Luke Jaffe
Microprocessor-Based Design
Due 03/03/16

Pre-Lab 4 (Part A)

Pre 4.1) Describe the functionality of an audio codec. In your description briefly outline the characteristics of an audio signal in the real world and how it is represented digitally.

An audio codec is a device or computer program capable of coding or decoding a digital data stream of audio. In the physical world, sound exists as longitudinal waves which travel in the form of compression and rarefaction. A microphone is used to convert sound into an electrical signal. This is done using electromagnetic transducers. Most microphones today use electromagnetic induction, capacitance change, or piezoelectricity to produce an electric signal from air pressure variations. An analog-to-digital converter (ADC) then converts the analog signal to a digital signal, typically using pulse-code modulation (PCM). In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals, and each sample is quantized to the nearest values within a range of digital steps. A typical audio sampling rate is 44.1 kHz.

Pre 4.2) The ADAU1761is exposed through an I2C and an I2S interface to the Zynq SoC. Describe the purpose of each interface. Why do you think two separate interfaces are used instead of just one?

The I2S interface is used for serial data I/O, i.e. sending/receiving audio data. The I2C interface is used for control i.e. read-write operations on registers. Two separate interfaces are used for these purposes because real-time audio streaming applications could be constantly using the serial data lines, leaving no room for control signals.

Pre 4.3) Describe the functionality of a FIFO. What is the sequence in which data is traveling in and out of the FIFO? Consider a writer to the FIFO and a reader from the FIFO; under which conditions should either a reader or a writer block?

A FIFO, standing for “first-in first-out”, is also known as a queue. Data enters a FIFO through one end, and exits through the other end. Elements which enter first leave first, hence the name. When writing to a FIFO, the writer will block if the FIFO is full. When reading from a FIFO, the reader will block if the FIFO is empty.

Pre 4.4) The audio application operates on 16bit signed samples. However, the FIFO is 32bit wide (i.e. requires that 32bit are written / read for each sample). However, out of those 32bit from the FIFO, the CODEC only interprets the 24 most significant bits. Show an example code snippet to convert a 16 bit sample (stored in variable sample_16) into the 32bit before writing it to the FIFO.

```
u32 sample_32 = (sample_16 << 8) & 0x00FFFF00;
```

Pre 4.5) Make a picture of the TX FIFO filled with 4 16bit stereo samples (i.e. 8 samples in total). Distinguish between left and right samples.

Each block contains a 16 bit sample, encoded in 32 bits as detailed in the previous question. It is assumed that the "right" sample is transmitted first, and the "left" sample afterwards.

Input	s4_left	s4_right	s3_left	s3_right	s2_left	s2_right	s1_left	s1_right	**Output**

Pre 4.6) How is the FIFO filled? Show a C code excerpt for defining a variable of the proper type, setting the address and filling the FIFO with one sample.

```
*(volatile u32 *) (FIFO_BASE_ADDR + FIFO_TX_DATA) = sample_32;
```

Pre 4.7) The processor might fill the FIFO faster than the codec can take them out (as the CODEC outputs samples at 48kHz). Consider the Transmit Data FIFO Vacancy Register (TDFV) register. Show a driver to fill the FIFO with samples from an array sampleA, before writing a sample, it should check if there is sufficient space in the FIFO and delay for 10ms if the FIFO is full.

```
void txData(short* sampleA, int len)
{
      int i = 0;
      while (i < len)
      {
            if (*(volatile u32 *) (FIFO_BASE_ADDR + FIFO_TX_VAC)  < 0x000001FC)
            {
                  u32 sample_32 = (sampleA[i] << 8) & 0x00FFFF00;
                  *(volatile u32 *) (FIFO_BASE_ADDR + FIFO_TX_DATA) = sample_32;
                  i++;
            }
            else
            {
                  vTaskDelay(2); //delay for 10-20 ms
            }
      }
}
```

Pre 4.8) What would happen if the loop above would not check for the full condition? Make a prediction about how the sound would be. In addition, how does the FIFO indicate this condition? Please focus on the on the FIFO's Interrupt Status Register (ISR) which serves as a status register as well.

If the loop above did not check for the full condition, samples pushed in while the FIFO was full would be lost. If this happened frequently, and in a randomly distributed way, the sound might be like white noise. If it happened infrequently, the sound would have ugly blips. The FIFO indicates that it is full by asserting an interrupt and setting the TFPF bit in the status register.