Design and Realization of practical FIR filter based on CSD and DA algorithms

R Thaviti raju1, K Chitambararao2, T Viswanadham3, B. Chinnarao4
1M.TechScholar, Department of ECE, AITAM
2Assoc. Professor, Department of ECE, AITAM
3 Asst. Professor, Department of ECE, AITAM, 4Assoc. Professor, Department of ECE, AITAM

Abstract: FIR digital filters find immense applications in mobile communications systems such as channel equalization, channelization, matched filtering and pulse shaping, due to their absolute stability and linear phase properties. In this paper it is proposed to design a practical FIR filter using MATLAB tool to obtain the response. After that the filter will be designed and analyzed based on canonical signed digits and compared with the distributed arithmetic algorithm in order to minimize the power consumption and fast implementation of the filter. The design filter will be simulated and synthesized using Xilinx ISE 13.1 software.

Keywords: FIR, CSD, DA, VERILOG HDL

I. Introduction

Recently many technologies have come out in those technologies electronic technology plays a vital role with development of marvelous speed. As of late, digital signal processing (DSP) is utilized as a part of a considerable measure of uses, for example, computerized set-top box, acoustic pillar formers, computerized adaptable plate, versatile video frameworks/PCs, advanced sound, computerized radio, media and remote interchanges, advanced still and system cameras, discourse handling, transmission frameworks, video pressure, link modems, radar imaging, worldwide situating frameworks, and biomedical sign preparing. The field of DSP has dependably been oversee by the advances in DSP applications and in scaled very-large-scale-integrated (VLSI) innovations. Filtering procedure is utilized as a part of the sign handling area to achieve a coveted recurrence band from a framework as the o/p by giving some contribution to it. The framework that is utilized for the separating operation is known as the channel. In DSP, there are essentially two sorts of channel, IIR and fir channel. The motivation reaction of the IIR channel is of unbounded span where as in fir channel it is of limited term if there should a rise an occurrence of fir channel. The fir channel requires no criticism way and along these lines it has no recursion and subsequently the fir channel is non_recursive. Fir channels detail incorporate greatest middle of the road stop band swell, pass band and stop band edge recurrence. The coefficients of fir channel requires impressive measure of figurings. Along these lines it is by and large performed by utilizing different PC supported configuration apparatuses, for example, channel outline and examination device of MATLAB. So for a continuous applications, for example, separating, combinational multipliers are utilized in view of fast. The vast majority of the equipment many-sided quality is because of multipliers, as channels require expansive no of augmentation, prompting extreme region, postpone and control utilization regardless of the fact that executed in a full exceptionally coordinated circuits now the issue confronted is that how lessen the equipment multifaceted nature of a multiplier. The principle anxiety is on the lessening of multipliers in fir channel the real impediment of higher request need. The higher request forces more equipment necessities, number-crunching operations, territory use and power utilization when outlining the channel. Accordingly, minimizing or diminishing these parameters, is significant objective in advanced channel outline assignment. It is yearning to discover productive calculation that require as couple of math operations as could reasonably be expected, as this in the zone and minimizes the gadget size and vitality utilization. To evacuate the repetitive calculation which prompts more proficient calculations the procedures picked are CSD, DA. This is utilized to streamline the region of high pass fir channel. In CSD structure the channel coefficients are settled. In CSD structure multiplier region get lessened. DA is essentially somewhat serial computational operation that structures and inward result of a couple of vectors in a one direct stride the upside of DA is its productivity of mechanization.
II. Existing Methodology

Fig 1 Direct form of FIR filter structure

Till now FIR filters are designed and implemented in VLSI domain without taking real coefficients. Our proposed method is to represent the FIR filters with real coefficients and implement the same in VLSI. Apart from the coefficients, they will be obtained by using MATLAB command window.

III. Proposed Methodology

Fig 2 Practical FIR filter

Block Diagram:

- Based on specifications design a FIR high pass filter
- Obtain the coefficients using MATLAB
- Realize the filter in direct form structure
- Apply the CSD and DA Algorithms
- Compare the results in terms of area, power, and delay

Flow Chart:

1. Take filter specifications
2. Based on specifications design a FIR filter using window technique
3. If symmetrical coefficients:
   - Apply the double precision floating point format
   - Obtain the final version using Q1.15 format
   - Binary representation of numbers
   - Coefficients in CSD algorithm
   - Coefficients in DA algorithm
   - Compare the results in terms of power area and delay
Realization:

Floating Point Format:

Floating point representation works well for numbers with large dynamic range based on the no.of bits. This standard is almost exclusively used across computing platforms and hardware designs that support floating point arithmetic. In this standard a normalized floating point number \( x \) is stored in three parts: the sign \( s \), the excess exponent \( e \), and the significand or mantissa \( m \), and the value of the number in terms of these parts is:

\[
x = (-1)^s \times 1 \times m \times 2^{-e}
\]

The format is written with the significand having an indirect integer bit of value 1 (except for special data, see the exponent encoding below). With the 52 bits of the fraction significand become visible in the memory format, the total precision is therefore 53 bits. The bits are laid out as follows:

\[
\begin{align*}
\text{exponent} & \quad \text{sign} \quad (11 \text{ bit}) \\
\text{fraction} & \quad (52 \text{ bit})
\end{align*}
\]

From the matlab command window the real filter coefficients are 0.0038, -0.035, -0.2278, 0.610, 0.0037, -0.331, -0.2291. These filter coefficients are converted to double precision floating point number. The converted coefficients are 0.003775018138711, -0.03354142857911, 0.227792539932163, 0.618026583319437, -0.229088808118480, -0.033110787270075, 0.003737449345339.

i) Q-FORMAT:

Q is a fixed point format where the number of fractional bits (and optionally the number of integer bits) is specified. For example, a Q15 number has 15 fractional bits: a Q1.15 number has 1 integer bit and 14 fractional bits. Q format is commonly used in hardware that does not have a floating-point unit and in applications that require constant resolution.

The Q\(_{n,m}\) format of an N bit number sets \( n \) bits to the left and \( m \) bits to the right of the binary point. In case of signed numbers, the MSB is used for the sign and has negative weight. A two’s complement fixed point number in Q\(_{n,m}\) format is equivalent to \( b=b_n b_{n-1} b_{n-2} b_{n-3} \ldots b_1 b_0 b_{-1} b_{-2} \ldots b_{-m} \) with equivalent floating point value \( -b_n 2^{-1} + b_{n-1} 2^{-2} + \ldots + b_1 2^1 + b_0 + b_{-1} 2^{-1} + \ldots b_{-m} 2^{-m} \).

A floating point number format is simply converted to Q\(_{n,m}\) fixed point format by bringing \( m \) fractional bits of the number to the integer part and then dropping the rest of the bits with or without rounding. This conversion translates a floating point number to an integer number with an implied decimal the implied decimal needs to be remembered by the designer for referral in further processing of the number in different calculations:

\[
\text{Num\_fixed} = \text{round}(\text{num\_float} \times 2^m)
\]

Or

\[
\text{Num\_fixed} = \text{fix}(\text{num\_float} \times 2^m)
\]

The coefficients are converted to double precision floating number into fixed point format. The coefficients are in decimal numbers that are 124, -1099, -7464, 20252, -7507, -1085, 123. These decimal numbers are converted to binary then hexadecimal number.
IV. Csd Algorithm

The CSD code is a ternary number system with the digit set \{1, 0, 1\}, where 1 stands for 1. Given a constant, the corresponding CSD representation is unique. CSD representation of a number can be recursively computed using the string property and has two main properties:

1. The number of nonzero digits is minimal
2. No two consecutive digits are both nonzero, that is, two nonzero digits are not adjacent.

The first property implies a minimal Hamming weight, which leads to a reduction in the number of additions in arithmetic operations. The second property provides its uniqueness characteristic. However, if this property is relaxed, this representation is called the minimal signed digit (MSD) representation, which has as many nonzero as the CSD representation, but which provides multiple representations for a constant. It enables the reduction of the number of partial products that must be calculated fast and also low-power consumption and low area structure of a multiplier for DSP applications or self-timed circuits. From the practical point of view, the traditional approach to generate the CSD representation. All of these algorithms generate the CSD code recursively from the least significant bit (LSB) to the most significant bit (MSB).

The CSD representation of an integer number is assigned and unique digit representation that contains no adjacent non-zero digits. Given an n-digit binary unsigned number \(X=\{x_0, x_1, \ldots, x_{n-1}\}\) expressed as

\[
X= \sum_{i=0}^{n-1} x_i 2^i, \quad x_i \in \{0,1\}
\]

Then the \((n+1)\)-digit CSD representation \(Y=\{y_0, y_1, \ldots, y_n\}\) of \(X\) is given by

\[
Y = \sum_{i=0}^{n-1} x_i 2^i = \sum_{i=0}^{n} y_i 2^i, \quad y_i \in \{1,0,1\}
\]

The condition that all non-zero digits in a CSD number are separated by zero implies that

\[y_{i+1}y_i = 0, \quad 0 \leq i \leq n - 1\]

From this property, the probability that a CSD n-digit has a non-zero value is given by

\[P(\{y_i = 1\}) = 1/3 + 1/9n[1 - (-1/2)^n]\]

As \(n\) becomes large, this probability tends to 1/3 while this probability becomes 1/2 in a binary code. Using this property, the number of additions/subtractions is reduced to minimum in multipliers and as a result, an overall speed-up can be achieved. Encoding 2 is preferable since it satisfies the following relation.

\[y_i = y_i^d - y_i^s\]

Where \(y_i^s\) represents the sign bit and \(y_i^d\) the data bit. This encoding also allows an additional valid representation of 0 when \(y_i^s=1\) and \(y_i^d=1\), which is useful in some arithmetic implementations. In the whole paper, this encoding is used.

CSD representation for binary form:

\[
\begin{align*}
\text{h(0)} &= 124 = 0000 0000 0111 \ 1100 = 0000 0000 1000 \ 0100 \text{(CSD form)} \\
\text{h(1)} &= 1099 = 0000 0100 0100 1011 = 0000 0100 0101 \ 0101 \text{(CSD form)} \\
\text{h(2)} &= 7464 = 0001 1101 0010 1000 = 0010 0101 \ 0010 \ 1000 \text{(CSD form)} \\
\text{h(3)} &= 20252 = 0100 1111 0001 0111 = 0011 0101 \ 0011 \ 0101 \text{(CSD form)} \\
\text{h(4)} &= 7507 = 0001 1101 0101 0011 = 0010 0101 \ 0101 \ 0101 \text{(CSD form)} \\
\text{h(5)} &= 1085 = 0000 0100 0011 1101 = 0000 0100 0100 \ 0100 \text{(CSD form)} \\
\text{h(6)} &= 123 = 0000 0000 0111 1011 = 0000 0000 1000 \ 0101 \text{(CSD form)} \\
\end{align*}
\]

\[
\begin{align*}
X_n(0) \cdot h(0) + x_n(1) \cdot h(1) + x_n(2) \cdot h(2) + x_n(3) \cdot h(3) + x_n(4) \cdot h(4) + x_n(5) \cdot h(5) + x_n(6) \cdot h(6)
\end{align*}
\]

V. Distributive Algorithm

DA is another way of implementing a dot product where one of the arrays has constant elements. The DA can be effectively used to implement FIR, IIR and FFT type algorithm. In the case of an FIR filter, the coefficients constitute an array of constants in some signed Q format where the tapped delay line forms the array.
of variables which changes every sample clock. The DA logic replaces the MAC operation of convolution summation of into a bit serial look up table read and addition operation. The architecture of FPGAs, time/area effective designs can be implemented using DA techniques. The DA logic works by first expanding the array of variable numbers in the dot product as a binary number and then rearranging MAC terms with respect to weights of the bits. Let the different elements of arrays of constants and variables are \( A_k \) and \( X_k \) respectively. The length of both the arrays is \( K \). Then their dot product can be written as:

\[
y = \sum_{k=0}^{K-1} A_k x_k
\]

Let us assume \( x_k \) is an \( N \) bit \( Q_1.(N-1) \) format number:

\[
x_k = x_{k0} 2^0 + \sum_{b=1}^{N-1} x_{kb} 2^{-b}
\]

The dot product can be written as:

\[
y = \sum_{k=0}^{K-1} (-x_{k0} 2^0 + \sum_{b=1}^{N-1} x_{kb} 2^{-b}) A_k
\]

Rearranging the terms yields:

\[
y = -\sum_{k=0}^{K-1} x_{k0} A_k 2^0 + \sum_{b=1}^{N-1} \sum_{k=0}^{K-1} x_{kb} A_k
\]

For \( k=3 \) and \( N=4 \), the rearrangement forms the following entries in the ROM:

\[
- (x_{00} A_0 + x_{10} A_1 + x_{20} A_2) 2^0 \\
+ (x_{01} A_0 + x_{11} A_1 + x_{21} A_2) 2^1 \\
+ (x_{02} A_0 + x_{12} A_1 + x_{22} A_2) 2^2 \\
+ (x_{03} A_0 + x_{13} A_1 + x_{23} A_2) 2^3
\]

The DA technique pre computes all possible values of \( \sum_{k=0}^{K-1} x_{kb} A_k \)

The ROM is \( P \) bits wide and \( 2^P \) deep and implements a look up table. The value of \( P \) is:

\[
P = \left\lfloor \log_2 \sum_{k=0}^{K-1} |A_k| \right\rfloor + 1
\]

<table>
<thead>
<tr>
<th>( x_{20} )</th>
<th>( x_{10} )</th>
<th>( x_{00} )</th>
<th>Contents of ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( A_0 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( A_1 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( A_1 + A_0 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( A_1 + A_0 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( A_2 + A_1 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( A_2 + A_1 + A_0 )</td>
</tr>
</tbody>
</table>

**Table1**: ROM for Distributed Arithmetic

All the elements of the vector are stored in the Shift Register. The architecture considers in each cycle the \( b^{th} \) bit of all the elements and concatenates them to from the address to the ROM. From the MSB, value in the ROM is subtracted from a running accumulator, and for the rest of the bit locations values from ROM are added in the accumulator. The size of the accumulator is set to \( P+N \), where a \( P \) bit adder adds the current output of the ROM in the accumulator and \( N \) bits of the accumulator are kept to the right side to cater for the shift operation.
It is obvious from the configuration of a DA based design that the size of ROM increases with an increase in the number of coefficients of the filter. This size is prohibitively large and DA techniques are used to reduce the ROM requirement.

VI. Simulation Results

![Simulation result of FIR filter without CSD algorithm](image)
Design and Realization of practical FIR filter based on CSD and DA algorithms

VII. Synthesis Report

**Device Utilization Summary**

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>128</td>
<td>9,312</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>188</td>
<td>9,312</td>
<td>2%</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>143</td>
<td>4,656</td>
<td>3%</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>143</td>
<td>143</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
<td>143</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>188</td>
<td>9,312</td>
<td>2%</td>
</tr>
</tbody>
</table>

**Fig9** Synthesis report of FIR filter without CSD algorithm

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>159</td>
<td>9,312</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>1,406</td>
<td>9,312</td>
<td>15%</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>784</td>
<td>4,656</td>
<td>16%</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>784</td>
<td>784</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
<td>784</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>1,413</td>
<td>9,312</td>
<td>15%</td>
</tr>
</tbody>
</table>

**Fig10** Synthesis report of FIR filter with CSD algorithm
VIII. Power Analysis

Fig 11: Synthesis report of FIR filter with Distributive arithmetic algorithm

Fig 12: Power estimator of FIR filter without CSD algorithm

Fig 13: Power estimator of FIR filter with CSD algorithm

Fig 14: Power estimator of FIR filter Distributive arithmetic algorithm
The static power can directly obtained from synthesis results but Xilinx software can’t provide dynamic power information. For this dynamic power analysis the Xilinx provides plugin support such as XPower Estimator (XPE). XPower Estimator-11.1 is Microsoft Excel spread book, which provides detailed power analysis by using mapping report file generated during the synthesis using Xilinx synthesizer. The dynamic power is 165mW and static power is 79mW. The total power is 244mW whereas the total power consumption for DA based FIR filter is 244mW.

IX. Conclusion

The FIR filters are extensively used in digital signal processing and can be implemented using programmable digital processors. With the advancement in VLSI technology as the DSP has become increasingly popular over the years, the high speed realization of FIR filter with his power consumption has become much more demanding. In this paper, FIR high pass filter is designed by using hamming window and obtain the coefficients using MATLAB. Moreover the CSD and DA algorithms are applied and compared the results interners of area, power and delay.

References


AUTHOR PROFILE:
Mr. R. Thavitiraju is presently pursuing his M.Tech in VLSI system design in Electronics and Communication Engineering Department, AITAM, Tekkali. His areas of interest are Low Power VLSI system design and digital filter optimization. He has attended for one national level workshop. He is membership in GSM IEE. The author may be reached at rtraju1577@gmail.com.
Mr. K. ChitambaraRao is presently working as Associate Professor in Electronics and Communication Engineering Department, AITAM, Tekkali. He completed M.Tech from Sathyabhama University in the specialization of VLSI Design and registered in Ph.D in Andhra university (his research area antennas and wave propagation). He has 12 years’ experience in teaching and research. He published more than 11 research papers in National/International Journals and Conferences. He is a life member of ISTE, IETE.

Mr. T. Viswanadham is presently working as Assistant Professor in Electronics and Communication Engineering Department, AITAM, Tekkali. He completed M.Tech from JNT University in the specialization of VLSI-System Design and registered in Ph.D in Andhra university (interested research areas are Biomedical signal processing and VLSI). He has 12 years’ experience in teaching and research. He published 8 research papers in National/International Journals and Conferences. He is a life member of SEMCE(I).

Mr. B. Chinnarao is presently working as Associate Professor in Electronics and Communication Engineering Department, AITAM, Tekkali. He completed M.Tech from JNTU, Hyd. in the specialization of signal processing and registered in Ph.D in JNTU, Hyd. (his research area image processing). He has 15 years’ experience in teaching and research. He published more than 30 research papers in National/International Journals and Conferences. He is a life member of ISTE and member in IEEE.