A NOVEL APPROACH FOR HIGH SPEED CONVOLUTION OF FINITE AND INFINITE LENGTH SEQUENCES USING VEDIC MATHEMATICS

M. Bharathi¹, D. Leela Rani²

¹Assistant Professor, ²Associate Professor, Department of ECE, Sree Vidyanikethan Engineering College, Tirupati, India, bharathi891@gmail.com, dlrani79@gmail.com

Abstract

Digital signal processing, Digital control systems, Telecommunication, Audio and Video processing are important applications in VLSI. Design and implementation of DSP systems with advances in VLSI demands low power, efficiency in energy, portability, reliability and miniaturization. In digital signal processing, linear-time invariant systems are important sub-class of systems and are the heart and soul of DSP.

In many application areas, linear and circular convolution are fundamental computations. Convolution with very long sequences is often required. Discrete linear convolution of two finite-length and infinite length sequences using circular convolution on for Overlap-Add and Overlap-Save methods can be computed. In real-time signal processing, circular convolution is much more effective than linear convolution. Circular convolution is simpler to compute and produces less output samples compared to linear convolution. Also linear convolution can be computed from circular convolution. In this paper, both linear, circular convolutions are performed using vedic multiplier architecture based on vertical and cross wise algorithm of Urdhva-Tirayabhyam. The implementation uses hierarchical design approach which leads to improvement in computational speed, power reduction, minimization in hardware resources and area. Coding is done using Verilog HDL. Simulation and synthesis are performed using Xilinx FPGA.

Keywords: Linear and Circular convolution, Urdhva - Tirayagbhyam, carry save multiplier, Overlap –Add/ Save Verilog HDL.

1. INTRODUCTION

Systems are classified in accordance with a no. of characteristic properties or categories, namely: linearity, causality, stability and time variance. Linear, time-invariant systems are important sub-class of systems. Urdhva-Tirayagbhyam sutra is used in developing carry save multiplier architecture to perform convolution of two finite and infinite length sequences [1]. Linear and circular convolutions, which are fundamental computations in Linear time-invariant (LTI) systems are implemented in Verilog HDL. Simulation and Synthesis are verified in Xilinx 10.1 ISE.

Multiplications, in general are complex and slow in operation. The overall speed in multiplication depends on number of partial products generated, shifting the partial products based on bit position and summation of partial products. In carry save multiplier, the carry bits are passed diagonally downwards, which requires a vector merging adder to obtain final sum of all the partial products. In convolution, fundamental computations includes multiplication and addition of input and impulse signals or samples[2],[3].

2. CIRCULAR CONVOLUTION

Let \(x_1(n)\) and \(x_2(n)\) be two finite- duration sequences of length N. Their respective N-point DFT’s are

\[
X_1(K) = \sum_{n=0}^{N-1} x_1(n) e^{-j2\pi nk/N}, \quad k = 0, 1… N-1 \quad (1)
\]

\[
X_2(K) = \sum_{n=0}^{N-1} x_2(n) e^{-j2\pi nk/N}, \quad k = 0, 1… N-1 \quad (2)
\]

If two DFT’s are multiplied together, the result is a DFT, \(X_3(k)\) of a sequence \(x_3(n)\) of length N.

The relationship between \(X_3(K)\) and sequences \(X_1(k)\) and \(X_2(k)\) is

\[
X_3(k)=X_1(k)X_2(k), \quad k=0,1,\ldots, N-1 \quad (3)
\]
The DFT of \( x_k(k) \) is

\[
x_k(n) = \sum_{m=0}^{N-1} x_1(n)x_2((m-n))_N \quad m=0,1...N-1
\]

(4)

Here

\[
x((m-n))_N = x(m-n+N)
\]

(5)

The above expression has the form of a convolution sum. It involves the index \((m-n)\)_N and is called circular convolution[4].

It is not the ordinary linear convolution which relates the output sequence \( y(n) \) of a linear system to the input sequence \( x(n) \) and the impulse response \( h(n) \). Thus it can be concluded that the multiplication of the DFT’s of two sequences is equivalent to circular convolution of two sequences in the time domain.

The methods that are used to find the circular convolution of two sequences are

a. Concentric circle method

b. Matrix multiplication method

Let \( x_1(n) \) and \( x_2(n) \) be two sequences of length \( L \) and \( M \) respectively.

Let \( x_3(n) \) be the output sequence. The length \( N \) of the output sequence, \( N = \text{Max} \ (L, M) \).

2.1. Concentric circle method

The length of \( x_1(n) \) should be equal to length \( x_2(n) \) in order to perform circular convolution using concentric circle method.

We have three cases here.

- The length \( L \) of sequence \( x_1(n) \) is equal to length \( M \) of sequence \( x_2(n) \), then the procedure explained below can be followed directly.
- If \( L>M \) then \( M \) is made equal to \( L \) by adding \( L-M \) number of zero samples to the sequence, \( x_2(n) \).
- If \( M>L \), then \( L \) is made equal to \( M \) by adding \( M-L \) number of zero samples to the sequence \( x_1(n) \).

After making the lengths of two sequences equal to \( N \) samples the circular convolution using concentric circle method between two sequences is performed using following steps. The \( N \) samples of sequence \( x_1(n) \) are graphed as equally spaced points around an outer circle in counter clockwise direction.

- Starting at the same point as \( x_1(n) \) the \( N \) samples of \( x_2(n) \) are graphed as equally spaced points in clockwise direction around an inner circle.

- The corresponding samples are multiplied on two circles and the products are added to produce first sample of output sequence, \( x_3(n) \).

The samples on the inner circle are rotated one position in counter clockwise direction successively and step 3 is repeated to obtain the next sample of output sequence \( x_3(n) \).

- Step 4 is repeated until the first sample of inner circle lines up with the first sample of outer circle once again. Hence all the samples of output sequence \( x_3(n) \) are collected.

2.2. Matrix Multiplication Method

Circular convolution of two sequences \( x_1(n) \) and \( x_2(n) \) is obtained by representing the sequences in matrix form as shown below

\[
\begin{bmatrix}
    x_1(0) & x_1(N-1) & \ldots & x_1(1) \\
    x_2(1) & x_2(0) & \ldots & x_2(2) \\
    \vdots & \vdots & \ddots & \vdots \\
    x_1(N-1) & x_2(N-2) & \ldots & x_2(0) \\
\end{bmatrix} \times 
\begin{bmatrix}
    x_2(0) \\
    x_2(1) \\
    \vdots \\
    x_2(N-1) \\
\end{bmatrix} = 
\begin{bmatrix}
    y_1(0) \\
    y_1(1) \\
    \vdots \\
    y_1(N-1) \\
\end{bmatrix}
\]

(6)

The columns of \( NxN \) matrix is formed by repeating the samples of \( x_2(n) \) via circular shift. The elements of column matrix are the samples of sequence \( x_1(n) \). The circular convolution of two sequences, \( x_3(n) \), is obtained by multiplying \( NxN \) matrix of samples of \( x_2(n) \) and column matrix which consists of samples of \( x_1(n) \).

3. LINEAR CONVOLUTION OF SHORT DURATION SEQUENCE

In discrete time, the output sequence \( y[n] \) of a linear time invariant system, with impulse response \( h[n] \) due to any input sequence \( x[n] \) is the convolution sum of \( x[n] \) with \( h[n] \) and is given as

\[
y[n] = x[n] * h[n] = \sum_{\infty} x[k] h[n-k]
\]

(7)

\( h[n] \) is the response of the system to impulse sequence, \( \delta[n] \).

To implement discrete time convolution, the two sequence \( x[k] \) and \( h[n-k] \) are multiplied together for \( -\infty < k < \infty \) and the products are summed to compute output samples of \( y[n] \). Convolution sum serves as an explicit realization of a discrete-time linear system. The above equation expresses each sample of output sequence in terms of all samples of input and impulse response sequence.
Let the length of input and impulse sequences, $x[n]$ and $h[n]$ be $L$ and $M$. The starting time of input and impulse sequences are represented by $n_1$ and $n_2$ respectively.

Therefore, the length $N$, of output sequence $y[n]= L + M - 1$ and the starting time $n = n_1 + n_2$

The samples of output sequence is computed using convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]$$

(8)

4. LINEAR CONVOLUTION OF LONG DURATION SEQUENCE

In real time signal processing applications concerned with signal monitoring and analysis linear filtering of signals is involved. The input sequence $x(n)$ is often a very long sequence[5].

Practically, it is difficult to store a long duration input sequence. So, in order to perform linear convolution of such a long duration input sequence with the impulse response of a system, the input sequence is divided into blocks. The successive blocks are processed one at a time and the results are combined to obtain the output sequence. The blocks are filtered separately and results are combined using overlap save method or overlap adds method [6].

Linear filtering performed via the DFT involves operations on a block of data, which by necessity must be limited in size due to limited memory of digital computers. A long input signal sequence must be segmented to fixed-size blocks prior to processing.

4.1 Overlap-Save Method

Let the length of long duration input sequence be $L_4$. The length of impulse response = $M$

The input sequence is divided into blocks of data. The length of each block is $N= L+M-1$

Each block consists of last $(M-1)$ data points of previous block followed by $L$ new data points for first block of data. The first $(M-1)$ points are set to zero.

Therefore blocks of input sequence are

$$x_1(n)= \{0,0,\ldots,0, x(0), x(n),\ldots x(L-1)\}$$

The first $(M-1)$ samples are zeros.

$$x_2(n)= \{x(L-M+1), x(L-1), x(L), x(2L-1)\}$$

$x(L-M+1), x(L-1)$ are the last $(M-1)$ samples and from $x_1(n)$ and $x(L), x(2L-1)$ are L new samples

$$x_3(n)= \{x(2L-M+1), x(2L-1), x(2L), x(3L-1)\}$$

$x(2L-M+1), \ldots, x(2L-1)$ are the last $(M-1)$ samples from $x_2(n)$

$x(2L), \ldots, x(3L-1)$ are the L new samples

The length of impulse response is increased by appending L-1 zeros

Circular convolution of $x_i(n)$ and $h(n)$ is computed for each block, which leads to blocks of output sequences $y_i(n)$

Because of aliasing the first $(M-1)$ samples of each output sequence $y_i(n)$ is discarded.

The final output sequence after discarding first $(M-1)$ samples of each output sequence $y_i(n)$ consists of samples of all blocks arranged in sequential order.

4.2. Overlap-Add Method

In this method also the long direction input sequence is divided into blocks of data.

The length of each block is $L+M-1$

The first L samples are new samples taken from long duration input sequence and the last $(M-1)$ samples are zero appended to have total length of samples as $L+M-1$

The data blocks are represented as

$$x_1(n)= \{x(0), x(1),\ldots x(L-1), 0,0,\ldots\}$$

$$x_2(n)= \{x(L), x(L+1),\ldots x(2L-1), 0,0,\ldots\}$$

$$x_3(n)= \{x(2L), x(2L+1),\ldots x(3L-1), 0,0,\ldots\}$$

The last $(M-1)$ samples in each sequence are zeros appended to have total length as $L+M-1$
Similarly the length of impulse response is increased to L+M-1 by appending L-1 zeros to it.

Circular convolution is performed on each block of input sequence with the impulse response to have blocks of output sequences.

The last M-1 samples of each block of output sequence is overlapped and added to the first M-1 samples of succeeding block. The samples thus obtained are arranged in sequential order to have the final output sequence y(n).

So this method is called as Overlap-Add method.

5. MULTIPLICATION TECHNIQUE

Jagadguru Swami Sri Bharati-Krisna Swamiji introduced his research on mathematics based on sixteen sutras for multiplication. A multiplier is the key block in Digital Signal processing. In the increasing technology, researchers are trying to design multipliers which offer high computational speed, less delay, low power and area efficient arithmetic building blocks [7].

In Linear Convolution, the multiplication is performed using Urdhva-Tiryagbyham Sutra of Vedic mathematics[8]. The Comparison between number of multiplications and additions in Conventional Mathematical approach and vedic mathematics is shown. [9]

<table>
<thead>
<tr>
<th>Normal multiplier</th>
<th>Vedic multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>For 2 bit multiplication</td>
<td>For 2 bit multiplication</td>
</tr>
<tr>
<td>No. of multiplications : 4</td>
<td>No. of multiplications : 4</td>
</tr>
<tr>
<td>No. of additions :2</td>
<td>No. of additions :1</td>
</tr>
<tr>
<td>For 3 bit multiplication</td>
<td>For 3 bit multiplication</td>
</tr>
<tr>
<td>No. of multiplications : 9</td>
<td>No. of multiplications : 9</td>
</tr>
<tr>
<td>No. of additions :7</td>
<td>No. of additions :5</td>
</tr>
<tr>
<td>For 4 bit multiplication</td>
<td>For 4 bit multiplication</td>
</tr>
<tr>
<td>No. of multiplications :16</td>
<td>No. of multiplications :16</td>
</tr>
<tr>
<td>No. of additions :15</td>
<td>No. of additions :9</td>
</tr>
<tr>
<td>For 8 bit multiplication</td>
<td>For 8 bit multiplication</td>
</tr>
<tr>
<td>No. of multiplications :64</td>
<td>No. of multiplications :64</td>
</tr>
<tr>
<td>No. of additions :77</td>
<td>No. of additions :53</td>
</tr>
</tbody>
</table>

Example

Multiplication of 1234 and 2116

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array}
\]

\[
\begin{array}{cccc}
2 & 1 & 1 & 6 \\
\end{array}
\]

Adder

\[
\begin{array}{cccc}
2 & 5 & 9 & 9 \\
0 & 0 & 1 & 1 \\
2 & 6 & 1 & 1 \\
\end{array}
\]

Step 1:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 1 & 1 & 6 \\
\end{array}
\]

4x6=24. 2, the carry is placed below the second digit.

Step 2:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 1 & 1 & 6 \\
\end{array}
\]

(3x6) + (4x1) = 22. 2, the carry is placed below the third digit.

Step 3:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 1 & 1 & 6 \\
\end{array}
\]

(2x6) + (4x1) + (3x1) = 19. 1, the carry is placed below the fourth digit.

Step 4:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 1 & 1 & 6 \\
\end{array}
\]

(1x6) + (2x4) + (2x1) + (3x1) = 19. The carry 1 is placed below the fifth digit.

Step 5:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 1 & 1 & 6 \\
\end{array}
\]
(1x1) + (3x2) + (2x1) = 9. The carry 0 is placed below the sixth digit.

Step6:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 1 & 1 & 6 \\
\end{array}
\]

(1x1) + (2x2) = 5. The carry 0 is placed below seventh digit.

Step7:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\downarrow & & & \\
2 & 1 & 1 & 6 \\
\end{array}
\]

(1x 2)=2.

6. SIMULATION RESULTS

6.1. Circular Convolution

Here input sequence is \( a(n) = [a_3, a_2, a_1, a_0] \)

Impulse sequence is \( b(n) = [b_3, b_2, b_1, b_0] \)

In this each value is of 4 bit length.

The given inputs are \( a(n) = [1, 2, 3, 4] \)

Impulse sequence is \( b(n) = [2, 3, 4, 5] \)

Output in hexadecimal format is (Each of 8 bit length)

\[ Y(n) = [8'\text{h} 06, 8'\text{h}06, 8'\text{h}04, 8'\text{h}05] \]

6.2. Linear Convolution for Short Duration Sequence

Fig. 3 Linear convolution for short duration sequence

Here input sequence is \( x(n) = [x_3, x_2, x_1, x_0] \)

Impulse sequence is \( h(n) = [h_3, h_2, h_1, h_0] \)

In this each value is of 4 bit length.

The given inputs are \( x(n) = [1, 2, 3, 4] \)

Impulse sequence is \( h(n) = [2, 3, 4, 5] \)

Output in hexadecimal format is (Each of 8 bit length)

\[ Y(n) = [8'\text{h} 14, 8'\text{h}1f, 8'\text{h}22, 8'\text{h}1e, 8'\text{h}10, 8'\text{h}07, 8'\text{h}02] \]

6.3. Linear Convolution for Long Duration Sequence

Overlap-Add Method

Fig. 4 Linear convolution for long duration sequence Overlap-Add method

Here sequence is \( a(n) = [a_3, a_2, a_1, a_0] \)

Impulse sequence is \( b(n) = [b_3, b_2, b_1, b_0] \)

In this each value is of 4 bit length.

The given inputs are \( a(n) = [1, 2, 3, 4] \)

Impulse sequence is \( b(n) = [1, 1, 0, 0] \)

Output in hexadecimal format is (Each of 8 bit length)

\[ Y(n) = [8'\text{h} 06, 8'\text{h}06, 8'\text{h}04, 8'\text{h}05] \]
Here, convolution is applied between sequences of lengths 12 and 2 respectively.

Input sequence \( x(n) = [i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11}] \)

Impulse sequence \( h(n) = [h_0, h_1] \)

Convolved sequence \( y(n) = [g_0, g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}, g_{11}, g_{12}] \)

The example taken here is

\( x(n) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] \) (Decimal format)

\( h(n) = [4, 5] \) (Decimal format)

\( y(n) = [8'h3C, 8'h67, 8'h5E, 8'h55, 8'h4C, 8'h43, 8'h3A, 8'h31, 8'h28, 8'h1F, 8'h16, 8'h0D, 8'h04] \)

6.4. Linear Convolution for Long Duration Sequence

**Overlap-Save Method**

The example taken here is

\( x(n) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] \) (Decimal format)

\( h(n) = [4, 5] \) (Decimal format)

\( y(n) = [8'h3C, 8'h67, 8'h5E, 8'h55, 8'h4C, 8'h43, 8'h3A, 8'h31, 8'h28, 8'h1F, 8'h16, 8'h0D, 8'h04] \)

**CONCLUSIONS**

Circular and Linear convolution of discrete finite and infinite length sequences are performed using carry save multiplier based on Vedic multiplication. The multiplier proposed in this paper, using Vedic mathematics results in high computation speed and minimum critical path, which results in less delay, when compared to normal multiplier. Speed can be further optimized by high performance adders.

**REFERENCES**


BIOGRAPHIES:

Ms. M. Bharathi, M.tech, is currently working as an Assistant Professor in ECE department of Sree Vidyanikethan Engineering College, Tirupati. She has completed M.tech in VLSI Design, in Satyabhama University. Her research areas are Digital System Design, VLSI Signal Processing.

Ms. D. Leela Rani received the M.Tech. Degree from Sri Venkateswara University, Tirupati. She is currently working towards the Ph.D. degree in the Department of Electronics and communication Engineering, SVU College of Engineering, Tirupati. Currently she is working as an Associate professor in Sree Vidyanikethan Engineering College (Autonomous). Her research areas include, Atmospheric Radar Signal Processing and VLSI Signal Processing.