Design of a High Speed FFT Processor
Belma Joseph¹, Dr. S. Jayanthy²

¹ME VLSI Design student, Sri Ramakrishna Engineering College, Coimbatore, India
²Professor ECE, Sri Ramakrishna Engineering College, Coimbatore, India

Abstract—The Discrete Fourier Transform (DFT) is used to transform the samples in time domain into frequency domain coefficients. The Fast Fourier Transform (FFT) is a widely used algorithm that computes the Discrete Fourier Transform (DFT) using much less operations than a direct realization of the DFT. FFTs is of great significance to a wide variety of applications such as data compression, spectral analysis etc.

This paper proposes a design that implements a Fast Fourier transform (FFT). The module is developing by Radix-2, Radix-4 decimation in time algorithm structure. The operation of the FFT processor performs three main processes i.e. data load, compute and result unload. The processing cycle starts with the data load process. In this process sampled data is read in and stored in memory. During the compute process computation of FFT on the stored data is performed and result unloaded process makes the FFT results available at its output. This paper also compares the performance of Radix-2 algorithm with Radix-4 algorithm.

Keywords-- DFT, FFT, Processor, Radix-2 algorithm, Radix-4 algorithm

I. INTRODUCTION

The Fourier transform is a mathematical procedure that was discovered by a French mathematician named Jean-Baptiste-Joseph Fourier in the early 1800’s [7]. It has been used very successfully through the years to solve many types of engineering, physics, and mathematics problems. The Fourier transform is defined for continuous (or analog) functions, and is usually applied in situations where the functions are assumed to be continuous. When computing spectra on a computer it is not possible to carry out the integrals involved in the continuous time Fourier transform. In its place a related transform called the discrete Fourier transform is used

The implementation of the Discrete Fourier Transform, or DFT, became practical in 1965 when Cooley and Tukey described an algorithm for computing the DFT very efficiently. Their algorithm (and others like it) has become known as the Fast Fourier Transform (FFT).

Basically, the computation problem for DFT is to compute the sequence \( \{X(k)\} \) of \( N \) complex valued number given another sequence of data \( \{x(n)\} \) of length \( N \) according to the formula

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

which is known as twiddle factor and substitute this in equation (1), then the equation can be simplified as,

\[
X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, 0 \leq k \leq N-1
\]

Direct computation of the DFT on an \( N \)-point complex valued function requires \( N^2 \) operations; where an operation is defined as one multiplication plus an addition [4]. The Cooley- Tukey algorithm takes approximately \( N \log_2 N \) operations; where \( N \) is a power of 2.

Many other methods for efficiently computing the DFT have since been discovered. However, all methods which require on the order of \( N \log N \) operations have become known as FFTs. Hence, in this project the Decimation in time FFT radix2 and Radix-4 architectures are implemented to compute the DFT sequence in processor unit [3].

The remaining part of paper is organized as follows. In Section II, introduces architectural design of the FFT processor. Section III contains simulation results and performance analysis. Section IV concludes this paper.

II. ARCHITECTURAL DESIGN OF FFT PROCESSOR

The FFT processor will calculate a 16-point FFT on incoming data [5]. The complete operation of the FFT processor mainly divided in to three processes. Data load, Compute and Result unload as shown in Fig.1. By data load process the processing cycle will start, where the sampled data is read in and stored in a memory.

Fig.1. FFT Computation Process
In computation process FFT is computed on stored data. And finally in Result unload process the FFT results are available at its output. A block diagram of general structure for FFT processor is shown in Fig.2. It has four blocks.

A. Bit Reversal Unit
B. Butterfly Unit
C. Address generation Unit
D. Control Unit

The input data to the algorithm is N complex value with b bits for real part and b bits for imaginary part. In this work we have considered b=16 as default value. A butterfly unit consists of Radix-2 or Radix-4 butterflies and Rom memory to generate the twiddle factor, the address unit to provide the synchronized addresses to extract the data from Rom and finally control unit.

**B. butterfly unit**

A butterfly unit consists of Radix-2 or Radix-4 butterflies.

1) **radix 2 algorithm:** The Radix-2 algorithms are the simplest algorithm used. In Radix-2 algorithm, split the N-point sequence in to two sequence (odd numbered and even numbered) [3]. This process is carried out until the initial transform is reduced to a set of two-point transforms of the initial data. The DFT of the x(n) is given by,

\[
X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, 0 \leq k \leq N
\]

\[
= \sum_{m=0}^{N/2-1} x(2m) W_N^{2mk} + \sum_{m=0}^{N/2-1} x(2n + 1) W_N^{2k(n+1)}
\]

\[
= \sum_{m=0}^{N/2-1} f1(m) W_{N/2}^{mk} W_N^k + \sum_{m=0}^{N/2-1} f2(m) W_{N/2}^{mk}
\]

But \( W_N^2 = W_{N/2} \)

\[
X(k) = F_1(k) + W_N^k F_2(k)
\]

Since \( F_1(k+N/2) = F_1(k) \) and \( F_2(k+N/2) = F_2(k) \) and \( W_N^{kN/2} = -W_N^k \)

\[
X(k+N/2) = F_1(k) - W_N^k F_2(k)
\]

Where \( k=0, 1\ldots(N/2)-1 \)

The calculation implies for the basic computational element (butterfly) for radix -2 algorithms will be introduced next. Two complex numbers A and B are represented as:

\[
A = x + Xj
\]

\[
B = y + Yj
\]

Then A transform (A') and B transform (B') are calculated as shown

\[
A' = x' + X'j = A + BW_N^k
\]

\[
B' = y' + Y'j = A - BW_N^k
\]

\[
W_N^k = e^{j2nk/N} = \cos(2nk/N) - j \sin(2nk/N)
\]

Then the equation (9) and (10) can be rewritten as

\[
A' = [(x + y\cos(2nk/N) + Y \sin(2nk/N) + j (X + Y \cos(2nk/N) - y\cos(2nk/N))]
\]

\[
B' = [(x - y\cos(2nk/N) - Y\sin(2nk/N)) + j (X - y\cos(2nk/N))]
\]
Where \( k \) depends on the number of stages and number of samples. Implementation of equation (13) and (14) is shown in Fig. 3.

\[
X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} + (-j)^k x(n+N/4) + (-1)^k x(n+N/2) + (j)^k x(n+3N/4) W_N^{kn} --- (17)
\]

This equation not an FFT of N/4 because the twiddle factor depends on N not on N/4. To make it an N/4-point FFT, the sequence X(k) is broken into four sequence for the case where

\[ k = 4r, 4r+1, 4r+2, 4r+3\]

2) radix-4 algorithm: In a Radix 4, each butterfly has four inputs and four outputs instead of two as in the case of Radix-2 FFT. This is also mean that the length of input sequence \( N \) should be a power of 4 [1][2]. To introduce the Radix-4 FFT equation (1) is broken into four summations.

\[
X(4r) = \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + x(n+N/4) + x(n+N/2) + x(n+3N/4) W_N^{kn} --- (18)
\]

\[
X(4r+1) = \sum_{n=0}^{N/4-1} x(n) - j x(n+N/4) + x(n+N/2) + j x(n+3N/4) W_N^{kn} W_N^{nN/4} --- (19)
\]

\[
X(4r+2) = \sum_{n=0}^{N/4-1} x(n) + x(n+N/4) + x(n+N/2) - x(n+3N/4) W_N^{kn} W_N^{3nN/4} --- (20)
\]

\[
X(4r+3) = \sum_{n=0}^{N/4-1} x(n) + j x(n+N/4) - x(n+N/2) - j x(n+3N/4) W_N^{kn} W_N^{nN/4} --- (21)
\]

Fig. 5 shows the corresponding butterfly diagram.

C. address unit and control unit

The purpose of address unit is to provide the ROMs with correct address to access its content. It also keeps track of which butterfly is being computed in each stage. [3] The correct address depends on the mode of operation which is being performed by the processor.
In the input and output process the address goes from 0 to N-1. But in FFT computation address goes from 0 to (N/2)-1 or 0 to N/4-1 depending upon the algorithm [2] [3]. The control unit supervises all the operation of the processor.

III. SIMULATION RESULTS AND PERFORMANCE ANALYSIS

The architecture designed with VHDL is synthesized, placed, and routed by using Xilinx ISE 12.3 software. The width of data in the design is 2bit and 16 bit. Fig.5(a), is the FFT waveform with 2 point input and (b) is for 16 point input. Fig.6 shows the output waveform for 16 point FFT using Radix-4 algorithm.

Fig.5. waveform of the (a) 2 point FFT (b) 16 point input using Radix-2

Fig.6. waveform of 16 point input using Radix-4
Fig.7. Device utilization summary of 16 point Radix-2 FFT

Fig.8. Device utilization summary of 16 point Radix-4 FFT

Fig.7 shows the device utilization summary for Radix-2 and Fig.8 shows for Radix-4. By comparing these two results we can say that the Radix-4 algorithm uses less no of devices than Radix-2 algorithm.

IV. CONCLUSION

This paper presented a high speed FFT processor based on Radix-2 and Radix-4 algorithm. A VHDL based methodology has been used for memory efficient design.

From this project it is found that Radix-4 algorithms will work more efficient than Radix-2. Therefore, implemented architecture has fast computing time and low complexity. So a high speed FFT processor can implement using Radix-4 algorithm.

REFERENCES


