A 32-Point Fft using Vedic Mathematics

Abstract:- This paper “FFT using Vedic mathematics” proposes a novel technique in order to improve speed and reduce the time delay in FFT Multiplication process. Fast Fourier Transform is widely used in various fields like image and signal processing, Voice recognition systems etc. Vedic mathematic comprises of 16 sutras in that one of sutras namely Urdhwa Tiryakbhyam sutra used here to improve the performance of FFT multiplier. Here, the proposed method “FFT using Vedic mathematics” compared with the one of the popular multiplier namely modified Radix-2 Booth multiplier in terms of delay which shows better result.

1. INTRODUCTION

FFT is a highly efficient procedure for computing the DFT of finite series and requires less number of computation than that of direct evaluation of the DFT. FFT reduces the computations that of direct the calculation of the coefficients of the DFT can be carried out iteratively. FFT reduces the computation time required to compute a Discrete Fourier Transform and improve the performance by a factor 100 or more over direct evaluation of the DFT. The complex multiplication in FFT algorithm is performed based on the Array and modified Booth’s algorithm. These algorithms are quite popular in modern VLSI design. A new approach to multiplier design based on ancient Vedic Mathematics is used to improve the speed of multiplication. Urdhwa Tiryakbhym sutra in Vedic mathematics explores a novel method for implementation of 32-point FFT.

2. FFT ALGORITHM

The Fast Fourier Transform is a highly efficient procedure for computing the Discrete Fourier Transform (DFT) of a finite series and requires less number of computations than that of direct evolution of DFT. The FFT is based on computation on decomposition and breaking the transform into smaller transforms and combing them to get total transform. The DFT of a sequence can be evaluated using the formula

\[ X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} \]

Here FFT multiplication was implemented based on Radix-2 algorithm where bits are taken as input and we get two output bits dependence upon the arrow operations. The basic block diagram of DIF-FFT was shown in Figure.2.1.

3. 32-POINT FFT

In 32-point FFT, there are five stages, which contain unique butterfly structure. Input is taken in normal order at first stage and the inputs for other stages dependence upon output generated by preceding stage butterfly structure. The output is taken in bit-reversal order at last stage. The block diagram for 32-point FFT is shown in Figure.2.2.
4. 32-POINT FFT BUTTERFLY STRUCTURE:

The butterfly structure of 32-point FFT is shown in Figure 2.3. For 32-point input, we require five butterfly stages to get the required output. In each stage of butterfly structure, we need to operations as shown in the Figure 2.1.

![Figure 2.3: 32-point FFT butterfly structure](image)

5. MODIFIED RADIX-2 BOOTH ALGORITHM

Booth's multiplication algorithm is an algorithm which multiplies 2 signed or unsigned integers in 2's complement form. This approach uses fewer additions and subtractions than more straight forward algorithms.

Procedure to multiplication can be stated as follows: Load multiplicand and multiplier in the registers B and Q. We also need third register A, which is initialize to 0(zero). Set sequence counter value equal to number of bits in the register. If the multiplier Qn,Qn+1 digits are either 00 or 11, then simply decrement sequence counter (sc). Then all bits of the A, Q are shifted to the right one bit. If the multiplier Qn,Qn+1 digits are 10, then subtract multiplicand to A and decrement sequence counter (sc). Then all bits of the A, Q are shifted to the right one bit. If the multiplier Qn,Qn+1 digits are 01, then add multiplicand to A and decrement sequence counter (sc). For every operation compare sequence counter (sc) value with ‘0’. if it is not equal to zero then continue the process. If it is equal to zero then end it result is in register AQ.

The flow chat for booth multiplier is shown in Figure 3.1.

![Figure 3.1: flow chat for radix-2 booth multiplier](image)

6. PROPOSED VEDIC MATHEMATICS

Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. The proposed Vedic multiplier is based on the Vedic multiplication formulae (Sutras). These Sutras have been traditionally used for the multiplication of two numbers in the decimal number system. The multiplier is based on an algorithm Urdhva Tiryakhbyam (Vertical & Crosswise) of ancient Indian Vedic Mathematics. Urdhva Tiryakhbyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means “Vertically and crosswise”. It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products. The parallelism in generation of partial products and their summation is obtained using Urdhava Tiryakhbyam explained in figure below. The algorithm can be generalized for $n \times n$ bit number. The proposed $8 \times 8$ Vedic multiplier produces was explained in Figure 4.1.
Figure 4.1: Multiplication Process in Urdhva Tiryakbhyam

Equations obtained in Vedic multiplier based on UrdhavaTiryakbhyam sutra are

P0 = A0 * B0

C1P1 = (A1 * B0) + (A0 * B1)


C7C6P4 = (A4 * B0) + (A3 * B1) + (A2 * B2) + (A1 * B3) + (A0 * B4) + C3 + C4


C25C24C23P10 = (A7 * B3) + (A6 * B4) + (A5 * B5) + (A4 * B6) + (A3 * B7) + C16 + C18 + C20

C27C26P11 = (A7 * B4) + (A6 * B5) + (A5 * B6) + (A4 * B7) + C19 + C21 + C23

C29C28P12 = (A7 * B5) + (A5 * B6) + (A5 * B7) + C22 + C24 + C26

C30P13=(A7 * B6) + (A6 * B7) + C25 + C27 + C28


7. RESULTS & CONCLUSION

Here the proposed method “FFT using Vedic mathematics” is compared with existing multipliers in terms of delay with different families of Xilinx 10.1 shows that the proposed method has a better result shown in below table.

<table>
<thead>
<tr>
<th>Families</th>
<th>Radix-2 modified booth algorithm</th>
<th>Proposed Urdha tiryabhyam</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Spartan-3</td>
<td>26.877</td>
<td>24.865</td>
</tr>
<tr>
<td>On Spartan-3A &amp; Spartan-3AN</td>
<td>24.089</td>
<td>20.360</td>
</tr>
<tr>
<td>On Spartan-3E</td>
<td>23.703</td>
<td>19.333</td>
</tr>
<tr>
<td>On Virtex-2</td>
<td>17.962</td>
<td>16.213</td>
</tr>
<tr>
<td>On Virtex-4</td>
<td>15.255</td>
<td>13.938</td>
</tr>
<tr>
<td>On Virtex-5</td>
<td>12.295</td>
<td>8.879</td>
</tr>
</tbody>
</table>

REFERENCES


[7] “Efficient implementation of 16-bit Multiplier-accumulator using radix-2 modified booth algorithm and spst adder using verilog” by addanki puma ramesh, Dr.a.v.n.tilak and Dr.a.m.prasad, international journal of vlsi design & communication systems (vlsics) vol.3, no.3, June 2012.
