MP3 Player

CSEE 4840 SPRING 2010 PROJECT DESIGN

Zheng Lai zl2215@columbia.edu

Zhao Liu zl2211@columbia.edu

Meng Li ml3088@columbia.edu

Quan Yuan qy2123@columbia.edu

I. Overview Architecture

The purpose of the system is to design a comprehensive hardware system that can decode the MP3 format audio file and automatically analyze the digital audio signal in frequency domain used to control floating histogram on screen. The MP3 format files are stored in SD card. With instruction from niosII processor the MP3 decoder module starts to fetch the audio bit stream in MP3 files from SD-card and execute a series of algorithm. The decoded discrete time audio signal is then transmitted to WM8371 control module that restores the digital signal to analog signal. At the same time the frequency analyzer acquires the digital audio signal from the decoder as input of some kind of digital processing method. As a result, the output of the analyzer, this contains information of how fast its input is changing, will direct the floating histogram effect displayed on screen. In addition we will implement a PS2 keyboard controller for future software application (probably a game software).

To sum up, the system has the following function:

- A MP3 hardware decoder
- A discrete-time-domain of discrete-frequency-domain transformation component
- WM8371 audio output control module
- VGA output control module
- SD-card control module and PS2 control module (optional)

The top level design is shown as follows:

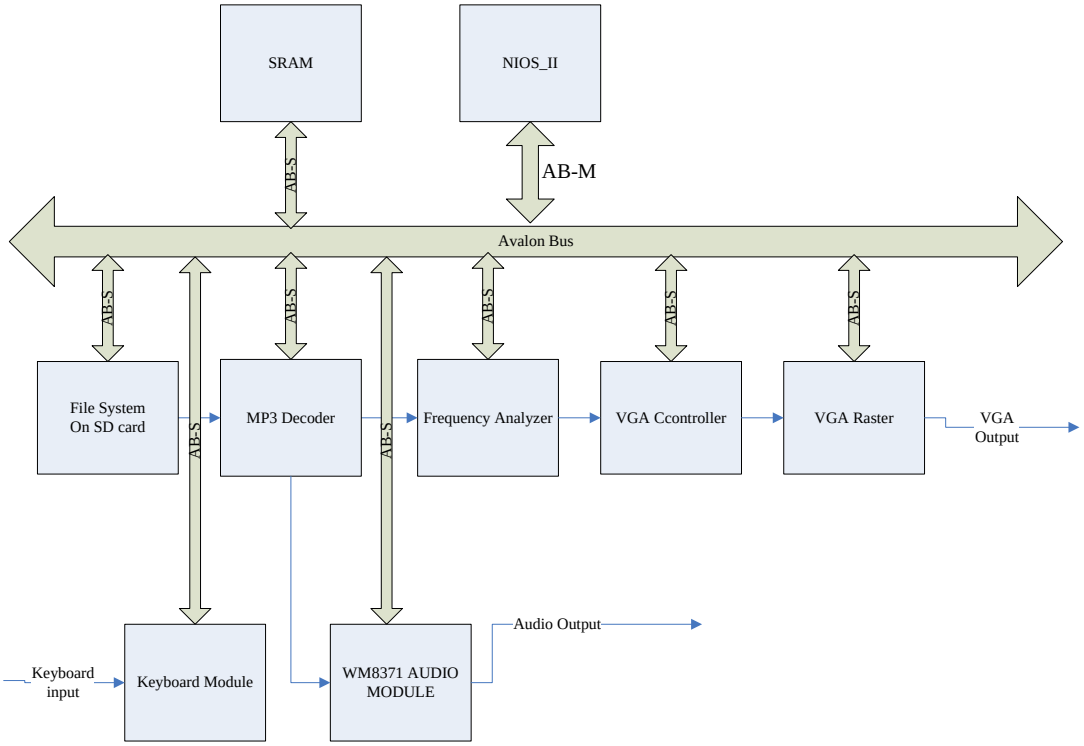

Fig. 1 Overall architecture ofMP3 player

II. MP3 Decoder

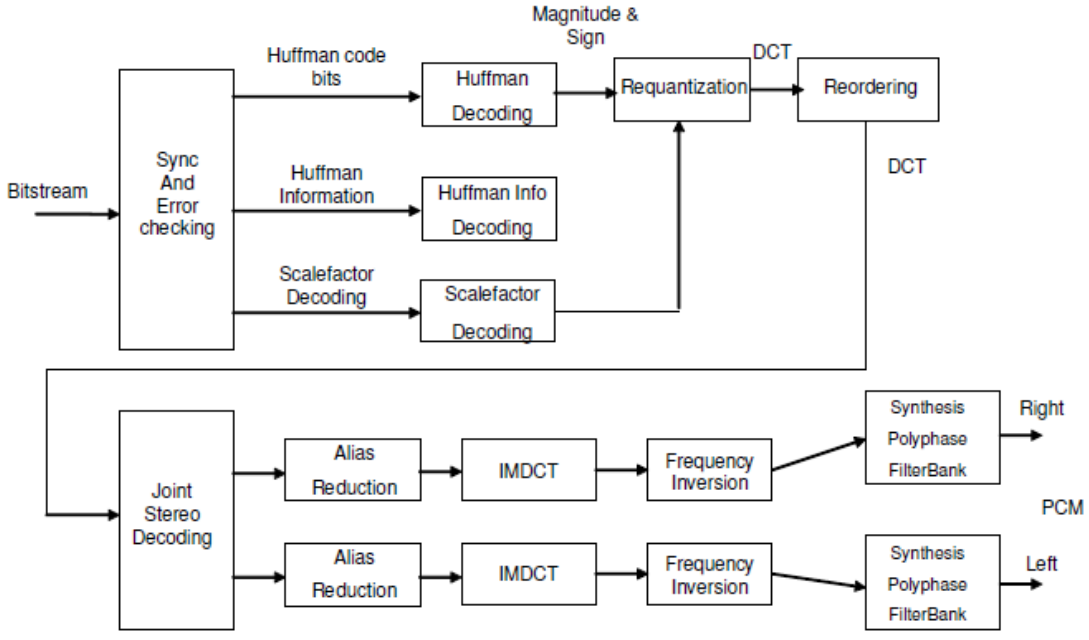

Fig. 2 Flowchart of MP3 decording

The block diagram in Figure 4 shows the data flow within the MP3 decoder. The incoming data stream is first split up into individual frames and the correctness of those frames is checked using Cyclic Redundancy Code (CRC) in the sync and the error checking block shown in Figure 4. Further, using the scale factor selection information in the side information, scale factors are decoded in the Scalefactor decoding block. Scale factors are used to scale up the re-quantized samples of a subband. Subband is a segment of the frequency spectrum. Subbands are introduced in the encoder to selectively compress the signals at different frequencies. These subbands are chosen to match the response of human ear. The main data of the frame is encoded as a Huffman codes. The quantized samples are derived from the huffman codes in the Huffman decoding block. The necessary side information needed for huffman decoding is obtained from Huffman Info decoding block. Since the huffman codes are variable length codes, the huffman encoding of the quantized samples results in a variable frame size. In order to optimize the space usage in a frame, the data from the adjacent frames are packed together. So, the Huffman Decoding stage refers to the previous frames data for its decoding. The next step after Huffman decoding, is the re-quantization. The re-quantizer, re-quantizes the huffman decoder output using the scalefactors and the global gain factors. The re-quantized data is reordered for the scalefactor bands. The re-quantized output is fed to the stereo decoder, which supports both MS stereo as well as Intensity stereo formats. The alias reduction block is used to reduce the unavoidable aliasing effects of the encoding polyphase filter bank. The IMDCT block converts the frequency domain samples to frequency subband samples. The frequency subbands were introduced by the encoder. This allows treating samples in each subband differently according to the different abilities of the human ear over different frequencies. This technique allows a higher compression ratio. Finally, the polyphase fillter bank transforms the data from the individual frequency subbands into PCM samples. The PCM samples can now be fed to a loudspeaker or any other output device through appropriate interface.

The main challenge of the project is a MP3 decoder consists purely of hardware. As depicted in the block diagram, each block can be abstracted as hardware module described by VHDL. It's expected that the only software involvement of the MP3 decoder will be the 'enable' signal sent from the Avalon bus.

III. Audio Output

In the audio signal processing, sampling is to get the discrete-time-domain signal from reduction of the continuous-time-domain signal. To convert analog signals to digital signals, the Analog to Digital Converter (ADC) is used to convert the sampling data to binary bit streams. As the sampling theorem stating, the sampling frequency must be greater than twice the bandwidth of the signal being sampled or the re-sampling will be distortion. For the re-sampling process, the Digital Converter (DAC) is used to convert the digital signals to analog signals.

For our audio out portion, we use WM8731 Audio CODEC (combining ADCs and DACs) provided on the DE2 board. The WM8731 are stereo CODECs which consumes low power. It is designed specifically for portable MP3 audio and speech player and recorders. It supports for 16-32 bits digital audio input word lengths and the supporting sample rates are from 8 kHz to 96 kHz.

It can be controlled by the serial I2C bus interface. While the MP3 file is being decoded, we can feed in the decoding bit stream into WM8731 Audio CODEC through I2C bus interface. The WM8731 Audio CODEC will do the digital to analog signal conversion. It has controller to control the buffer in the WM8731 Audio CODEC through BLCK signal. The master clock (MLCK) will synchronize the internal operations of the conversion. And the Left Right Clock (LRCK) will select the left or right channel to the output portion of the data. We use line-out port on the DE2 board to output the audio.

IV. Frequency Analyzer

After the decoding from the MP3-decoder block, we are ready to have 16-bit wide discrete signal. These data is first to be buffered in some kind of FIFO waiting to be processed.

As one of the deliverable we are to implement a simple DSP as a peripheral to the niosII processor. The main function of this computing element is to output information in frequency domain based on its input signal. Since all we want is a floating histogram which shape is controlled by frequencies of some audio signal, we choose a relatively simple and fast algorithm—known as the 16-bit radix-2 decimation-in-time(DIT) FFT algorithm(one of the most common Cooley-Tukey FFT algorithm) as the principle of the computing mission. It's required by the algorithm to build hardware components which can execute complex addition and multiplication. These components will make use of the internal multiplier and adder of the FPGA. To turbo computation speed all the twiddle factors are pre-calculated and stored in ROM. Finally this simple DSP will be designed to output a 16-bit-wise data.

With the frequency information we now have some way to make the floating histogram. Our idea is implementing a hardware that can manifest a rectangular on the screen whose length varies as the input varies. The output of the frequency analyzer is then directly transmitted to the input of the rectangular generating hardware to control the shape of the rectangular. We expect to make use of the result in Lab3 to aim our VGA design.

V. VGA display

The result that frequency analyzer generates is to be displayed as histogram on the screen. Therefore it is necessary to build one or more Avalon slave peripherals to control VGA display. We are going to build two peripherals to implement the display: VGA controller and VGA raster. VGA controller is in charge of transforming the frequency-amplitude information from the frequency analyzer into parameters of histogram that we are going to display while VGA raster's job is to take these parameters from VGA controller and show the histogram on the screen (similar to what the raster in the lab3 did).

The output signals of frequency analyzer should be a vector, which correspond to the amplitudes of certain frequencies at a certain time and these signals should vary from time to time. The task of VGA controller is to receive output signals from frequency analyzer, change these signal to the heights of different rectangles (about 20 in total) in proportion. These rectangles represent different frequencies and their width should be set as the same. The whole histogram should look like those on the classic media players.

VGA raster will specify the synchronous area, blank area and most importantly the area of rectangles representing amplitudes of different frequency in the histogram. To be concrete, there is a pointer with x and y coordinates scanning the screen from left to right, up to down. When the pointer is moving in the area of rectangles in the histogram, the height and width of which are set by VGA controller, certain signal will turn from 0 to 1 and thus this area on the screen will light up.

VI. MP3 file Storage Options

To test whether our MP3 player can actually work or not, we need to store MP3 files in the memory and play it out. There are several options for MP3 file storage. The first one is SRAM, which is the simplest one but also the most easy one to implement. It can only store a really short part of music but we think it is enough for the test in the beginning. After we get positive testing result using SRAM, we then will try to store MP3 file in the device with bigger storage capacity so that we can play out entire piece of music. Choice will be made between SDRAM and SD card. If we have more spare time, we will go for SD card otherwise we will use SDRAM since SD card is a much bigger memory but is much harder to implement.

a. SRAM

First we will try to use SRAM on the DE2 board. It is organized as 262,144 words by 16 bits and has high performance. The SRAM can store the MP3 file to decode and it may store the MP3 decoding data buffer from the Avalon bus. When decoding process and analyzing the frequency spectrum, the buffer can be load into the Avalon bus. While analyzing, the frame to display the frequency analyzer on the screen can also be buffered into the SRAM and then read from SRAM to display the frame on the screen.

b. SDRAM

Since the SRAM is only 512 KB, which is really limited space for MP3 file. We may then try use SDRAM to store MP3 files as well as the buffers for decoding on the DE2 board if necessary. It has organization as 1M by 16 bits by 4 banks, totally 64Mbits.

All the signals of the SDRAM can be generated by using the SOPC Builder to provide the SDRAM Controller, except for the clock signal. The clock signal is very important for the proper operation of the SDRAM and it is provided separately. The clock skew arises from the physical properties of the DE2 board. If we have to use SDRAM, we will use the Phase-Locked Loop to meet the clock skew requirements for SDRAM. A phase-locked loop circuit can be generated from Quartus II Mega function called ALTPLL.

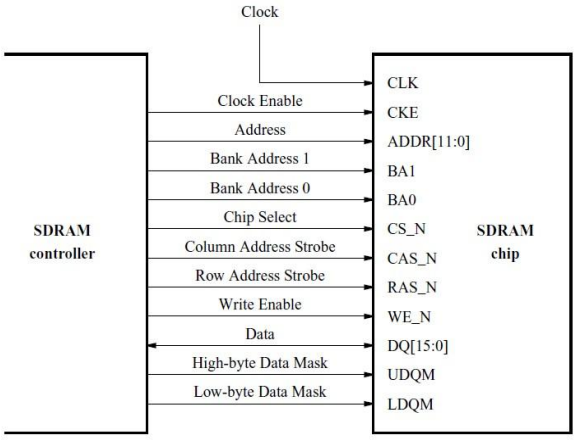

Fig. 3 SDRAM schematic

c. SD Card

If time permit, we will store MP3 files into SD card instead of SDRAM for this project. The SD Card

supports three modes, one of which is called SPI (Serial Peripheral Interface) mode and can be supported by DE2 board. So we will use this mode for our project. We need an Avalon slave peripheral to control the SD card reading actions base on the SPI protocol. This SD card controller should be able to receive command from CPU and then send command to SD card as well as read and buffer data from SD card.

In addition, in order to distinguish MP3 file from files of other type and select the file we want, we need to use hardware to build up a FAT file system.

VII. Milestones

Milestone 1 MP3 decoder: Huffman decoding and IMDCT

Milestone 2 MP3 decoder: The rest part of the decoder and Audio output

Milestone 3 The Frequency analyzer and VGA display

If Time Permitted SD card storage