Third Prize

Auto Audio Equalizer Using Digital Signal Analysis

Institution: **Hanyang University**

Participants: **Sung-Wook Kim, Eun-Chan Kim, Bum-Su Jeong**

Instructor: **Professor Jae-Myoung Jeong**

Design Introduction

Our project, an auto audio equalizer using digital signal analysis, is an equalizer that enables low-end speakers to deliver high-quality audio like that found in expensive speakers. Consumers can copy the characteristics of speakers with the desired audio quality and apply those characteristics to their own low-cost, low-audio-quality speakers. The structure and application of our project is simple, and it allows users of low-cost speakers to enjoy better sound quality.

Our project is an economical device that can also be applied to high-end monitoring equipment to achieve a flatter response without upgrading the existing system.

Function Description

The equalizer has three major functions:

- Measure and save the response of the speaker system.
- Analyze and compare the saved speaker's response with that of the current speaker system, process the input signal, and adjust the current speaker system's response characteristics to mimic those of the selected speaker system.
- Measure the applied system's response and evaluate the performance.

To execute these functions, we implemented the design in four parts:

- We implemented the fast Fourier transform (FFT)/inverse FFT (IFFT) block's input signals using functions from the MegaCore® library.
- We configured the FFT/IFFT hardware as an equalizer by connecting the coefficients of each frequency ranges to the registers of a Nios® II processor.
- We included an SDRAM controller, flash controller, user logic FIFO buffer, and audio CODEC interface logic using SOPC Builder. We ported µClinux for the system that uses the Nios II processor.
- We included PC monitoring software that has a graphical user interface (GUI) that displays and saves the stored speaker's response and the measured response. The response can also be downloaded to the Cyclone® device on the development and education (DE1) board.Communication between the device and the PC uses the DE1 board's RS-232 interface.

We chose µClinux as the system's operating system because many programmers are familiar with the Linux environment, which helps shorten the development period, and there are plenty of available development resources. Additionally, µClinux dramatically reduces the need for low-level coding such as interrupt handling without overloading the CPU.

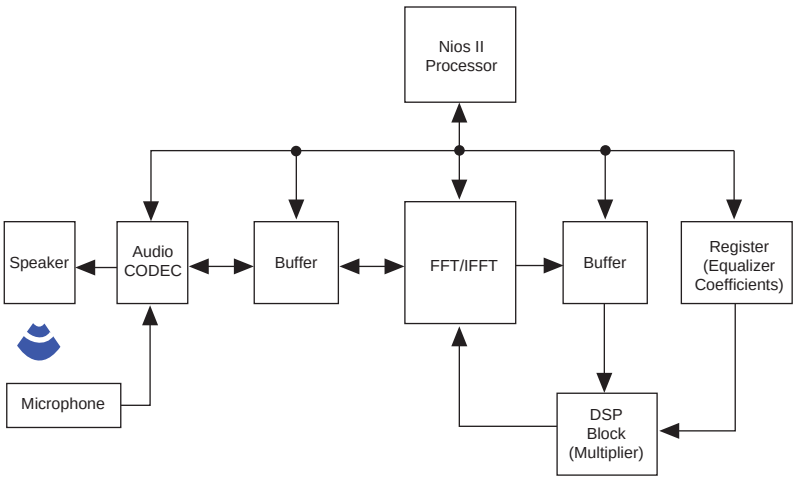
Performance Parameters

The equalizer applies the response of an already measured separate speaker to the current speaker system and then measures the current system's response a second time. The system numerically displays the correlation between the desired response and the improved response. Users can monitor how close the responses are using a PC-based GUI.

Design Architecture

Because the contest required us to use the DE1 development board, the system has an abbreviated structure that only implements the core functions. Figure 1 shows the system's block diagram. The digital signal processing (DSP) block performs most of the signal processing. The Nios II processor implements the module control and monitoring. We could expand the tasks performed by the Nios II processor later using the C-to-Hardware Acceleration (C2H) Compiler or user-defined custom instructions.

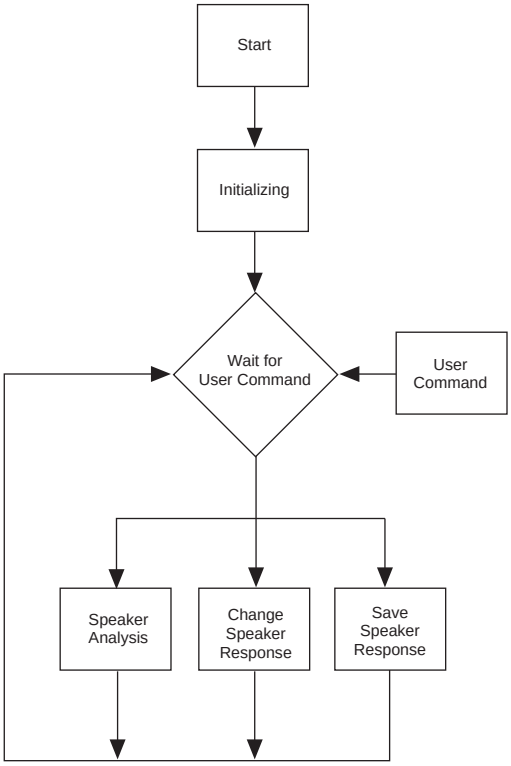
Figure 1. System Block Diagram



We inserted a digital signal analyzer (DSA) to measure the speaker's response. The microphone is not speaker dependent; therefore, once the frequency characteristics are met, other response characteristics are not critical.

Figure 2 shows the software flow chart.

Figure 2. Software Flow Chart



The system performs three key functions as instructed by the user:

- *Speaker analysis*—The system measures the response of a speaker that is currently connected to the system. After measurement, it graphically display the result so that the user can decide whether to save it.
- *Save speaker response*—The system can store the measured speaker response in flash memory; later, the system can change a speaker's response based on this information.
- *Adjust speaker response*—If the user selects the response of the speaker currently connected to the system and the desired speaker response, the system can change the current speaker's response through DSP.

Equalizers currently available in the marketplace can perform similar functions, but the user must manually adjust each variable. In the future, we would like to add a feature that imports calculated variable values (like those used in commercial equalizers) to the PC. This feature would allow users to use Winamp or Windows Media Player without a commercial system.

Design Description

The most difficult part of designing the hardware is debugging and verifying the system operation at every step. We only added an unverified block after the rest of the system had been fully verified. We also verified the intellectual property (IP) features using simulation. This process helped us avoid rechecking existing parts of the system while debugging and increased our development efficiency.

Fortunately, the Altera® Quartus® II software, SOPC Builder, and Nios II Integrated Development Environment (IDE) allowed us to use already verified IP functions and design methodologies, significantly reducing the debugging burden. For example:

- Because we used the FFT-IFFT block without modifications, we only needed to perform a simple data format check and timing simulation.
- We implemented the Nios II processor, memory controller, FIFO interface, and audio CODEC interface using SOPC Builder. These blocks are basic features of SOPC Builder; therefore, we easily designed and tested the firmware using the Nios II IDE. Test programs are readily available at communities such as the Nios II forum, speeding up the entire process.
- The JotSpot Wiki (nioswiki) provides a good explanation of how to port µClinux. We just needed to follow the instructions to implement all of the IP blocks with SOPC Builder.
- We used Windows-based programming to design the PC monitoring program. After we developed the monitoring program, we simulated the monitored signals using the MATLAB software and other tools that manipulate data easily. Then, we ported the compiled code to the DE1 board and tested the hardware.

Design Features

Our design has the following features:

- *Creative*—Using DSP, the design makes a low-end speaker produce high-end speaker sound.
- *Inexpensive*—All functions except the audio CODEC and microphone are implemented in an Altera FPGA, reducing the bill of materials (BOM). Also, the system performance is not affected by using a cheap microphone, further reducing the total system cost.
- *Easy and prompt development*—The Nios II processor and user-defined logic (buffer, FFT/IFFT block, register, and audio CODEC) connect directly to the Avalon® bus, which controls them. Therefore, we did not need to provide controls using a general-purpose I/O (GPIO). In the future, we will not need to design the DSP block separately; instead we can implement it with the Nios II processor using user-defined custom instructions.

Conclusion

The Cyclone FPGA on the DE1 board was sufficient to implement a 2,048-sample FFT/IFFT with a 16-bit, 44.1-kHz sampling rate, the Nios II processor, and other IP functions. However, our design only measured and processed the magnitude as a speaker's response, ignoring phase delay and other distortions, which limits the system performance. In the future, improving the signal processing will provide better performance.

We often perform projects using the DE2 board, mainly for fast prototyping, because Altera's FPGA solution provides proven tools and IP blocks, enabling us to implement systems quickly. Additionally, active Internet communities provide quicker, higher quality feedback than local distributors. We ask Altera to continue events like the Nios II design contest to support and encourage these communities.