

CECS 561 Spring 2018

Basic System

By

Nolan Luckett

04/06/2018

An audio recorder was built on the Zybo board using a new I2S IP.

# Introduction

An audio recorder was built on the Zybo board using a new I2S IP. This new IP (called LogicBricks LOGII2S) is available from Xylon through Vivado’s IP catalog. A free evaluation license can be obtained from the vendor’s website. The LOGII2S IP allows for 8x Rx/Tx channels to be configured for recording/playback on external audio hardware (i.e. the Zybo’s SSM2603 audio codec) via the I2S protocol.

# Operation

When button 1 is pressed, the Zybo will record audio data from “line in” until it is released. When button 2 is pressed, audio data will be played back from the headphone jack. When button 3 is pressed, audio data will be streamed to the host PC for plotting. If the host python application is running, the streamed data will be plotted in the GUI.

# Theory

Audio data is recorded from the “line in” connector by the Zybo’s audio codec. The audio codec converts the analog signal into a digital PCM (pulse code modulation) signal that is transmitted via I2S to the Zynq SoC. The LOGII2S IP that is instantiated in the Zynq’s PL is connected to the I2S wires from the audio codec.

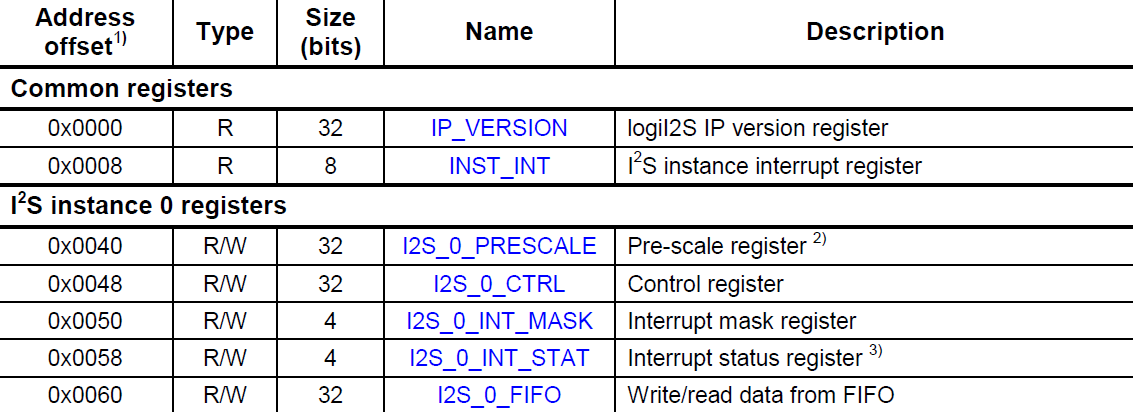
The LOGII2S IP is configured to have 1 receive channel and 1 transmit channel. Each of these channels has registers associated with it as shown in Figure 1.

Figure The LOGII2S's register map. The table is taken from Xylon's documentation.

The LOGII2S IP uses interrupts to inform the system about the state of its registers. Interrupts can be configured to occur when a channel’s FIFO is empty, almost empty (user defined), almost full (user defined), and full. When any channel raises an interrupt

the INST\_INT register can be read to determine the source. The indicated channel’s I2S\_n\_INT\_STAT register can be read to determine which of the four interrupt types caused the event. The I2S\_n\_INT\_STAT register is also used to clear a given interrupt by writing a ‘1’ to any given position. The I2S\_n\_INT\_MASK register can be used to enable/disable the four interrupt conditions.

The I2S\_n\_PRESCALE value can be used to set the bit clock relative to the system clock. For example, if the system clock is 100 MHz then a pre-scale value of 4 would result in a bit clock value of 100 MHz / (2 \* 4) = 12.5 MHz. The factor of 2 is included for the case of stereo transmission, indicating that each channel was operate at an interleaved bit rate of 12.5 MHz.

I2S\_n\_CTRL register is used to enable/disable serial data transfers and configure the values of the word select clocks among other functions minor functions.

The I2S protocol—Inter-IC Sound—uses 1+ data lines, a word select clock, and a bit clock. The word select clock is used to alternate between the L/R channels for stereo playback. For transmission, when the output is a stereo line, the data line consists of interleaved L/R samples as indicated by the word select clock with a serial data rate indicated by the bit clock.

# Hardware Design

Figure The hardware platform for the basic system. It is based on Lab 2 with the addition of the I2C interface and the LOGII2S IP.

The hardware platform is based on Lab 2 with the addition of an I2C interface on the PS, the LOGII2S IP, and an additional AXI GPIO used for the digital mute control on the audio codec. The AXI GPIO and I2S IP have interrupts that are connected to the GIC (generic interrupt controller) on the PS though a concatenation IP.

# Software Design

The software is divided into 3 categories. The top level embedded software is written in C++. The C++ code has a class called “Audio” which is used to configure and interact with the audio codec in an object-oriented manner. The top-level code makes use of low level C code that is written in C. Most of this C code is provided by the I2S IP’s vendor as driver code for the Linux platform. After minor modifications the Linux drivers can be used with the Standalone OS. The third category is the GUI code which is written in Python, JavaScript, and HTML/CSS.

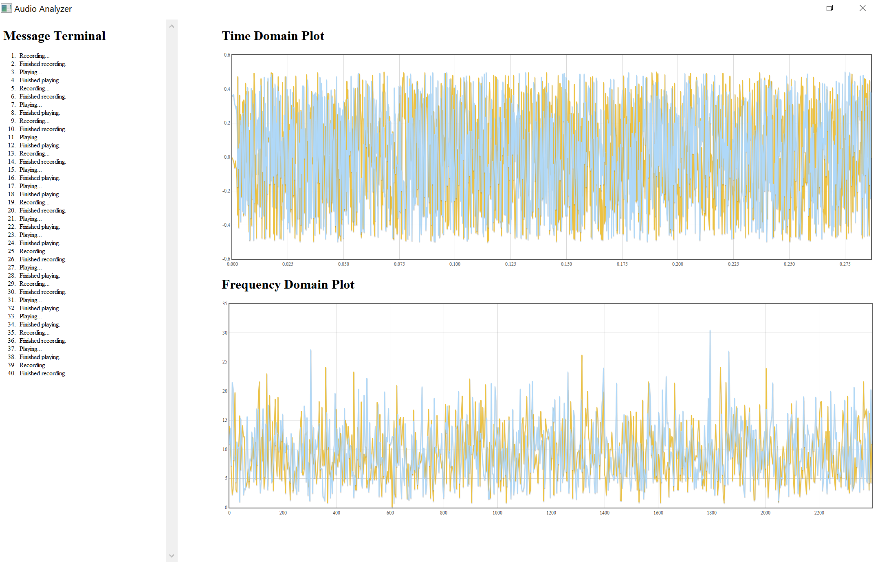
The GUI is generated using a Python framework called htmlPy. This framework allows for backend Python code to be called from the JavaScript code running in the frontend (and vice-versa). It also translates the HTML/CSS code into graphics. The GUI that is generated has 3 components: the message terminal, the time domain plot, and the frequency domain plot (Figure 3).

Figure The GUI that runs on the host PC.

The beginning of the top-level code runs initialization routines. Before using the Zybo’s audio codec it must be configured over I2C. The I2C functionality is enabled in the PS and accessed through the system software. The necessary configuration values are available in the SSM2603’s datasheet. The I2S IP is configured next for a 48 kHz sample rate. Finally, the digital mute is disabled to allow for audio playback.

The playback function performs the following procedure:

1. Set the channel’s pre-scale value to obtain the appropriate bit rate.
2. Set the channel’s word select values.
3. Fill the FIFO with data.
4. Clear the interrupt mask register.
5. Reset the interrupt sense register by writing ‘1’.
6. Enable data transfer by writing a ‘1’ to the enable bit in the control register.
7. Wait for the FIFO almost empty interrupt to occur.
8. Repeat from step 3.

# The record function performs the same procedure with a different FIFO interaction:

1. Set the channel’s pre-scale value to obtain the appropriate bit rate.
2. Set the channel’s word select values.
3. Clear the interrupt mask register.
4. Reset the interrupt sense register by writing ‘1’.
5. Enable data transfer by writing a ‘1’ to the enable bit in the control register.
6. Wait for the FIFO almost full interrupt to occur.
7. Read from the FIFO.
8. Repeat from step 3.

The advanced system uses a handler registered with the GIC (generic interrupt controller) to process the FIFO almost full and the FIFO almost empty events. It also uses interrupts on the AXI GPIO to process button events.

# Conclusion

This project enabled a Zybo board to record and playback audio data when the appropriate buttons are pressed. The board can also stream audio data over the serial interface for plotting in the host application at the press of a button.