Music Synthesizer Designed on FPGA

Han Liu, Rong Su, and Hui-Min Dai
Sun Yat-Sen University
Guangzhou, the People's Republic of China
liu16th@gmail.com

Abstract — **As touch technology has become a hot spot of human-computer interaction in recent years, we accomplished the design--Music Synthesizer on LCD touch screen. We took use of the resources from board DE2-115, combined with Verilog HDL and C Language under the environment of Quartus II and Nios II Eclipse of Altera's. In the time when digital music develops in such a high speed, our design can achieve four different kinds of instruments playing and the control of changing the timbre, making more people able to create music themselves under a lower cost.**

Keywords — **music synthesizer; FPGA design; timbre; SOPC Builder; Nios II**

I. INTRODUCTION

Human-computer interaction, studying communications and correspondence between human beings and computers, to the utmost extent, makes information management, services, processing and other functions possible to work[1]. Along with electronic products like Apple's iphone, and the brand Android phone market, panel computers emerged and predominated in market shares, touch technology has been widely applied and rapidly developed, providing the public with a more natural, direct and efficient interactive pattern. Touch technology, which has broad application prospects, becomes not only a research but also an application hot spot. Nowadays it has an enormous consumer market and application platform. Touch screen has been widely used, such as private digital assistants, mobile phones and other applications, gradually turning into an ordinary and familiar mode of human-computer interaction.

Appeared on the market in recent years, a lot of musical instruments simulation software has become popular among consumers, such as "Magic Piano". As can be seen, the simulation of instrument entities is implemented by the design of the touch screen interface. We use our hands to touch the simulate keyboard and the system will scan signals through hardware to find the corresponding musical notes. Every note corresponds to a code, which can be decoded and output a corresponding note by the audio decoder. Thus, we can achieve the process of instrument playing.

Our ideas for the design begin from here. So, here comes the question, what should we do to design a multi-instrument playing system which is both adjustable and timbre-controllable?

First, we learned the function of the devices used in the design.

A. Introduction of FPGA platform

1) DE2-115 demo board

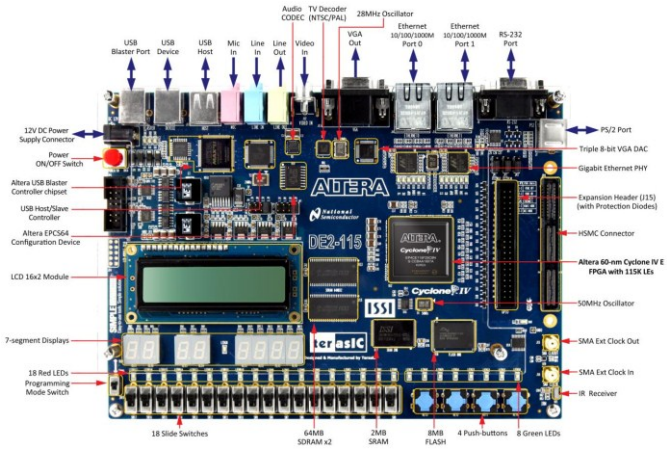

Fig. 1 Structure of DE2-115 demo board [2]

Fig. 1 shows the structure and layout of DE2-115 demo board. We mainly took use of the Audio CODEC (WM8731) chip, 24-bit ADC and DAC, whose powerful features help achieve the audio codec processing. And with Cyclone IV E FPGA driving VGA synchronization signals and a three-channel 10-bit high-speed video DAC chip ADV7123 transforming the output digital signals to analog 8-bit RGB signals, we can achieve the interface picture display.

2) Touch Screen

The design uses the 7-inch TFT-LCD touch screen provided by the Terasic, with a resolution of 800×3 (RGB) ×480[3], shown in Fig. 2.

Fig.2 Terasic touch screen [4]

On this platform we can make musical instrument entities turn into virtual interface display. And with hardware and software driving together, we can make multi-touch and audio corresponding output available and controllable.

B. Introduction of language

We take Verilog HDL and C Language to achieve the design, using a combination of hardware and software. Thus, we can take advantage of either hardware language's high speed, accuracy of hardware description, convenience to maintain and simulate or software language's flexibility, high-reliability, error-checking function. The two languages are inseparably interconnected, which is greatly conducive to achieve the hardware circuit system better and faster.

C. Design Review

This design uses two softwares Quartus II and Nios II IDE, with the resources on FPGA development platform DE2-115 demo board. Using Verilog HDL to build the system hardware platform on Quartus II, we accomplished the designation of each function module. With embedded SOPC development tools, integrated soft core, CPU, PLL, memory, input and output components on FPGA chip. Data controlled by the Avalon bus, developed with software Nios II IDE, then we can finish the drive for multi-touch module and the processing of all data.

Therefore, the highly integrated music synthesizer chip combines hardware design and software design, with the abundant resources on FPGA, powerful and efficient ability to deal with data.

II. MODULE AND FUNCTION

Our design uses the combination of Verilog HDL and C language to accomplish a playing system of analog electronic organ, piano, guitar, drums etc., with either automatic demo play mode or personal play mode on a touchable LCD screen under the environment of Quartus II and Nios II IDE (11.0) .

The entire system includes five main modules: image display on LCD interface, real-time touch control, sound synthesize, instruments switch, play and auto-play.

A. Image Display on LCD Interface

The device used for display module is the 7 inches TFT a-Si Active Matrix Color LCD, whose resolution is 800×3(RGB) × 480[3]. Image showed under the circumstance of right horizontal scanning and field scanning frequency.

1) LCD Display Module

The port definition of LCD display module as Fig. 3, in which iCLK represents clock signal of display, connected to the 33MHz clock signal generated by PLL. iRST_n represents reset signal, iNote for the currently playing note signal detected from the touch screen, oHD for horizontal synchronization signal, i.e., the line sync signal, oVD for vertical sync signal, i.e., the line sync signal, oDEN for the data valid signal, oLCD_R, oLCD_G, oLCD_B for red, green and blue component signals required for LCD display.

```
//=======================================================
module ltp_controller(
                    iCLK,               // LCD display clock
                    iRST_n,          // systen reset
                    iNote,

                    oHD,              // LCD Horizontal sync
                    oVD,              // LCD Vertical sync
                    oDEN,             // LCD Data Enable
                    oLCD_R,             // LCD Red color data
                    oLCD_G,               // LCD Green color data
                    oLCD_B,               // LCD Blue color data
                    );
//=======================================================
```

Fig. 3 Port definition of LCD display module

When each line scan begins, the line synchronizing signal oHD is set to low level signal for the first 30 clock cycles, which shows in Fig. 4 with *thpw*. And then returns high level signal, the appearance of each low level represents the start of every line scan.

The LCD display has a resolution of 800×480, which means each line has 800 pixels to display, and a field includes 480 lines of pixels. Data can be found in Table I that, the time required to scan each line is 1056 clock cycles, and each line of the display contains a horizontal blanking signal and the active display signal, of which the trailing edge time *tnb* is about 16 clock cycles, and *thpw* for 46 clock cycles, cutting front edge time *thfp* for 210 clock cycles, effective line show time *thd* for about 800 clock cycles.

Field synchronizing signal is similar as line synchronizing signal, the length of oVD signal is 13 low level signals *tvpw* of line cycle, which represents the start of every field scan. Then return to high level signal, which we can see from Table I that every field contains 525 line cycles, of which the trailing edge time *tvbp* is 10 line cycles, and *tvpw* for 23 line cycles, *tvfp* for 22 line cycles, effective field display time is 480 line cycles which can be seen in Fig. 5.

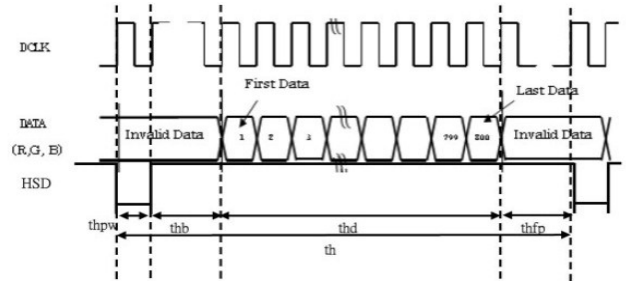

Fig. 4 Line sync signal and RGB data

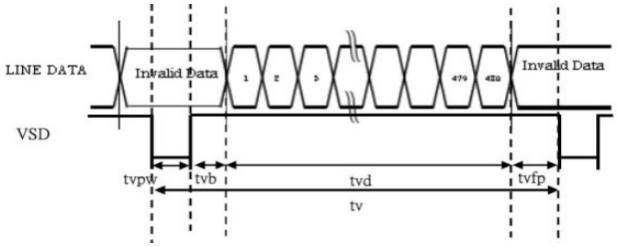

Fig. 5 Field sync signal and RGB data

TABLE I

REFERENCES FOR LCD DISPLAY

ITEM		SYMBOL	MIN.	TYP.	MAX.	UNIT	NOTE
	Dot Clock	1/tCLK		33		MHZ	
DCLK	DCLK pulse duty	Tcwh	40	50	60	%	
	Setup time	Tesu	8			ns	
	Hold time	Tehd	8			ns	
	Horizontal period	tH		1056		tCLK	
	Horizontal Valid	tHA		800		tCLK	
	Horizontal Blank	tHB		256		tCLK	
	Vertical Period	tV		525		tH	
	Vertical Valid	tVA		480		tH	
DE	Vertical Blank	tVB		45		tH	
SYNC	HSYNC setup time	Thst	8			ns	
	HSYNC hold time	Thhd	8			ns	
	VSYNC Setup Time	Tvst	8			ns	
	VSYNC Hold Time	Tvhd	8			ns	
	Horizontal Period	th		1056		tCLK	
	Horizontal Pulse Width	thpw		30		tCLK	thb+thpw=46DCLK
	Horizontal Back Porch	thb		16		tCLK	is fixed
	Horizontal Front Porch	thfp		210		tCLK	
	Horizontal Valid	thd		800		tCLK	
	Vertical Period	tv		525		th	
	Vertical Pulse Width	tvpw		13		th	tvpw + tvb = 23th
	Vertical Back Porch	tvb		10		th	is fixed
	Vertical Front Porch	tvfp		22		th	
	Vertical Valid	tvd		480		th	
	Setup time	Tdsu	8			ns	
DATA	Hold time	Tdsu	8			ns	

After synthesizing on Quartus II, the RTL view is as below:

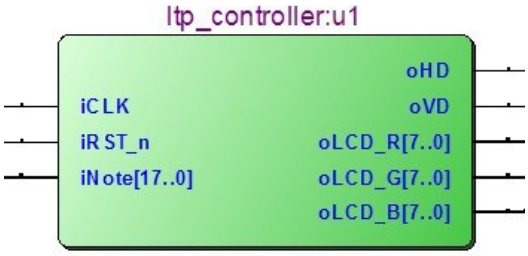

Fig. 6 RTL view for LCD display module

2) Piano (Electronic Organ) Display Module

Piano keys are composed of white keys and black keys. This module will control pixels in sections to display eight white keys and six black keys in Verilog HDL. The white are 1, 2, 3, 4, 5, 6, 7, +1. The black are #1, #2, #4, #5, #6, #+1, represent the half-steps.

When white keys are pressed, the pressed key section will be displayed grey shadows, and when black keys are pressed, the edge of the key shadow will fade away.

After synthesizing on Quartus II, the RTL view is as below:

Fig. 7 RTL view for Piano display module

3) Guitar Display Module

Different from playing piano, the structure and operation for guitar is more complex.

It requires both hands simultaneously press and toggle the strings so that we can change the pitch. Considering the particularity, we dislodged the grade part, making the constant chord name replaced.

This module will display six strings and eight chords. It can be able to choose chords, and touch corresponding strings with the correct pitch to play.

The image of this module is transferred mainly by master interface, with coding C language in Nios II, to set up register to controller. LTM needs to ask SDRAM, where image stored, for data with a constant frequency 33MHz. LTM controller sustains master interface, so that LTM controller can initiatively read SDRAM without Nios II CPU involved.

After synthesizing on Quartus II, the RTL view is as below:

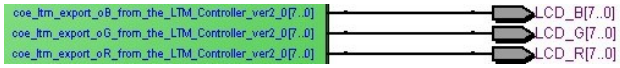

Fig. 8 RTL view for Guitar display module

B. Real-Time Touch Control Module

1) Multi-Touch Module

Multi-touch module contains simple gestures control and touch input, using DE2-115 general input and output (General Purpose Input / Output, GPIO) to interconnect. We utilize IP, for user's input, provided by Terasic to design the module, and connect it to Avalon bus through SOPC Builder, processing data with Nios CPU.

After synthesizing on Quartus II, the RTL view is as below:

Fig. 9 RTL view for Multi-touch module

2) Touch Block

Take piano module as an example, we split the screen into blocks with keys; calculate for their edge location and their (x, y); set the scan area in accordance with the corresponding coordinate length.

If the position pressed belongs to the exact block, then use the statement "if/ else" to switch to a certain signal, which is connected to the corresponding output signal that represents the frequency of the key in order to achieve a total of 14 keys letting out sounds independently.

C . Sound Synthesize

1) Digital Music Theory

Digital music, using dots, lines and eight Arabic numerals 0-7, constituting a variety of notes, are the basic elements of music tracks. Here we use this notation to illustrate the way how the synthesizer functions.

Combined according to certain relations, these notes constitute a melody. Notes 1-7 represent seven different pitches in an octave, do, re, mi, fa, so, la, ti. The dot over these seven musical alphabet means it is 8 degrees higher than the pitch, if below, means 8 degrees lower than the pitch. Arabic numeral 0 means rest, which means to stop pronunciation. Lines below numeral indicate the duration of the sound; numerals without lines behind are called a crotchet, whose sound length is the basic unit of measure. Every one increase of the segments is equal to a decrease of playing time. As segments after numeral increase, sound length increase[7]. Table II lists out correspond relations for different notes.

TABLE II

RELATIONSHIP BETWEEN NOTES AND BEATS

Name	1	1/2	1/4	1/8	1/16	1/32
Note	1---	1-	1	1	1	1
Beat	4	2	1	1/2	1/4	1/8

TABLE III

RELATIONSHIP BETWEEN NOTES AND FREQUENCY

Note	1	#1	2	#2	3	4	#4
f/Hz	523. 3	554. 4	587. 3	622. 3	659. 3	698. 5	740. 1
Note	5	#5	6	#6	7	1	#1
f/Hz	784. 1	830. 7	880. 1	932. 4	987. 8	1046. 5	1108. 8

Sound length is a relative concept of time, the length of time is not restricted, may be one second or two. If a quarter-note is recognized as a second, then play time for other notes should refer to it as the basic unit of measure. A half beat is 0.5 second.

The Twelve-tone Equal Temperament stipulates that every two octaves (such as a midrange 1 and treble 1) have a frequency difference between them. And every two half-note have a frequency different rate of 1.059. In addition, the pitch A (6)'s frequency is 440Hz, between 3, 4 and 7, 1, there is semitones; the rest are whole tones. Therefore can we calculate the frequency from 1 to treble 1, as what is indicated in Table III.

2) Pitch, Timbre and Loudness

The pitch of a sound is also known as its frequency. When the frequency is high, the wavelength of the sound is shorter. According to the experimental data from Table III, we can know that the higher the frequency, the higher the pitch.

Loudness is the characteristic of a sound that is primarily a psychological correlate of physical strength (amplitude). The greater the amplitude is, the greater the energy, the greater the loudness.

Timbre is the characteristic quality of a sound, independent of pitch and loudness, from which its source or manner of production can be inferred. Timbre depends on the relative strengths of the components of different frequencies, which are determined by resonance.

To be simple, when we play the same pitch on piano and guitar, they are not sound alike.

Among these three decisive characteristic, timbre is the most difficult one to simulate, for pitch and loudness can be expressed by digital number, but timbre is decided by the entire wave form. Therefore, timbre is a comprehensive response to all kinds of frequency and intensity.

The reason why different instruments sound ever-changing, we can draw a conclusion through studying their waves:

- Harmonic wave spectrum of the sound wave and envelope shape determine the timbre of specific instrument.
- Overtones generated by tiny vibrations of harmonic waves are decisive to timbre ineligibly.
- Humans can hear a range of sound between 20 and 20,000 Hz, while musical instruments' overtone frequency is far beyond the scope of this.
- What we hear most clearly is the basic frequency of the wave, which represents the pitch.
- The key to simulate timbre is to simulate harmonic waves and overtones.

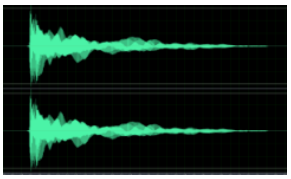

Fig. 10

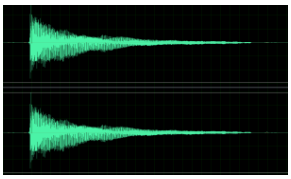

Fig. 11

Fig. 10 shows the image of the vibration waveform for piano, pitch 1. Fig. 11 shows the same pitch for guitar. Contrast these two pictures can we find that when the amplitude of the waveforms are substantially the same pitch, the structure of their envelope differs a lot.

As for guitar is a string plucked instrument, we play with our fingers plucking the strings; piano is a string hit instrument. So the sound of piano is softer, decays faster, which can be seen from the waveform.

3) Karplus-Strong algorithm

Karplus-Strong algorithm is simulation behavior through simulating sine wave attenuation. It is constituted by a delay line of a short length, which would be decided by the frequency generated. Its waveform changes all the time in cycle, and the delay line after filtering simulates the hit or plucked string.

To achieve the algorithm, the main components are: delay line, percussion, note attenuation and feedback.

The length of the delay line depends on the expected frequency of the note generated. The fundamental frequency of the signal, which is the non-zero lowest resonant frequency, whose phase delay is -2π.

Relative to a given fundamental frequency F0 of the required delay D can be calculated by the following formula:

$$D = F_s/F_0$$

Fs is sample rate. In this design, the sample rate is set as 48KHz. The reason why we adopt the high sample rate is that research shows signals decay faster with short delay line. Choosing higher sample rate allows to use longer delay line, so that it can achieve a better attenuation characteristics[8]. For instance, the central C (seen from Table III), whose frequency is 523.3Hz, has a delay line length of about:

$$D = 48000/523.3 \approx 93$$

Karplus-Strong algorithm defines a low-pass filtered random noise signal as an initial strike chord signal, similar to the sound generated randomly from slightly plucked guitar strings. And the real piano uses the hammering way to generate sound, so the hit is softer and smoother. To simulate this situation, hitting triangular waveform is defined as the pulse waveform, rather than random strike. Therefore, the audio output produced by this hammer way is smoother than that plucked guitar sound.

In addition, each of the real piano's notes is generated not only by a simple string. Usually two are needed, and slightly difference in length between them. In order to obtain better

simulation results, the two strings are coupled together, and fed back after averaging the output.

D. Instruments Switch

1) Initialize Selection Menu

As shown in Fig. 2, the root directory of selection mode is a sample of DE2-115 demo board from ALTERA. Application Selector interface shows when SD card inserted. We now modified the sample, storing some user-defined modules in SD card.

2) Instrument Modules Implantation

The SD card provided by Terasic have a memory of 2GB. If you want to read the modules of different musical instruments, the program needs to be converted to a binary BIN file and stored in the SD card.

While the program is divided into two operative parts, the software part and the hardware part. It needs to turn compiled SOF file and ELF file to BIN file into the SD card. The conversion process requires the Nios II Command Shell:

Turn SOF to BIN

```
sof2flash --epcs --compress --input=hardware.sof --output=hardware.flash
nios2-elf-objcopy -I srec -O binary hardware.flash hardware.bin
```

Turn ELF to BIN

```
Elf2flash --epcs --after=hw.flash --input=sw.elf --output=sw.flash
nios2-elf-objcopy -I srec -O binary sw.flash sw.bin
```

Turn ELF to BIN within one step

```
nios2-elf-objcopy -O binary test_sw.elf test_sw.bin
```

Put hardware bin and software bin into a folder, storing in the root of SD card. When getting power, selection interface appears as Fig. 12. Choose PIANO and click LOAD to get into piano personal playing mode.

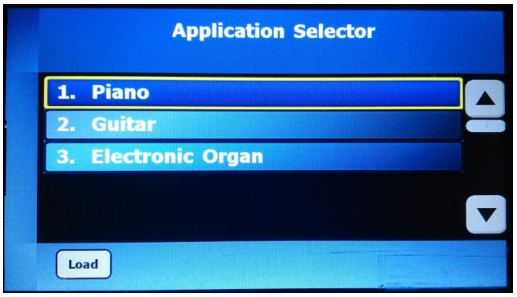

Fig. 11 Instrument Switch Mode

E. Playing System

Playing system includes sample tracks auto playing mode and personal play mode. When switch SW[1] is set to 1, the system turns out to be sample tracks auto playing mode; when SW[1] is set to 0, turns personal playing mode. LCD display control module controls the line sync signal and vertical sync signal and RGB signals. Playing on the screen, the LCD brings the touch signal to processor via the touch control module, and shows the effect of multi-touch. After obtaining the touch signals, transfer them to the generation module, and make corresponding key note data, ultimately through the audio codec module to broadcast sounds.

1) WM8731 Audio Codec Chip

The audio control chip is provided by Wolfson, low-power, 24-bit stereo audio codec chip WM8731[5].

We turned the demo WMA file into WAV file with required parameters under the operating environment of Cool Edit. As for WAV file is consist of sample values and file headers, usually representing sound with quantization bits, sample rate and amplitude of sample points. We adopt the standard formatting sample frequency of 44.1KHz, 16-bit quantized digital stereo.

WAV file will be converted into SD card, and then read the data out from the SD card into DAC.

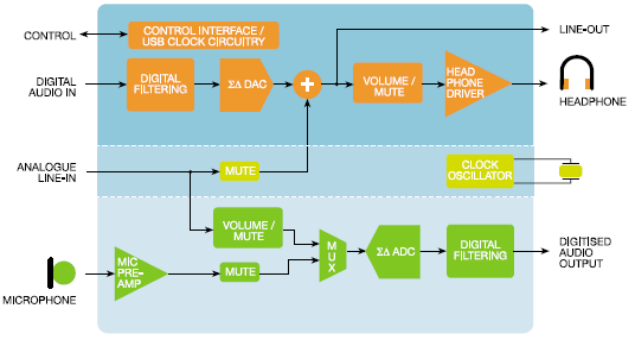

Fig. 12 Inner Structure of WM8731

It can be seen from Fig. 12 that the digital audio input enters the digital filter first, and then processed by the DAC, forming 24-bit files, with little distortion, then put out through the headphone driver.

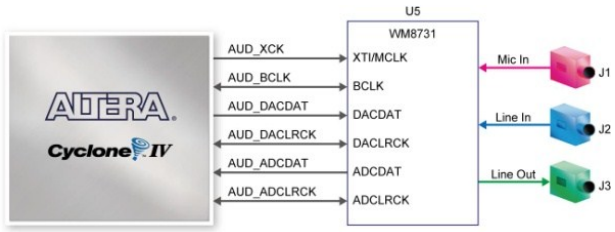

Fig. 13 Connection between FPGA and WM8731

Fig. 13 shows the connection diagrams between the WM8731 audio chip and FPGA. To control WM8731, the I2C bus is needed. The module's ports definitions are as below:

```
//=============================================================
module I2C_Controller (
   CLOCK,
   I2C_SCLK,//I2C CLOCK
   I2C_SDAT,//I2C DATA
   I2C_DATA,//DATA:[SLAVE_ADDR,SUB_ADDR,DATA]
   GO,      //GO transfor
   END,     //END transfor
   W_R,     //W_R
   ACK,     //ACK
   RESET,
   //TEST
   SD_COUNTER,
   SDO
);
//=============================================================
```

Fig. 14 Audio I2C Module Port Definitions

In this case, I2C controller uses 33 clock cycles to transfer a 24-bit data. The first clock cycle to initialize the controller, the second and third start transmission, 4th-30th cycle transfer data (which contains 24-bit data and 3 ACK), the last three cycles are used to stop transmission. Controller uses a 6-bit counter SD_COUNTER to transfer cycle count.

Before starting the transfer and after the transfer, I2C_SCLK signal should remain high, the start condition (START) and termination condition (STOP) being completed with I2C_SCLK and I2C_SDAT. There is a register SCLK added to the code to generate START and STOP conditions. Before START and after STOP, SCLK=0. During data transmission, I2C_SCLK is provided by the CLOCK.
After synthesizing on Quartus II, the RTL view is as below:

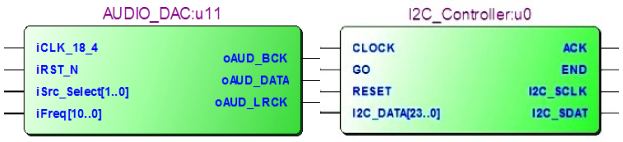

Fig. 15 Fig. 16

III. Integrated System Design

Fig. 17 illustrates the framework of the overall system design diagram. The system includes the following eight sections:

- Touchable LCD display
- SDRAM/Flash memory, for program storage and operation
- Parallel input and output ports (PIO), mainly for testing and communication with the Nios II processor. Our design uses the 18 switches, 4 buttons, 18 red LEDs on DE2-115 demo board.
- Sounds generation module
- Audio codec control, for play corresponding audios through LINEOUT connected to headphones or loudspeakers.
- Phase-locked loop (PLL), DE2-115 on 50MHz clock as input, putting out other clock of different frequency as the input of other modules. Among, 33MHz as LCD display clock, 100MHz as the input clock for SDRAM, 18.432MHz as the input clock to audio codec module.
- Nios II processor, as the pivot of the whole system, complishes the core function, including the touch screen and data processing.
- JTAG UART, mainly for observation to the results, and debugging the program.

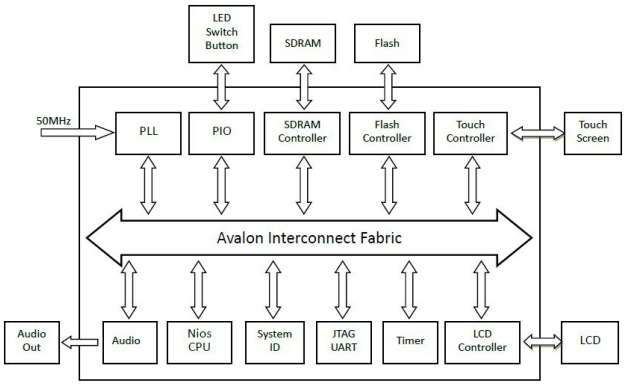

Fig. 17 Framework of overall system

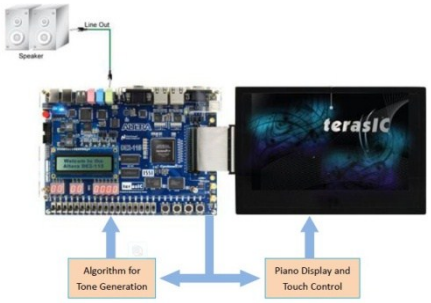

Fig. 18 System working environment

Fig. 18 shows the overall system operating environment, the DE2-115 demo board with LCD touch screen and loudspeaker connected, Quartus II and Nios II IDE programmed, then we can achieve a multi-instrument playing system including piano, electronic organ, guitar and Jazz drums.

A. System Hardware Design

Hardware part mainly controls two modules : LCD display and audio output.

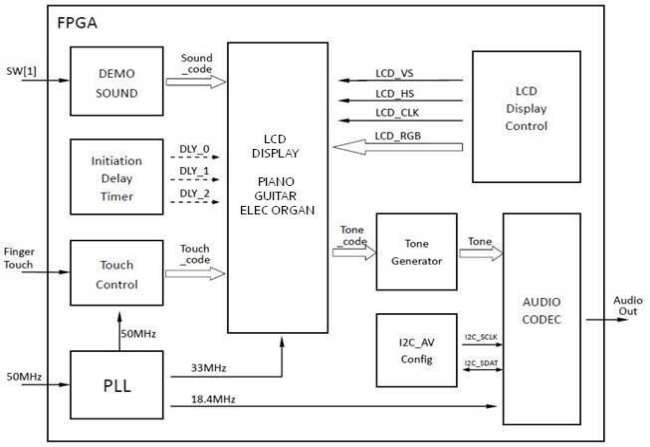

Fig. 19 Hardware system and data flow

In electronic organ module,we use Verilog HDL to describe the interface and control relative areas to drive pitch frequency signal.

In piano, guitar, and drum module, we use SDRAM to read the BIN file of the image, and call for SD card to play audio module with Nios II.

As Fig. 19 shows the output from the PLL clock inputs to the different frequencies in each module. LCD display module outputs scan signal to the screen, and outputs the audio signal to audio module through display module.

B. System Software Design

The design in SOPC Builder mainly contains the following components:

- Nios II processor, optional full-featured CPU cores (Nios II/f), with optimal performance;
- SDRAM, CFI Flash, EPCS memory devices;
- PIO, including the buttons, switches and LED lights;
- PLL, to generate different frequency clock as the input clock of different modules;
- System ID, provide a unique identifier for the SOPC Builder system ;
- I2C module, generate I2C_SCLK, I2C_SDAT signal;
- Touch module, access to the touch point position and other data;
- LCD module, can display characters;
- JTAG UART module, to facilitate the observation results of the proceedings and functional debugging.

After finishing adding the components, click the Generate button to build SOPC system, generate .sopcinfo files. Then, Nios II IDE will automatically generate HAL (Hardware Abstraction Layer) system libraries. HAL system library is a simplified operating environment, providing simple device driver interface for underlying hardware communication program. HAL application programming interface (API) and ANSIC standard library are integrated. Through HAL API, we can use the familiar C language function library to access devices and files.

HAL system library provides the following services:

- Integrated with the new ANSIC standard library, providing familiar C standard function library;
- Device drivers, providing access to each device in the system;
- HAL API, providing a consistent and standard interface;
- System initialization, initializing processor and operating environment before the main ();
- Device initialization, initializing each device in system before the main ();

System.h file, providing software description for hardware (Nios II), is the basis for the HAL system library. The document describes every peripheral devices in the system, and the details are as follow:

- Peripheral hardware configuration;
- Base address;
- IRQ priority;
- Peripherals symbolic name[9];

HAL system library as the underlying, based on the principle of each module and working agreement, creates their own drivers, including I2C, Multi-touch, LCD driver modules and application functions, and then develope through main () function. The software system structure is shown in Fig. 20.

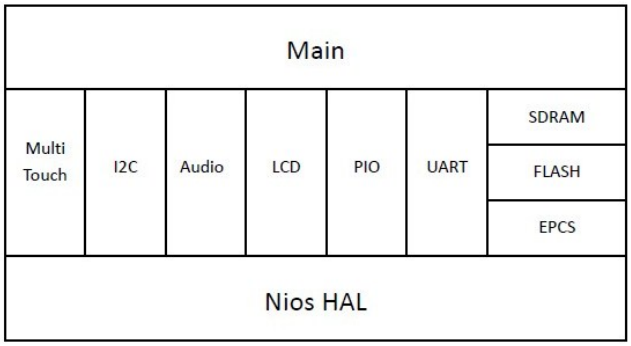

Fig. 20 Structure of software system

IV. Results and Show

The physical platform for the design is shown in Fig. 21. Firstly, run Quartus II, analysis and synthesis, after showing the interface, run Nios II Eclipse/IDE to get software started, thus, we can achieve the whole program with controllably operating the system.

Fig. 21 Physical System Platform

A. Electronic Organ Playing System

When SD card inserted and DE2-115 be turned on, choose mode “Electronic Organ”, and click LOAD, then we will come to the electronic organ interface, as Fig. 22 shows.

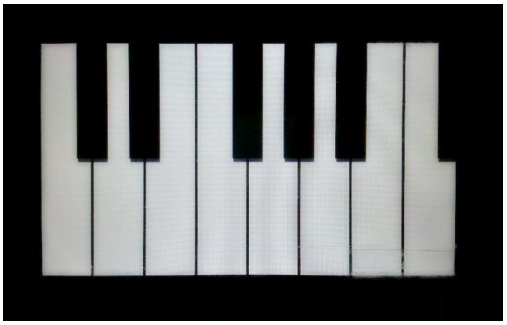

Fig. 22 Electronic Organ Playing Interface

When the key is not pressed, the shadow of the black key at edge remains, represents the 3D effect, and the white key stays white. The keyboard has an octave, a total of 14 key tones in all.

When the white key is pressed, it is displayed as shades of gray, which is showed in Fig. 23;when the black key is pressed, it is displayed as shadow edge disappearing, which is showed in Fig. 24. And the corresponding LED lights up. When the switch SW[1] turns to automatically play mode, it begins to play demo sound, and the keys will change their color in time.

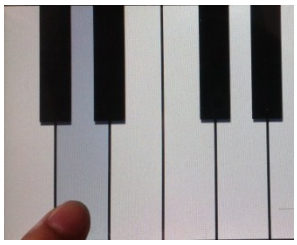

Fig. 23

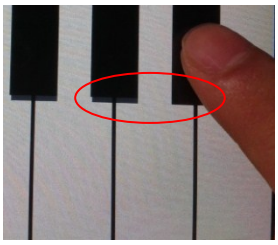

Fig. 24

B. Piano Playing System

The piano module is quite similar to electronic organ's, for their interface display is exactly the same. What makes them differ from each other is that the way to generate their waves is of no similarity.

The principle of piano playing calls for complex echos and harmonic waves, which contributes a lot to its unique timbre. At the top right corner, there is a semi-transparent section indicating the interface of Jazz drum. When press the section, the interface will switch into Jazz drum playing mode.

Fig. 25 Piano Playing Interface

C. Guitar Playing System

The guitar interface consists of 8 chords and 6 strings, and we turned these function objects into touch area on LCD screen.

Whenever choosing a chord, the strings will be prepared, and make specific sounds according to the chord. Therefore, we can achieve a 8-chord guitar playing system.

At the bottom right corner, there is a semi-transparent section indicating the switch to piano interface.

Fig. 26 Guitar Playing Interface

D. Jazz Drum Playing System

The Jazz Drum Playing system has 4 drums and 4 cymbals and a bass drum. The way it works is just the same as piano and guitar. When the semi-transparent guitar at the bottom left corner pressed, the system will change to the guitar playing mode.

Fig. 27 Jazz Drum Playing Interface

E. System Resource Utilization

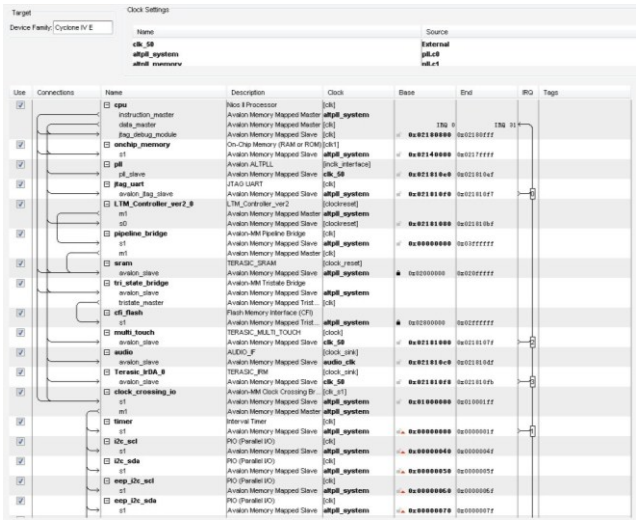

Fig. 28 SOPC Components and Connections

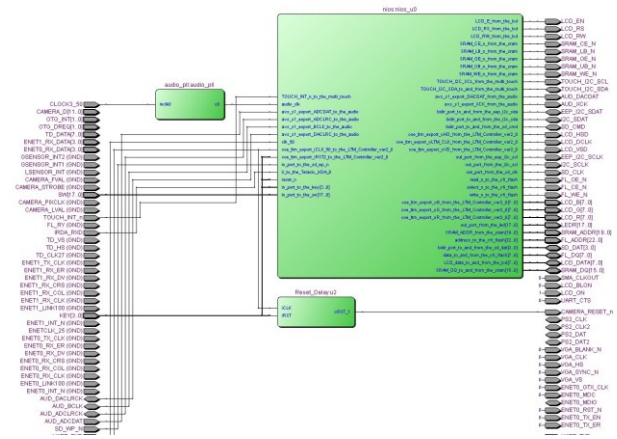

Fig. 29 RTL View fot the Entire System

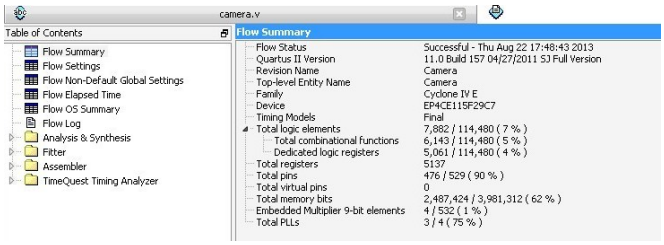

Fig. 30 System Resource Utilization

V. CONCLUSIONS

Hardware design completed on Quartus II 11.0 and software design developed on Nios II 11.0, combining the two ALTERA development platform, we have build a system using SOPC on DE2-115 demo board, and accomplished a real-time controllable and multi-instrument switchable music synthesizer.

The proposed simulation algorithm sounds, compared with the real sounds, still needs to be improved. It is unrealistic to make the simulate same as the real instrument. The intensity of the strength, plucked location and the surrounding environment such as temperature and humidity will affect subtle changes to the sound, and the data simulated will also be huge. The further improvement for sound envelope algorithm will help make progress to the efficiency as simulating.

Sufficient hardware resources on the platform provide great convenience for further development. This design can be developed in many recreational aspects, such as adding some game elements and cool effects, making it more interesting, more attractive, and more valuable and competitive in current electronic markets.

REFERENCES

[1] Bao-zong Yuan，Qiu-qi Ruan，Yan-jiang Wang，etc．Conceptual Framework Features and Key Technologies for a new generation(the 4th generation) of HCI[J]．Electronic Journal，2003，31(12A)：1945-1954．

[2] Altera Corporation. DE2-115 User Manual[EB/OL]. Available: http://www.altera.com/literature/ lit-cyclone-iv.jsp. 2010.1.

[3] Altera Corporation. Video and Embedded Evaluation Kit – Multi-touch User Manual[EB/OL]. Available: http://www.altera.com/literature/. 2010.1.

[4] MULTEK Corporation. LC-CPT 7inch PS-9928-01 Rev01[M/CD]. 2011, 1.

[5] Wolfson Corporation. WM8731/WM8731L[M/CD]. 2009, 4.

[6] WM8731EDS portable digital audio solutions[EB/OL]. Available: www.wolfsonmicro.com

[7] Pei-yuan Guo, Mei-hua Qiao. Music Storage and Playback System Based on FPGA[J]. Beijing Technology and Business University Academia Journal (Natural Science Edition),2004,(06):18-22,26

[8] Jaffe D A, Smith J O. Extensions of the Karplus-Strong plucked-string algorithm[J]. Computer Music Journal, 1983, 7(2): 56-69.

[9] Lan-ying Li. SOPC Design Principles and Application for Embedded Soft-cored Nios [M]. Beijing: Beijing Aerospace University Press, 2006, 11.