ISSN: 2394-7659
IMPACT FACTOR- 2.785

International Journal of Engineering Researches and Management Studies

FFT IMPLEMENTATION OF HARDWARE

Amandeep Arora[*1], Gaurav Nama[2] and Deepak Birt[3]
[*1,2]Department of Electrical Engineering, IIT Delhi,
[3]Department of Physics, IIT Delhi

ABSTRACT

This paper explains various techniques of hardware implementation of Fast Fourier Transform (FFT). FFT narrows down thousands of computational steps to calculate DFT by simple n*n multiplications to lesser number of steps by exploiting the symmetries in the Fourier terms. Radix-2 and Radix-4 implementations have been explained in detail. Further, various constituents of hardware implementation of FFT are explained. A pipelined FFT architecture for Radix-2 is also mentioned. Other than 2 and 4 radix computations, radix 3 and radix 5 can be also calculated. Mixed radix computations which extend the calculation of FFT for N which are perfect powers of 2 to many possible values of N.

Keywods:- *FFT, Radi-2, Radix-4, Pipelined Architecture.*

I. INTRODUCTION

Discrete Fourier Transform (DFT) is one of the most important tools in wireless and signal processing. The Discrete Fourier Transform is mathematically defined as:

$$Y(k) = \sum_{n=0}^{N} x(n) W_N^{kn}$$

Where x(n) is a sequence of n input data, X(k) is the transformed sequence and $W_N^{kn} = e^{-j2\pi kn/N}$ also known as twiddle factor.
Calculation of DFT using the equation (1) requires n multiplications for a particular k(frequency) and k too varies from 0 to n-1 . Total number of operations would be $O(n^2)$ which is quite large. To reduce the complexity, Fast Fourier Transform (FFT) is used which exploits the symmetries in the calculations. The basic concept behind FFT is that DFT is divided into two (or more) smaller DFTs. These DFTs are computed independently and then combined using twiddle factor. This process is recursively continued until the prime RADIX for N is reached. When N is a power of two, total number of computations is equal to NlogN. The reason for hardware implementation is that by using Multiplier Accumulator chip, we can compute one butterfly (later explained) in hardware .The time required for these computations is lesser than time required for software butterfly computation.

A. The Radix-2 Computation

To compute FFT for more than two-point sequences, N is equally divided into two halves until the basic butterfly (fig) is reached. This can be done both with the time-domain, known as Decimation in Time(DIT) as well as frequency domain, known as Decimation in frequency(DIF).The trick behind this FFT computation is to divide x(n) into an even sequence x(2m) and an odd sequence x(2m+1) and then perform DFT on these sequences separately.

$$Y(k) = \sum_{m=0}^{N/2-1} x(2m) W_{N/2}^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1) W_{N/2}^{(2m+1)k}$$

$$= \sum^{N/2-1} x(2m)W_{N/2}^{2mk} + \overset{kN/2}{\underbrace{W_N}} \sum^{N/2-1} x(2m+1)W_{N/2}^{(2m)k}$$

m=0

So, the total number of multiplications would be 2*(N/2)^2 = N^2/2 in this case. Now, further divide both the even parts and odd parts into new even and odd sequences. Do this until 2 point DFTs are reached. This process is called divide and conquer. When n is a power of 2, number of multiplications would be $n\log_2(n)$. A butterfly diagram is used to compute 2-point DFT.

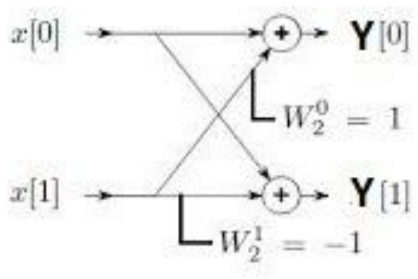

Fig1. Radix -2 Flow Chart

CALCULATIONS IN RADIX-2UNIT

Y(0) = x(0) + x(1) Y(1) = x(0) - x(1)

Each radix-2 butterfly requires 1 complex multiplication.

B. Radix-4 FFT computation

This is similar to Radix-2 computation. In this case N is divided into 4 subsets and each sub-group is transformed separately. The division is done until the basic radix-4 butterfly is reached.

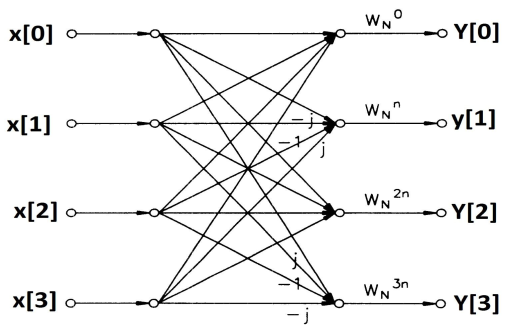

Fig. 2 Radix -4 Flow Graph

IJERMS
International Journal of Engineering Researches and Management Studies

The arithmetic kernel of radix-4 kernel is the butterfly operation (fig.2) defined as:

$$Y(0) = x(0) + x(1) + x(2) + x(3) \quad (2)$$

$$Y(1) = (x(0) - jx(1) - x(2) + jx(3)) * W_N^k \quad (3)$$

$$Y(2) = (x(0) + x(1) + x(2) + x(3)) * W_N^2 k \quad (4)$$

$$Y(3) = (x(0) + x(1) + x(2) + x(3)) * W_N^3 k \quad (5)$$

One such radix-4 butter-fly is capable of doing computations equivalent to 4 radix-2 butterflies. Number of multiplications would be only 3 as compared to 4 in case radix-2 implementation is done.

C. *Mixed Radix Computations*

Other than Radix-2 and radix-4 computations, other radix index are also possible. Radix-3 (DFT is factored into radix 3 units) and Radix-5(DFT is factored in multiples of 5) [3] algorithms are also practiced for some values of N.

It is also possible to have mixed-radix butterflies in which there are different radices of butterflies involved in the computation process. This technique covers a larger range of N and also many possible configurations in which FFT could be calculated. So for example to create a 1200 point FFT, it can be done by $N1200 = 243152$,$N1200 = 524231$ or

$N_{1200} = 2^2 5^1 3^1 2^2 5^2$.All such combinations lead to same FFT.

D. *Hardware Implementation*

The fundamental constituents of hardware implementation of FFT are as follows.

1) *Memory:* In memory, we need to store the input sequence and the twiddle factors. Memory elements are sequential elements. Latches, counters and ROMs are used for this purpose. The input sequence is fed into the input latches and at the correct time decided by counters, this data along with twiddle factors is fed to multiplier accumulator for the butterfly computation.

The results of multiplier accumulator are then stored in the output latches. The butterfly computation needs to be performed for a large number of times depending on the size of the input sequence and radix index used for the computation.

2) *Input Unit:* The input unit is made up of several latches which temporarily stores the input data for processing. The twiddle factors are also stored in latches. In each step of butterfly computation, the input sequence and the twiddle factors from the corresponding latches are fed into corresponding registers of the multiplier accumulator. Theses latches are enabled by EPROMs. The input unit is the foremost step in the hardware implementation of FFT.

3) *Butterfly Computation Unit:* It uses a multiplier accumulator and performs complex multiplication given a radix-2 or radix-4 unit. Flow graph of butterfly operation is already mentioned in section 2 and 3. For radix-2, only one complex multiplication is needed while for radix-4, three complex multiplications are needed.

FFT flow graphs are not directly implemented on hardware because the area usage would be too large. Instead, the FFT implementation is then folded to compress the multipliers in a smaller area. A common folded structure is the pipeline structure is shown in Fig. 3 for a radix 2 FFT.

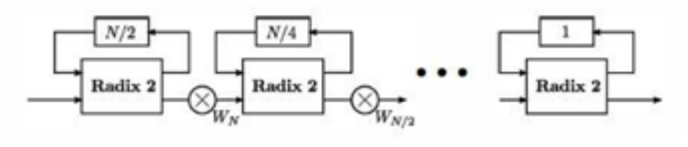

Fig. 3. Pipelined Radix-2 FFT

The first stage registers the incoming data by feeding it into the latches of input unit. Then sign-magnitude conversions are performed in case of signed bits and then fed into the multipliers. The second stage includes butterfly operations. First the data is received from the FIFO queue. The converted input data along with the twiddle factors is fed into the butterfly multipliers. The twiddle factors in the pipelined architecture are different but they are same in case of parallel architecture. To reduce area usage by considerable amount, twiddle factors are made different. After butterfly computations, the output is fed into the output latches.
The third stage involves conversion to floats for complex multiplications since the twiddle factors are complex for radix3 and 4 computations. Then the reverse procedure of first stage is done. The unsigned bits are converted back into signed bits. At last in stage 4 final additions of various complex multiplications is performed and we get the desired FFT in reverse order.
Each step would have different twiddle factors to account for larger computations in a smaller area. Also the FIFOs are longer for the ones in the starting as compared to those in the end. As shown for radix-2 , the first FIFO is fed with half of the input data and the last FIFO stores only a single
The same pipelined structure can also be made for other radices. Since higher radix values require more inputs as compared to radix-2 units, the input entries for each FIFO will increase. Two FIFOs are needed for a radix 3 unit whereas four FIFOs are needed for radix 5 units. The generalised pipeline structure for radix 3 and radix 5 is shown in fig(4).

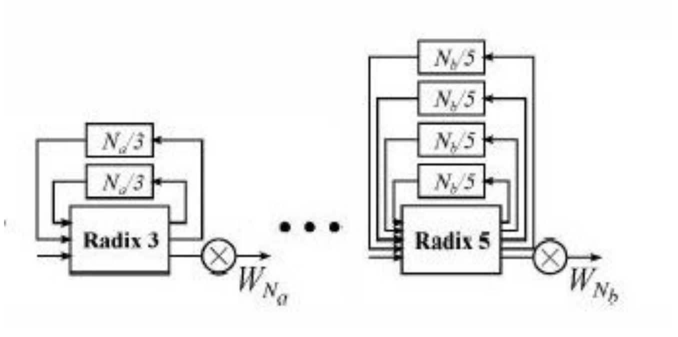

Fig. 4 Pipelined Mixed Radix FFT

So higher radices involve larger storage units. Therefore in a mix radix, higher radices should be put at the end of the pipeline. At the end the multiplication units are unity. This would require lesser computations. Among $N_{1200} = 2^4 3^1 5^2, N_{1200} = 5^2 4^2 3^1$, the former type is selected because it deals with higher radices later.

4) *Output Unit:* This unit consists of latches which temporarily stores the output for every butterfly from the multiplier. EPROMs control the loading of results from output registers to output latches. The output of these latches are connected to microprocessor data lines. Then bit reversal is done using software.

5) Control Unit: The operating sequence of one butterfly is controlled by microprogramming which is stored in EPROMs as shown in fig.5. Addressing of these EPROMs is done by using counters. The counter starts as soon as the complete input data is loaded and therefore, control sequence in data lines of EPROMs is executed.

The counter stops after one butterfly computation is over. At this stage, the microprocessor reads the result from output latches and store them in appropriate memory locations for further access.

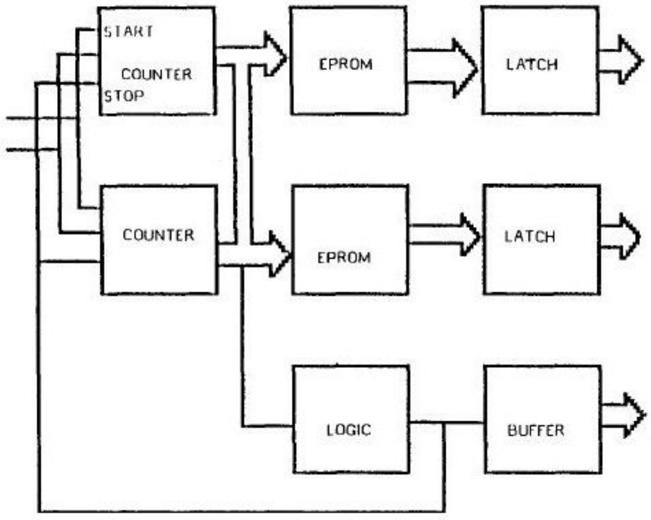

Fig. 5 Control Unit

The hardware implementation of FFT is used for faster calculations of DFT to solve real life problems using DFT calculations. To compute FFT, radix-2 and radix-4 implementations have been discussed. 1 radix-4 butter-fly is capable of doing computations equivalent to 4 radix-2 butterflies and the good fact is that the former requires only 3 multiplications in comparison to 4 multiplications required in radix-2 multiplications. So by increasing the radix index you can do similar computations in lesser number of multiplications. So it would be great to design FFTs with higher index butterflies. But such things are not done because of the complication in control and dataflow of such butterflies. There are lesser number of multiplications but with increased time delay in the circuit which would not be beneficial. Moreover radix 3 and radix 5 computations are also practiced.

REFERENCES

[1] *Leena Thomas MaKiamma Chacko Babu P. Anto Chetlur S. Sridhar "Hardware implementation of fft-8086 based system",1989 IEEE*

[2] *Mitra Nasserbhakt "Efficient hardware implementation of FFT"*

[3] *Johan Lofgren and Peter Nilsson "On hardware implementation of radix 3 and radix 5 FFT kernels for lte systems" 2011 IEEE*

[4] *Senthilkumar Ranganath,Ravikumar Krishnan,H S Sriharsha "efficient hardware implementation of scalable FFT using configurable radix-4/2" 2014 IEEE*

[5] *FT,Wikipedia*