Hardware Realization of FIR Filter Implementation through FPGA

ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT, TEKKALI, A. P., INDIA

ABSTRACT: - Distributed Arithmetic (DA) is an important technique to implement digital signal processing (DSP) functions in FPGA. It is a powerful technique for reducing the size of a parallel hardware. When DA (Distributed Arithmetic) algorithm is directly applied to the FPGA (field programmable gate array) to realize FIR (finite impulse response) filter, it is difficult to achieve the best configuration in the coefficient of FIR filter, the storage resource and the computing speed. According to this problem, the paper provides the detailed analysis and discussion in the algorithm, the memory size and look-up table speed. Also, the corresponding optimization and improvement measures are discussed and the concrete hardware realization of the circuit is presented.

The required size of memory with improved algorithm is \(2^{M/4} + 2^{M/4} = 2^{M/2} + 1\), where it is with traditional one is \(2^{M-1}\), its memory scale is only \(2^{3M/4 + 2}\) times of the original. Through the algorithm improvement, the hardware resource is reduced and the operation speed is improved. In this project a 16th order FIR filter is proposed to be implemented. Design, Implementation and Verification are aimed in this project. XILINX's Spartan 3E FPGA is targeted for this implementation. XILINX ISE Foundation (9.1ISE (or) 10.1ISE (or) 12.2ISE) software is used for the FPGA design flow which includes Synthesis, Translation, Mapping, Floor planning, Placing and Routing, Post Place and Route simulation and Bit file generation.

The results of simulation and the test show that this method greatly reduces the FPGA hardware resource and the high speed filtering is achieved. The design has a big breakthrough compared to the traditional FPGA realization.

KEY TERMS: Improved DA algorithm, FPGA, Xilinx 10.1SE, Look-Up Table and Bit Level Rearrangement.

1. INTRODUCTION: DA algorithm is simply known as “Distributed Arithmetic” algorithm. Which is invented and proposed by “Crosier” in the year of 1973? Distributed arithmetic algorithm is best and efficient technique for calculation of sum of products or multiple and accumulation (MAC) applications. The main advantage of the distributed arithmetic algorithm is it’s the best analyzer of data path circuits while in designing. And one more fabulous advantage of distributed arithmetic algorithm is hardware required is reduced up to 80% while comparing with and without usage of (DA) Distributed arithmetic algorithm. Sometimes by using distributed arithmetic algorithm the total hardware requirement of design in a Digital signal processing circuit will be reduced up to less than 50%. Actually it’s an old technique that was introduced and proposed by the “Crosier” in the year of 1973.

But, in recent days, digital signal processing (DSP) circuits are implementation using field programmable gate array (FPGA) has a great advantage. But by using the (DA) distributed arithmetic algorithm, it gives great advantage for the hardware implementation of Digital signal processing circuits using field programmable gate array (FPGA). Due to this only now-a-days (DA) distributed arithmetic algorithm having great demand. By using (DA) distributed arithmetic algorithm, we can implement (MAC) multiple and accumulator system. For implementing (MAC) multiple and accumulation system, (DA) distributed arithmetic algorithm uses basic building of (FPGA) field programmable gate array like (LUTs) look-up tables.

1.1 DESCRIPTION OF MAC OPERATION:

The name itself stands for (MAC) multiplier and accumulation operation. The name “Multiply” stands for the operation of the multiplication and “Accumulation” stands for the addition. Both the operations of multiplications and accumulation are done simultaneously is known as (MAC) multiply and accumulation operation. The following expression represents that the (MAC) multiply and accumulation operation
\[ y = A_1 \times x_1 + A_2 \times x_2 + \cdots + A_K \times x_K \]

\[ i.e. \quad y = \sum_{k=1}^{K} A_k x_k \]

Where ‘A’ is a matrix of \textit{Constant} values.

‘X’ is a matrix of input variables.

Each ‘A_k’ is having M-bits.

Each ‘X_k’ is having N-bits.

‘y’ should be a memory element

‘y’ should be able to store the resultant value of an expression.

\textbf{Example:} where \( A = [2, 4, 6, 8] \) and \( X = [1, 3, 5, 7] \) where \( K = 4 \).

\textbf{Solution:}

\[ y = 2 \times 1 + 4 \times 3 + 6 \times 5 + 8 \times 7 \]

\[ y = 2 + 12 + 30 + 56 \]

\[ y = 14 + 86 = 100. \]

Below figure shows that the hardware requirement for (MAC) multiplier and accumulator.

\textbf{1.2 POSSIBLE HARDWARE:}

Let \( A = [C_1, 2, C_3, C_4] \) and \( X = [A, B, C, D] \) where the value of \( K = 4 \).

\textbf{3. REDUCING THE MEMORY SIZE:}

\textbf{3.1 Memory Partitioning:} One of several possible ways to reduce the memory size is to partitioning the memory into smaller pieces of memories that are added before the shift accumulator. The amount of memory reduced from \( 2^N \) words to \( 2.2^{N/2} \) words if the original memory is partitioned into 2 parts. Below figure shows that
3.2 Memory Coding: The second approach is based on a special coding of the ROM content. Memory size can be halved by using the ingenious scheme based on the identity \[ X = \frac{1}{2} [x - (x')] \]

In two’s compliment representation the identity can be written
\[
x = \frac{1}{2} \left[ x_0 + \sum_{k=1}^{d-1} x_k 2^{-k} - \left( x_0 + \sum_{k=1}^{d-1} x_k 2^{-k} - \frac{2}{d} \right) \right] = -(x_0 - x'_0)2^{-1} + \sum_{k=1}^{d-1} (x_k - x'_k)2^{-k} - 1 - 2^{-W_d}
\]

Notice that \((x_k - x'_k)\) can only take on the values of (-1) or (+1). By inserting this expression into the inner product yields
\[
y = \sum_{k=1}^{W_d} W_k \left[ F_k(x_1k, x_2k, ..., x_Nk) = \sum_{l=1}^{N} (x_{lk} - x_{lk'}) \right]
\]

The function \(F_k\) is shown in the table for \(N = 3\)

<table>
<thead>
<tr>
<th>(X_1)</th>
<th>(X_2)</th>
<th>(X_3)</th>
<th>(F_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>-A1-A2-A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>-A1-A2+A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>-A1+A2-A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>-A1+A2+A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>A1-A2-A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>A1-A2+A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>A1+A2-A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>A1+A2+A3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Anti-Symmetry can be occurs at ‘011’. Notice that only half the values are needed, since the other half can be obtained by changing the signs. The pixels that are multiplied by the same coefficient area added (or subtracted).

The ROM content is …

If \(a_i \text{ XOR } b_i = 1\) the F values are applied directly to the accumulators, and IF \(a_i \text{ XOR } b_i = 0\) the F values are interchanged. The F values are either added to, or subtracted from, the accumulator’s registers depending on the data bits \(a_i\) and \(b_i\).

4. IMPROVED DESIGN OF THE DA ALGORITHM:

From Eq. (2), \(X_m\) can be expressed as Eq. (4).
\[
x_k = \frac{1}{2} [x_k - (x'_k)] \quad \ldots (6)
\]

Where the –\(X_m\) can be expressed as Eq. (1) according to the binary compliment operation [3].
\[
-x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} + 2^{-(N-1)} \ldots (6)
\]

The step by step derivation can be calculated and then the result could be estimated like..
\[
x_k = \frac{1}{2} \left[ -(x - b_{k0}) + \sum_{n=1}^{N-1} (b_{kn} - b_{kn'}) 2^{-n} - 2^{-(N-1)} \right]
\]

For convenience, two variables are defined as follows:
\[
\varphi_{mn} = -(x_{mn} - x'_{mn})
\]

In which, as the value of \(x_{mn}\) is 0 or a, so the value of \(\varphi_{mn}\) and \(\varphi_{mn}\) is ±1. Then Eq. (6) can be expressed as Eq. (7).
\[
x_k = \frac{1}{2} \sum_{n=0}^{N-1} c_{kn} 2^{-n} - 2^{-(N-1)} \]

As there are \(2^m\) different kinds of results of
\[
y = \sum_{k=1}^{K} A_k x_k \]
\[
y = \sum_{k=1}^{K} A_k \left[ \sum_{n=0}^{N-1} c_{kn} 2^{-n} - 2^{-(N-1)} \right]
\]
\[
y = \sum_{n=0}^{N-1} \sum_{k=1}^{K} A_k c_{kn} 2^{-n} - \frac{1}{2} \sum_{k=1}^{K} A_k 2^{-(N-1)}
\]

And the value of \(\varphi_{mn}\) is ±1, so the results show positive and negative symmetry property. If the positive and negative sign are not considered, there are only \(2^{m-1}\) different kind of results and the size of storage will reduce by half.
\[
y = \sum_{n=0}^{N-1} \sum_{k=1}^{K} A_k c_{kn} 2^{-n} - \frac{1}{2} \sum_{k=1}^{K} A_k 2^{-(N-1)} \ldots (9)
\]
In which, \( z \geq y, y \geq b, b \geq a+1, a>1 \), so an inner product operation with the scale of \( 2M \) will be realized through several LUTs with different or same depth and adders. The scale of the memory is \( 2^a + 2^b - a + \cdots + 2^z - y + 2^M/2 \). For example, if using two LUTs with depth of \( M/4 \) and adders to achieve it, namely, Then the size of memory is \( 2^M/4 = 2^{M+1} \). Compared with the memory size which is \( 2^M - 1 \) before optimizing, its memory scale is only \( 2^{-3M/4} \) times of the original. The simplified hardware circuit structure is as shown in below Fig.1.

![Fig.2: The circuit structure of FIR system](image)

**5. THE CIRCUIT DESIGN OF FIR FILTER:**

**A. Design Index And Parameters Extraction:**

A 16\(^{th}\) order FIR filter is designed. Its parameters are as follows: the sampling frequency is 2.25MHz; the pass band cut-off frequency is 100 KHz; the width of the input data, the output data and the filter coefficient is 8, 16 and 20 bits respectively. It adopts Hamming window to design and MAT Lab simulation to calculate its unit-sampling response \( h(k) \) and simply it \( 2^{16} \) times. The \( h(k) \) is as follows:

\[
\begin{align*}
H(0) &= H(15) = 298D; \\
H(1) &= H(14) = 578D \\
H(2) &= H(13) = 1364D; \\
H(3) &= H(12) = 2718D \\
H(4) &= H(11) = 4503D; \\
H(5) &= H(10) = 6400D \\
H(6) &= H(09) = 7996D; \\
H(7) &= H(08) = 8908D \\
\end{align*}
\]

**B. The Hardware Circuit Unit:**

The address maker circuit generated the LUT address. The upper half of the address looks up its corresponding pre-storing value.

The hardware circuit is shown in Fig. 2.

**C. Circuit Simulation And Testing:**

The input sequence is \( x(n) = [0, 3, 1, 0, 2, 1, 4, 3, 2, 2, 0, 1, 2, 2, 3] \) and the simulation waveform is shown in figure 3.

![Fig 3: The simulation waveform](image)
6. HARDWARE ARCHITECTURE:

The below figure shows that the internal architecture of the FIR filter design present here using techniques. It will give us the realistic view of the internal architecture of the FIR filter design using Improved DA algorithm implementation using Field Programmable Gate Array (FPGA) is can be developed using Verilog hardware Descriptive Language and it can be developed by using the Spartan 3E S350E hardware kit. It can realize the hardware requirement of the FIR filter while developing with and without Improved DA algorithm.

CONCLUSIONS:

DA is a very efficient means to mechanize computations that are dominated by inner products. DA has always fared well, not always (but often) best, and never poorly. DA is a very efficient mechanism for computations that are dominated by inner products (Convolution). If performance/cost ratio is critical, DA should be seriously considered as a contender.

The complicated multiplication-accumulation operation is converted to the shifting and adding operation when the DA algorithm is directly applied to realize FIR filter. Aiming at the problems of the best configuration in the coefficient of FIR filter, the storage resource and the calculating speed, the DA algorithm is optimized and improved in the algorithm structure, the memory size and the LUT speed. The arithmetic expression has clear layers of derivation process and the circuit structure is reasonable, which make the memory size smaller and the operation speed faster. The design improves greatly compared to the conventional FPGA realization and it can be flexible applied to implement high-pass, low-pass and band-pass filters by changing to the order and the LUT coefficient.

REFERENCES:


Author Description:

This is Eshwararao Boddepalli, completed my Bachelor of technology in Electronics and Communication Engg. Pursuing master of technology in the stream of VLSI System Design. My research area is VLSI and FPGA using DSP implementations.

This is Lokeshraju Vysyaraju, completed my Master of Technology. Now, I am working as an Assoc. professor in the Department of Electronics and Communication Engineering, Aditya Institute of Technology and Management, Andrapradesh, India.