FM Radio Project

Background

FM radio is a ubiquitous medium for information and entertainment, but the intricacies of how it functions are not widely known. This FM radio project delves into the engineering principles that allow us to harness radio waves for audio transmission.
Radio Wave Fundamentals

Radio waves, part of the electromagnetic spectrum, are all around us, though beyond human auditory perception due to their frequency range. FM radio stations operate within a designated spectrum, continuously transmitting signals at various frequencies. These frequencies are higher than what human ears can detect, yet FM radios are designed to receive and process these specific bands, enabling us to listen to the audio content being broadcasted.

Signal Conversion and Processing
To interpret these signals, FM radios must perform a series of conversions and processing steps:

1. **Analog-to-Digital Conversion (ADC)**: Radios initially capture analog signals, which are continuous waves. An ADC converts these waves into digital form by sampling the amplitude at discrete intervals, producing a binary representation of the wave's amplitude over time.
2. **Fourier Transform**: Digital processing commences with a Discrete Fourier Transform (DFT), a mathematical technique that transforms time-domain data (our sampled signal) into the frequency domain. This is essential to isolate and manipulate the specific frequencies of interest.
3. **Stereo Signal Decoding**: FM radio provides mono audio within a base frequency range (30Hz to 15kHz). Frequencies just above this range, particularly around 19kHz, often carry a 'pilot tone'—a signal indicating stereo broadcast. This project assumes stereo and processes accordingly.
4. **Bandpass Filtering and Channel Separation**: We employ a 32-tap Finite Impulse Response (FIR) filter to isolate the 19kHz pilot tone. This tone is then squared to generate a 38kHz signal used to separate the stereo channels (Left and Right). High-pass filtering removes any remnants of the pilot tone at 0Hz.
5. **Demodulation to Baseband**: The left-right channel is multiplied by the squared pilot signal and passed through a low-pass FIR filter to demodulate it back to baseband. This step also includes downsampling by a factor of ten.
6. **Channel Reconstruction**: We reconstruct the stereo channels by combining and processing the L+R and L-R signals, resulting in two distinct channels with a proper

stereo effect.

7. **Digital-to-Analog Conversion (DAC)**: Once processing is complete, the separated left and right channels are converted back into analog signals using a DAC. This allows the processed audio to be played through speakers or headphones.

Implementation

For this project, the signal processing algorithm is designed to run on a Cyclone IV E Field-Programmable Gate Array (FPGA). FPGAs are ideal for such applications due to their high-speed processing capabilities and reconfigurability, which allow for real-time signal processing and manipulation.

Through this project, we explore the seamless transition from radio waves to audible sounds, unveiling the technical artistry behind FM radio broadcasting and reception.

The system architecture

The basic architecture of this project was mainly designed with streaming architecture. Refer to the work process of FM radio which was introduced in last part. There is a block diagram which could present the functional modules and how to connect them. FIFOs applied between each modules to confirm the synchronization of the inputs and outputs of the design.

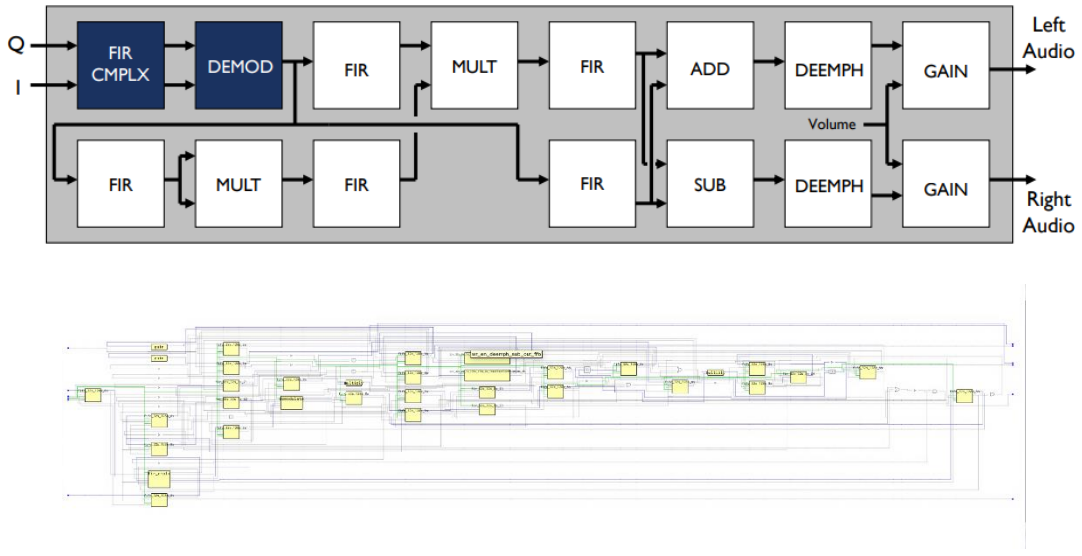

Figure1. Block Diagram and RTL

Design process

Modular Approach and Testing

Our FM radio project was developed with a focus on modularity and exhaustive testing,

ensuring that each component functions correctly and efficiently. The design process began by breaking down the radio software into distinct functions. By doing so, we could individually stream output values from these functions to text files, which allowed for isolated testing and validation of each step's functionality.

Code Snippet for Testing

To facilitate this, we wrote specific code blocks (see Figure 8 in the original document) to generate test data. These blocks were designed to output demodulated data, effectively converting the complex radio signal processing algorithms into readable formats for analysis and debugging.

Systematic Function Analysis

With our testing framework in place, we meticulously analyzed each function within the C++ codebase. This analysis was twofold: understanding the function's logic and determining the state changes within the system. This helped us ensure that each function was not only logically sound but also that it transitioned between states as expected.

Schematic and State Machine Design

Next, we transitioned to a more visual phase, sketching out schematics and designing state machines. This visual planning was instrumental in conceptualizing how the entire system would function as a cohesive unit. Once the schematics were set, we developed corresponding modules and their respective testbenches.

Incremental Testbench Implementation

Each module was tested with a dedicated testbench that was fed input data from the output of the preceding function, creating a linked chain of validations. This incremental approach meant that each function's output was methodically verified to be accurate before moving on to the next.

Synthesis and Optimization

Upon achieving functional correctness, our attention shifted to synthesizing the modules. This synthesis allowed us to observe layout, resource utilization, and frequency characteristics. With this insight, we embarked on optimizing each module, enhancing the system's overall performance.

Top-Level Design Integration

Post-optimization, modules were integrated into a top-level design. The top-level design was architectured to facilitate seamless module integration, featuring a FIFO buffer between each module to manage data flow efficiently. This strategic setup simplified the integration of new modules by pre-establishing signal pathways.

Comprehensive Testbench for Top-Level Design

The culmination of our module testing and optimization was the development of a top-level design testbench. This was the ultimate test of our system's integrity, ensuring that all

components worked harmoniously. We ran a Universal Verification Methodology (UVM) testbench, renowned for its rigor in verifying hardware designs, and meticulously double-checked for correctness.

Iterative Design Philosophy
Our design process was not linear but cyclical. After each test, we revisited our designs to iterate and refine. This iterative cycle of planning, coding, testing, optimizing, and validating was repeated for each aspect of the design, a testament to our commitment to precision and quality.

Optimizations

An effective optimization strategy employed in the FM radio project was the use of pipelining and resource sharing, specifically within the Deemphasis module. By implementing a state machine that divides the computation into multiple states, we managed to significantly improve the clock speed.
Additionally, in the demodulate module, we integrated a hardware divider to further streamline the process. Hardware dividers can perform division operations more efficiently than software-based solutions, as they are optimized for such calculations. By offloading this task to dedicated hardware, we were able to reduce the computational load on the main processing unit, thereby achieving a higher clock frequency and improved performance.

In essence, the optimization leveraged the principle of dividing the processing workload across different states of a state machine, coupled with the use of specialized hardware, to enhance clock speeds without incurring the cost of additional resources. This meticulous approach to optimization highlights the balance between performance, latency, and resource utilization that is critical in complex system designs like FM radio receivers.

Simulation: Throughput analysis

In the simulation phase of our FM radio project, the primary task was to validate the functionality of each module within the top-level system. This was meticulously executed using the Universal Verification Methodology (UVM), ensuring data integrity between inputs and outputs.

The simulation revealed that the cmplx FIR (Complex FIR) module presented a bottleneck in throughput. The intricate nature of this module, inclusive of feedback loops and decimation processes, resulted in longer latencies, slowing down the processing speed. Despite the module's capability to perform simple arithmetic operations, its internal complexity hinders

data processing speed, thus affecting the subsequent modules that are dependent on its output.

Throughput was measured by the cycle count required to output all data, and this was correlated with the frequency synthesized by the module to evaluate performance. Moreover, the dual audio channel configuration doubled the demand for data processing, further impacting overall throughput.

In summary, this analysis underscores the significance of optimizing the Complex FIR module to enhance the performance of the entire FM radio system.

fir cmplx	
Data	100
byte	100*4
Cycle	2303
frequence	100Mhz
throughput	138.9MB/s

Demodulate	
Data	100
byte	100*4
Cycle	1640
frequence	100Mhz
throughput	195.1MB/s

fir with fifo	
Data	100
byte	100*2
Cycle	507
frequence	100Mhz
throughput	315.6MB/s

iir with fifo	
Data	100
byte	100*2
Cycle	403
frequence	100Mhz
throughput	372.1MB/s

gain with fifo	
Data	100
Byte	100*2
Cycle	103
frequence	100Mhz
throughput	1553.4MB/s

Total design	
Data	366149
byte	366149*4
Cycle	/
Time	127972295ns
throughput	91.6MB/s

Synthesis: Resource and Timing analysis

Resource Utilization	
I/O pins	136
LUT	17428
DSP block	62
Registers	8244
Memory bits	84864
Top_clk freq	52.5/44.6Mhz

We synthesized each module individually with the following predicted frequencies

Timing Frequency of each modules	
Module Name	Frequency
fir	79.7MHz
Fir_cmplx	78.2 MHz
demodulate	44.6MHz
arctan	44.3MHz
dividor	45.7MHz
iir	88.7MHz
gain	86.8MHz

Thus, it is clear that the overall frequency performance is limited by the div critical paths in the demodulate module also corroborates this point insert msb algorithm in which the circular adder limits the overall performance.

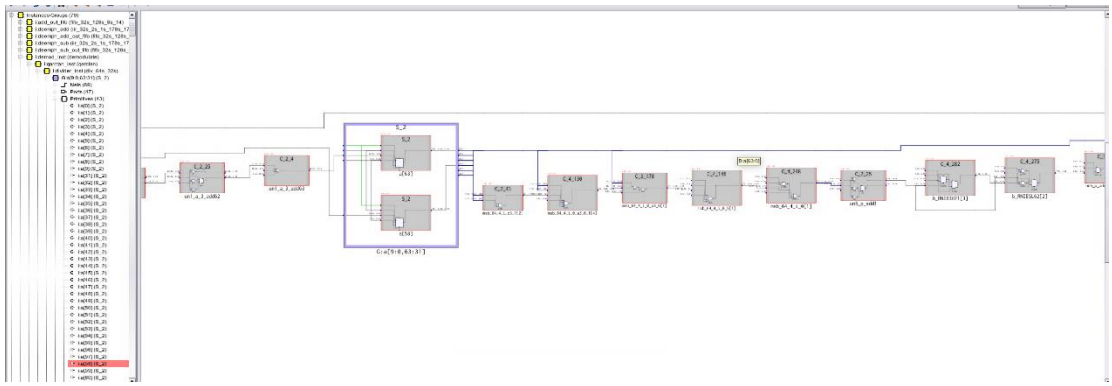

Figure2. Critical path analysis