Brady Lee

EECS 395

FM Radio Project Summary

Introduction

FM radios and digital signal processing in general is classically performed using analog hardware, but using the background and techniques described within, one can easily create similar performance using digital logic in software or hardware.

Background

Sampling

Continuous signals can be digitized by sampling, which is the process of reading the value of the signal at given time intervals. Sampling creates a discrete signal that represents a continuous signal. Some data is lost during sampling, so if the digital signal is converted back to a continuous signal, the quality may be decreased. The sampling rate determines how much data is captured and in turn the quality of the resulting signal.

Quantization

Quantization is a mapping of a set of numbers to a different, typically smaller, set. Quantization is similar to sampling, in that the resulting value is representative of the original value, but it is possible that some information may be missing. Quantization can be used to map fixed point real numbers to integer numbers. One of the benefits of quantizing fixed point numbers is to allow for easier arithmetic operations. Integer operations are typically simpler and quicker to compute than real number operations, real number operations may even require additional hardware, such as a GPU in a computer, which is specialized for floating point operations.

Fixed point numbers have three components: a sign bit, integer bits, and fractional bits. As expected, the sign bit determines if the number is positive or negative, the integer bits determine the value of the mantissa, and the fractional bits determine the value of the fraction. Fixed point numbers can be quantized by left shifting by the number of fractional bits and dequantized by shifting right by the number of fractional bits. The total bit width of the number and the fractional bit width determine the overall accuracy of quantized fixed point operations, as well as the overall hardware cost to support said bit widths.

Filtering

Filters are often used to isolate and extract information, or remove unwanted information, from a signal. Common filters include low pass, high pass, and band pass. Low pass filters filter out data at a frequency higher than some cutoff frequency and keep the low values. High pass filters are the opposite, keeping high frequency data and removing low frequency data. Band pass filters are a combination of low and high pass filters, used to select a range of frequencies. Typically, a filter would be implemented with a combination of capacitors, resistors, and inductors, but it is possible to implement them digitally.

Finite impulse response filters comprise of a history of input samples, each scaled by some coefficient, then summed to produce an output. Digitally, the input history can be created using a shift register. FIR filters can be used to simulate low pass, high pass, and band pass filters.

Infinite impulse response filters are similar to FIR filters, but they also keep a history of previous outputs as feedback. The sum of the previous inputs and previous outputs form the new output. IIR filters can be used to deemphasize a signal, improving the signal-to-noise ratio, or SNR. The transfer function used to deemphasize in this design is $H(s) = \frac{1}{1+s}$.

<u>FM Radio</u>

FM radio uses the principle of frequency modulation to broadcast signals. The original signal is added to a carrier wave before it is broadcast, which is less prone to signal deterioration by noise than the older amplitude modulation method. The original signal can be derived by removing the carrier frequency from the received signal. Useful information is encoded in the original signal, such as mono audio, stereo audio, and a stereo pilot tone (Figure 1).

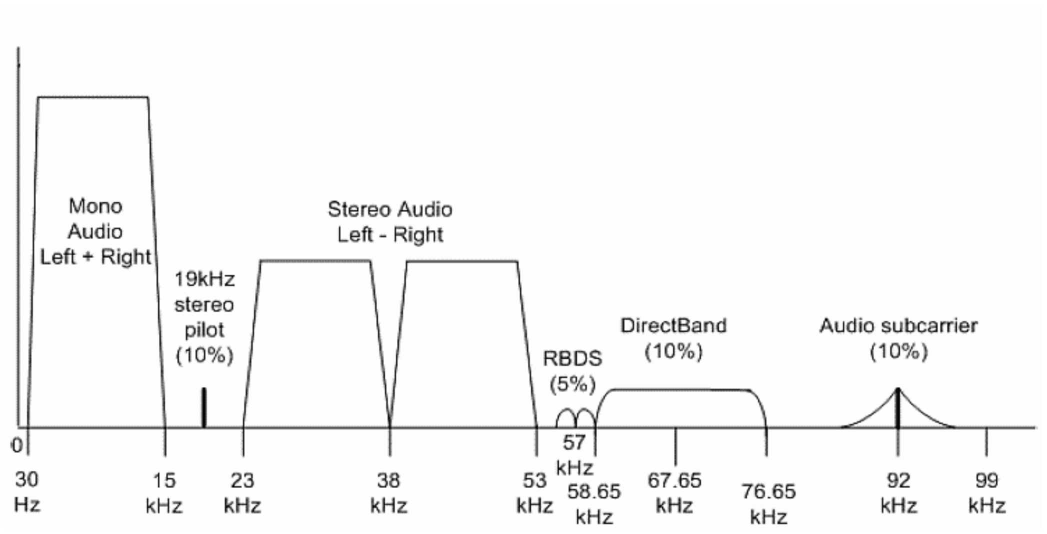

Figure 1: Radio signal encoding

System Architecture

The design is a streaming architecture consisting of ten different types of functional units and fifteen units overall (Figure 2). Each functional unit is connected by a FIFO to buffer the data, bringing the overall component count to thirty-five. The components can be grouped into four broad sections: input, pilot determination, left branch, and right branch.

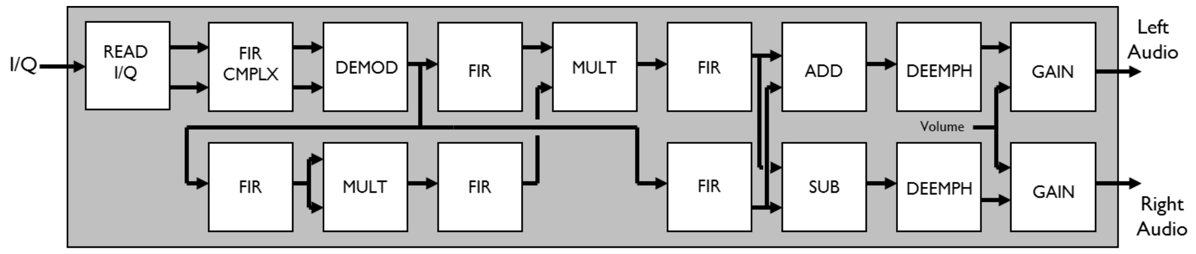

Figure 2: Functional units of architecture

Input

The input section involves reading, splitting, filtering, and demodulating samples to set up for the rest of the design. The functional units are an IQ-reader, complex FIR, and a demodulator respectively. The IQ reader takes input samples and splits them into an I and Q component. The complex FIR treats each I/Q pair as a complex number then filters them. The demodulator is used to find the phase change between I/Q samples to determine the original frequency, which in turn determines pitch of the sound. The output of the demodulator is piped to the other three sections of the design for further processing.

Pilot Determination

This is a small section used to determine the pilot tone. The demodulated signal is passed through a 32-tap FIR band pass filter to isolate the 19 kHz tone. If the tone exists, it indicates that stereo data is available in the 23 kHz to 53 kHz range. The tone can be squared to use as a reference point at 38 kHz, splitting the stereo range into two components. The act of squaring also creates a 0 Hz tone that is removed with a second FIR as a high pass filter.

Left Branch

The left branch is used to generate the left component of stereo audio. The branch begins with a 32-tap FIR band pass filter to isolate the 23 kHz to 53 kHz stereo audio range. The band is multiplied by the squared pilot tone to shift the frequencies down by 38 kHz, thereby centering the band on 0 Hz and making further calculations on the band easier to perform without offset. The result is passed through a decimating low pass FIR to isolate the stereo component from the rest of the data and to reduce the overall sampling rate by a factor of the decimating constant. The stereo data is added to the mono sound data generated by the right branch at an adder-subtractor component that joins the two branches temporarily. The addition isolates the left audio, which is deemphasized by passing it through an IIR filter. The filter result is passed through a final gain component, which multiplies the signal by a constant to change the volume level.

Right Branch

The right branch is fairly similar to the left branch. The demodulated input signal is put through a decimating 32-tap low pass FIR filter to isolate the mono sound. The stereo sound from the left branch is subtracted the mono sound to isolate the right audio. As done in the left branch, the audio is pushed through IIR and gain modules to deemphasize the signal and scale by the proper volume.

Design Process

A software description of the design was provided. The software demonstrated how each component functioned as well as the basic relationships between components. These components were fairly easy to convert to hardware, maintaining equivalent or better functionality. FIFOs were added to facilitate the flow of data throughout the design, as is done with any streaming architecture.

Optimizations and Impact

A few basic optimizations were added to the design to gain benefits over the original software implementation.

Streaming

The design had a streaming architecture. Each data sample is piped through each of the components, one after the other, as opposed to calculating the result of each component on the entire data set, which is typical of software. Streaming allows the design to perform using fewer cycles overall. The software implementation uses at least N cycles per component to calculate the result for N samples, while the streaming implementation uses N closer to N cycles total, plus some setup and drain cycles.

Parallelism

The hardware implementation allows for easier parallelism in design. As mentioned before, the output of the demodulator component is piped to three different branches. Processing on the demodulator output in each branch can be performed concurrently in the hardware implementation and only serially in the software implementation. Parallelism also reduces the overall cycle count.

Loop Unrolling

In the software implementation, loops are used to process large chunks of data at a time. The higher order loops were removed when converting the architecture to a streaming architecture, but some of the components had loop-based designs as well, such as the FIR and IIR components. The loops in these components were converted into shift registers, since the filters operate by saving a buffer of previous data and shifting new data in and old data out. By using a shift register instead of creating a state machine to simulate a loop, many cycles were saved. The consequence of loop unrolling, however, is the use of a large number of multiplier elements.

Decimation

Decimation is the process of ignoring some samples to allow for reduced processing, while maintaining a decent quality output signal. In the design, decimation could also double as a method for synchronizing signals. The data was split into multiple paths, but each path was not of equal length, so without ignoring some of the data, the FIFOs in the latter stages of the design would become full, consequently backing up the entire design, and slowing down the overall flow.

FIFO Sizes

The size of each of the FIFOs in the design is based on the expected speed of reads and writes on that specific FIFO. FIFOs in the beginning of the design are larger because they are expected to read new input data every cycle, while the components ahead of them are slower to read the data out of the FIFOs, because of delays in data processing. FIFOs near the end of the design can be smaller because they read new data slowly and write out quickly.

Simulation and Performance

Performance

The simulated design can process 10000 input, creating 1250 output samples due to decimation, in 78757 cycles, which is about 8 cycles per input sample, or 63 cycles per output sample. The FPGA processor is 50 MHz, so the design is capable of producing 793650 samples per second (Eq. 1), which more than exceeds the necessary 44 kHz typically used in digital audio processing.

$$Equation\ 1\colon \frac{50000000\,\frac{cycles}{second}}{63\,\frac{cycles}{sample}} = 793650\,\frac{samples}{second}$$

Bottlenecks

The design architecture is not strictly linear, so it is prone to experiencing bottlenecks. The overall path from demodulator through pilot to the left branch is significantly longer than the path directly from the demodulator to the right path. Without any form of optimization this leaves the right branch waiting a while for the left branch to catch up before data can be sent through to the output. Unfortunately, the demodulator itself complicates the design, as it contains a divider, which does not run for a set number of cycles. Depending on the input to the demodulator, the divider could run from about 3 to 8 cycles. This variation causes increased waiting in the latter components of the design.

Synthesis Results

The design uses 29020 logic elements overall (Table 1), 28985 combinational and 6598 register elements. On a Cyclone IV EP4CE115F29C7, the chip on Altera DE2 115 boards, about 25% of the combinational elements are used, while 6% of the register elements are used. 10752 memory bits are used, which is less than 1% of the available memory. All 532 of the multipliers are used, so the loop unrolling should potentially be cut back in the design, or a bigger board should be used.

Element	**Count**	**Total**	**Percentage**
Combinational elements	28985	114480	25%
Register elements	6598	114480	6%
Memory bits	10752	3981312	1%
Multipliers	532	532	100%

Table 1: Design compilation results