Amptraxx 2

Daren Ruben, Earl Maier, Talitha Rubio, Matt Webb

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Amptraxx 2 can be best described as a network enabled, centralized DSP, distributed speaker system. This system is not like just any other distributed speaker system. The System allows a user to switch output channels on the fly at an unlimited number of locations while monitoring the status of each location all through a mobile device. Amptraxx 2 comprises of a DSP unit performing filtering and digital conversion over Ethernet, while each breakout box receives the audio over Ethernet and outputs eight independent audio channels.**

***Index Terms* — Audio system, Audio user interface, Client server system, Digital filters, Digital signal processing, Ethernet network, Microcontroller**

I. Introduction

Amptraxx 2 requires an existing Ethernet network to utilize its full potential. The use of Dante audio-over-Ethernet by Audinate technology enables the distributed speaker system to utilize existing standard Ethernet networks to transmit up to 512 channels simultaneously. Audio can be transmitted to the Dante network from any computer with a 100/1000Base-T Ethernet port or from the centralized DSP box.

The centralized network digital signal processing unit is responsible for all signal processing tasks. It contains all of the logic necessary to apply equalization and dynamics processing to input and output audio channels. The unit also contains all user interface devices pertaining to signal processing. The physical user interface is through the LCD screen and rotary-encoder knob located on the front panel of the unit. The unit is also the home of the web server that serves the status of the processing, the processing parameters, and the identity of each breakout box. The Internet based user interface is available on any mobile device and is optimized for mobile phones running iOS or Android.

Audio is extracted from the Ethernet network via what we are calling "Break-out Boxes". Each break-out box houses a microprocessor, a CODEC, a digital audio transceiver, a LCD screen, a rotary encoder, a single button, a 7-port gigabit switch, and a stereo class-D amplifier. A breakout box is able to output eight independent audio channels. Two of those channels are amplified up to 90 Watts RMS output. The integrated 7-port gigabit switch means that the boxes are easily daisy-chainable, providing two extra gigabit RJ-45 jacks. Breakout boxes can easily be strewn across a large area with a minimal amount of cable compared to systems that require analog signals to be driven from a centralized node.

II. Specifications

Amptraxx 2 was designed to satisfy audiophiles, while catering to professionals through its reliability, flexibility, and low maintenance. Below is a list of specifications for the audio system.

A. DSP unit

- Input and output 8 channels at 96kHz/24-bit audio words
- Digital signal processing for each channel.
  - Four EQ Bands (Parametric, LPF, HPF)
  - Three filter topologies (Bessel, Butterworth, and Linkwitz – Riley)
- Web-based monitoring:
  - View breakout box status and channel selections.
  - Change equalization, routing, and compression.
  - Check channel clip status.
- Dante Audio-over-Ethernet
- Front Panel controls and Display Screen
- 1U rack size.

B. Break-out Box

- Output any eight channels via Dante Audio-over-Ethernet
- Stereo Class-D amplifier (2x90W RMS @40 Ohms)
- Simple human interface to modify settings
- Front Panel controls and display screen allow for volume change and channel display.
- Compact size for mounting on rear of a speaker

III. Digital Signal Processing

For the digital signal processing (DSP) component of this project, various processors were considered and researched in order to select the ideal part that corresponds adequately to the specifications. The Blackfin processor, BF537, by Analog Devices Incorporated has been determined as most fitting of the signal processing criteria for this project. The development of the signal processing applications for the ADSPBF537 processor takes place on

a PC running a Linux OS. The applications are compiled with an open source Linux alternative (uClinux) and the image is loaded onto the board. The uClinux distribution used for this board supports application developed in C, C++, and assembly language as does the full Linux OS. A PC is used to program the digital signal processing algorithms such as parametric equalization, high and low pass filtering, and band pass filtering. Such algorithms are developed in C and debugged on the PC, which is running a Linux OS containing the appropriate C and C++ interpreters.

A. UART

In order to control the channel routing and filter parameters that the user needs in real time, the Stellaris MCU needs a method to relay this information to the DSP chip. Our design uses the Universal Asynchronous Receive and Transmit (UART) protocol to allow this communication. The UART protocol is an essential method for sending bytes of information between different components of a system by sequentially transmitting bits of data and reassembling them into the original format. It uses special bits to indicate the start and end of a byte of information which eliminates the need for the sender to send a clock signal to control the receiver. As long as both components agree on a communication baud rate, this property can be taken advantage of. An initial requirement for the interface between Blackfin and Stellaris is the development of a standardized data stream. This data stream needs to identify all of the characteristics of the incoming data. Three basic operations are needed. The first is for Stellaris to have the ability to receive digital signal processing parameters from Blackfin. The second requirement of the standardized data stream is the ability to write a digital signal processing parameter to Blackfin. The third requirement of the standardized data stream is the ability for Blackfin to issue notifications to Stellaris.

The decision was made to create data stream types modeled after microprocessor instruction types. Based on the interface functionality requirements outlined above it was decided that four (4) data stream types should exist. The first two bits define the data stream type. Each data stream is 8 bytes long and always starts with 0xFF. The four types are Read Value, Write Value, Acknowledge/Respond, and Notification. The first two bits of all four types dictate the type. These first two bits are the only bits common to all four types. The read data stream type is used to request digital signal processing parameter values from Blackfin. Software on Blackfin is used to find the current value of the requested parameter. After the value is found it is populated into an Acknowledge stream data type. This acknowledge data type contains the unique message identifier sent to it by Stellaris in the read data type as well as the floating point representation in IEEE-754 of the requested value.

The write data type is used to change the digital signal processing values on the Blackfin when a change is initiated by Stellaris. The channel identifier, feature, parameter, a unique identifier, and the floating-point value are sent in that data type. Software in Blackfin changes the value appropriately, recalculates the digital signal processing IIR filter coefficients, substitutes the new values into the system while maintaining zero down-time, and then constructs an acknowledge data type to send back to Stellaris. This acknowledge data type contains the unique identifier initially sent as well as the IEEE-754 representation of the floating point value that actually written to Blackfin. If the floating point number returned to Stellaris is not equal to the value intended to be sent then the message will be sent again, up to five times. Upon failing five times the system will log the failure.

B. TDM

Time-Division Multiplexing allows multiple channels of data to be transmitted on the same line seemingly simultaneously. TDM is actually nothing more than multiple channels of data taking turns on one data line. TDM technology was created to multiplex multiple channels of data onto a single line. A telephone network with a one to one call to wire ratio would be absurd. For that very reason Bell Labs began encoding 24 calls onto a 4-wire copper trunk between switching stations. TDM divides time into discrete intervals that provides the framework for a logical system capable of carrying multiple channels of data simultaneously. Each sub-channel, or data channel, fits into a time slot. Sub-channel 1 takes time slot 1 and sub-channel 2 would take time slot 2. Each TDM frame consists of an integer number of sub-channels, a synchronization channel, and occasionally an error-correction channel. In TDM, Time-Division Multiplexing, a slot (or channel or time slot) is the data word as well as possible padding bits that provide a convenient interface between DSP and device. Multiple slots make up a TDM frame. A frame begins with a Frame Sync pulse that defines the length of the slot. Frame sync delay also plays a part in the TDM format. Zero, one, and two bit delays are common for audio information transfer.

IV. DIGITAL FILTERING

Various filters useful for audio processing are programmed and the operations are optimized for efficiency and run time using appropriate data structures. The C programming language is used for developing these

algorithms since the Blackfin processor not only supports it, but is designed to optimize the running of C applications. The program that was created is hardcoded with multiple s domain section which contain polynomials for various filters types and orders. The program then uses the bilinear transformation to transform the s-domain coefficients to z-domain coefficients. A direct form II filter is returned and applied to Blackfin with a preexisting function (iir_fr16). This function implements a biquad, transposed direct form II, infinite impulse response filter.

A. Parametric Equalizer

Parametric equalizers allow finer and more sophisticated control of audio than other equalizers. There are three parameters associated with it: amplitude, center frequency, and bandwidth. The sound engineer can adjust the gain specify the center frequency and adjust the bandwidth around this frequency on which to apply the gain. The precision of parametric equalization allows for targeting and removal of unwanted noise or feedback in an audio transmitting system and for enhancing and sharpening the desired portions of an audio signal. The s-domain transfer function for parametric boost (1) and cut (2) are listed below.

$$H_B = \frac{s^2 + \frac{K\Omega_o}{\Omega} s + {\Omega_0}^2}{s^2 + \frac{\Omega_o}{\Omega} s + {\Omega_0}^2} \qquad (1)$$

$$H_C = \frac{s^2 + \frac{\Omega_o}{\Omega} s + {\Omega_0}^2}{s^2 + \frac{K\Omega_o}{\Omega} s + {\Omega_0}^2} \qquad (2)$$

$\Omega_o = 2\Pi F_o$, F_o is the center frequency in Hz.

$Q = \frac{F_o}{BW}$, BW is the 3dB bandwidth in Hz.

$K = 10^{\frac{G}{20}}$, G is the desired boost or cut in dB

B. Filter Topologies

There are three filter topologies to choose from, from the web UI; Bessel, Butterworth and Linkwitz-Riley. Bessel functions are widely used in the audio industry for crossover. Bessel filters are characterized by almost constant group delay across the entire pass band, thus preserving the wave shape of filtered signals in the pass band. Below in Figure 1, you can see the comparison for group delay between the three filter topologies, the Butterworth filter has the longest delay, while the Bessel as the shortest, which makes for a quick, precise filter application. Linkwitz-Riley is similar to the Bessel; it is also used in audio crossovers. Looking at Figure 2 you can compare the three topologies magnitude for a 4th other filter. Linkwitz-Riley filter is created by cascading two Butterworth filters resulting in a -6 dB gain at the cutoff frequency or cascading a low pass filter and high pass filter yields an all pass filter with 0 dB gain. The Butterworth has the sharpest initial cutoff frequency and a 3 dB sum at crossover.

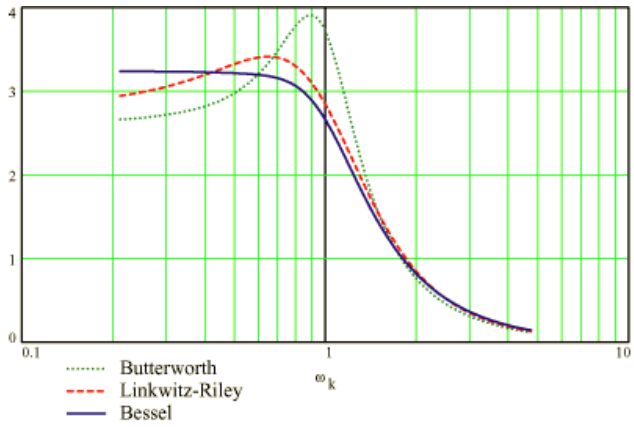

Fig. 1. Fourth-Order Group Delays

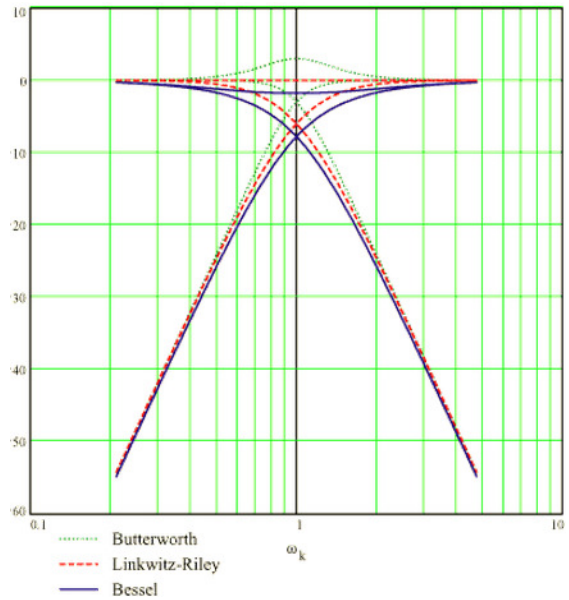

Fig. 2. Fourth-Order Magnitudes

V. Microcontroller

The Stellaris microprocessor is responsible for everything besides audio processing. As noted above, all audio processing is taken care of by the very capable Blackfin digital signal processor. The Stellaris

microprocessor was chosen for several reasons. The reasons include, but are not limited to: the availability of great development tools, ease of display interfacing, on-board Ethernet stack, and the large array of available interface formats (I2C, CAN, SPI, etc). The Stellaris chip also interfaces with the Dante Brooklyn-II module to set channel selections, to retrieve channel names, and to check system status bit.

The availability and accessibility of a web server on Stellaris really opens a lot of possibilities. Instead of the centralized DSP unit only being physically accessible via an LCD screen and a few rotary encoders, it will always be available on a network and possibly even on the World Wide Web. The web server is capable of serving the status of every breakout box on the same local area network (LAN). It will also give the user the ability to change the routing matrix. In addition, the user is able to change the settings of equalizers, and dynamics, processing for every input and output channel.

VI. CODEC AND DIGITAL AUDIO TRANSCEIVER

The CS42448 codec by Cirrus was designed for home theaters and automotive audio systems to encode high quality multichannel audio. This CODEC has six 24-bit resolution A/D converters along with eight 24-bit D/A converters and has a sampling rate up to 192 kHz. The CODEC also allows for digital volume controls for each individual DAC and ADC. Stellaris utilizes this feature by allowing the web UI to control the volume for each of the converters. I2C is used to communicate between the CODEC and Stellaris, data sent and received depending on the SCL clock. The serial audio port allows up to six DAC channels and eight ADC channels in a Time-Division Multiplexed interface format. The CODEC is connected to all the outputs of the Breakout box through output filters.

The DIX9211 is used to input SPDIF into the DSP and output SPDIF to the CODEC in the breakout box. The DIX 9211 is used to communicate between the Blackfin DSP and Dante. The DIX makes it possible for SPDIF input to be inputted and routed anywhere.

The CODEC is connected to the Stellaris and the DIX in both Breakout Box and DSP Box. The CODEC only connects to Dante in the Breakout Box, while in the DSP box it only connects to Blackfin. This connection is done by the resistor MUX; by not populating the corresponding resistor. This resistor MUX allows for the two boxes to have identical PCB layouts, but different device populations (i.e. Blackfin is not populated in the Break-out box).

VI. DANTE

An NDA agreement was required in order to use this technology. Dante was designed to make professional audio networking easily accessible for audiophiles. Dante creates the ability to manage a network and makes plug and play networking possible. A Dante system can send data from 100 Mbps to 1Gbps making it a modern high speed performance digital media transport system that runs over standard IP networks. With Dante, each channel can be given a name that corresponds with its location or audio type, which makes troubleshooting effortless. Dante has the ability to send a maximum channel depth of 1024 or 512x512 bidirectional channels to be transmitted and received over a single Ethernet cable. Dante enabled devises have the ability to discover one another over the network and learn each other's capabilities such as sample rates and bit depth, which makes adding a new zone literally as easy as plugging in a new break-out box.

Dante produces our master clock throughout the system. This clock harmonization is completely autonomous of the audio data and the sample rates being executed on the network, which gives the ability to process the incoming or outgoing audio data quickly without waiting. The local clock is used to time stamp network packets, to control the rate at which audio sample are transmitted and received.

VII. POWER SUPPLY

Amptraxx power supply was designed around two specifications, low noise and low maintenance. The same power supply is used for both the DSP box and Break-out box. For our project, both power supplies need to deliver constant DC voltage, while the break outbox needs an addition 120 VAC power signal to oblige the class D amplifier power specifications.

The power supply is obtaining its electrical power from an ordinary 120VAC outlet at 60 Hz. We are using a 120VAC to 24 VAC center tapped transformer to accomplish +/- 12 VAC. TI's TL750M12 is a fixed linear regulator that regulates at a 12V voltage and has the ability to supply a current of 750mA. For the negative 12 volt supply we are using the UCC284-12, which is a negative linear-regulator. These regulators is used to drive op-amps that balance outbound audio.

We chose the LM26003 synchronous switching regulator for many reasons; it allows for external clock synchronizations, it has a wide input voltage range, it contains "power good" flag and it has an adjustable switching frequency. The LM26003 is used to produce the

3V and 5V digital and analog voltages. All of the switching regulators are synchronized with an external clock. The analog and digital switching regulator are regulating inversely with respect to each other at 384 kHz (four times our sampling frequency), which insures that minimal noise is generated.

A blue LED located in the front panel illuminates when all of the switching regulators are working properly. Two NAND gates are used to NAND the output signal from the three switching regulators. The switching regulators output a low signal on the "power good" (PGOOD) pin when the output falls below 89% of the normal operating supply. When the output voltage returns to within 95%, measured by the feedback pin, PGOOD returns back to its high state. Other voltages such as 2.5V, 1.8V and 1V will be using low-dropout regulators (LDO). We are using the TLV1117 family that offers positive low-dropout voltage for 2.5V and 1.8V and TLV71210 for 1V.

VIII. User Interface

There are several ways to meet the base user interface requirements. They can be divided up into three categories: network control, wireless control, and local control. Network control would involve the device being monitored and configured via Ethernet over a local-area-network. The device would be assigned an IP address that would be accessible by any other web-enabled device on the network. This could include iPhones, iPads, desktop computers, Android phones, and Android tablets. Wireless control would involve a connection with another device via Bluetooth or an Ad-hoc network. Local control would entail physical controls and displays on the digital signal processing box. This would most likely come in the form of multiple rotary encoders, several push buttons, and a 3-4" diagonal LCD screen.

The third and final control option for the system works over Ethernet on a local area network (LAN). There are several inherit advantages to operating on a LAN. For instance, the system can be configured by any computer or device connected to the LAN. This alleviates the need to be physically close to the system. The largest advantage is that the system can be configured on a multitude of devices in a limitless variety of form factors. A mouse or a keyboard or a 20" LCD monitor is not outside of the realm of possibility. There are also devices such as the iPad, iPhone, Android phones, and Android tablets that can configure the system easily using the exact same software. The same software can be run on each device because there is a common runtime environment, a web browser that meets international web standards. Due to the numerous key advantages of Ethernet control via web browser, this project will utilize local area networks and existing devices with web browsers for configuration and monitoring.

The decision to utilize control over Ethernet requires the end user to be able to view and change the network settings of the DSP box. If the end user cannot view or edit the settings then they will be unable to control the DSP box. There are three parameters that must be viewable and editable via physical controls on the DSP box. They are IP address, gateway address, and subnet mask. Each address is in the format "xxx.xxx.xxx.xxx" where x is a number between 0 and 9. A 16-character display must be used to be able to view the whole address simultaneously. A method of changing between different addresses must be included in the user interface design in addition to a way of rotating through the numbers. It would also be beneficial to the end user to see signal and/or clip LED's on the front of the DSP Box. If a clip LED is lit up on a particular channel the end user would know that there is a problem that must be addressed. A signal LED would assure the end user that their Dante virtual sound card audio is actually reaching the DSP Box.

A. User Interface Technology

Web interfaces in 2011 are generally a combination of more than three high-level programming languages. Two of the three languages are almost always HTML and JavaScript. HTML has been around since the beginning of the Internet browser and has been revised over the years to support languages such as JavaScript which allow the user to manipulate the browser Document Object Model (DOM), containing the elements on the web page, in helpful ways. The DSP project interface will utilize HTML and JavaScript.

The real user interface magic occurs when jQuery Mobile is added into the equation. jQuery Mobile is a "touch-optimized web framework for smartphones & tablets" according to their website splash page. jQuery Mobile supports a very wide range of device operating systems including iOS, Android, Blackberry, webOS, and Windows Phone. jQuery Mobile makes a pass on the page after the HTML has been initially generated and makes it much more mobile interface friendly. For example, it is able to convert standard hyperlinks to buttons that are easy to touch with your and that are aesthetically pleasing to the eye. Not only does it do a great job at converting things easily, it is also incredibly easy to implement. Another neat feature of jQuery Mobile is its ability to change the look of typically ugly forms with mouse-required interaction to touch friendly forms that actually look like

native device user interfacing. Another core feature of jQuery Mobile is AJAX page loading. AJAX is an acronym for Asynchronous JavaScript and XML. AJAX enables the end user to make page requests in the background, meaning that the browser never loses responsiveness or leaves the screen blank while the next page is downloaded. The end user's experience is much more fluid when AJAX is used to load pages. jQuery Mobile also works very well on a standard desktop computer browser.

As previously mentioned, most websites use three programming languages. The two most common languages are HTML and JavaScript. These two languages allow the Document Object Model (DOM) to be generated and manipulated with great ease. Typically the third language is very dynamic in nature, such as Ruby, ASP.NET, or PHP. These third languages typically possess the ability to create HTML and JavaScript themselves. They also easily facilitate data transfer between a large data structure and the DOM. JavaScript and HTML are not inherently good at this. Therefore, the DSP project needs a third language to help with data transfer between the embedded Stellaris system and the DOM. The Common Gateway Interface (CGI) enables a web server to delegate web requests to executable files. The executable file in this project will be function calls in the firmware of the Stellaris microprocessor. The executable code will control the Stellaris hardware according to the web server request. A web-based application programming interface (API) was created to facilitate structured web requests which correspond to C based function calls in the Stellaris firmware.

B. User Interface Implementation

The user interface is divided into several different screens. The main screen, as shown in Fig 3 gives you the option to change the routing matrix, process inputs, process outputs, view processing status', or change labels.

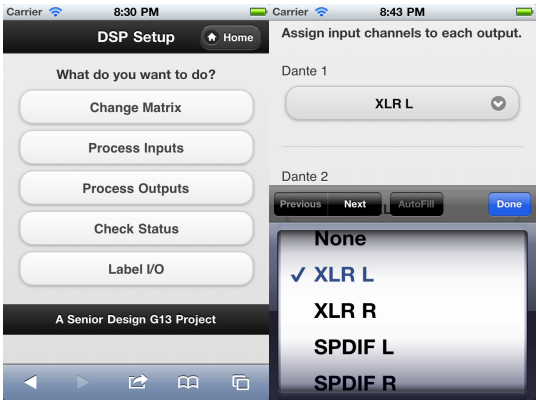

Fig. 3. Left, "Home". Right, "Change Matrix".

Fig 4 left shows the screen immediately after selecting "Process Inputs" on the Home menu. The end user is given the ability to select an input channel from all of the active channels (inputs which lead to an output). After selecting a channel the end user is brought to a processing type selection screen, not shown. If the end user clicks on equalization they are taken to the screen shown on the right half of Fig 4. From there the user is able to view or modify the filter parameters. When the user hits update UART commands will be sent to Blackfin to change the filter parameters.

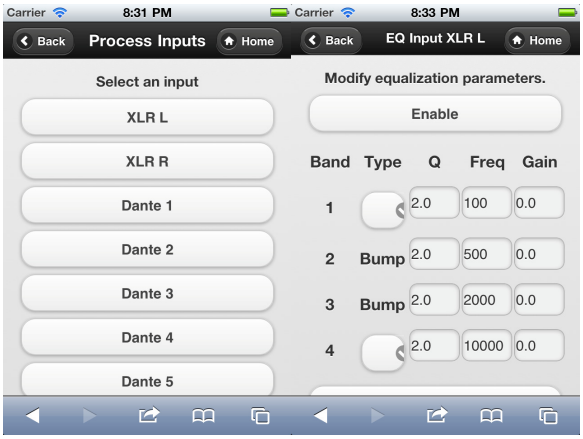

Fig. 4. Left, "Process Inputs". Right, equalization parameters view, where a user can view or change equalization on single input or output channel.

All of the data structures, which provide data encapsulation for a channel's routing, equalization, and other digital signal processing are accessible as, needed by the web browser. Data is fed to the DOM (web browser) from the Stellaris microcontroller in JavasScript Object Notation (JSON). A JSON library was written to turn key-value maps into a string that is JSON compliant. Key-value maps were created for arrays of channels and their

attributes. These attributes include their active/inaction state, their channel number, and their assigned name. Maps were also created for equalization parameters, compression parameters, and routing parameters.

A single function parses every file system request and decides whether it is a valid API function. If it is a valid API function there is logic to return a pointer to the appropriate JSON that needs to be returned to the web browser. The web browser requests this JSON data via HTTP GET requests. The URL is of the form "/a/{i or o}/{channel number}/eqparams" to request the equalization parameters for a specified input or output channel. After the asynchronous HTTP GET request is returned to the browser the JSON is parsed by jQuery. After it has been parsed the data is populated in the appropriate HTML tags for the end-user to view and interact with. In order to modify digital signal processing parameters HTTP GET requests are made asynchronously to the Stellaris microcontrollers, but this time at a different API endpoint with query parameters attached. For example, to modify the first equalization band on a input or output channel a GET request would be sent to "/a/{i or o}/{channel number}/modeq?band=1&type=1 &freq=1000&gain=3.15&e=1&q=4". When a digital signal processing parameter is changed that change must be reflected in the Blackfin processor which is actually processing the audio. Otherwise the user is making a visual change on the user interface without any actual processing difference occurring. This is handled by generating ApiWrite data structures that are queued in a circular buffer. A timer interrupt fires every 50ms and transmits every byte in the circular buffer over UART to the Blackfin. Once the change has occurred on Blackfin, Blackfin sends an ApiAck data structure back to Stellaris over UART to inform Stellaris that the change has, in fact, occurred.

The last concern with user interface technology centers around the end user's ability to view and change the current network settings on the front of the DSP box. A VFD 2 line by 16 character display was chosen in order to show the IP address, gateway address, and subnet mask. This particular display is very simple to interface with a processor and does not require writing a complex driver. A single rotary encoder with push button is used to change the value of any of the three addresses. Enabling the end user to switch between the three addresses is a single push button that simply changes the number shown on the screen to the next address. Clip and signal LED's in the audio industry are generally green and red, respectively. One clip and signal LED per channel would be appropriate. The clip light would turn on if that particular output or the corresponding input were clipping. The signal light would turn on if that particular output exceeds a certain signal level threshold.

IX. Amplification

After the processed audio channel is sent to the breakout box it needs to be reproduced over a speaker. Since designing a quality audio amplifier is a senior design project all in itself we purchased an amplifier from ABLETEC. ALC0180 is a high quality stereo amplifier that is bridged capable. This particular amplifier can deliver 90Wrms to two 4 ohm speakers or 50Wrms to two 8 ohm speakers. When the amplifier is bridged it delivers 180 Watts to a single 8 ohm speaker. Being able to be bridged greatly increases our design option. This amplifier has its own power supply that only requires 120 VAC. Other features of the ALC0180 consist of over current protection, over temperature protection and over voltage protection. These features can aid in design of temperature sensor or overall monitoring system.

X. Audio Inputs and Outputs

A. Incoming Audio

Incoming RCA signal are sent through a low pass to protect from any attenuating noise energy. A voltage is created to lift the signal by 2.2V to prevent clipping. Once the signal passes the filter it is sent through a single to differential active filter. This filter rejects signal within the stop band and create a differential signal.

B. Outgoing Audio

The CODEC is converting digital audio into two balanced analog signals. We need to add a buffer and line driver in order to send the audio signal with minimal noise and protect the Codec. To drive the XLR balance signal we are using the THAT 1606 configuration. To protect the signal against radio frequency interference, diodes, ferrite leads and capacitors are used.

To drive the RCA we are using a LMV358 to create a low pass filter because of the possibility of aliasing from the CODEC. The low pass filter is followed by a ferrite bead along with a transistor which triggers if incoming current is detected. This will protect the CODEC if one was to mistakenly hook up an incoming signal to the output port.

XI. Conclusion

The network enabled, centralized DSP, distributed speaker system is designed to simplify the distribution of

audio. Any existing Gigabit or 100Mbps Ethernet infrastructure can accommodate this system and can easily be used to distribute the audio over the desired location. Not only is this speaker system easy for setup but it is able to change the audio being distributed through multichannel audio distribution capabilities of Dante.

ACKNOWLEDGEMENT

We would like to acknowledge and thank all of the support we received from Alcorn McBride Inc. Alcorn McBride's Director of Engineering, Jim Carstensen, has been instrumental in the success of this project throughout every step of the process. Without him this project surely would have been impossible. A special thanks must also be given to Adam Rosenberg of Alcorn McBride for his all-hours programming support. **Daren Ruben would like to acknowledge the support by WORKFORCE CENTRAL FLORIDA. Earl Maier would like to acknowledge the support by WORKFORCE CENTRAL FLORIDA. Talitha Rubio would like to acknowledge the support by WORKFORCE CENTRAL FLORIDA. Matt Webb would like to acknowledge the support by WORKFORCE CENTRAL FLORIDA.**

REFERENCES

[1] Chris Hanna, "Real-Time Control of DSP Parametric Equalizers," THAT Corporations, Paper 13-041, AES 13th International Conference (Dec 1996).

[2] Rane Corporation. "Linkwitz-Riley Crossovers: A Primer." *Rane Corporation*. Rane Corporation. Web. 30 Mar. 2012. <http://www.rane.com/note160.html>.

[3] "JQuery: The Write Less, Do More, JavaScript Library." *JQuery: The Write Less, Do More, JavaScript Library*. Web. 30 Mar. 2012. <http://jquery.com/>.

Daren Ruben is a 23-year-old Electrical Engineering major graduating with honors. He has accepted an engineering position with Niagara Bottling LLC and will start working in May 2012. He hopes to extend his Electrical Engineering knowledge by working with PLC in the field and potentially return for his Master's Degree.

Earl Maier, 22, is graduating in Electrical Engineering with a minor in Mathematics in May 2012. He plans to continue onto graduate school in the fall of the same year at the University of Central Florida. He currently is searching for a job in the Electrical Engineering field. He plans to stay within his home state of Florida.

Talitha Rubio is a 23 year old electrical engineering major graduating in May 2012. She has accepted a position with Intel Corporation as a Component Design Engineer. She would like to complete her Master's Degree in the area of VLSI Design.

Matt Webb, 22, is graduating with a degree in Electrical Engineering in May 2012. He has accepted a Systems Engineering position at Texas Instruments on the Apple Products Team. In addition to graduating and moving to Texas in May, he is marrying his beautiful bride, Emily. He plans to one day acquiring his Master's Degree in an Electrical Engineering related field. His passions include programming, live sound

production, audio recording, music, and serving Jesus.