Third Prize

Nios II-Based Audio-Controlled Digital Oscillograph

Institution: **Xian Jiao Tong University**

Participants: **Wan Liang, Zhang Weile, and Wang Wei**

Instructor: **Penghui Zhang**

Design Introduction

The oscillograph is a common instrument that plays a key role in many experiments. Because of its design structure, the oscillograph cannot work in some environments. For example, it may not work when a waveform must go through projector for a demonstration or if the waveform needs to be adjusted while the user is performing other operations.

With these issues in mind, we designed an audio-controlled digital oscillograph that uses an audio recognition algorithm to control the oscillograph; specifically, it controls the waveform display size attenuation and amplification as well as the scanning speed. The oscillograph display uses a standard VGA interface that can display waveforms on a PC monitor, projector, or other standard output device. Additionally, the system leverages the advantage of FPGAs for digital signal processing (DSP) and hence, for computing the period, valid value, and average value for the input signal and fast Fourier transform (FFT) algorithm-based frequency spectrum analysis.

The whole system is designed with Nios® II-based system-on-a-programmable-chip (SOPC) technology. The design has a front-end acquisition circuit on the Altera Development and Education (DE2) development board. After the acquisition circuit collects analog signals, it outputs the signals to the FPGA for processing. The corresponding output is generated according to the control information.

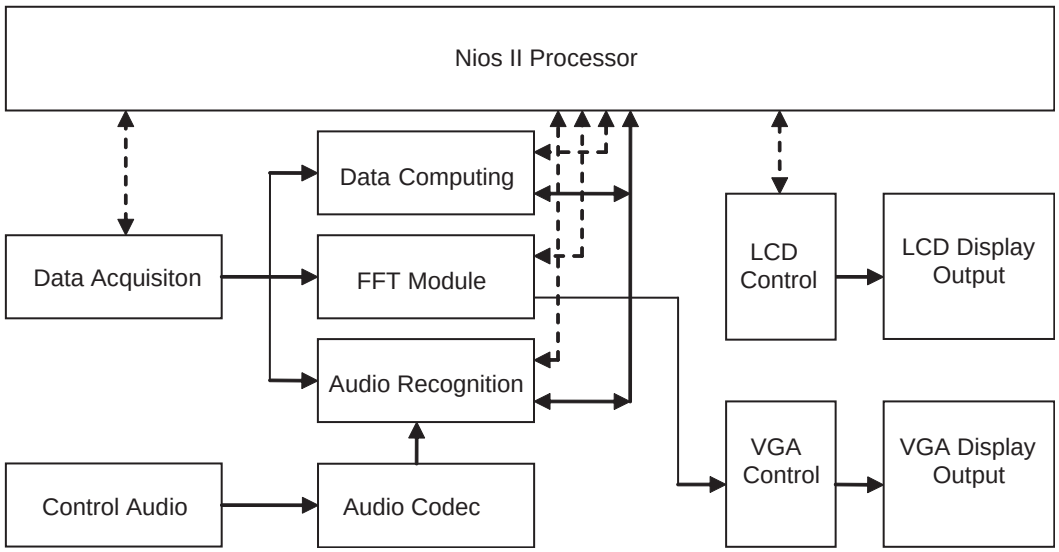
We use SOPC technology for the design and we integrate the Nios II processor into the FPGA. Our solution implements a system-on-chip (SOC) solution, which reduces the system size and overall power consumption. Additionally, we can change the system model and easily make upgrades because the main structures of the system are all implemented within the FPGA. This implementation makes the design universal and prolongs the system life. The system requires high-speed data processing such as a fast FFT function and real-time hardware implementation of the audio recognition algorithm, which we could not solve using a traditional microcontroller (MCU). With an FPGA solution, we could customize Avalon® bus-based peripherals via custom instructions and implement them with a high-

speed FPGA and corresponding development environment, significantly improving the system's DSP capability.

Function Description

The whole system is composed of the front-end acquisition circuit and DE2 development board. The front-end acquisition circuit collects analog signals, and then processes and displays them using the FPGA on the board. The system contains control modules for the Nios II processor, audio recognition, data computing, FFT, and display. Figure 1 shows the overall system structure.

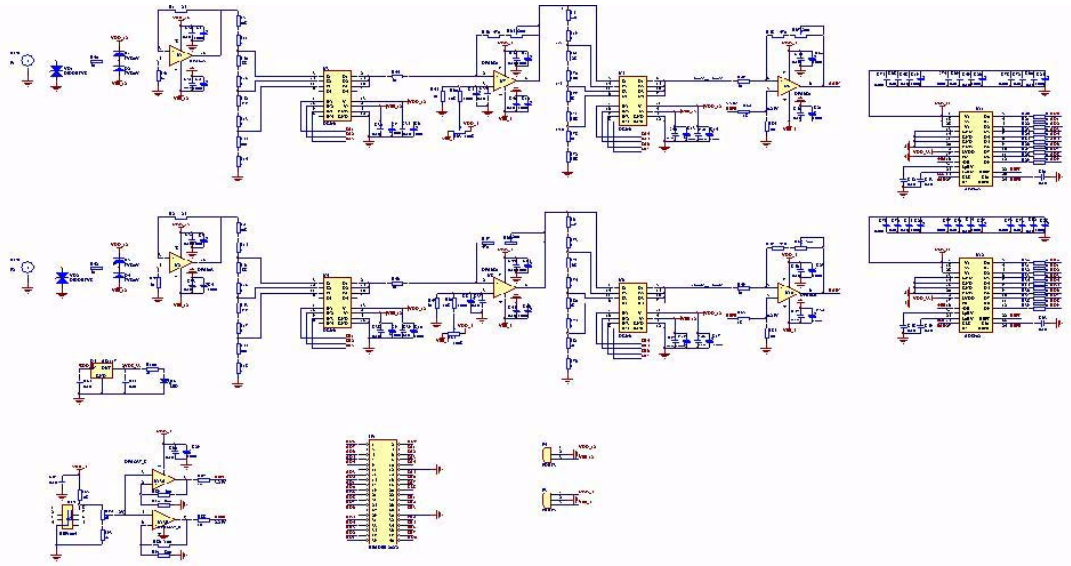
Figure 1. Overall System Structure



Front-End Acquisition Circuit

The data acquisition circuit is composed of attenuation control and the amplified circuit with a fixed multiple. Figure 2 shows the circuit design. The first-level operational amplifier is the voltage follower that provides high-input impedance. The following output connects to a fixed-multiple amplifier and then connects with a resistor ladder that varies the attenuation signal. The next switch selects the signal with different attenuation. The selected signal is sent to the analog-to-digital converter (ADC) for quantitative output after amplification and attenuation by the same fixed multiple.

Figure 2. Front-End Acquisition Circuit



Audio Control Module

The audio control function implements an audio recognition algorithm and collects external analog audio signals through a microphone. It transforms the audio signal using the WM8731 audio encoding/decoding device on the DE2 development board, and sends the collected digital signal to the audio recognition module for recognition. If it is recognized as a control command, the system automatically implements the corresponding operations; otherwise, the system does not respond.

FFT Module

The FFT module changes the input signal in real time, and can synchronously display the spectrum map on the output terminal. The general FFT algorithm has a butterfly operation structure; the input is ordered in code-reversed sequence and the output is ordered in natural sequence. The operation structure is clear and simple. However, in this structure, each butterfly output data is still saved in the data storage unit of the original input. The input data addressing in the same location on different levels is not fixed; therefore, it is hard to implement cycle control. Additionally, it is hard to operate the function in parallel when using an FPGA.

In this design, we use a fixed geometric structure FFT operation. The operational structure is quite similar to the general butterfly structure, but it has some differences in the data and operation sequence. After modification, there is no change in the sequence of input and output data in each level; in this way, the geometric structure in each level can be fixed. With this structure, the design can perform cycle addressing, facilitate programming with the FPGA, and implement an internal parallel FFT hardware structure that reduces the FFT operation speed dramatically. The “Design Architecture” section describes the fixed geometrical structure FFT in more detail.

Data Computing Module

The general digital oscillograph can compute the signal’s real-time period, average value, and peak-to-peak value. The FPGA provides speed advantages for DSP applications, making real-time computing of these parameters easy.

VGA Display Module

Cathode ray tube (CRT) monitors have three colors: RGB. The color is created by shooting the CRT-generated focusing electrode (Eb) on the screen. The scan begins from the top left of the screen to the

right and from the top to the bottom. When a line finishes scanning, the Eb goes back to the starting point of the next line on the left of the screen. The CRT masks the Eb during this period and prepares for the next scan. Therefore, a general VGA monitor has 5 signals:

- *R, G, and B*—The 3 basic color signals (10 digits).
- *HS*—Horizontal line synchronized signal.
- *VS*—Vertical field synchronized signal.

The monitor must strictly abide by the VGA industry standard, i.e., 640 x 480 x 60 Hz. The standard's corresponding frequency is:

- *Clock frequency*—25.175 MHz
- *Line frequency*—31,469 Hz
- *Field frequency*—59.94 Hz

The key to implement VGA display is to provide the clock signals, which drive the monitor's line and field scanning.

The DE2 development board contains the ADV7123 control chip, which can be used for VGA display control. The chip implements the time sequence of all standard signals required for VGA display, masks the bottom-level structure, and enables the user to implement the VGA display easily. Refer to the "Design Architecture" section for more details.

LCD Display Module

The VGA display module displays waveforms and the graphical user interface (GUI). It also displays simple system information that is not convenient to display on the oscillograph, such as initialization and system status settings. We display this information on the DE2 board's LCD monitor. The Nios II hardware abstraction layer (HAL) provides a convenient interface for using the LCD: the designer can use it by simply calling the interface function.

Performance Parameters

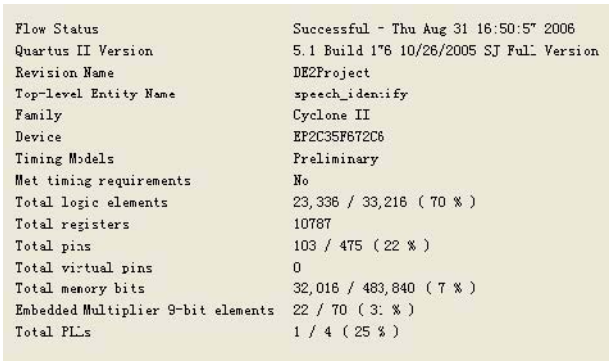
Figure 3 shows the resource usage for the VGA oscillograph module.

Figure 3. VGA Oscillograph Module Resource Usage

Flow Status	Successful - Thu Aug 31 12:39:01 2006
Quartus II Version	5.1 Build 176 10/26/2005 SJ Full Version
Revision Name	DE2Project
Top-level Entity Name	DE2Project
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Preliminary
Met timing requirements	No
Total logic elements	14,417 / 33,216 (43 %)
Total registers	6425
Total pins	228 / 475 (48 %)
Total virtual pins	0
Total memory bits	407,192 / 483,840 (84 %)
Embedded Multiplier 9-bit elements	42 / 70 (60 %)
Total PLLs	2 / 4 (50 %)

Figure 4 shows the resource usage for the audio recognition module.

Figure 4. Audio Recognition Module Resource Usage



Flow Status	Successful - Thu Aug 31 16:50:57 2006
Quartus II Version	5.1 Build 176 10/26/2005 SJ Full Version
Revision Name	DE2Project
Top-level Entity Name	speech_identify
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Preliminary
Met timing requirements	No
Total logic elements	23,336 / 33,216 (70 %)
Total registers	10787
Total pins	103 / 475 (22 %)
Total virtual pins	0
Total memory bits	32,016 / 483,840 (7 %)
Embedded Multiplier 9-bit elements	22 / 70 (31 %)
Total PLLs	1 / 4 (25 %)

The design has the following system parameters:

- *CPU parameter*—The system uses a 50-MHz Nios II clock frequency.
- *Sampling circuit*—The sampling frequency of the front-end analog signal is 25 MHz. The bandwidth is designed as 1 MHz, and the gear is designed as x100, x50, x20, x10, x5, x2, x1, x1/2, x1/5, and x1/10, or 10 gears in all.
- *VGA display module*—We use the VGA industry standard with a data refresh rate of 4 Hz.
- *FFT operation module*—We use a fixed geometric structure algorithm and pipelined output. One butterfly operation is completed every 8 clock cycles, and the 512-point FFT requires about 370 µs.
- *Audio recognition module*—We use the Durbin algorithm with a recognition rate of more than 90%.
- Input signal parameters:
  - *Analog signal input extent*—±5 V
  - *Bandwidth*—300 KHz
  - *Analog signal sampling frequency*—25 MHz

Design Architecture

This section describes the architecture of our design.

Sampling Circuit Design

To process and display the analog signal, the system must first convert the signal from analog to digital using the front-end sampling circuit. The system design displays general waveforms with 1-MHz bandwidth and an amplitude of less than ±2 V, such as sine waves, square waves, and sawtooth waves. According to the Nyquist sampling theorem, the sampling frequency must be greater than 2 MHz to convert the 1-MHz signal without distortion. For non-sine waves, the sampling frequency is 30 MHz. In this case, a signal within the bandwidth could be created with harmonic waves more than 30 times, resulting in a better waveform.

Figures 5 and 6 shows the sampling circuit schematic diagrams.

Figure 5. Sampling Circuit

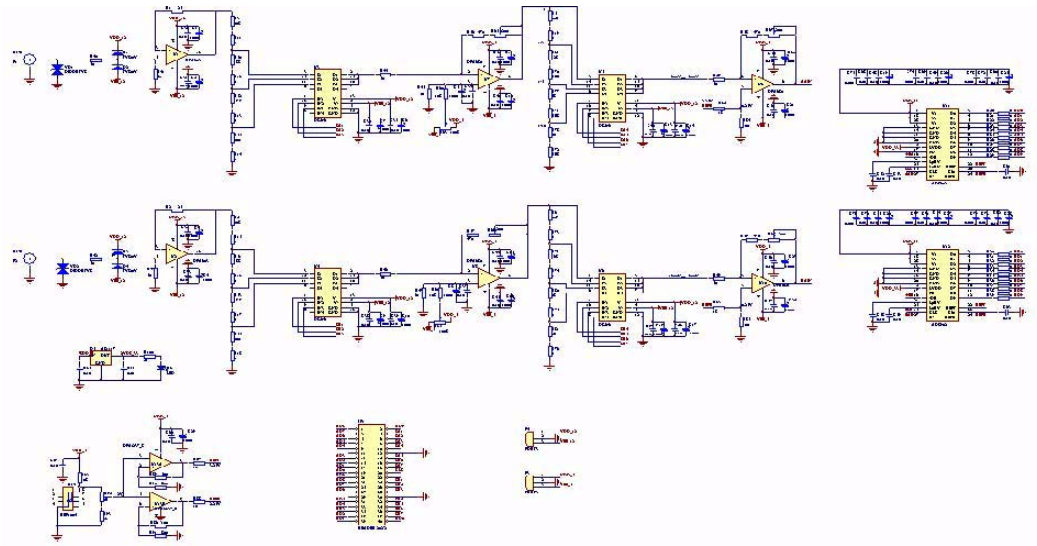
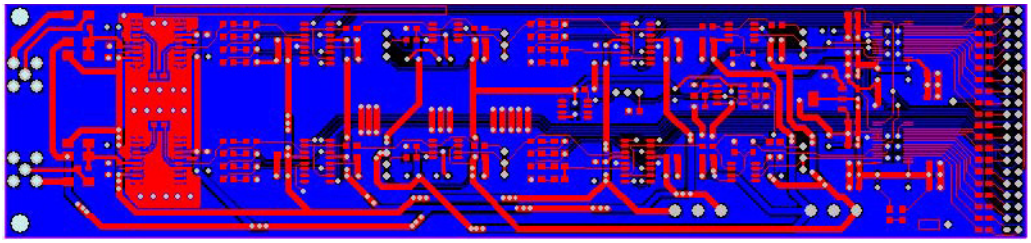


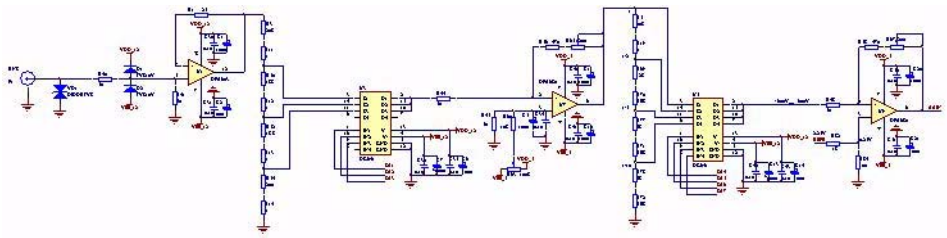
Figure 6. Sampling Circuit Design Layout



Forestage Protection, Amplification, and Shift Circuit

High-speed A/D conversion has strict requirements for the level and quantification ranges. Therefore, we need to do some processing on the signal before A/D conversion, including limited amplitude protection, variable attenuation control, amplification, and level conversion. Figure 7 shows the forestage processing circuit.

Figure 7. Sampling Forestage Amplification and Shift Circuit



The SMA connector inputs the analog signal and provides limited amplitude protection for positive and negative power supplies through a ground TVS pipe and two Schottky diodes. A 1-MΩ ground resistor provides high-input impedance. The operational amplifier is the OPA603 device. The device supports

a 100-MHz bandwidth gain to ten times the gain in the ±9 V input amplitude, which generates the oscillation's large input amplitude. The OPA603 signal output is connected to the ground with the resistor ladder to obtain different attenuation signal outputs by shift tapping. The system uses three attenuations in this level: x1, x1/10, and x1/100.

The shift function is implemented by selecting different attenuation signals using the broadband simulation switch, DG641. The bandwidth of the DG641 device is 500 MHz and the 5-MHz crosstalk is at -85 dB, which can implement on-off control of the 4-channel PN power supply. The signals output by the resistor ladder access the three-input gateway. To select different signals, we connect the appropriate output terminals.

We connect the selected output signal to the level-1 amplifier for a fixed number of times (10) to perform broadband operational amplification using the OPA680 device. The gain bandwidth is 400 MHz and the power supply is ±5 V, which we can use to amplify the ±4 V signal.

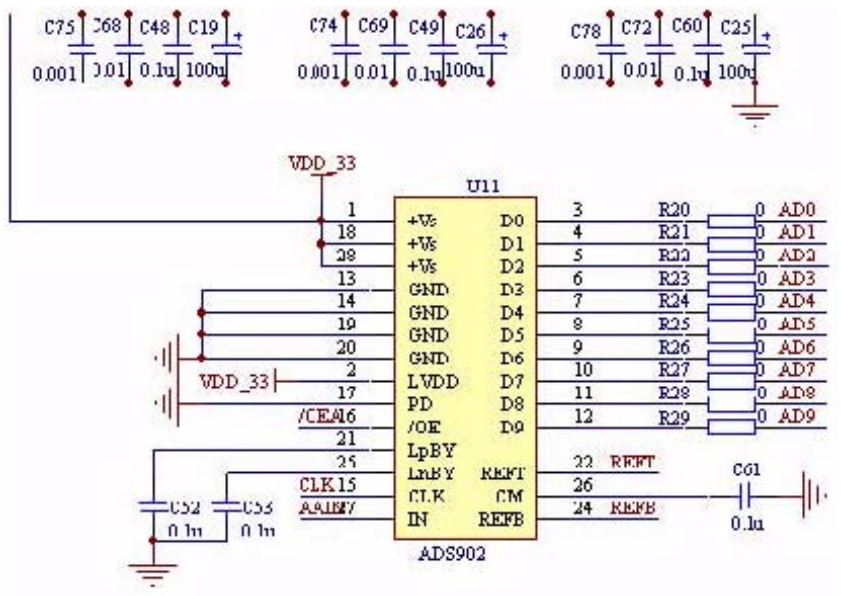
Using the level-1 attenuation, we can obtain four other attenuation types: x1, x1/2, x1/5, and x1/10 combined with the previous x1, x1/10, and x1/100. We can choose from ten attenuations: x1, x1/2, x1/5, x1/10, x1/20, x1/50, x1/100, x1/200, x1/500, and x1/1,000.

The level-2 attenuation output is amplified 10 times. The amplification of this level implements the signal amplitude adjusting and direct current (DC) level transfer, which changes the signal into the level appropriate for A/D operation. With the above-mentioned attenuation, it can implement 10 shifts with the range of ±5 V, ±2 V, ±1 V, ±500 mV, ±200 mV, ±100 mV, ±50 mV, ±20 mV, ±10 mV, and ±5 mV.

AD Conversion Circuit

The ADC uses the BB ADS902 device with operating parameters of 10 bits, 30 MHz, and a quantization amplitude of 1 V, which requires an external reference voltage norm. Figure 8 shows the A/D conversion circuit.

Figure 8. ADS902 Conversion Circuit

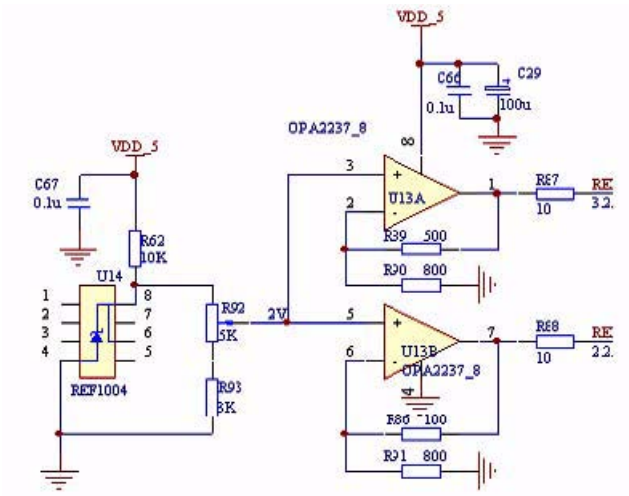


The ADS902 device has several power supply and grounding pins. To lower the interference, it performs multi-point filtering for each power supply. At the same time, it filters the reference voltage norm and the common mode voltage output terminal. The ADC analog input terminal and the data output terminal match the resistance.

Voltage Normalizing Circuit

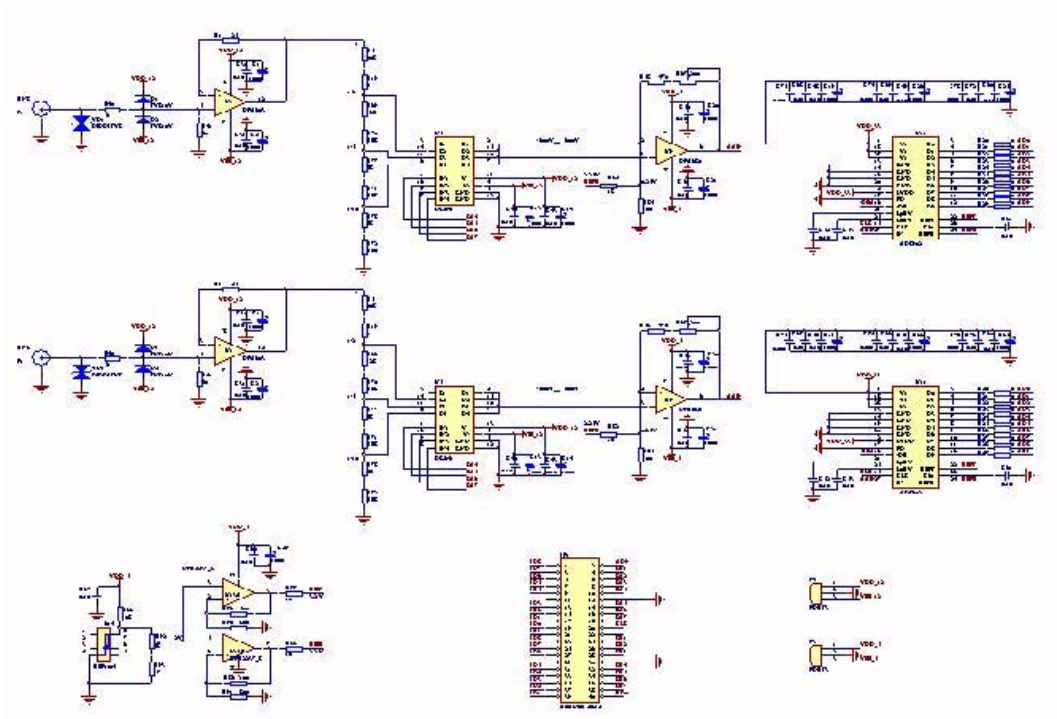
We implement the OPA680 level norm conversion and the AD operation reference evaluation norm with the REF1004-2.5 and OPA2237. The REF1004-2.5 device provides the 2.5-V voltage norm; the OPA2237 device provides a single power supply and dual-operational amplification. As shown in Figure 9, the 2.5-V voltage output by REF1004 is divided and receives the 2-V voltage norm, which gets two reference voltages, 2.25-V and 3.25-V, with the 1-V pressure difference for the ADS902 transfer. The 2.25-V voltage norm also provides OPA680 for level raising. Figure 9 shows the REF1004 and OPA2237 norm circuit.

Figure 9. Circuit Provided by Voltage Norm



During debugging, the input impedance of the broadband operational amplifier is low, so the two input ends' current deviation in the shift operational amplifier is high. Therefore, the zero-deviation is correspondingly high. Additionally, because the signal-to-noise ratio (SNR) is less than 5 mV and is produced by a general signal source, setting 5 mV and 10 mV is meaningless. So, the level 2 attenuation is changed into level 1 and the shift selections are x1/10, x1 and x10, that is, to remove the level 1 attenuation and amplification. The changed circuit is shown in Figure 10.

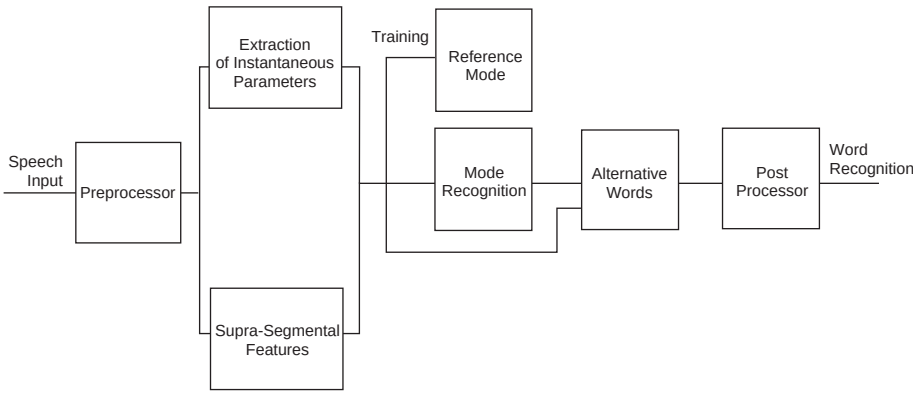
Figure 10. Changed Circuit



Audio Control Design and Implementation

Audio recognition means that machines understand human speech, i.e., they recognize the content of speech accurately and implement the requests according to the information. Figure 11 depicts a typical isolated word recognition system.

Figure 11. Isolated Word Recognition System



An audio recognition system has two stages: training and recognition. Both stages require pre-processing of the original input speech and extraction of features.

- The pre-processing module processes the original audio signal input, removes unimportant information and background noise, makes audio signal endpoint detection (i.e., determines the start and end positions within the valid range), and processes speech framing and pre-emphasis.

- The feature extraction module calculates the acoustic parameters of speech and the features to extract the key feature parameters, reducing dimensions, and facilitating post processing. The feature parameters frequently used in audio recognition systems include: amplitude, energy, zero cross, linear predictive coefficients (LPC), LPC coefficients (LPCC), line spectrum pair (LSP), short-time spectrum, mel-frequency cepstrum coefficients (MFCC) to reflect physiological characteristics of the human ear, etc. The selection and extraction of features are critical to system construction.

In the training stage, the user inputs training speech several times. The system obtains feature vector parameters (sequence) through the pre-processing and feature extraction, and builds a reference mode library (which can be reference template or model) for training speech through the feature modeling module.

In the recognition stage, a similarity measure comparison is made between the feature vectors of the input speech and the modes in the reference mode library. The mode category with the highest similarity is output as the intermediate alternative result.

The post-processing module processes the alternative recognition result and gets the final recognition result through more knowledge constraints.

Pre-Processing

Pre-processing mainly includes pre-emphasis, framing, and forming processes. We apply pre-emphasis to the original audio signals input to emphasize the high-frequency components of the speech and increase the high-frequency resolution of the speech. We usually use a filter with transfer function of $H(z) = 1 - \alpha Z^{-1}$ for filtering, where α is the pre-emphasis coefficient ($0.9 < \alpha < 1.0$), which is usually 0.95, 0.97, or 0.98. Suppose the speech sampling value at moment n is x(n), the result of pre-emphasis is $y(n) = x(n) - \alpha x(n-1)$ (0.9 , α , 1.0).

The speech has short time and stability and the short time feature can be improved through a framing operation to facilitate modes. The frame length is usually 30 ms and the frame shift is 10 ms.

We multiply the frame signals with a Hamming window to reduce the signal discontinuity at the beginning and end of each frame. The function of Hamming window is:

$w(n) = 0.54 - 0.46 \cos(2\pi n/N-1)$ for $(1 \leq n \leq N-1)$

where N is the sampling number of the current speech frame.

Expression and Extraction of Features

Linear forecasting and analysis technology is the most widely adopted technology for extracting feature parameters. Most successful systems use the LPCC extracted by linear forecasting technology as feature vectors of the system. In our system, we calculate the speech frame's self-correlation value, obtain the linear forecasting coefficients of this speech frame using the Durbin algorithm, and obtain the LPCC as the feature parameter vectors.

Audio Recognition

During audio recognition, the duration for pronouncing a word or each part of a word changes when it is spoken, even though the user tries to keep them unchanged during training or recognition. Therefore, using a feature vector sequence does not provide the best similarity comparison result. The project provides time-alignment for the feature parameter sequence mode using dynamic time warping (DTW), which efficiently solves the problem.

The starting point of a word corresponds to that of the path. The distance between the starting point and ending point of the best path is the distance between the speech to be recognized and the template speech. The recognition result is the template word with a minimum distance from the speech to be recognized. The method has a high computational load, but it is technically easy with a high recognition

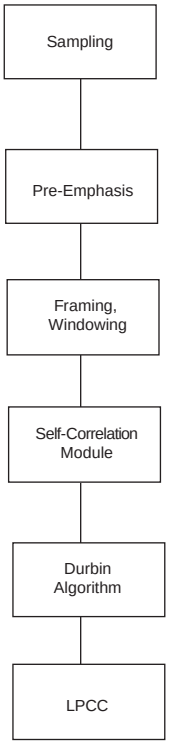
rate. In point matching, the distortion measure for short-time spectral or cepstrum parameter recognition systems can adopt a Euclidean distance while the distortion measure for a recognition system adopting the LPC parameter may use a log likelihood ratio distance. The nearest neighbor criterion is usually adopted for these decisions.

In the oscillograph's audio control module, the speech signal is sampled at 8,000 Hz and 16 bits; the frame length is 20 ms, i.e., 160 sampling points, and the frame shift is 10 ms. The speech feature parameter uses a 12-order LPCC. Both the audio signal pre-processing and feature parameter extraction use FPGA hardware, thereby implementing parallel operation with the CPU and reducing the CPU load to a certain extent. The peripheral processing circuit gets the speech frame feature parameter vector once every 10 ms and interrupts the CPU. The CPU makes real-time endpoint detection and matches the template with the DTW algorithm once a complete speech feature sequence is obtained, thus obtaining the best recognition result.

Hardware Design and Implementation

The audio recognition design's front-end acquisition module uses the WM8731 audio encoding/decoding chip on DE2 development board. The chip is connected directly to the analog audio interface on the development board, and is controlled through I^2C bus. The AD/DA converters sampling rate and depth can be controlled easily by controlling the reading/writing of the register. The speech feature parameter adopts LPCC, which can be obtained as shown in Figure 12.

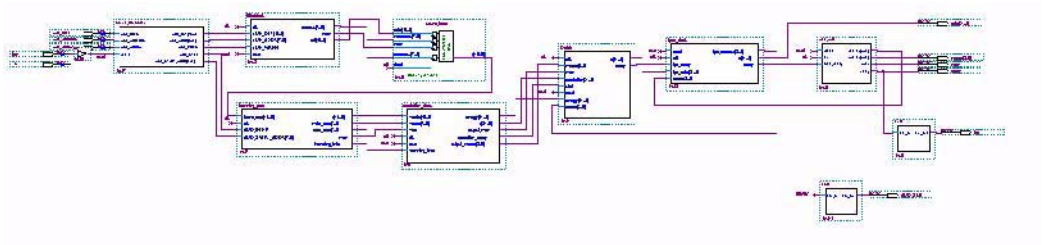
Figure 12. Feature Extraction Flow Chart



We implemented the parameter extraction module in the FPGA using Verilog HDL. The module's functionality includes sampling from the audio chip through the I^2C bus, pre-emphasis of audio signals, framing, windowing, extraction of self-correlation functions of speech frame, transform of the self-correlation value into LPC using the Durbin algorithm, and conversion of LPC into LPCC, which is used frequently in audio recognition.

Figure 13 shows the parameter extraction module's top layer, which uses the pipeline operation mode. The internal modules are initiated by a rising signal edge. After operation, the module provides a rising edge indicating the status change that drives the next-level pipeline. Each module has an internal cache to ensure continuous data transmission.

Figure 13. Speech Feature Parameter Extraction Hardware Structure



As the value varies greatly during the whole operation, fixed-point operation reduces the result's precision. Therefore, our design uses floating-point operation. The addition and subtraction of each floating point number uses 18 clock cycles; multiplication requires 14 clock cycles. Division requires more time, 34 clock cycles.

LPC Analysis and Durbin Recursive Algorithm

LPC analysis estimates the value to be input by linear combination of the past values of output signals in a certain time domain discrete linear system. That is, if the estimated value of the audio signals at time n is:

$$s'(n) = -\sum_{i=1}^{P} a_i s(n-i)$$

The prediction error is:

$$e(n) = s(n) - s'(n) = \sum_{i=0}^{P} a_i s(n-i)$$

According to the minimum mean square error principle (LMS algorithm), the predictor's optimal prediction coefficient a_i can be calculated using the following equation:

$$\sum_{i=0}^{P} a_i R(k-i) = -R(k),\ k = 1,\ 2,\ ..P$$

where:

$$R(l) = \sum_{n=0}^{N-1-l} s(n+1)s(n),\ l = 0,\ 1,\ ..P$$

These equations are called LPC canonical equations. R(l) is the self-correlation function, which is the basis of LPC analysis.

Durbin Recursive Algorithm

The Durbin algorithm starts from zero-order prediction when p = 0, $E^0_n = R_n(0)$, $a^0 = 1$. The complete recursive process is shown below:

$E_n^0 = R_n(0)$

$$k_i = \left[R_n(i) - \sum_{j=1}^{i-1} a_j^{i-1} R_n(i-1) \right] / E_n^{i-1}$$

$a^i_i = k_i$

$a^i_j = a_j^{i-1} - k_i a_{i-j}^{i-1}$, $1 \leq j , i - 1$

$E_n^i = (1 - k_i^2) E_n^{i-1}$

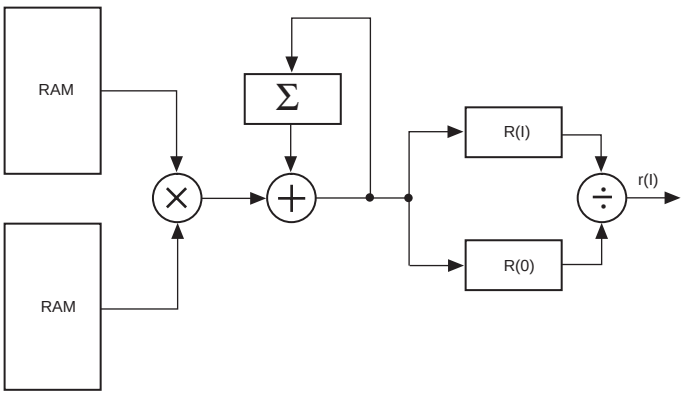
if i < p, goto (1)

$a_j - a_j^p$, $1 \leq j \leq p$

Self-Correlation Calculation Module

Figure 14 shows the self-correlation calculation module block diagram.

Figure 14. Self-Correlation Calculation Module

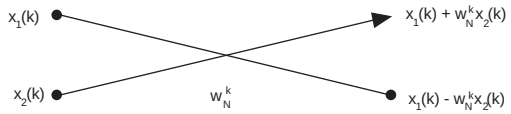


Self-correlation functions require many multiplier-accumulator (MAC) operations. Therefore, calculating these functions are a key factor that affects system speed. To improve system speed, we store the windowed audio signal in two RAMs and use a parallel load to obtain two sets of data for multiplication. The result is sent to the accumulator for accumulation. Then the multiplier calculates the next data set. The multiplier and accumulator perform parallel operations throughout the whole process, ensuring pipeline continuity and maximum system speed.

FFT Design Implementation

We use the radix-2 decimation-in-time FFT algorithm in our design. The butterfly operation is the basic operational unit of the FFT algorithm. Figure 15 shows the butterfly operation symbols, in which W_n^K is the twiddle factor. We set the length of the sequence x(n) as N, and $N = 2^M$ (where M is an integer), so the FFT operation flow diagram of the sequence consists of M levels of butterfly operations.

Figure 15. Butterfly Operation Blocks



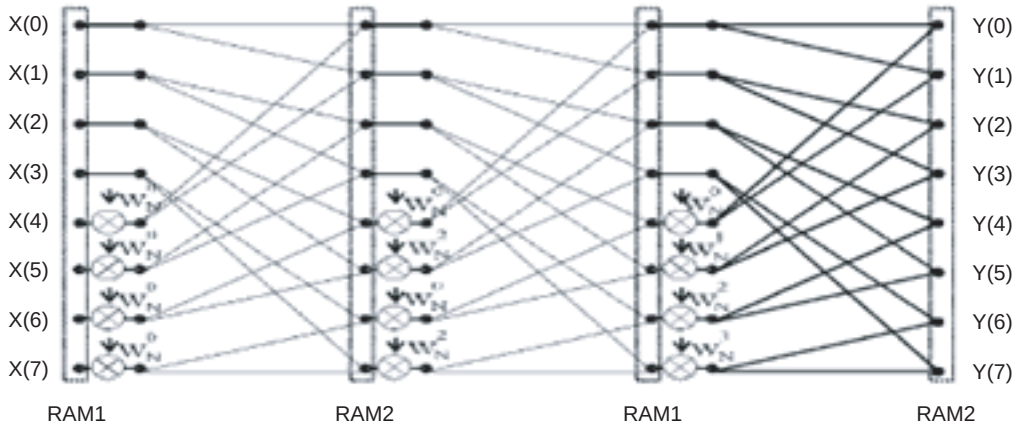
The traditional flow diagram features reversed input and natural output. The algorithm has several advantages, for example, for operations of the same level, the two input data of each butterfly are only useful to their own operation. Additionally, the I/O data nodes of each butterfly are on the same line; therefore, the output data after one butterfly operation can be immediately stored in the storage unit of the former input data, i.e., in-place operation. This function saves storage units in the project implementation. The disadvantage of the algorithm is that the changeable geometrical structure of each algorithm cannot be cyclically controlled in the program. Therefore, it is not easy to expand.

Another FFT algorithm flow diagram features natural input and reversed output. See Figure 16 (use N = 8 as an example). The advantages of this algorithm are that the fixed geometric structure of each operation facilitates the cyclical program control. We only need to change the twiddle factor for operations that involve different levels. This algorithm is also more scalable: for an FFT with different points, we only need to change the corresponding address generator. Therefore, this FFT structure is more suitable for processing in an FPGA.

The disadvantage of this method is that the butterfly I/O data nodes are not on the same line. Therefore, data obtained through each butterfly operation cannot be stored in the storage unit of the original data. To implement the same N-point complex FFT with 16-bit data width, we need a 4N x16 bit storage space, which is twice the size required by the traditional in-place operation.

We decided to use the second structure (i.e., natural input and reversed output) in our design. See Figure 16.

Figure 16. Natural Input, Reversed Output 8-Point FFT Structure



FFT Block Modules

The RAM module consists of two dual-port RAM blocks, `RAM1` and `RAM2`, which store the input operation result's real and idle component or that of different levels. The input data is first stored in `RAM1` and the result of the first-level operation is stored in `RAM2`. The `RAM2` data is extracted and placed in `RAM1` after the second-level operation. We repeat the process for $M = \log_2 N$ times (where N is the number of FFT points). When N is different, the butterfly operation levels may also be different. Therefore, the result of the last-level operation may be stored in either RAM block. The input data is stored in `RAM1` first, so the operation result may be overlaid by the input data before it is output. For this

reason, the result of the last-level operation is stored in RAM1 when M is an even number. Before writing new data, we store the result in RAM2 and the final result is output through RAM2. Because the design's input data is regular (512 points), we read from RAM2 after receiving the FFT operation end signal.

In the twiddle factor module,we generated the twiddle factors as real and idle components before the MATLAB software. Then we converted them into 16-bit fixed points and stored the result in hexadecimal format. We used the Quartus® II MegaWizard® Plug-In Manager to generate two N/2 x16-bit ROMs, which are initialized by the corresponding Hexadecimal (Intel-format) Files of the twiddle factor real and idle components, respectively. In this way, the FPGA contains the twiddle factor values. For butterfly operations of different levels, the address generation unit generates corresponding addresses for ROMs to read the factor values. For the natural input and reversed output structure, the twiddle factor addressing is important. As can be seen from the FFT structure diagrams, we can infer the twiddle factor address rule at each level. For the N-point FFT ($N = 2^k$, k is the level), there are:

$W_N^0, W_N^1, W_N^2, \ldots W_N^{N/2-1}$

or N/2 twiddle factors. Arranging them them as an array with N/2 elements:

$w(N/2) = \{W_N^0, W_N^2, W_N^4, \ldots W_N^7\} (N = 16)$

Thus, the order of the twiddle factor at level i is obtained by repeating w(0), w(1), w(2), ...$w(2^i)$ for 2^{k-i-1} times.

For the N-point complex FFT processing, the RAM and twiddle factor two modules require total memory resources of (2 x N + N/2) x 2 x 16 bits (the data are in plural form, so each point needs two storage units).

The butterfly operation unit consists of four multipliers, one adder, and one subtracter. The Cyclone® II FPGA contains rich multiplier resources, which can implement one butterfly operation in one clock cycle. This performance is better than a single-multiplier DSP in which each butterfly operation requires four clock cycles. Theoretical analysis concludes that to finish the FFT with the same number of points, the clock cycles required by the FPGA implementation are one-fourth of those required for the DSP implementation.

For the N-point FFT, the operation can be divided into $N = \log_2 N$ levels. Operation between two levels works serially. Operation at each level has N/2 butterfly operations, which work serially with each other. So M x N/2 butterfly operations should proceed successively, with each operation deploying the same butterfly operation unit. Because the operation is serial, only one butterfly operation unit block is needed, even for an FFT with more points. Therefore, we do not require additional DSP resources in the FPGA. The design uses the 50-MHz system clock as the operating clock. To obtain stable data, we added latches before and after operations at each level. One butterfly is finished in 8 clock cycles and an N-point FFT needs M x N/2 x 8 clock cycles. A 512-point FFT needs roughly 370 µs.

The address generation unit is mainly used for generating RAM data and storing read/write-enable signals and read/write addresses of the twiddle factors ROMs. For example, the final operation result is reversely stored in the RAM. To enable the natural output of the result, we use a few simple statements to realize the bit-reverse addresses in the address generation module. See the following code (take N = 32 as an example):

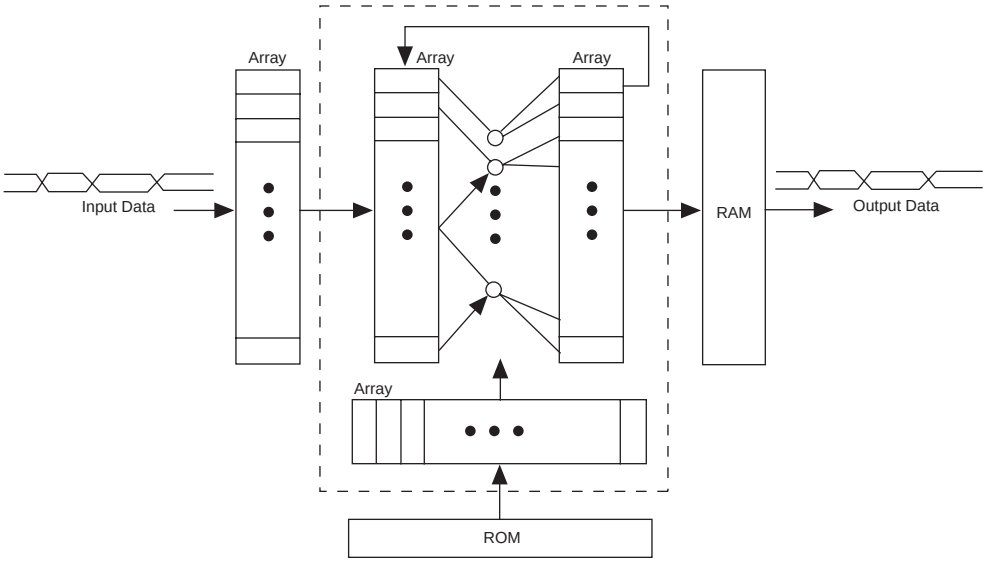
```
process (clock)
   begin
      if (clock' event and clock = '1') then
         if (done = '1') then
            count < = count + 1;
         end if;
         address < = count (0) & count (1) & count (2) & count(3) &count (4)
      end if;
end process;
```

Because we used the natural input, reversed output structure for the RAM block that provides data to the butterfly operation blocks, the address generation logic only requires the natural generation addresses - n (0 - (N/2 - 1)) and corresponding n + N/2 to address the RAM and send it to the butterfly blocks in mid-operation. The address generation logic only needs the natural generation addresses - n (0 - (N - 1)) for the RAM block that receives the butterfly blocks output data. For the twiddle factor table addressing, the butterfly operation at each level reads one section naturally according to its needs because the twiddle factors are stored in reverse.

FPGA Hardware Implementation

FPGAs can be flexibly programmed using hardware description languages (HDLs). Because we can simulate the FPGA operation, the hardware design is as flexible and convenient as the software design, which shortened our research and development phase. We used the JTAG interface to configure the device in system, enhancing the system's flexibility. We used the Altera Quartus II software to emulate the hardware, perform logic analysis, and compare the output result to the MATLAB emulation result. See Figure 17 for the system architecture.

Figure 17. FFT Hardware Implementation



We wrote the design using VHDL. The 512-point FFT requires a 9-level butterfly operation; each level includes 256 butterfly operation blocks. The operations input data bit width is 16 bits. Each butterfly operation block needs four multiplications and then four additions (2 each for the real and idle component). Because the twiddle factor is 16 bits, we select the higher 16 bits of the 32-bit data after multiplication and then perform addition. The real and idle components of each butterfly operation block need two additions, so data can overflow. However, we only need the relative size of the energy value of each signal spectrum harmonic, so we extend the 16-bit data into 18 bits and calculate. Additionally, we set two sign bits for each level. If data overflows in one or more butterfly operations (real or idle component) at the same level and if the overflow happens at the 16th bit, we set sign bit 1. If the overflow happens at the 16th bit and it leads to the overflow of the 17th bit, we set sign bit 2. We store the operation result in RAM as 18-bit data. Therefore, when we next read the RAM data for calculation or output, the selected data is the second higher 16 bits if the sign bit is 1. If sign bit 2 is set, the selected data is the higher 16 bits; otherwise, the data is less than 16 bits.

The direct digital synthesizer (DDS) acts as the front-end signal generation module. Figures 18 through 21 shows spectrograms sampled by the DDS-generated square wave signals (frequency is 48.828125 kHz) at different frequencies).

Figure 18. 50-kHz Sampling Frequency

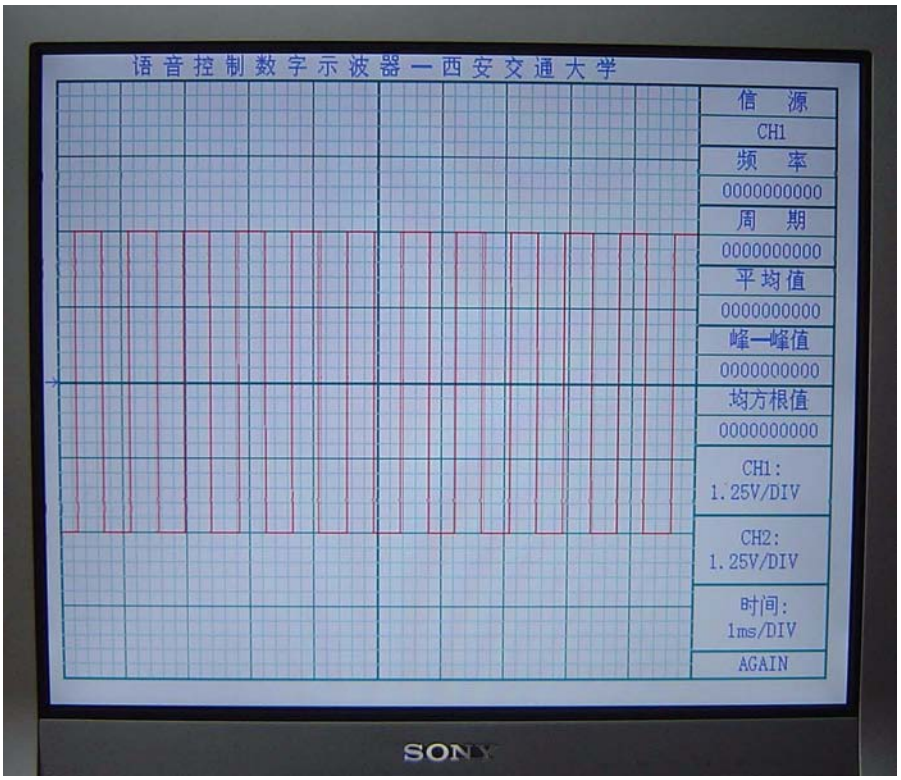


Figure 19. 50-kHz Sampling Frequency Spectrogram

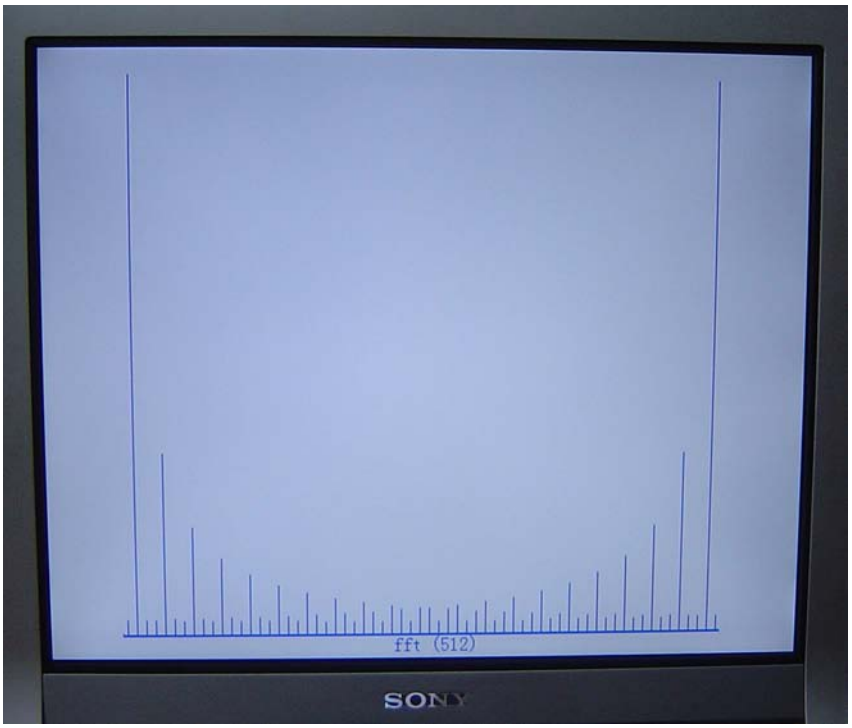


Figure 20. 100-kHz Sampling Frequency

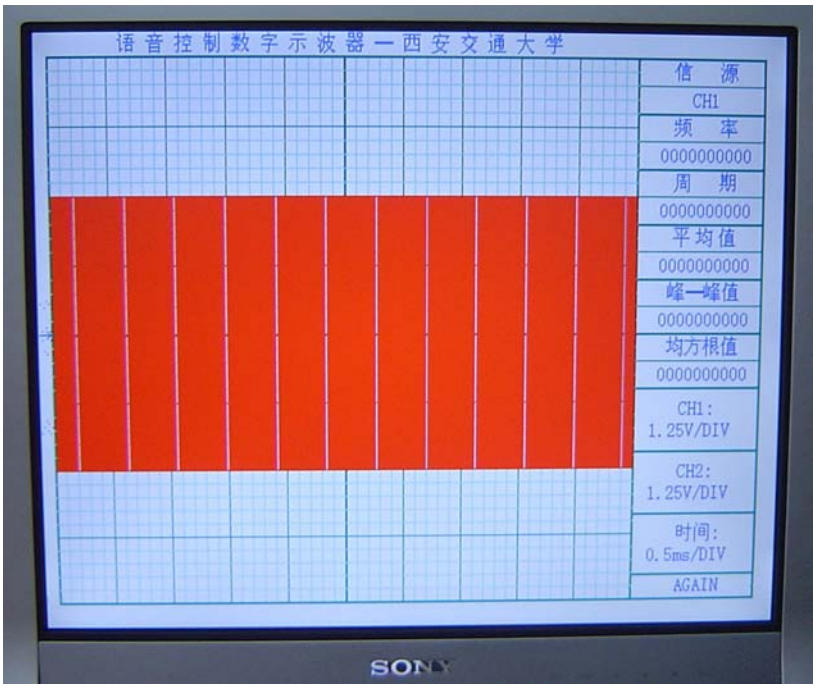
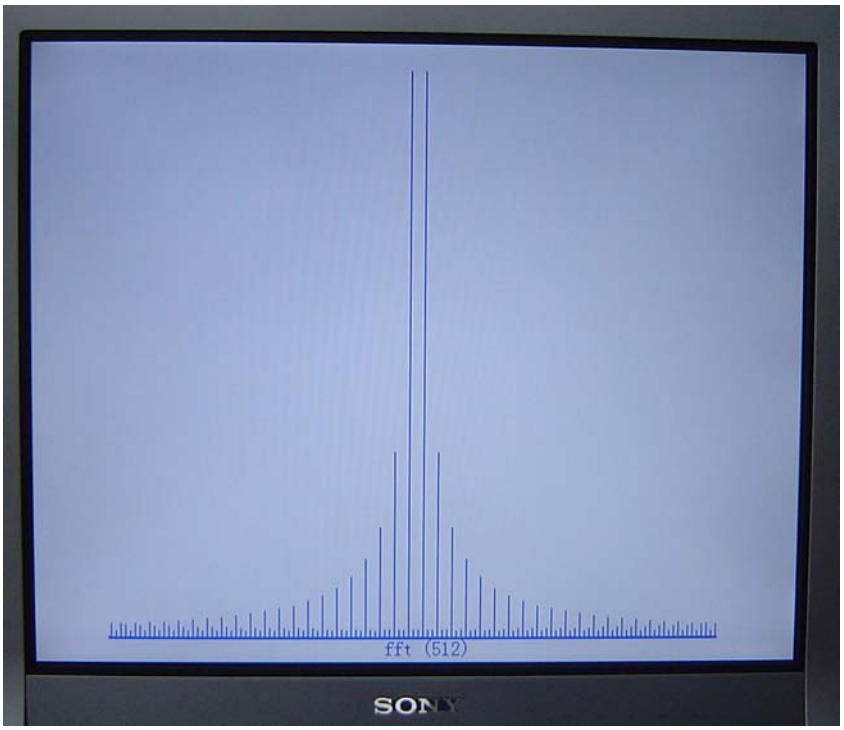


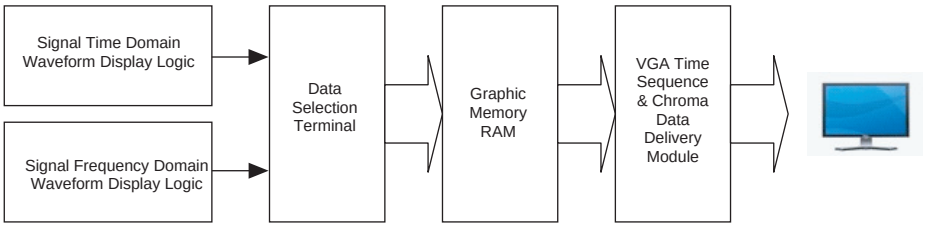
Figure 21. 100-kHz Sampling Frequency Spectrogram



VGA Display Implementation

The VGA display module is composed of two display control modules. One controls the time domain waveform that displays the two signal channels and the channel parameters, including signal sources, frequency, cycle, average value, peak-to-peak value, and root mean square (RMS). The other controls the frequency domain waveform that displays either of the two signal channels obtained by a 512-point FFT. We use a 4-Hz frequency to update the storage display RAM as it relates to the current time domain waveform or frequency domain spectrum. See Figure 22.

Figure 22. VGA Display Control Module



Time Domain Waveform Display Control Module

The time domain waveform display screen has the following five general display areas:

- *Top*—The top area is the title panel of the digital oscillograph, which displays **Audio Control Digital Oscillograph—Xi'an Jiaotong University** in blue characters.

- *Left*—The design uses the left area to display the 0-level symbol of the two channel signals. A blue arrow points at the 0 level of the current channel.

- *Waveform (or middle)*—The middle area is the signal waveform display area, measuring 500 pixels (horizontally) by 440 pixels (vertically). It can simultaneously display the time domain waveform signal of the two channels or that of either channel. We update the middle area's memory storage unit using the display signal value that is converted from the original signal in the column. The signal waveform is displayed using vectors, i.e., the pixels that correspond to the current column signal's amplitude value track the pixels that correspond to the next column signal's amplitude value. The dynamic scope of the original signal data is 0 to 1,023, and the scope of the display signal data is 0 to 439. We use the `fr` control signal synchronous with the graphic RAM refresh frequency to update the RAM display signal. When `fr` rises, it continuously collects 500 data bits on the A/D port at the given frequency, `fs`, and adjusts the dynamic scope every time new data is collected. The module then saves the data into the display signal RAM.

- *Right*—The right area displays the signal attributes and values, and has six big grids and one small grid. Each big grid is further divided into two small ones.

  - *First grid*—The first small grid displays messages and the second small grid displays the signal channel of the current attribute values.

  - *Second grid*—The first small grid displays the frequency and the second small grid displays the frequency value.

  - *Third grid*—The first small grid displays the cycle and the second small grid displays the cycle value.

  - *Fourth grid*—The first small grid displays the average value and the second small grid gives the current value.

  - *Fifth grid*—The first small grid displays the peak-to-peak value and the second small grid gives the current value.

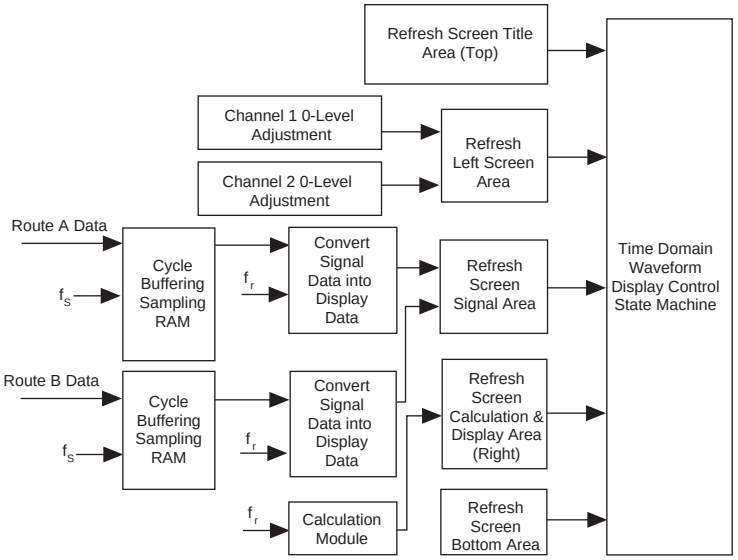
  - *Sixth grid*—The first small grid displays the RMS and the second small grid gives the value.

  - The last grid displays the audio control command's recognition result. It displays `OK` if the current command is recognized; otherwise it displays `AGAIN`.

- *Bottom*—We do not use the bottom area. The white point refreshes when update signals occur.

On the screen, the blue Chinese characters and code are placed against a white background. In the signal area, the red signal waveform of the two channels appears on the light or dark green squared grid. In the value display area on the right, dark green grids distinguish the values with different attributes. See Figure 23 for the control module functions of each area.

Figure 23. VGA Time Domain Waveform Display Control Module



The state machine of the time domain waveform screen controls the five parts of the time domain waveform display screen. Every time the `fr` signal rises, the state machine initiates each module to update the relative RAM; a complete memory refresh updates the screen.

Frequency Domain Waveform Display Control Module

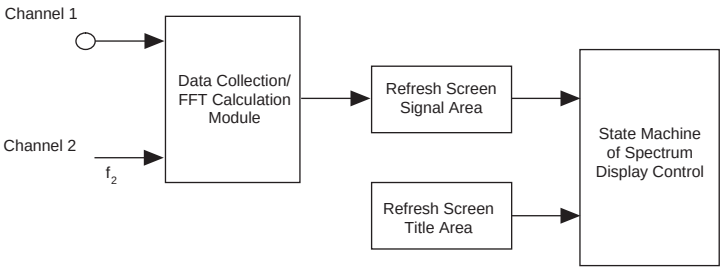
The frequency domain waveform display screen is composed of two areas that correspond to the two control modules:

- *Spectrum*—The spectrum area displays the channel signal frequency spectrum diagram that corresponds to the current message source. The rising edge of the `fr` control signal starts the FFT calculation module, which continuously collects 512 signal values on the A/D port at the given frequency `fs`. Then it performs the FFT calculation. The spectrum area control module obtains a signal when the calculation completes. The block updates current memory according to the 512-point frequency domain energy value obtained by the FFT.

- *Bottom*—The bottom area displays the title panel of frequency domain waveform screen in blue characters, e.g., FFT(512).

The frequency domain waveform display also has a control state machine. At the frequency domain waveform screen state, the module updates the bottom area. Otherwise, it only updates the spectrum area. Figure 24 shows the control module.

Figure 24. VGA Frequency Domain Display Control Module



Design Methodology

Our design methodology consisted of the following aspects:

- *SOPC design*—The SOPC design is divided into hardware and software. The hardware design includes the basic Nios II SOPC environment and customized peripheral circuits using the Avalon bus. The design uses hardware to implement the main system calculation units, including audio recognition attribute extraction, the FFT calculation module, and the VGA display control module. Software design includes the system control and audio recognition algorithm.

- *Structure design*—The system runs on the DE2 development board; the board resources implement the majority of the design. Additionally, the system has a front-end acquisition circuit board connected with other parts of the system via a development board expansion slot.

Design Features

The audio control implementation and real-time FFT spectrum display are the two highlights of our system.We used the Nios II processor to design and develop the audio control digital oscillograph. Except for the front-end acquisition circuit and some control devices, the system design is implemented in a single FPGA. Compared with a hard-core design, the soft-core system is compact, uses fewer circuit connections between devices, lowers power consumption, and improves system stability.

Because the system has many real-time calculations, CPU-based software calculation cannot meet our speed requirements. Therefore, most of our data processing is implemented using hardware design. With the FPGA's data processing capability, we achieved real-time processing of high-speed, complex data.

Conclusion

After tuning, our system can display a waveform like a standard oscillograph for a given frequency scope. However, the design has some shortcomings. Due to improper design of the front-end acquisition circuit, the system's signal input bandwidth is 300 KHz instead of 1 MHz.

As mentioned previously, the amplitude has 10 gears. However, the low-input impedance of the broadband operation during tuning causes a large deflection current on the two input ports when shifting,which creates a larger zero drift. Additionally, the signal-to-noise ratio (SNR) of lower than 5 mV arising from ordinary signal sources is so poor that it is not necessary to re-configure the full-range gears of 5 mV and 10 mV. Therefore, we changed the second-level attenuation to the first level when tuning, and choose x1/10, x1 and x10 in the shift selection, i.e., we removed one-level attenuation and one-level amplification. Due to the limited memory resources of the EP2C35F672C6 device, we could not add our audio recognition module into the system, which affected our design.

This competition gave us a deeper understanding of the Nios II processor, particularly new design concepts and development features. The SOPC design method enables us to customize peripherals flexibly. We could easily set up the interface between peripherals and the CPU without the development limitations of a monolithic processor.

Finally, and most importantly, we thank Altera Corporation for offering us this valuable learning and hands-on opportunity.