide the ADC/DAC serial bit clock (BCK) and left/right channel clock (LRCK) to the audio codec. The audio DAC controller is integrated into the Avalon bus architecture, so that the Nios II processor can control the application [5].

During operation the Nios II processor will check if the FIFO memory of the Audio DAC controller becomes full. If the FIFO is not full, the processor will read a 512-byte sector and send the data to the FIFO of the Audio DAC Controller via the Avalon bus. The Audio DAC Controller uses a 48 kHz sample rate to send the data and clock signals to the audio CODEC. The design also mixes the data from microphone-in with line-in.

To play a music file with this system, the file must use the 48KHz sample rate and 16-bit sample resolution wav format. Copy one or more such wav files onto the FAT16-formatted

or FAT32-formatted SD card. Due to a limitation in the software used for this system, it is necessary to reformat the whole SD card if any wav file that has been copied onto the card needs to be later removed from the SD card.

D. SD Card Module

SD card is a flash memory device based on a new generation of semiconductor memory devices. SD card with high memory capacity, fast data transfer rates, great flexibility and good mobile security. Considering of system size, power and storage capacity, it is easy to find out there are great advantages to choose SD card as the storage device for those systems which are based on the FPGA chip [6]. The more things about SD card reading and writing can also be find in [6].

The mode we use is 4-bit SD mode and the file system is FAT16. In order to use this mode we need first realize some command for SD card. The commands we use includes: CMD0, CMD8, CMD55, CMD2, CMD3, CMD9 etc.

In order to use FAT16 file system, we also need to know the structure of this system. There are four parts in FAT16 file system which can be seen in Table.1.

TABLE I
LOGIC STRUCTURE OF SD CARD

Master boot sector	32 sector
Partition boot record(PBR)	1 sector
FAT Table 1	According to the capacity of the SD Card
FAT Table 2	According to the capacity of the SD Card
Data section	According to the capacity of the SD Card

The structure of FAT16 file system in SD card contains four parts. It is the Partition Boot Record (PBR), File Allocation Table (FAT), File Directory Table and Data Section.

In the PBR, the BIOS parameter record block in which recorded some most important parameters. The position of FAT, file directory table and data area (which sector in SD card) can be calculated through these parameters. Following the partition boot record is FAT area. FAT table records the link information of files which store between clusters. This is the chain store of files. Following the FAT table is the file directory table FDT (File Directory Table), which takes over 32 fixed sectors; each sector can hold 16 register items. The contents of the registration contain file name, file attributes, modification time, file length, etc Following FDT is the data area which takes up most of the disk space and is used to store file.

After test, the system we designed can detect the number of terminal online and let terminal play special song on the time we specified in schedule.

V. CONCLUSIONS

We introduced a system use both software and hardware knowledge to combine a system called on time music broadcast system. This system used Qt technology to implement GUI and functions on PC part which will make it very easy to change it's platform from Ubuntu to Windows. Also we use UDP as the protocol when we communicate between PC and DE2 board. UDP can easily implement broadcast communication in local area network (LAN).

But after deeper thinking about our system, we can find some weaknesses about our system.

- 1) If there is no network available, how to deal with the bell ring for teaching work?

So, it's better to and a configuration file to ensure that no network condition system can work normally. The configuration file can use XML format or plain text. Also we need an external clock circuit to get the current time since DE2 board doesn't support this function.

- 2) If we need the function of speech can be supported in our system?

Maybe, the biggest problem is on the software side since first thing we need to capture data from sound card and then we will need to encode the data to wav format which is supported by our system and send it to the online terminals.

- 3) If we need to group the terminal by grade 1, 2...

We can think about more additional functionality. Our try just is a start. The more important and the main meaning of our work is to expand the use of FPGA in daily application.

ACKNOWLEDGMENT

Many thanks must give to Altera Corp for supplying excellent FPGA chips and development board. We want to thank TERCASIC providing a solid background for our work.

REFERENCES

- [1] (2013)Qt, <http://qt-project.org/>.
- [2] (2013)SQLite, <http://www.sqlite.org/>.
- [3] DM9000 ISA to Ethernet MAC Controller with integrated 10/100 PHY, AVICOM Inc, 2001.
- [4] Hai, X., Zhao, C., & Jiang, X. (2012). Train station classification for passenger dedicated line. *International Journal of Advancements in Computing Technology*, 4(15), p.328-335.
- [5] (2013)DE2_UserManual_1.6.pdf, <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=China&CategoryNo=60&No=31&PartNo=4>
- [6] Zhenlin LU, Jingjiao LI, Yao Zhang, "The reading/ writing SD card system based on FPGA", in *First International Conference on Pervasive Computing, Signal Processing and Applications*, 2010, p.419-422
- [7] TetsutaroUEHARA, Takashi SATO, Katsunori YAMAOKA, "The design and implementation of a music broadcasting system via IP multicast with user-authentication", *Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference*, vol(2), p.984-987
- [8] LIANG Hong-wei, LI Jian-ai, KAN Ling-ling, "Implementation of SD Card Music Player Using Altera DE2-70", *International Conference on Multimedia and Signal Processing*, 2011, p.150-153