Design and Implementation of FIR Filter Using Multiply and Accumulate (MAC)

Param Shah[1], Dharman Patel[2], and Avadh Nandasana[3]
School of Engineering and Applied Science (SEAS), Ahmedabad University, Ahmedabad, Gujarat – 380009[1,2,3]

***Abstract*—This paper focuses on the design, implementation, and simulation of a Finite Impulse Response (FIR) filter using one of the most important operations in DSP; Multiply and Accumulate (MAC). FIR filters are particularly important for such applications as noise filtering, smoothing, and feature extraction, because they can possess stability and linear-phase property. We describe our Verilog HDL code of the filter and, by applying a sine wave input signal to the filter and using the Verilog HDL simulator to analyze the filter's function, we assess the feasibility of the filter in terms of throughput, latency, and frequency response. This work seeks to offer insight regarding the design of FIR filters and suggest several concerns pertinent to deploying these filters in hardware for real-time DSP applications.**

I. Introduction

Digital signal processing (DSP) is a key enabling technology for today's communications, multimedia and control systems. Again, FIR filters are real time signal processing filters and fall under the category of Digital Signal Processing filters and characterized by their simplicity and stability. While FIR filters are of their own category in that they do not use feedback, they provide stability and can give linear phase responses.

A. Motivation

This paper aims at presenting improved techniques for implementing efficient and flexible hardware solutions for real-time filtering of signals in various applications such as communication systems, audio processing and biomedical systems. Although the algorithmic implementations of the filters are quite general-purpose for flexibility, implementations of FIR filters using electronics hardware offer real-time performance with little or no delay, which makes them ideal for use with techniques where speed is of essence. For example, when using High Level Design Languages, like the hardware description language Verilog it is possible to take advantage of the MAC operation's parallelism to better performance filters.

B. Objectives

This paper aims to:

- Design a 9-tap FIR filter based on the Multiply and Accumulate (MAC) operation.
- Implement the FIR filter in Verilog HDL, focusing on scalability and efficiency.
- Simulate the filter response using different test signals and measure performance metrics.
- Compare the performance of the hardware implementation against software-based solutions.

II. Background and Related Work

A. Overview of FIR Filters

Finite Impulse Response (FIR) filters are discrete-time filters characterized by a finite number of coefficients. These filters do not have feedback, which guarantees stability. The FIR filter's output is determined by convolving the input signal with the filter coefficients, as shown in the equation:

$$y[n] = \sum_{i=0}^{N-1} h[i] \cdot x[n-i]$$

where $h[i]$ are the filter coefficients, $x[n-i]$ are the delayed input samples, and $y[n]$ is the output. FIR filters are typically used when a linear-phase response is required, as they ensure that all frequency components of the input signal have the same delay.

B. Advantages of FIR Filters

- **Stability:** Since FIR filters do not include feedback loops, they are always stable.
- **Linear-phase response:** All frequency components of the signal experience the same time delay, which ensures the preservation of the waveform.
- **Ease of implementation:** FIR filters are straightforward to implement both in hardware and software, making them suitable for a wide range of applications.

C. Applications of FIR Filters

FIR filters are used in a wide range of applications where signal integrity and performance are critical. Some prominent applications include:

- **Signal smoothing:** FIR filters are used to remove high-frequency noise from audio or sensor signals.

- **Audio processing:** FIR filters are commonly employed in equalization and noise reduction in audio systems.
- **Biomedical signal processing:** FIR filters help clean up noisy ECG, EEG, and EMG signals to improve diagnostic accuracy.
- **Image processing:** FIR filters are used to reduce the image noise, and improve the feature in the multiple computer vision process.

D. Previous Work on FIR Filter Implementation

This paper presents the various techniques that have been employed to build FIR filters in hardware. Prior studies have been developed based on approaches for the selection of filter structures for FPGA and ASIC architectures. Especially the application of Multiply and Accumulate (MAC) units has been researched in terms of throughput as well as in terms of latency for on-line real-time DSP systems.

III. Filter Design Methodology

A. Design Steps

The design of a FIR filter follows these steps:

1) **Filter Specification:** The specifications to be met by the desired frequency response include the cutoff frequencies, the passband ripple and the stopband attenuation.
2) **Coefficient Calculation:** Parks -McClellan algorithm or windowing technique we can determine the filter coefficients depending on the given specifications.
3) **Hardware Design:** Use MAC units to perform the filtering to get the weighted sum total within the inputs samples adapted to the filter and use Verilog HDL for synthesis.
4) **Simulation and Validation:** Using sine and noise signals, as well as actual data, as input to mimic the filter to discover its capabilities.

B. Filter Coefficients

In this design, we use a 9-tap FIR filter with the following coefficients, chosen for a low-pass frequency response:

$$coeff[0] = 16'h04F6; (126.6indecimal)$$

$$coeff[1] = 16'h0AE4; (278.8indecimal)$$

$$coeff[2] = 16'h1089; (423.1indecimal)$$

$$coeff[3] = 16'h1496; (527.0indecimal)$$

$$coeff[4] = 16'h160F; (564.7indecimal)$$

$$coeff[5] = 16'h1496; (mirrorscoeff[3])$$

$$coeff[6] = 16'h1089; (mirrorscoeff[2])$$

$$coeff[7] = 16'h0AE4; (mirrorscoeff[1])$$

$$coeff[8] = 16'h04F6; (mirrorscoeff[0])$$

These coefficients are for the impulse response of the FIR filter and defines in what manner the particular filter will respond to, process or act on the chosen frequency parts of the input signal.

C. Hardware Architecture

The hardware implementation of the FIR filter employs a shift register for register delay of the input stream, a multiplier for multiplying the delayed sample with its coefficient value, and an adder to add up the products. The MAC operation is done in parallel fashion and then the result is computed in one clock cycle only.

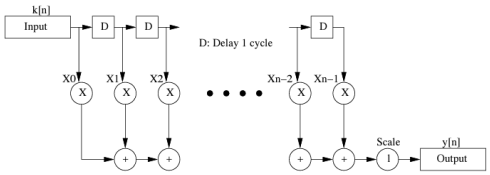

Fig. 1: FIR Filter Block Diagram

The key components of the FIR filter, which include:

- **Input Signal:** The noisy signal is passed into the system.
- **Shift Register:** To generate the required input samples for MAC operation, the input signal is delayed across the multiple pipeline stages.
- **Multiplier:** Delayed input signal is multiplied with the corresponding filter coefficient.
- **Accumulator:** The products of the delayed input signal and filter coefficients are summed to produce the final output signal.
- **Output Signal:** The accumulated result is the filtered version of the input signal.

D. Finding FIR Filter Coefficients

Filter coefficients ($h[i]$) are critical parameters that determine the behavior of an FIR filter. These coefficients are calculated based on the desired frequency response of the filter, such as low-pass, high-pass, band-pass, or band-stop characteristics. Several methods can be used to design FIR filter coefficients:

1) Windowing Method: In this approach, an ideal filter's impulse response is truncated and multiplied by a window function (e.g., Hamming, Hanning, or Blackman window). This limits the impulse response length, ensuring finite coefficients. For example, the Hamming window is defined as:

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$

The coefficients are then obtained as:

$$h[n] = w[n] \cdot h_{\text{ideal}}[n]$$

This method provides a good trade-off between computational efficiency and frequency response accuracy.

2) Frequency Sampling Method: This method directly samples the desired frequency response $H(e^{j\omega})$ at discrete points and computes the inverse discrete Fourier transform (IDFT) to get the filter coefficients. The advantage of this approach is the ability to approximate arbitrary frequency responses, but it may require more design effort for complex specifications.

3) Parks-McClellan Algorithm: This is an optimization based approach that seek to minimize approximately an error between the frequency response and the actual frequency response based on the Chebyshev approximation. This is ideal in the design of filters for complex or very strict applications. The algorithm is done in others software's like MATLAB or Python libraries like `scipy.signal.remez` to get the filter coefficients for the filter.

4) Least-Squares Method: This method reduce the squared error of the actual solution and the desired frequency response. Its advantage is high selectivity at lower frequency errors and is useful for applications were high selectivity is desirable although it is not more accurate than the Parks McClellan method in this area.

E. Multiply and Accumulate (MAC) Operation

In the FIR filter design, the basic and most important operation is the Multiply and Accumulate (MAC) operation. It is involved in sample by filter coefficient multiplication of the various input sample that have been delayed to complete the filter length and accumulation of the product of the two will yield the final output.

Mathematically, the MAC operation can be represented as:

$$y[n] = \sum_{i=0}^{N-1} h[i] \cdot x[n-i]$$

Where:

- $x[n-i]$: delayed input samples.
- $h[i]$: filter coefficients.
- $y[n]$: output of the MAC operation, which represents the filtered signal.

In the hardware the MAC operation would require specialized MAC units which are easily implemental in FPGA and ASIC architecture of the modern systems. Such units can work in a pipeline way, which at one time, synthesizes the speed and performance of FIR filters.

1) Advantages of MAC in Hardware: Using MAC units in hardware offers several advantages:

- **Parallelism:** MAC allow for parallel execution of multiplication and accumulation processes, improving throughput.
- **Reduced Latency:** The MAC process can be completed in a single clock cycle, reducing the processing time.
- **Efficient Resource Usage:** Specialized MAC consume less power and require lesser area compared to general-purpose multipliers and adders.

2) Hardware Architecture: A shift register circuit for delaying the input signal, a multiplier for the product of delayed input samples with corresponding coefficients and adder for summing up the products are developed for the hardware architecture for the FIR filter. Since the MAC operation is occurring in parallel the output is calculated within the time of one clock cycle.

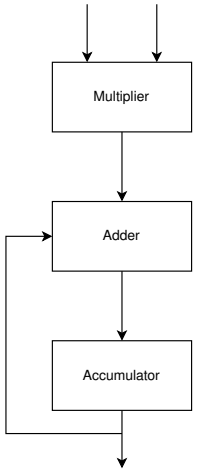

Fig. 2: MAC Block Diagram

IV. FIR Filter Pseudocode

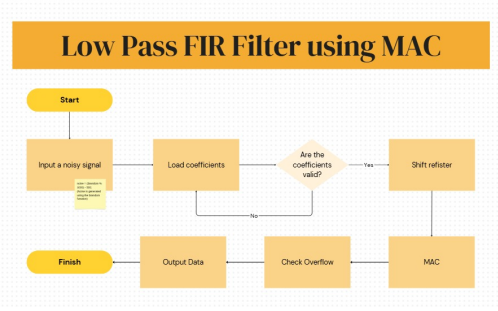

Fig. 3: FIR Filter Psuedocode Diagram

A. Initialization

1) Define arrays and registers:
   - `coeff[0:8]`: Stores filter coefficients.
   - `delayed_signal[0:8]`: Stores delayed input samples.
   - `prod[0:8]`: Stores products of coefficients and delayed signals.
   - `sum_0[0:4]`, `sum_1[0:2]`, `sum_2[0:1]`, `sum_3`: Registers for intermediate summation.
2) Initialize coefficients:

$$\texttt{coeff} = \{16'h04F6, 16'h0AE4, 16'h1089, \\ 16'h1496, 16'h160F, 16'h1496, \\ 16'h1089, 16'h0AE4, 16'h04F6\}$$

B. Main Logic (Triggered on Clock Edge)

1) **Update Delayed Signals:**

$$\texttt{delayed\_signal[0]} = \texttt{noisy\_signal}$$

For $i = 1$ to 8 :

$$\texttt{delayed\_signal[i]} = \texttt{delayed\_signal[i-1]}$$

2) **Compute Products:**

For $i = 0$ to 8 :

$$\texttt{prod[i]} = \texttt{delayed\_signal[i]} \times \texttt{coeff[i]}$$

3) **Perform Summations:**

a) **Stage 1: Add Pairs of Products**

$$\texttt{sum\_0[0]} = \texttt{prod[0]} + \texttt{prod[1]}$$
$$\texttt{sum\_0[1]} = \texttt{prod[2]} + \texttt{prod[3]}$$
$$\texttt{sum\_0[2]} = \texttt{prod[4]} + \texttt{prod[5]}$$
$$\texttt{sum\_0[3]} = \texttt{prod[6]} + \texttt{prod[7]}$$
$$\texttt{sum\_0[4]} = \texttt{prod[8]}$$

b) **Stage 2: Combine Stage 1 Results**

$$\texttt{sum\_1[0]} = \texttt{sum\_0[0]} + \texttt{sum\_0[1]}$$
$$\texttt{sum\_1[1]} = \texttt{sum\_0[2]} + \texttt{sum\_0[3]}$$
$$\texttt{sum\_1[2]} = \texttt{sum\_0[4]}$$

c) **Stage 3: Combine Stage 2 Results**

$$\texttt{sum\_2[0]} = \texttt{sum\_1[0]} + \texttt{sum\_1[1]}$$
$$\texttt{sum\_2[1]} = \texttt{sum\_1[2]}$$

d) **Stage 4: Final Sum**

$$\texttt{sum\_3} = \texttt{sum\_2[0]} + \texttt{sum\_2[1]}$$

4) **Compute Filtered Output:**

$$\texttt{filtered\_signal} = \texttt{sum\_3[35:14]}$$

(Extract 16 MSBs)

V. Implementation

A. Verilog Code

The Verilog code for implementing the 9-tap FIR filter includes the necessary components: delay registers for input samples, multipliers for the MAC operation, and an accumulator for the summing process. The code is optimized for hardware synthesis to minimize resource usage.

```
`timescale 1ns / 1ps

module fir (
  input clk,
  input signed [15:0] noisy_signal,
  output signed [15:0]
      filtered_signal
);

  integer i;

  reg signed [15:0] coeff[0:8];
  reg signed [15:0] delayed_signal
      [0:8];
  reg signed [31:0] prod[0:8];
  reg signed [32:0] sum_0[0:4];
  reg signed [33:0] sum_1[0:2];
  reg signed [34:0] sum_2[0:1];
  reg signed [35:0] sum_3;

  // Initialize coefficients
      manually
  initial begin
    coeff[0] = 16'h1000; coeff[1] =
        16'h1F0F; coeff[2] = 16'h2A3C
        ; coeff[3] = 16'h3D7B;
    coeff[4] = 16'h4E9F; coeff[5] =
        16'h5CA2; coeff[6] = 16'h6B92
        ; coeff[7] = 16'h7A63;
    coeff[8] = 16'h0AE4;
  end

  // Shift input signal through
      registers
  always @(posedge clk) begin
    delayed_signal[0] <=
        noisy_signal;
    for (i = 1; i < 9; i = i + 1)
        begin
      delayed_signal[i] <=
          delayed_signal[i - 1];
    end
  end

  // Perform multiply-accumulate
      operation
  always @(posedge clk) begin
    for (i = 0; i < 9; i = i + 1)
        begin
      prod[i] <= delayed_signal[i] *
          coeff[i];
    end
  end

```

```
  // Summing the results
  always @(posedge clk) begin
    sum_0[0] <= prod[0] + prod[1];
    sum_0[1] <= prod[2] + prod[3];
    sum_0[2] <= prod[4] + prod[5];
    sum_0[3] <= prod[6] + prod[7];
    sum_0[4] <= prod[8];

    sum_1[0] <= sum_0[0] + sum_0[1];
    sum_1[1] <= sum_0[2] + sum_0[3];

    sum_2[0] <= sum_1[0] + sum_1[1];
    sum_3 <= sum_2[0] + sum_0[4];
  end

  assign filtered_signal = sum_3;

endmodule
```

Listing 1: Verilog Code for FIR Filter

VI. Simulation and Results

A. Testbench

The Verilog testbench provides a noisy sine wave as input and verifies the FIR filter's output. A test case is set up where the sine wave input is sampled at regular intervals, and the output of the FIR filter is checked for correctness. The testbench is designed to drive the clock, apply the noisy signal, and capture the output.

B. Performance Metrics

Several performance metrics are considered to evaluate the efficiency of the filter implementation:

- **Throughput:** The rate at which data can be processed, measured in terms of filter output per clock cycle.
- **Latency:** The number of clock cycles required for an input sample to be filtered.
- **Area Efficiency:** The resource usage (e.g., LUTs, flip-flops) required to implement the FIR filter.

C. Waveform Analysis

The waveform analysis in Fig. 4 shows the noisy input signal and the filtered output. The FIR filter successfully attenuates the high-frequency noise while preserving the low-frequency components of the signal. The cutoff frequency of the filter corresponds to the transition between the passband and stopband in the frequency domain.

Fig. 4: Waveform Analysis of FIR Filter Output

VII. Conclusion

This paper presented the design and implementation of a 9-tap FIR filter using the Multiply and Accumulate (MAC) technique in Verilog HDL. The simulation results demonstrated that the filter effectively removes high-frequency noise from the input signal while preserving its low-frequency components. The design's efficiency was validated through simulation, and future work will focus on higher-order filters, FPGA implementation, and real-time signal processing applications.

VIII. Future Work

- **Higher-order FIR filters:** Implement filters with more taps to achieve finer frequency response control.
- **Parallel processing:** Investigate parallelizing the MAC operations for faster computation in hardware.
- **FPGA implementation:** Deploy the filter on an FPGA and evaluate its performance in real-time applications.

IX. Comparison Table: FIR Filter Using MAC in Verilog

Parameter	**Your Work**	**Reference 1: Optimized FIR Filter using CSA and Booth Multiplier**	**Reference 2: Symmetric Systolic FIR Filter using Multi-channel Technique**
Architecture	MAC-based FIR filter in Verilog	CSA and Booth multiplier optimization	Systolic MAC for multi-channel FIR filters
Implementation Tool	Verilog HDL, FPGA	Verilog HDL, Xilinx Vivado	Verilog HDL, FPGA
Optimization Focus	General MAC optimization	Area and delay reduction	Multi-channel modularity
Results Achieved	Custom to your data	Reduced area and delay	Enhanced resource utilization
Novelty	Unique MAC implementation	Booth multiplier-based architecture	Systolic MAC for reconfigurable design

TABLE I: Comparison of FIR Filter Implementations

References

[1] M. Morris Mano, "Digital Design," Pearson, 2012.

[2] S. Palnitkar, "Verilog HDL," Pearson, 2003.

[3] J. G. Proakis and D. G. Manolakis, "Digital Signal Processing: Principles, Algorithms, and Applications," 4th ed., Pearson, 2007.

[4] Kaur, R., Malhotra, R., & Deb, S. (n.d.). Mac based Fir Filter: A novel approach for low-power ... https://www.iiitd.edu.in/noc/wp-content/uploads/2017/11/07208065.pdf

[5] Rao, Dr. B. R. (n.d.). Design and implementation of 6-tap FIR filter using mac for low power applications. https://www.ijraset.com/research-paper/design-and-implementation-of-6-tap-fir-filter

[6] Wang, Z., Zhang, J., & Verma, N. (n.d.). Reducing quantization error in low-energy FIR filter accelerators — IEEE conference publication — IEEE Xplore. https://ieeexplore.ieee.org/document/7178126