Efficient Implementation & Comparison of Signed Complex Multiplier on FPGA using FFT Algorithm

Archana Fande\(^1\), Anil Sahu\(^2\)

\(^1\)ME IV Sem, Department of ET&T, FET, SSTC, Bhilai
\(^2\)Asst. Prof. Department of ET&T, FET, SSTC, Bhilai

ABSTRACT
Various applications based on Fast Fourier Transform (FFT) such as signal and image processing require a high computational power, plus the ability to choose the algorithm and architecture to implement it. This paper is devoted for the design of a low power complex multiplier design to reduce the hardware required to implement the FFT algorithms. This complex multiplier can also be used for any radix-N algorithms.

The aim is to design and implement a complex multiplier for Radix-4 FFT, which offer high speed, low power consumption and lesser area. Thus making them suitable for various high speeds, low power compact VLSI implementations. These three parameters i.e. power, area and speed are always traded off. The maximum path delay to implement radix-4 butterfly is 11.656ns

Keywords - Verilog HDL, FFT, DFT, Butterfly unit

I. INTRODUCTION
The fast Fourier transform (FFT) and discrete Fourier transform (DFT) has been widely applied in the analysis and implementation of communication systems such as OFDM-based wireless local area network (WLAN) [2]. The Fast Fourier Transform (FFT) has become almost ubiquitous in high speed signal processing. Using this transform, signals can be moved to the frequency domain where filtering and correlation can be performed with fewer operations. This paper presents complex multiplier architecture for radix-4 FFT architecture for use on field programmable gate arrays (FPGAs). A fully parallel hardware implementation of the FFT can be quickly derived from the data flow graph of the algorithm. If only one data sample is ready at a time only one butterfly per stage can be active at a time. Taking this input pattern into account, the entire column of butterflies is collapsed into a single butterfly [1].

The FFT algorithm eliminates the redundant calculation which is needed in computing Discrete Fourier Transform (DFT) and is thus very suitable for efficient hardware implementation. In addition to computing efficient DFT, the FFT also finds applications in linear filtering, digital spectral analysis and correlation analysis, Ultra Wide Band (UWB) applications, etc. [7].

Multiplication is one of the basic functions used in digital signal processing (DSP). It requires more hardware resources and processing time than addition and subtraction [18]. This paper presents complex multiplier architecture for radix-4 FFT algorithm to reduce the execution time and hardware.

The most commonly used FFT is the Cooley–Tukey algorithm. This is a divide and conquer algorithm that recursively breaks down a DFT of any composite size \(N = N_1N_2\) into many smaller DFTs of sizes \(N_1\) and \(N_2\), along with \(O(N)\) multiplications by complex roots of unity traditionally called twiddle factors. The radix-4 butterfly divide into two parts first part consist of a complex multiplier to multiply a twiddle factor with 8 bit input data and second part consist of addition/subtraction module to add/sub 16 bit data to obtain 8 output result of radix-4 BF in real and imaginary form. The proposed pipeline architecture consist of two multiplier and two add/sub module on other hand the conventional complex multiplier of radix-4 consist of four complex multiplication and three add/sub operation so this is very useful to minimize hardware. So trade-off between the reduction in delay due to pipelining and increase in area must always be considered.

II. RADIX-4 ARCHITECTURE
The butterfly of a radix-4 algorithm consists of four inputs and four outputs. The FFT length is 4M, where \(M\) is the number of stages. A stage is half of radix-2. The radix-4 DIT FFT divides an \(N\)-point discrete Fourier transform (DFT) into four \(N/4\) -point DFTs, then into 16 \(N/16\) -point DFTs, and so on. In the radix-2 DIT FFT, the DFT equation is expressed as the sum of two calculations. One calculation sum for the first half and one calculation sum for the second half of the input sequence. The radix-4 butterfly is depicted in Figure 1(a) below and in a more compact form in Figure 1(b). We also have to note that \(W_0N = 1\), which will give us three complex multiplications and 12
complex additions per Radix-4 butterfly. As the Radix-
4 algorithm consists of v steps \((\log(N)/\log(4))\) where 
each step involves \(N/4\) number of butterflies we will 
get \(3*v*N/4 = (3N/8)\log2N\) number of complex 
multiplications and \((3N/2)\log2N\) complex additions. If 
compared with the computational power used by the 
Radix-2 algorithm, it will find that a computer gain of 
25\% regarding the complex multiplications, but that 
the number of complex additions increases by 50\% 
[12].

Similarly, the radix-4 DIT fast Fourier transform 
(FFT) expresses the DFT equation as four summations, 
and then divides it into four equations, each of which 
computes every fourth output sample. The following 
equations illustrate radix-4 decimation in time.

\[
X(p,q) = F(l,q)
\]

\[
F(l,q) = \begin{cases} 
1 & l=0,1,2,3; \quad p=0,1,2,3; \quad q=0,1,2,\ldots, \\
-1 & \text{otherwise} 
\end{cases}
\]

And

\[
X(l,m) = x(4m+1)
\]

\[
X(p,q) = x ( )
\]

Thus the four \(N/4\)-point DFTs \(F(l, q)\) obtained from 
the above equation are combined to yield the \(N\)-point 
DFT. The expression for combining the \(N/4\)-point 
DFTs defines a radix-4 decimation-in-time butterfly, 
which can be expressed in matrix form as

\[
\begin{pmatrix}
X(0,q) \\
X(1,q) \\
X(2,q) \\
X(3,q)
\end{pmatrix} = 
\begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & -j & -1 & j \\
1 & 1 & 1 & 1 \\
1 & -j & -1 & j
\end{pmatrix}
\begin{pmatrix}
w^{2^0}_q F(0,q) \\
w^{2^1}_q F(1,q) \\
w^{2^2}_q F(2,q) \\
w^{2^3}_q F(3,q)
\end{pmatrix}
\]

Where \(x(n)\) is the time-domain discrete input signal 
and \(X(k)\) is the DFT. \(n\) represents the discrete time-
domain index, while \(k\) is the normalized frequency-
domain index.

### III. COMPLEX MULTIPLIER ARCHITECTURE

According to equation of radix-4 a complex multiplier 
is needed to multiply the twiddle factor. For 
accomplished this operation its required four 
multipliers, one adder, one subtractor.

This is given by the,

\[(a_+jai) \cdot (br + jbi) = (ar \cdot br - ai \cdot bi) + j(ai \cdot br + ar \cdot bi)\]

The implementation of complex multiplier for radix_4 
FFT for fast implementation of radix_4 FFT algorithm 
by modifying the above algorithm can be as follows,

- Let \(F_1 = br \cdot (ar + ai)\)
- \(F_2 = ar \cdot (bi - br)\)
- \(F_3 = ai \cdot (br + bi)\)

Real part = \(F_1 - F_3\)

Imaginary part = \(F_1 + F_2\)

To implement the above operation one extra data 
selector is needed to select the operation and to control 
the data flow.

Using the above approach the number of complex 
multiplier and adder required to implement the 
complex operation is reduced to \(\log N\). This is as 
follows:

<table>
<thead>
<tr>
<th>(N)</th>
<th>Radix-4</th>
<th>Project approach</th>
<th>Radix-4</th>
<th>Project approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>20</td>
<td>16</td>
<td>148</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>208</td>
<td>64</td>
<td>976</td>
<td>128</td>
</tr>
</tbody>
</table>

www.ijsret.org
The radix-4 butterfly operation completed in two module. The first module consist of advanced complex multiplier and the second module consist of add/sub module. At first the input real and imaginary data is added by adder and then multiplied by twiddle factor as shown in figure 3 parallely.

The real and imaginary data of twiddle factor is add or subtract according to the control signal add of data selector and then multiplied it by either real input data or imaginary input data as shown in above block dig. These two output are then fed back to another add/sub module by using control signal sub which is generated by inverting add control signal.

IV. ADDER & SUBTRACTOR MODULE

The output data is in the form of real and imaginary form which is separated by data distributor by using same control signal add. This data is then added or subtracted by using another control signal c1 and c2. Control signal c1 is used to add or sub the real data and c2 is used to add or sub the imaginary data. Out signal is the x-oring of c1 and c2 which is used to add or sub the real-real and img-img data to gate separated real and imaginary data.

V. IMPLEMENTATION

The module is implemented on Spartan 3 and on Spartan 6. The RTL module & test bench waveforms are as follows

VI. RESULT & ANALYSIS

For the paper data, implementation is done on Spartan 6 Trainer kit and compared it with Spartan 3 by using Verilog HDL on xilinx13.2 Project Navigator (ISE) for FPGA (support provided: HoD, Department of ET&T, SSITM, SSTC, Bhilai). As the Radix-4 FFT algorithm utilizes less complex multipliers than the Radix-2 FFT algorithm, the Radix-4 algorithm is preferable for hardware implementation.

Figure 4 Advance complex multiplier
Figure 5 Test Bench Waveform for Complex multiplier
Figure 6 Test Bench Waveform for Radix-4
Figure 7 Pin diagram for complex multiplier
The maximum path delay obtained is 11.656 ns to implement the complex multiplication in radix_4 FFT algorithm.

VII. CONCLUSION

As the Radix-4 FFT algorithm utilizes less complex multipliers than the Radix-2 FFT algorithm, the Radix-4 algorithm is preferable for hardware implementation. A parallel programming approach seems to be the model when a real-time system with high sampling rate is desired. This approach reduced complex multiplier to log N and complex adder to log N.

REFERENCES


[17] FPGA implementation of FFT for real time signal & image processing by I.S.Uzun, A. Amira & A.Bouridone, Ieee proceedings on vision, image & signal processing volume152no:3may2005