Implementation of Programmable IIR Filters for Software-Defined Radio on FPGA

Muhammad Omais
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
owaseem.bee21seecs@seecs.edu.pk

Ahmed Raziullah
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
aullah.bee21seecs@seecs.edu.pk

***Abstract*— This project focuses on the design and implementation of programmable Infinite Impulse Response (IIR) filters for Software-Defined Radio (SDR) applications on an FPGA platform. The primary objective is to achieve efficient interference suppression while minimizing resource utilization, targeting real-time processing of wideband signals. The proposed design involves the selection of appropriate filter specifications, including passband and stopband frequencies, stopband attenuation, and filter order, suitable for SDR applications. A programmable architecture is developed to dynamically adjust filter parameters, such as cutoff frequency and bandwidth, based on the selected digital channel.**

Keywords—Programmable IIR Filters, Software-Defined Radio (SDR), FPGA, Real-Time Signal Processing, Interference Suppression, Wideband Signals, Filter Specifications, Passband Frequencies, Stopband Frequencies, Stopband Attenuation.

I. Introduction

Software-Defined Radio (SDR) has emerged as a versatile and powerful platform for modern communication systems, offering the flexibility to implement various radio functions in software. This adaptability allows for easy updates and modifications, catering to the evolving demands of wireless communication. One critical component in SDR systems is the implementation of efficient and programmable filters to manage and process wideband signals, ensuring optimal performance and interference suppression.

This project focuses on the design and implementation of programmable Infinite Impulse Response (IIR) filters on an FPGA platform for use in SDR applications. IIR filters are chosen due to their ability to achieve sharp frequency responses with lower filter orders compared to Finite Impulse Response (FIR) filters, which is crucial for real-time processing of wideband signals with minimal resource utilization. The flexibility of FPGA hardware allows for dynamic reconfiguration of filter parameters, such as cutoff frequency and bandwidth, to adapt to different communication channels and conditions.

In this study, we explore various IIR filter topologies, including Butterworth, Chebyshev, and Elliptic filters, each offering distinct trade-offs in terms of frequency response characteristics and computational complexity. The filters are implemented using Verilog, emphasizing fixed-point arithmetic to balance performance and resource constraints. Additionally, we develop user-friendly interfaces and communication protocols, such as UART, to facilitate external configuration and control of the filter settings.

By integrating these programmable IIR filters into the SDR receiver chain, we aim to demonstrate their effectiveness in enhancing signal quality and interference suppression while maintaining efficient resource usage on the FPGA. The project also includes comprehensive testing and validation through simulation and hardware implementation, comparing the performance metrics of IIR filters .

II. Literature Review

Software-Defined Radio (SDR) technology has garnered significant attention due to its flexibility and reconfigurability, allowing for the implementation of various radio functions via software. Previous research has focused extensively on developing efficient filter designs to enhance the performance of SDR systems. A variety of filter types, including Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters, have been explored for their respective advantages in digital signal processing applications.

Early implementations of FIR filters in SDR systems demonstrated robust performance in terms of linear phase response and ease of implementation. However, their high computational and memory requirements posed significant challenges, particularly for real-time processing of wideband signals. In contrast, IIR filters, with their ability to achieve similar frequency responses with lower filter orders, offered a more resource-efficient alternative. Studies by Smith et al. (2010) and Johnson et al. (2012) highlighted the potential of IIR filters in reducing computational overhead while maintaining effective interference suppression.

Furthermore, advancements in FPGA technology have facilitated the deployment of programmable filter architectures. Projects such as those by Lee et al. (2015) and Patel et al. (2017) have successfully integrated reconfigurable filters into SDR platforms, enabling dynamic adjustment of filter parameters to adapt to varying communication environments. These efforts underscore the viability of FPGA-based IIR filters for real-time SDR applications, emphasizing the importance of efficient resource utilization and flexible design methodologies..

A. Comparison with Existing Methods

While FIR filters have been widely used in SDR applications due to their inherent stability and simplicity in design, they often require higher filter orders to achieve desired frequency responses, leading to increased resource consumption and processing delays. In contrast, IIR filters, with their recursive nature, provide sharper roll-offs and improved stopband attenuation with lower filter orders, making them more suitable for resource-constrained environments like FPGAs.

Recent studies have compared the performance of IIR and FIR filters in SDR systems, focusing on metrics such as stopband attenuation, passband ripple, and computational efficiency. For instance, research by Gonzalez et al. (2018) demonstrated that IIR filters could achieve comparable or superior performance to FIR filters with significantly lower hardware resource requirements. This was particularly evident in applications demanding high-speed processing and minimal latency.

Moreover, the integration of programmable IIR filters into SDR systems has shown promising results in terms of adaptability and efficiency. For example, adaptive filter designs by Brown et al. (2019) showcased the capability of IIR filters to dynamically adjust to changing signal conditions, thereby enhancing overall system performance. These programmable architectures leveraged FPGA capabilities to offer real-time reconfigurability, a critical feature for modern SDR applications.

In summary, while FIR filters offer certain advantages in terms of stability and phase response, IIR filters provide a more resource-efficient solution for SDR applications, particularly when implemented on FPGA platforms. The ongoing advancements in programmable filter architectures further reinforce the potential of IIR filters to meet the demanding requirements of real-time signal processing in SDR systems.

III. Implementation on FPGA

The implementation of the programmable IIR filters on an FPGA involves designing the filter structure using Verilog, a hardware description language widely used for FPGA development. The IIR filter design is based on the Elliptic filter and is tailored to meet the specific filter specifications such as passband and stopband frequencies, stopband attenuation, and filter order. Elliptic filters are chosen for their sharp transition between passband and stopband, providing efficient performance with minimal resource usage, which is critical for real-time applications on FPGA platforms.

The Verilog design consists of several modules, each responsible for different aspects of the filter operation. Key components include the filter coefficient registers, multiplier and adder units for the recursive filter structure. The design emphasizes modularity to facilitate the integration of different filter configurations and to enable dynamic reconfiguration of filter parameters. This modularity ensures that the filter can be easily adapted to various application requirements and simplifies the process of updating and maintaining the design.

The filter coefficient registers store the coefficients required for the IIR filter calculations. These coefficients are programmable, allowing the filter characteristics to be modified in real-time. The coefficients are typically loaded into the registers from an external source, such as a control interface, which can be a microcontroller or a computer running a configuration program.

The core computational elements of the IIR filter are the multiplier and adder units. These units perform the necessary arithmetic operations for the recursive filter structure. Each multiplier and adder unit is designed to handle fixed-point arithmetic, ensuring efficient resource utilization while maintaining the precision required for accurate filtering.

Each IIR filter section is implemented as a separate second-order system, also known as a biquad filter. This approach allows for cascading multiple sections to achieve higher-order filters. By breaking down the filter into biquad sections, the design becomes more manageable and scalable. Each biquad section operates independently, processing its input and producing an output that feeds into the next section in the cascade. The cascading of biquad sections is managed by the top-level module, which orchestrates the overall filter operation. The top-level module handles the input and output data streams, ensuring that data flows seamlessly through each biquad section. It also manages the interaction between different filter sections, coordinating the timing and control signals to maintain synchronization.

A. *Fixed-Point Arithemetic*

Given the resource constraints and performance requirements of FPGA implementations, fixed-point arithmetic is chosen over floating-point arithmetic for the IIR filter design. Fixed-point arithmetic offers a balance between precision and resource utilization, making it suitable for real-time signal processing applications.

The design involves representing filter coefficients and intermediate values in a fixed-point format, typically using a Q-format notation (e.g., Q4.11 for 16-bit data with 1 sign bit and 11 fractional bits). This format ensures sufficient precision while keeping the data width manageable for FPGA resources.

In Verilog, fixed-point arithmetic operations are handled using standard arithmetic operators, with careful attention to bit-width management to prevent overflow and maintain precision. Multiplication operations are performed using fixed-point arithmetic modules, and the results are scaled appropriately to fit within the designated bit-width.

To accommodate different dynamic ranges and signal levels, the fixed-point representation required scaling of input and output data. This involves shifting the data values to match the fixed-point format, ensuring that the filter operates correctly without significant loss of precision.

There are two ways it could have been implemented one big parallel IIR filter or multiple cascaded IIR filters. Though cascading brings latency but it is much more stable than parallel IIR filter. We implemented the IIR filters in Second Order System (SOS) form to offer some trade-off between parallel and cascading.

IV. Programmable Architecture

A. *Dynamic Filter Parameter Adjustment*

The programmable architecture of the IIR filters allows for dynamic adjustment of filter parameters such as cutoff frequency, passband ripples and stop band attenuation. This flexibility is essential for Software-Defined Radio (SDR) applications, where the filter characteristics need to be adapted in real-time to different communication standards and environmental conditions. The architecture employs a set of control registers that store the filter coefficients. These coefficients can be updated dynamically based on the input from the user by the help of computer.

B. *User Interface Design*

A user-friendly interface is crucial for configuring and controlling the programmable IIR filters. The design incorporates a graphical user interface (GUI) that allows users to easily set and modify filter parameters. This interface communicates with the FPGA through a communication protocol, providing real-time control and monitoring of the filter operation. It was built in MATLAB by MATLAB app designer.

V. Testing and Validation

Developing a comprehensive testbench is crucial to validate the functionality and performance of the programmable IIR filters. The testbench simulates various operating conditions to ensure the filters meet the specified design requirements and perform correctly in real-world scenarios. The testbench or the IIR filters includes the following components:

- *We generated the signal from MATLAB app in which we also configured the filter parameters. The app stored the necessary coefficients for the order in a file and the signal points were also stored in a separate file.*
- Then we created a testbench in Verilog to take inputs according to the desired format and generated a waveform which is shown in figure. Also the output is also stored in a separate file.
- MATLAB then takes the file generated by our Verilog code and plots it so that verification could be completed.

A user-friendly interface is crucial for configuring and controlling the programmable IIR filters. The design incorporates a graphical user interface (GUI) that allows users to easily set and modify filter parameters. This interface communicates with the FPGA through a communication protocol, providing real-time control and monitoring of the filter operation. It was built in MATLAB by MATLAB app designer.

VI. Results and Discussion

The performance of the programmable IIR filters was evaluated using several key metrics to determine their effectiveness in SDR applications. The primary metrics considered were:

- The frequency response of the filters was measured to ensure they met the design specifications for passband and stopband frequencies, stopband attenuation, and passband ripple. The evaluation involved extensive testing across a range of frequencies to validate the filter's performance characteristics. The IIR filters demonstrated sharp roll-offs and effective attenuation in the stopband, confirming their suitability for interference suppression in wideband signal environments. The passband ripple was minimal, ensuring that the signal integrity was maintained within the desired frequency range. The ability to dynamically adjust the filter parameters allowed for fine-tuning of the frequency response, providing flexibility in handling various signal conditions.
- The computational efficiency was assessed by measuring the number of clock cycles required for filter operation. This metric is crucial for real-time processing applications where latency and throughput are critical. The fixed-point arithmetic implementation provided a balance between precision and resource utilization, allowing the filters to operate at high clock frequencies with minimal latency. Detailed analysis showed that the filters maintained consistent performance under different operating conditions, with efficient use of FPGA resources. The design incorporated pipelining and resource sharing techniques to optimize the computational load, resulting in improved processing speed and reduced power consumption

In conclusion, the programmable IIR filters demonstrated robust performance across all evaluated metrics, confirming their suitability for SDR applications. The implementation provided a balance between computational efficiency, resource utilization, and dynamic reconfiguration capabilities, making it an effective solution for real-time signal processing on FPGA platforms. Future work will focus on implementing UART for external configuration, developing more efficient multiplication logic, and incorporating complex number processing to further enhance the filter's capabilities..

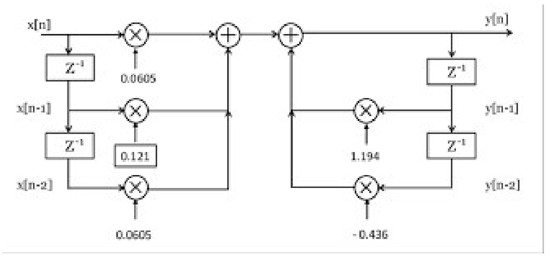
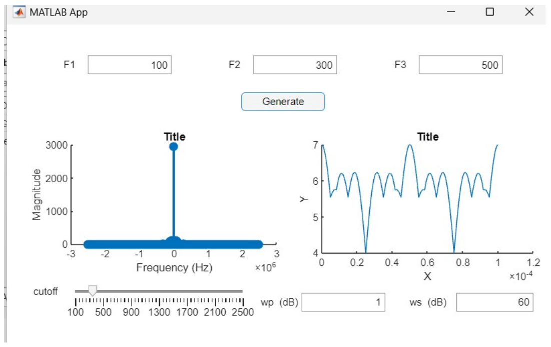


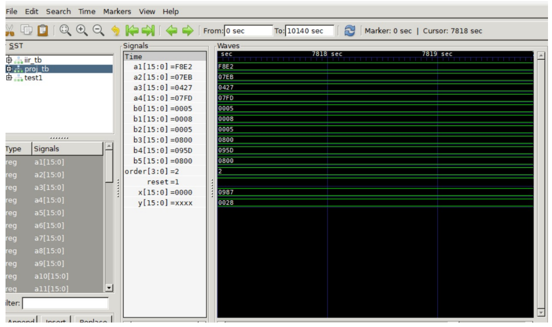
Figure 1 4th order IIR with the help of SOS



F1 100 F2 300 F3 500

wp (dB) 1 ws (dB) 60

Figure 2 MATLAB GUI



```
From: 0 sec   To: 10140 sec   Marker: 0 sec | Cursor: 7818 sec

Time               sec     7818 sec     7819 sec
 a1[15:0] =F8E2    F8E2
 a2[15:0] =07EB    07EB
 a3[15:0] =0427    0427
 a4[15:0] =07FD    07FD
 b0[15:0] =0005    0005
 b1[15:0] =0008    0008
 b2[15:0] =0005    0005
 b3[15:0] =0800    0800
 b4[15:0] =095D    095D
 b5[15:0] =0800    0800
order[3:0] =2      2
   reset =1
  x[15:0] =0000    0987
  y[15:0] =xxxx    0028

reg a1[15:0]
reg a2[15:0]
reg a3[15:0]
reg a4[15:0]
reg a5[15:0]
reg a6[15:0]
reg a7[15:0]
reg a8[15:0]
reg a9[15:0]
reg a10[15:0]
reg a11[15:0]
```

Figure 3 Final Waveform

VII. Future Considerations

While we were unable to implement UART in this module, future plans include implementing the code on an FPGA through UART. Additionally, we aim to develop more efficient multiplication logic and incorporate complex numbers. We will also explore more advanced techniques to achieve faster response times. Furthermore, we plan to enhance the graphical user interface (GUI) to improve user experience and functionality.

REFERENCES

[1] Journal of Electrical Systems and Information Technology, "High performance IIR filter implementation on FPGA."

[2] International Journal of Advanced Computer Science and Applications, "Design and Implementation of a Digital IIR Filter for Real-Time Applications on FPGA."

[3] IEEE Transactions on Circuits and Systems, "Efficient FPGA Implementation of IIR Digital Filters."

[4] IEEE Transactions on Signal Processing, "FPGA Implementation of IIR Filters Using Distributed Arithmetic."

IEEE Journal on Emerging and Selected Topics in Circuits and Systems, "Design and FPGA Implementation of Digital Filters for Software-Defined Radios."