A Survey on CORDIC Based FFT Implementation on FPGA

Jaimini Bhatt
PG Scholar
KITRC, Kalol, Gujarat, India.

Abstract— In this survey paper, a detailed study on architectures of FFT processor and decomposition methods of FFT is carried on. Also, how the calculation is made simplified using optimization techniques is illustrated. It is described in detail how calculation using CORDIC algorithm is done. Some techniques to improve results are also mentioned.

I. INTRODUCTION
A fast Fourier transform (FFT) is an algorithm to compute the discrete Fourier transform (DFT) and its inverse. Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. The Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. Due to their programmable nature, FPGAs are an ideal fit for many different markets. Due to flexible nature it is best suitable for FFT implementation. CORDIC is a class of shift-add algorithms for rotating vectors in a plane. It is a mechanism for computing the magnitude and phase angle of an input vector thus it is used to compute twiddle factor.

\[ X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \quad (1.1) \]

The Discrete Fourier Transform (DFT) is one of the most important tools used in Digital Signal Processing applications. It has been widely implemented in digital communication systems such as Radars, Ultra Wide Band (UWB) receivers and many other applications. Computing this operation has a high computational requirement and needs a large number of operations (N² complex multiplications and N (N -1) complex additions). This makes computing and implementation very difficult to realize. To reduce the number of operations a fast algorithm has been introduced by Cooley-Tukey [1] and called Fast Fourier Transform (FFT). The latter, reduces complexity from O (N²) to O (N log N).

II. ARCHITECTURE OF FFT PROCESSOR
A. General Architecture
The whole FFT processor architecture consists of butterfly unit, data storage unit and address generator unit. The butterfly operation is the heart of the FFT algorithm, which can influence system speed, power consumption and cost [1].

B. Digit Slicing Architecture
The concept behind the digit-slicing architecture is any binary number that can be sliced into a few blocks of shorter binary numbers, with each block carrying a different weight. In this study, the fixed-point 2’s complements arithmetic has been chosen to represent the input data, which are signed numbers with absolute value less than one.

Fig. 1: Butterfly to compute DFT.

Fig. 2: General Architecture of FFT Processor.

Fig. 3: Digit Slicing Architecture for FFT Processor

On-chip implementation of pipeline digit-slicing multiplier-less butterfly for FFT structure is an enabler in solving problems that affect communications capability in FFT and these algorithms.

There is a growing interest in Field Programmable Gate Arrays (FPGAs) because of their potential to substantially accelerate computational intensive algorithms such as FFTs. Unfortunately, high order FFT are almost implemented into high cost FPGAs. For example, it is not possible to instantiate higher point FFT with the Xilinx IP core to implement it in Spartan 3 family. To meet with this challenge, VLSI architecture is proposed to allow the implementation of high order FFT into low cost FPGAs.
possesses huge potentials for future related works and research areas [2]. Structures of Complex Adders and Multipliers are shown.

![Fig. 4: Structure of Complex Adder.](image45x76)

![Fig. 5: Structure of Complex Multiplier.](image60x633 to 274x746)

### III. DECOMPOSITION OF FFT

Decomposing a high length FFT to 8-point FFTs may be done in a spatial or in temporal distribution.

#### A. Spatial Distribution

It can be observed that computing 64-point FFT is composed on five levels. The first level is composed of two serial to parallel blocks used to store real and imaginary part of data presented in a serial way. The second floor is composed of 8 blocks of 8-point FFT Split Radix DIT. The third block contains 49 complex multipliers used to compute non-trivial complex multiplication. The fourth is similar to the second one. The last level is composed of two parallel to serial blocks gives data in a serial way.

The main advantage of this architecture is the high speed and low-latency. However, the implementation of this architecture on FPGA needs high memory, high number of complex multipliers and complex adders. Therefore, this architecture is not suitable for low cost FPGA such as Spartan 3 family.

#### B. Temporal Distribution

It is another possible realization of the 64-point FFT. According to this structure, the first stage is realized by one block of 8-point FFT rather than 8. Similarly, the third stage is performed by only one block of 8-point FFT rather than 8. Consequently, the control unit plays an important role to synchronize all the treatments. This architecture performs FFT in a pipeline way [3].

### IV. OPTIMIZATIONS IN FFT

#### A. Optimization of Complex Multiplier

Traditional complex multiplier has been obtained as follows.

\[
(a+jb)(c+jd)=ac-bd+j(ad+bc)
\]  \hspace{1cm} (4.1)

Where, \(a, b, c\) and \(d\) are four independent real numbers, four multipliers and two adders are used. After doing some optimization deformation, (4.1) can be moved to (4.2) as follows,

\[
(a+jb)(c+jd)=ac-bd+j(ad+bc)
\]  \hspace{1cm} (4.2)

Three adders are added, however one multiplex is reduced.

#### B. Optimization of Twiddle Factor

Due to the symmetric properties of the twiddle factor, \(W_8^1\) , \(W_8^3\) and \(W_8^4\) can be written as \((\sqrt{2}/2)(1+j)\) and \(-j\) respectively, \(W_8^1\), \(W_8^3\) and \(W_8^4\) are referred to as specific twiddle factor.

Others are referred to as common twiddle factor. Here, the complex multiplication can be computed as follows.

\[
(\sqrt{2}/2)(1+j)(a+jb) = (\sqrt{2}/2)(b-a)+j(\sqrt{2}/2)(a-b)
\]  \hspace{1cm} (4.3)

\[
(\sqrt{2}/2)(1+j)(a+jb) = (\sqrt{2}/2)(b-a) - j(\sqrt{2}/2)(a+b)
\]  \hspace{1cm} (4.4)

\[
(a+jb) (-j) = b - ja
\]  \hspace{1cm} (4.5)

Compare (4.3), (4.4) with (4.1), two multipliers are reduced. Compare (4.5) with (4.1), it does not require any multiplier to construct complex multiplication [4].

### V. PIPELINED ARCHITECTURE

Limited by available resources on FPGAs, only a single butterfly unit is used in our FFT processor. To fully pipeline the butterfly unit, two read and two write operations of complex are needed in each cycle. Usually just dual-port block RAMs are provided on FPGAs, and only two operations can be performed in a cycle [5].

[5] Designed and reorganized the RAM access, and use an extra RAM bank to implement two read and two write operations of complex in a single cycle. 32-bit IEEE 754 single precision floating-point FFT processor was designed.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Utilization of traditional FFT</th>
<th>Utilization of this FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice</td>
<td>2478 Out of 5120</td>
<td>2452 Out of 5120</td>
</tr>
<tr>
<td>MULT18x18</td>
<td>16 Out of 40</td>
<td>16 Out of 40</td>
</tr>
<tr>
<td>Block RAM</td>
<td>14 Out of 40</td>
<td>18 Out of 40</td>
</tr>
</tbody>
</table>

Table 1: Comparison of Resource Utilization

<table>
<thead>
<tr>
<th>Resource</th>
<th>Traditional FFT</th>
<th>Our FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>(\geq 100\text{MHz})</td>
<td>(\geq 150\text{MHz})</td>
</tr>
<tr>
<td>Number of cycles</td>
<td>10240</td>
<td>5220</td>
</tr>
<tr>
<td>Processing time</td>
<td>102.4 us</td>
<td>34.8 us</td>
</tr>
</tbody>
</table>

Table 2: Comparison of Performance

NEDA is one of the techniques to implement many digital signal processing systems that require multiply and accumulate units. Distributed Arithmetic (DA) has become an efficient tool to implement multiply and accumulate (MAC) unit in many digital signal processing (DSP) systems. It eliminates the need of a multiplier that is used as a part of MAC unit. DA implements MAC unit by pre-computing all possible products and by storing them using a read only memory (ROM). Usage of ROM can be eliminated if one set of the inputs has a fixed value. This is done by distributing the coefficients to the inputs of the unit. This approach is called New Distributed Arithmetic (NEDA). Thus, using NEDA, any MAC like unit can be implemented just by using adders and shifters [6]. Noise reduction is done using wiener filter [7].
VII. PROPOSED WORK

After surveying, one can implement higher point FFT of real type data by defining one basic FFT structure and making use of it in form of parallel architecture and can compare various parameters such as power consumption, area utilization, latency, processing speed. Also, one can implement CORDIC into VHDL code to make the design more efficient. Real time application can be developed. I have developed basic 8 point FFT on SPARTAN 3 E kit. Issues are accuracy, compatibility for real time data. It can be solved by using Floating point IP core.

REFERENCES


