Group Report

Digital Audio Workstation

ECE532 DIGITAL SYSTEMS DESIGN

Lee Tarnow – 993921547

Andrew Rosselet – 994731448

Pete Scourboutakos - 994786335

April 5, 2010

Table of Contents

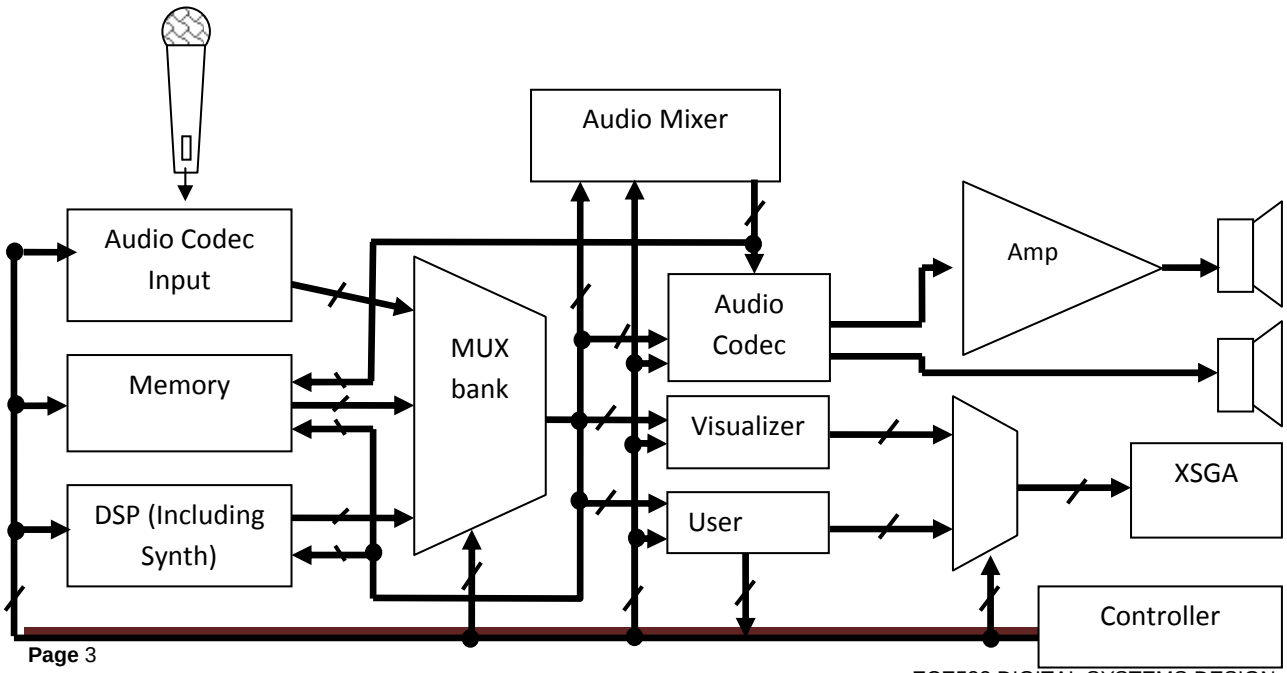
Overview

Digital Audio Workstations usually feature stereophonic sound reproduction, mixing and visualization as well as synthesis of audio effects. Stereophonic sound reproduction consists of using more than one audio channel to create the impression of sound from multiple directions as in natural hearing. Mixing consists of combining, routing, and changing the level and timbre of the signal.

We wanted to implement a mixer and sequencer with an easily accessible software user interface. The user interface will feature controls for transport (play, rewind, record, etc.), track controls, channel inputs (basic input controls, auxiliary send routing), master controls and a visualizer which renders a graphical waveform display.

A microphone and pair of speakers as well as a VGA monitor is required to test complete functionality of the system. The IP required to build the system comes mostly from Xilinx, however in the case of the audio codec, we needed to search around forums for find help making the hardware IP work and to find a working driver.

The DSP, MUX Bank and Audio Mixer blocks proposed in our proposed system block diagram (see below) have all been replaced with sofware control as we had insufficient time to complete implementation of seperate hardware modules.



Outcome

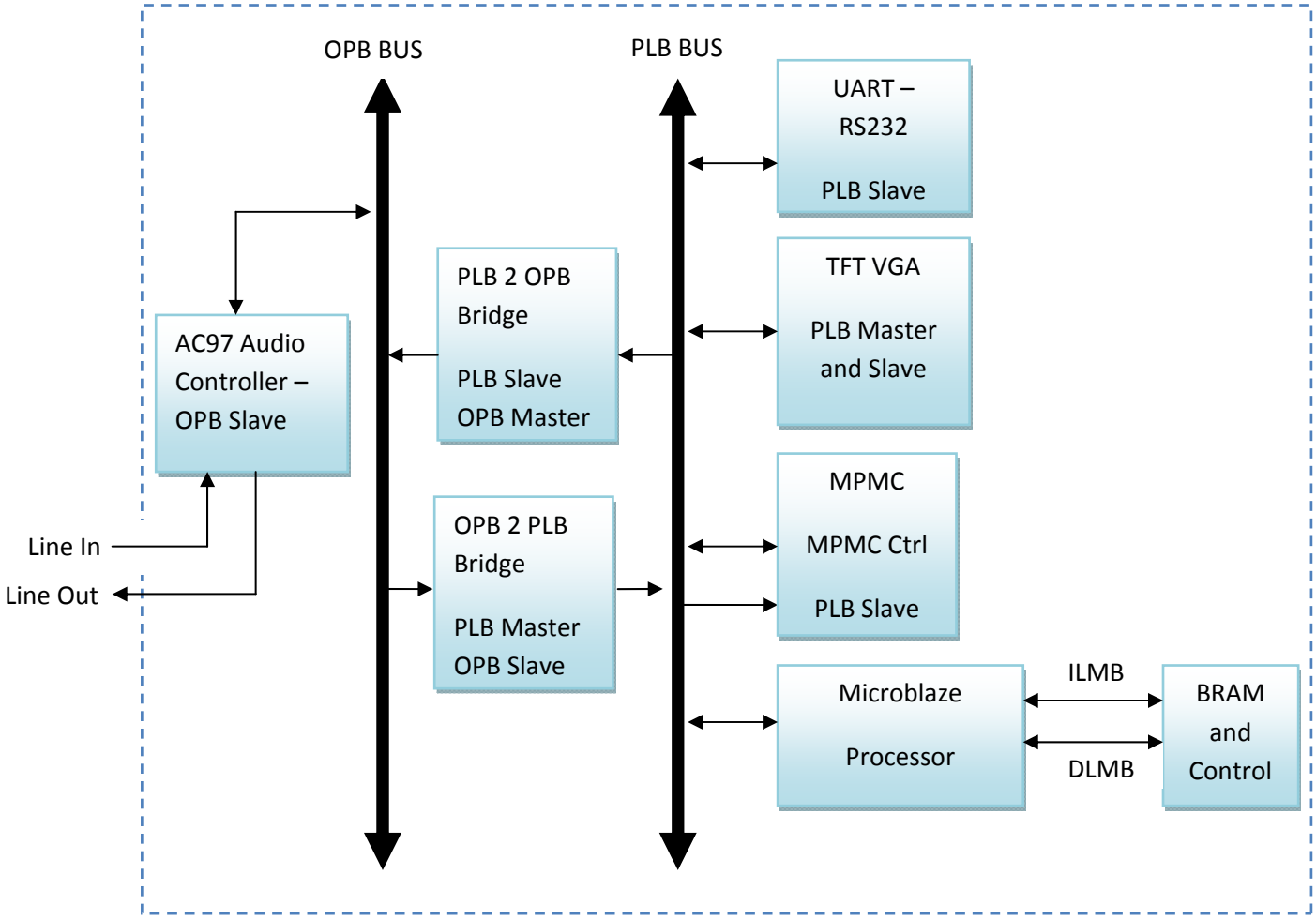


Figure 1: **Project Block Diagram**

Our final design demonstrates sequencing and synthesizing of several audio signals and generation of several audio effects and signals in software. Our interface allows us to record audio into user specified memory locations and we are implementing audio effects and filters in software which are applied to the saved samples in memory.

The visualizer averages a sequence of audio samples recorded in memory and calculates the location to plot a single sample value on the VGA display. The 16-bit audio samples are bit-shifted so the most significant 8 bits are used to render each point on the screen. These 8 bits shift samples plotted above or below the '0' or no signal line by +/- 128 pixels.

Several parameters are used to adjust the mood of the visualizer. Further work on this project could use DSP blocks to detect the tempo, timbre and mood of the audio sample being recorded and use this to set a few parameters which would combine to produce a large variety of visual effects.

The gradient can be switched from "normal" to "psychadelic" which causes the colour palette to be pushed through a shift register so it wraps around at a varying point along the gradient. The colour palette can also be set to any RGB value or a shifting palette between pairs of colours and a tempo detector could be used to set the rate of colour modulation.

The audio portion of this project consists of the AC97 controller and codec. The controller reads serial data frames from the codec for recording. It then writes decoded PCM samples to memory through the OPB, OPB2PLB Bridge, and PLB. This is all handled through the use of memory mapping. The PCM samples in memory are processed and sequenced. As well, synthesized tones can be processed and sequenced in memory. The controller then reads the processed PCM samples from memory by way of the PLB, PLB2OPB Bridge, and OPB, again using memory mapping. Finally, the controller writes serial data frames to the codec for playback.

Description of the Blocks

Microblaze version 7.10.d

The Microblaze soft processor was used to control the program flow, the audio recording and playback as well as output on the VGA display.

opb_ac97 version 2.00a

The audio codec controller receives commands from the microBlaze and transmits audio over the OPB bus. It uses mapped memory on the PLB bus.

xps_tft version 1.00a

The graphics controller also uses mapped memory on the PLB bus

mdm - debug module version 1.00d

Microblaze debug module

PLB bus version 1.03a

OPB_v20 - OPB bus version 1.10c

OPB bus arbiter

PLB2OPB bridge version 1.00a

Bridge which enables communication between the PLB and OPB buses, allows the Microblaze which sits on the PLB to send commands to the AC97 controller on the OPB bus and write recorded audio from memory to the codec.

OPB2PLB bridge version 1.00a

Bridge which enablers communication between the OPB and PLB, allows data to be read off the input codec into the mapped memory attached to the PLB bus.

Multi-Port Memory Controller (MPMC) version 4.03

Hardware IP for interfacing with the 512MB RAM controller

xps_uartlite - RS232 UART version 1.00a

Serial module for communicating with a Hyperterminal session for debugging and sending commands to the mixer/sequencer.

Intc - Xilinx Interrupt Controller version 1.00a

RS232 Interrupt controller

clock_generator version 2.01a

Used to generate clocks for timing both system buses, the audio codec as well as the vga controller.

bram block - version 1.00a

FPGA block ram controller for program instructions and data

dlmb - data local memory bus (BRAM) version 1.00.a

ilmb - instruction local memory bus (BRAM controller) version 2.10a

The TFT controller uses a pair of screen buffers stored in mapped memory in the DDR SDRAM. The pair of buffers enables screen transitions between atomic screen states. Frames are pre-computed and stored the hidden buffer and then the buffers are swapped to display the new image. The VGA header file contains constants defining the shape of standard ASCII characters from 0x20..0x7E and a print routine has been written to output a string on the VGA display.

The opb_AC97 (Wirthlin) is the audio codec controller. It is an OPB slave that provides a register-based interface for the codec. It was found as part of the labs for an ECE course at Iowa State University, and was created by Prof. Mike Wirthlin of Brigham Young University. As illustrated by Figure 1-2, the controller interacts with the audio codec using 5 ports:

- SDATA_OUT: From opb_AC97 to codec. Serial input of data frames for codec sampled on the falling edge of BIT_CLK.
- BIT_CLK: From codec to opb_AC97. 12.288 MHz clock.
- SDATA_IN: From codec to opb_AC97. Serial output of data frames for codec sampled on the rising edge of BIT_CLK.
- SYNC: From opb_AC97 to codec. Defines boundaries of data frames.
- RESET#: From opb_AC97 to codec. Active low hardware reset.

The data frames wrap Pulse-Code Modulaton (PCM) samples going to and from the codec. These are digital codes that are quantized from an analog signal sampled at normal intervals. By writing these PCM samples to and from memory, we are able to record, process, and playback audio.

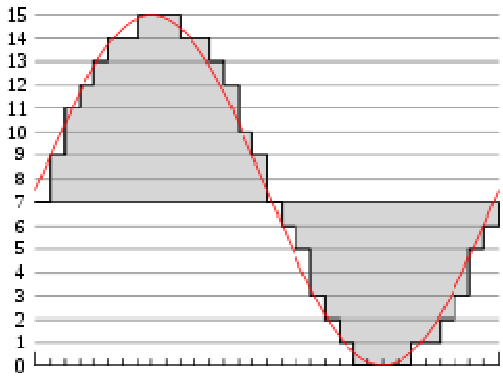


Figure 2: **4-bit PCM sampling and quantization** (Wikipedia)

For operation, the audio codec uses the following software control flow:

- ac97_init(): Initializes codec
- record(32-bit start address, 32-bit stop address):
  Writes data from output FIFO to OPB, then from OPB to the PLB, and then from the PLB into memory
- play(32-bit start address, 32-bit stop address):
  Reads data from memory to PLB, then from PLB to OPB, and then from the OPB to the input FIFO

- clear(): Clears the controller FIFOs that implement the streaming of data to and from the codec. This is important to do after every record and play because the FIFOs define the data frame. If clear is not called appropriately, the controller will behave unexpectedly. As an example, if the FIFOs are not cleared after a record, and record is called again, it will run the risk of overwriting existing Pulse Code Modulation (PCM) samples with new ones. Similarly, if the FIFOs are not cleared after a play, and play is called again, it will run the risk of sequencing old data and will not stop playback accordingly.

- reset(): Resets the codec

Synthesis effects are also available for sine, triangle, and rectangle waves:

- sin(16-bit frequency, 16-bit pulse height): Generates sine wave PCM samples

- rec(16-bit frequency, 16-bit pulse width, 16-bit pulse height): Generates rectangle wave PCM samples

- tri(16-bit frequency, 16-bit pulse width, 16-bit pulse height, 8-bit slope, 8-bit peak_x_loc): Generates triangle wave PCM samples

Subsequently, we are also able to synthesize sawtooth and square waves by virtue of our synthesizing triangle and rectangle waves.

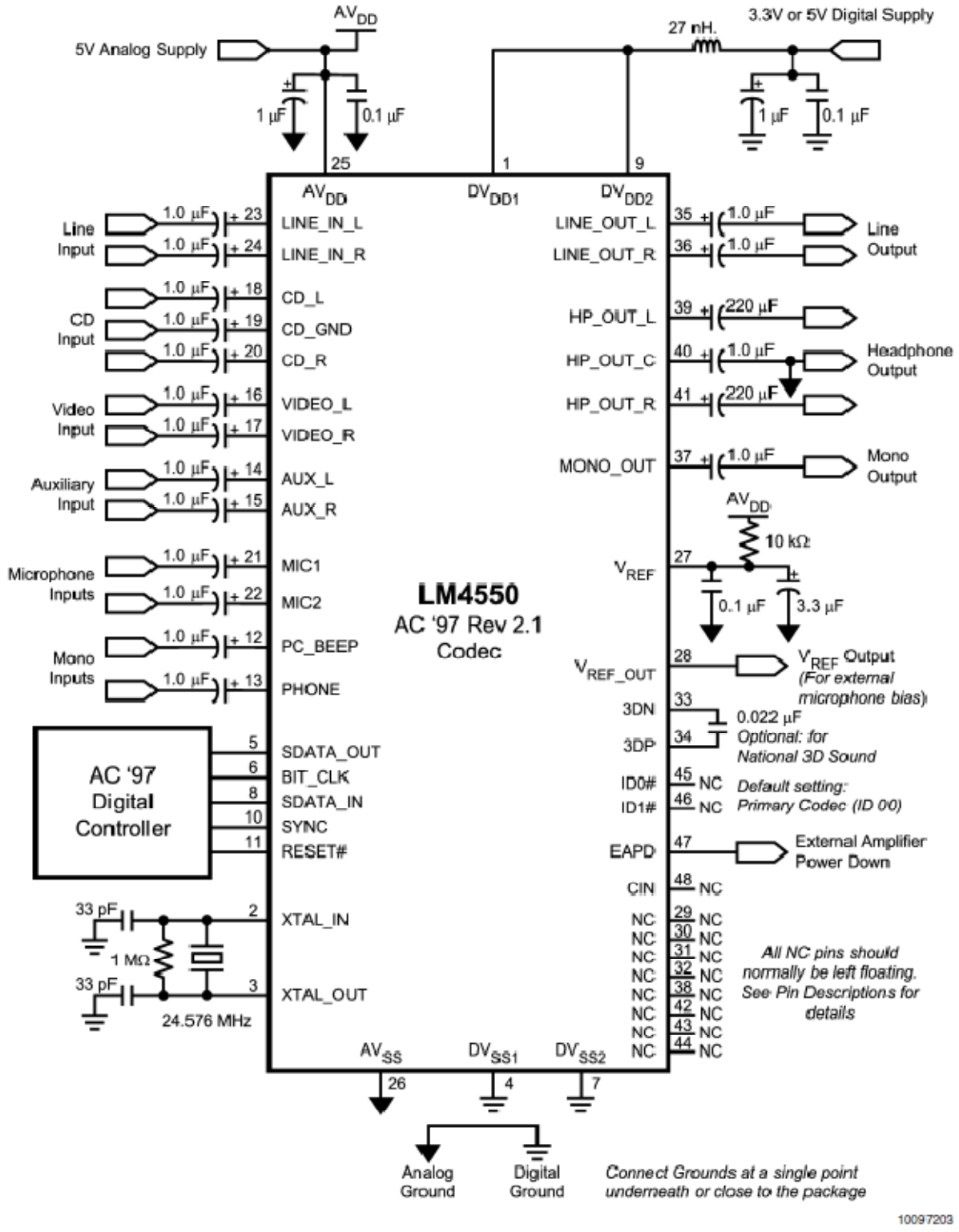


Figure 3: **AC97 Typical Application Circuit** (National Semiconductor)

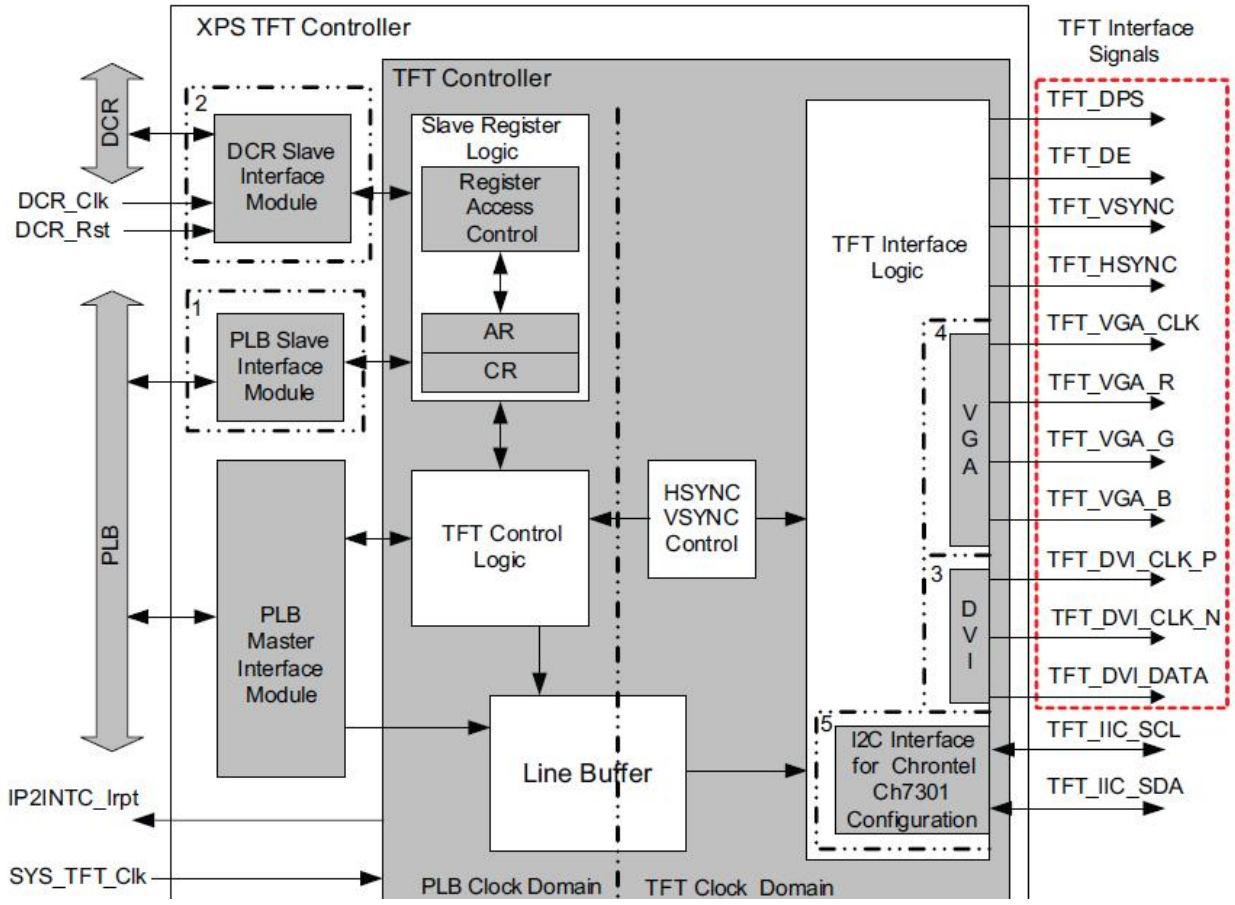


Figure 4: **Xilinx Thin Film Transistor (TFT) Controller**

Description of Your Design Tree

Directory/File	Description
./__xps	Option files for bitinit, libgen, simgen and platgen
./blkdiagram	Block diagram generated by XPS
./code/main.c	The main software control program
./code/header.h	Header file for AC97 controller including sine table, constant definitions, and codec interface
./code/VGA.h	Header file for graphics controller which includes constant definitions and VGA library routines
./data/system.ucf	System Constraints file; external pins' assignment
./drivers	Includes the drivers for AC97 and graphics controllers
./etc	Option files for bitgen and downloading
./microblaze_0	The processor
./pcores	Includes hard IP cores AC97 and graphics controllers
./system.xmp	XPS project file
./system.mhs	System Hardware Specification file
./system.mss	System Software Specification File
./README	Documentation

Table 1: **Design Tree**

References

National Semiconductor. (n.d.). *LM4550 AC '97 Rev 2.1 Multi-Channel Audio Codec with Stereo*. Retrieved from http://www.national.com/ds/LM/LM4550.pdf

National Semiconductor. (2004). LM4550 AC '97 Rev 2.1 Multi-Channel Audio Codec with Stereo.

Wikipedia. (n.d.). *Pulse-code modulation*. Retrieved from http://en.wikipedia.org/wiki/Pulse-code_modulation

Wirthlin, M. (n.d.). Retrieved from http://www.google.ca/url?sa=t&source=web&ct=res&cd=3&ved=0CA0QFjAC&url=http%3A%2F%2Fclass.ee.iastate.edu%2Fcpre488%2FLabs%2FLab_8%2Fopb_ac97.doc&rct=j&q=Mike+Wirthlin+BYU+AC97&ei=uaq5S6XyO4KC8gb7wOHhBw&usg=AFQjCNHu_-D2SyLy5CLW4OYNXm1BRCfPaw

Xilinx. (n.d.). *Xilinx TFT Controller Datasheet.* Retrieved 2010, from http://www.xilinx.com/support/documentation/ip_documentation/xps_tft.pdf