International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 9, November- 2012

Fixed Point Streaming Fft Processor For Ofdm

Sudhir Kumar Sa **Rashmi Panda** **Aradhana Raju**

Abstract

Fast Fourier Transform (FFT) processors are today one of the most important blocks in communication systems. They are used in every communication system from broadband to 3G and digital TV to Radio LANs. This paper deals with the pipelined hardware solution, algorithmic exploration for the FFT processor with the FFT size of 2^N points, This FFT processor is used in OFDM based BPSK/QPSK modulated communication system for the WHD WVAN standard at the Low Rate Physical (LRP) layer.

This Paper presents the design of the 128 point fixed–point FFT streaming processor. The final architecture used is the SDF (single path with delay feedback) that implements the radix-2 FFT algorithm. Since the FFT processor can not be used standalone, so it is employed in an OFDM Transmitter and the performance is measured for SNR over a range of PAPRs.

Introduction

Discrete Fourier transform (DFT) is one of the fundamental operations in the field of digital signal processing. The DFT with a length of N equal to power of 2 is usually implemented with the fast Fourier transform (FFT). The FFT plays an important role in the design and implementation of the discrete-time signal processing algorithms and systems. In general, the FFT implementation typically falls into one of the two categories

1) Method based on Direct Fourier Transform .
2) Method based on direct-hardware implementation of the FFT signal flow graph.

OFDM allows packet-based high data rate communication suitable for video transmission and mobile internet applications. Pipelined FFT processor is a type of real-time FFT architectures characterized by continuous processing of the input data, which for the reason of the transmission economy, usually arrives in the sequential format. However, the FFT operation is very communication intensive which calls for spatially global interconnection. Therefore much effort on the design of the FFT processors focuses on how to efficiently map the FFT algorithm to the hardware to accommodate serial input for computation.

Algorithms

FFT algorithms uses a divide and conquer approach to reduce the computational complexity for DFT, means one long calculation is divided into a number of smaller calculation and finally integrated to get the results. In a communication system that uses an FFT algorithm there is also a need for the IFFT algorithms. Since the DFT and the IDFT are similar both of these can be computed using basically the same FFT hardware as, swap the real and imaginary part of the input data compute the FFT and swap the real and imaginary part of the output. The output is now the IFFT of the input

data, except for the scaling factor in the IFFT algorithm 1/N. This is describes as
The inverse DFT of an N-point sequence X(k), k= 0, 1, 2,…..N-1, is defined as

$$x(n) = 1/N \sum_{k=0}^{N-1} X(k)W^{-nk} \ldots \quad (1)$$

If we take the complex conjugate of the above equation and multiply by N, we find

$$N^*x(n)=\sum_{k=0}^{N-1} X^*(k)W^{nk} \ldots (2)$$

The right hand side of the above equation is recognized to be the DFT of the sequence {X*(k)} and may be computed using any FFT algorithm. The desired output sequence {x(n)} can then be obtained by complex conjugation the DFT of eqn-2 and dividing by N to give

$$x(n) = 1/N \{\sum_{k=0}^{N-1} X^*(k)W^{-nk}\}^* \quad \ldots\ldots..(3)$$

Thus a single FFT algorithm serves for evaluation of both the direct and inverse DFTs.

Common Factor Algorithms:

Common factor algorithms are one way of dividing the problem, using the divide-and-conquer approach. This method is the most widely used way of computing FFTs. The number of points N are then divided into factors according to

$$N = \prod N_i$$

This means that they have one factor in common. In this way an FFT can be computed in i number of steps. There are two equally computational complex algorithms that can be derived from this, DIT (decimation-in-time) and DIF) decimation-in-frequency.

Radix-2 Algorithm:

The radix-2 algorithm is a special case of the common factor algorithm for N-point DFTs, where N is power-of-2.

Radix-r Algorithm:The radix-r algorithm uses the same approach as radix-2, but with the decomposition using base-r instead of base-2. N is factored as

$$N = r^a$$

The derivation of the radix-r is analogous to the derivation of redix-2. The computational complexity for the radix-r is $N\log_r N$ butterfly operations.

Butterflies:

Radix-four Butterfly:

The radix-4 butterfly involves three complex multiplications and four complex additions.

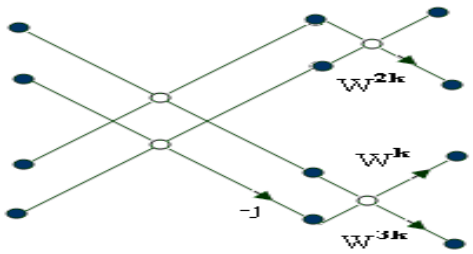

Fig-1 Radix-4 Butterfly

Split Radix Butterfly:

The split radix butterfly involves two complex multiplications and three complex additions.

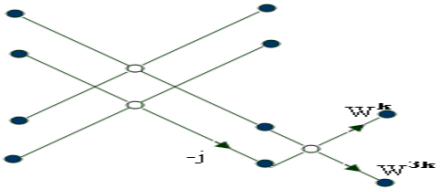

Fig-2 Split-radix Butterfly

The radix-4 algorithm is limited to the number of points, i.e. it is used for the N number of points, where N is divisible by 4. Although the split radix algorithm and radix-4 algorithm results in less multipliers and adders but are not used in general because of their irregular structure for the VLSI implementation point of view. So these algorithms will not be discussed in this thesis.

Pipeline Architectures:

The fastest architectures among all the architectures are the pipelined architecures. The amount of parallelism is $\log_r N$ for these architectures means that there will be $\log_r N$ butterfly stages for the radix-r algorithm. Because of their simple control mechanism and higher efficiency as compared to the general purpose processor,

Figure-3 Basic Pipeline FFT architecture

R4-MDC (Multi Path with Delay Commutator):

This architecture is inefficient for the N inputs (N is divisible by four) because the butterfly's (AE's) will be working only for one-fourth of the time

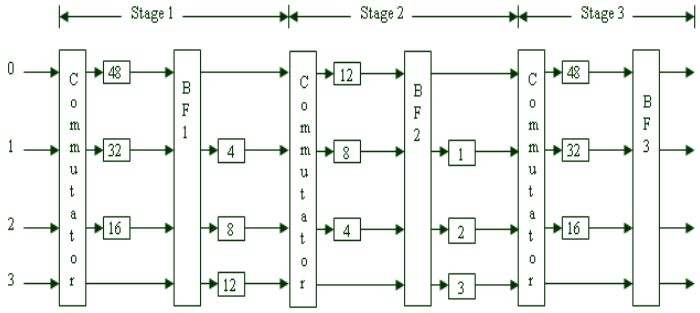

Figure-4 R-4 MDC (16-point) Pipeline FFT architecture

After every 16 clock pulses the commutator switches to the next position so that the subsequent 16 samples goes through a proper delay. At 48th clock time the samples x(0), x(16), x(32), x(63) appears to the input of the butterfly simultaneously and after the 63rd clock time the first stage of the R4-MDC is completed. Hence the Butterfly was used only for16 cycles out of 64. The rest of the algorithm can be completed in similar way. In the next stages the commutator switches four times as rapidly as its predecessor.

Fixed-point Representation:

The fixed-point representation has the following types

1) Wordlength
2) Integer Wordlength

This is illustrated in figure-5

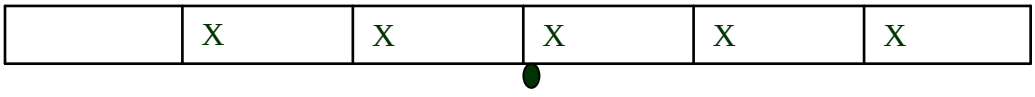

Figure-5 Fixed-point representation

The word length indicates the number of bits to represent a fixed-point number. The integer wordlength indicates the number of bits including the sign bit, used in representing the integer part of the fixed-point representation. The first bit represents the sign bit, the next bits before the decimal point are the integer bits i.e. the number of bits to represent the integer part, the bits next to the decimal point are the bits to represent the fractional part.

Designing of the fixed-point 128–point Pipelined FFT processor using SDF:

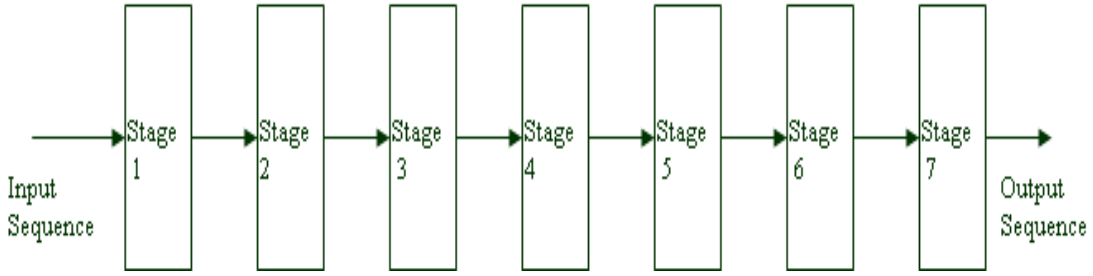

Figure-6 Pipelined-FFT satges

There are seven stages ($N=2^v$) in this pipelined SDF DIF architecture. Each stage contains the radix-2 butterfly element, which contains two adders and one multiplier. The one stage is shown in fig (7). The seven such stages connected in cascade form constitute the 128-point pipelined FFT architecture. The twiddle factor is nothing but the $\mathbf{exp^{-j\theta}}$ term whose magnitude ranges from -1 to 1. So to represent this range in 2's complement form it requires at least two integer bits. The twiddle factor is multiplied by the subtractor output data in the multiplier **M1,** since the twiddle multiplication factor is always less than or equal to 1, so there is no need to increase the integer precision bit for the multiplier. Because if any maximum number N1 multiplied by the number N2 that is less than 1 will result in a number which is lesser in amplitude than the number N1.

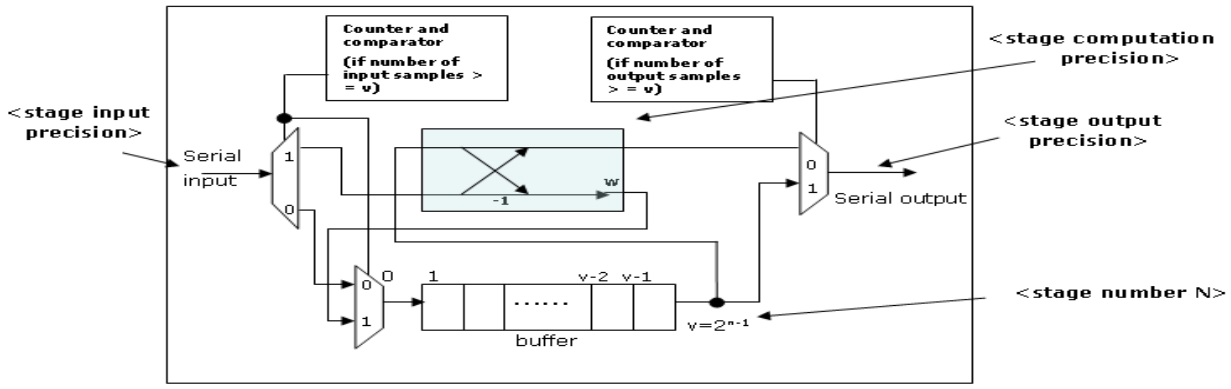

Figure-7 DIF n^{th} stage

are two output sequences available from the butterfly, one of which is fed to the buffer (memory) and another is forwarded to the next stage via the output switch. The size of the buffer decreases from the first stage to the last stage, since the input sequence is in-order and the 1st input sample is processed in the butterfly with the (N/2)+1th element as specified by the DIF algorithm. So the first stage contains a buffer of size N/2, second stage with a buffer of size N/4 and so on.

Precision Determination:

Input Precision = P
Adder Precision = P+1
Twiddle Precision = TP
Multiplier **M1** Precision = TP+P and truncated to P+1
Scale Factor Precision = SP
Multiplier **M2** Precision = P + SP and truncated to P

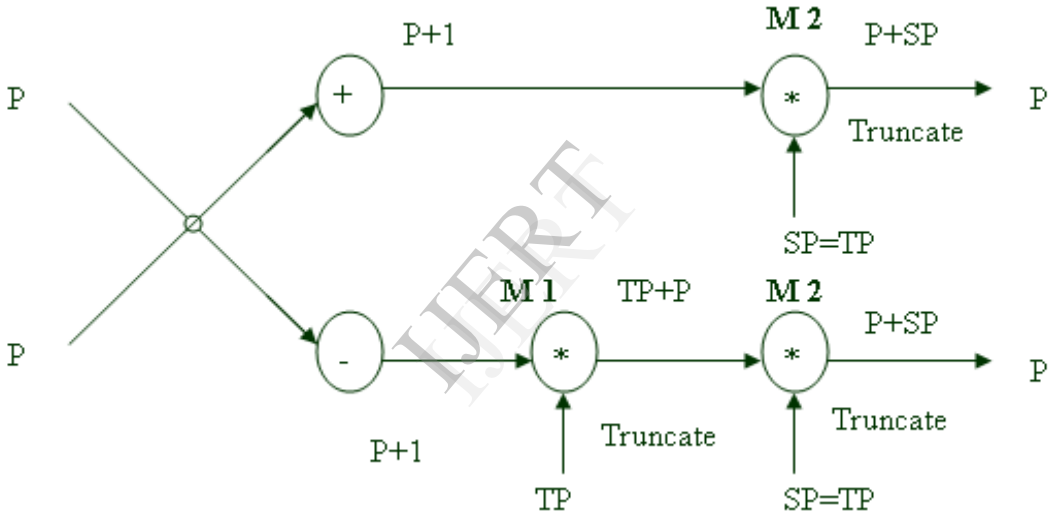

Figure-8 Precision Description for arithmetic elements

After fixing the precision for each of the arithmetic elements the noise calculation was done for the individual stage. For an SNR target of 45dB at 0dB PAPR (peak to average power ratio), the noise contributed by each individual stage was calculated in such a way that the noise generated by only one stage can be determined.

Noise Threshold Calculation

Noise threshold calculation is required for the stage wise precision determination in the pipelined architecture and to maintain the SNR aver the PAPR values. For the proper functionality of the system the noise level should not cross that threshold, otherwise the system will provide the erroneous result.
We know that SNR is given by

SNR = Signal power/Noise power

Or Noise power (dB) = signal power(dB) – SNR(dB) ……(4)

And according to the Parseval's Energy theorem

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

or in the other way,

input RMS (root mean square) = (1/√N)*output RMS
or
output RMS = √N *input RMS……(5)

This equation indicates that output signal rms is √N times the input signal rms, i.e. the FFT gives the gain of N, where N is the number of FFT points. This is the case when there is no scaling.

Let, N =128,
If the SNR target is 45dB at 0dB PAPR (peak to average power ratio), then noise level is -45 dB
From equation (1)

Case 1: No Scaling

From equation (2),
Output signal power (dB) = $10*\log_{10}(\sqrt{N}) + 20*\log_{10}$(input RMS)…….(3)
Noise power (dB) = output signal power (dB) – 45dB
= $10*\log_{10}(\sqrt{N}) + 20*\log_{10}$(input RMS) – 45dB
= - 45dB + $10*\log_{10}(\sqrt{128})$ + 0 {because the input PAPR is 0dB}
= - 45 + 21.07
= -24dB
This is the noise contributed by total number of stages, so the noise contributed by individual stage is given by
Number of stages (v) = $\log_2$ N, for N = 128, there are 7 stages

Noise power per stage (dB) = - 24 - $10*\log_{10}$(v)

= - 24 -$10*\log_{10}$(7)

= - 24 -8.45
= -32.5 dB
It indicated that the noise threshold for a single stage is -32.5dB, so the noise contributed by each stage should not be grater than this threshold.

Case 2: 1/N global Scaling

From equation (3),
output RMS = (1/√N) * input RMS
Noise power (dB) = output signal power (dB) – 45dB
= $10*\log_{10}(1/\sqrt{N}) + 20*\log_{10}$(input RMS) – 45dB
= - 45dB + $10*\log_{10}(1/\sqrt{128})$
= - 45dB - $10*\log_{10}(\sqrt{128})$
= - 45 - 21.07
= -66dB

Noise power per stage (dB) = - 66 - $10*\log_{10}$(v)

= - 66 -8.45
= - 74.5dB
So in this case the noise threshold is -74.5dB per stage

Case 3: 1/√N global Scaling

From equation (3),
output RMS = input RMS
Noise power (dB) = output signal power (dB) – 45dB
= -45 dB

Noise power per stage (dB) = - 45 - $10*\log_{10}$(v)
= - 45 - 8.45
= -53.5dB

By running the sweeps for the precision determination, vary the precision of the particular stage until the noise crosses the threshold. The other stages should be kept ideal i.e. should not contribute the noise, so to

make them ideal their precision is kept high say 25 bits.

Experiments done on 128 – point FFT:

With the global scaling of 1/N, the scale factor is selected as 1/2 each of the stage and the following precision are obtained with the input dynamic range of (2, -2). The SNR target was 45dB at 0dB PAPR.

Scaling	S1	S2	S3	S4	S5	S6	S7
1/(N)	2.8	3.8	3.8	3.9	3.11	3.12	3.12

The S1 to S7 are the stage numbers, and the second row in the table shows the first digit as the integer bit while the second digit indicates the fractional part precision. We see that there is a progressive increase in the stage precision from the first stage to the last stage. This is because the noise contributed by the first stage is accumulated to the output through the intermediate stages.

The S1 to S7 are the stage numbers, and the second row in the table shows the first digit as the integer bit while the second digit indicates the fractional part precision. We see that there is a progressive increase in the stage precision from the first stage to the last stage. This is because the noise contributed by the first stage is accumulated to the output through the intermediate stages. From the above results we see that there is increase in one bit in the integer precision from 3 bits to 4 bits, but the overall precision is constant and is 11 bits. There is no progressive increase in the stage precision.

Results for 1/√N scaling for the 128-point FFT:

Scaling	S1	S2	S3	S4	S5	S6	S7
1/√(N)	4.7	4.7	4.7	4.7	4.7	4.7	4.7

If we reserve the 3 bits for the integer part then we obtain that with 1/√N scaling the same design can work well for large values of PAPRs.

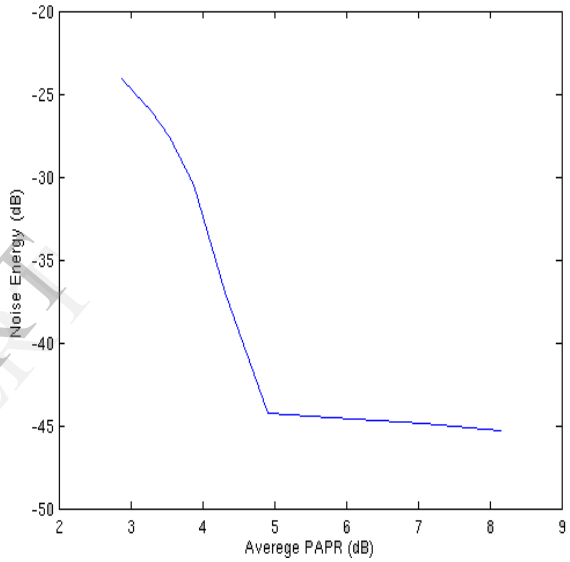

Figure-9 PAPR vs Noise energy for alternate scaling

The conclusion is that with the same integer bits for both the scaling methods, 1/root(N) scaling factor works fine over input PAPR of 9 db. The application for which the FFT processor is designed the input PAPR range was 9 – 14 dB. So in that case the 1/√N scaling method is the better choice.

The graph in fig(9) shows the PAPR vs noise energy in dB with the alternate scaling of 1/2 indicates that by reserving the 3 bits for the integer

precision and 8 bits for the fraction part precision and considering that the input dynamic range is between (-1, 1) the designed architecture can work fine for the PAPR range from 4.8dB onwards. If the scaling of $1/\sqrt{2}$ is done in all the stages to get the global scaling of $1/\sqrt{N}$ then only two bits are sufficient for the integer part for the same input

Conclusion

In this paper, a novel 128-point FFT/IFFT processor for OFDM-based WPAN systems has been proposed. In this architecture, high throughput rate can be dynamic range.The number of complex multiplications is reduced effectively by using a higher radix algorithm.

REFERENCES

[1] Ahmad R. S. Bahai,Burton R Sultzberg ‘‘Multicarrier digital communications theory and application of OFDM’’

[2] T. Xanthopoulos and A. P. Chandrakasan, “A low-power IDCT macrocell
for MPEG-2 MP@ML exploiting data distribution properties for
minimal activity,” *IEEE J. Solid-State Circuits*, vol. 34, pp. 693–703,
May 1999.

[3] E. Grass, K. Tittelbach, U. Jagdhold, A. Troya, G. Lippert, O. Krueger,
J. Lehmann, K. Maharatna, N. Fiebig, K. Dombrowski, R. Kraemer, and
P. Maehoenen, “On the single chip implementation of a hiperlan/2 and
IEEE802.11a capable modem,” *IEEE Pers. Commun.*, vol. 8, pp. 48–57,
Dec. 2001.

[4] K. Maharatna, E. Grass, and U. Jagdhold, “A novel 64-point FFT/IFFT processor for IEEE 802.11a standard,” in *Proc. ICASSP’03*, vol. II, pp.
II-321–II-324.

[5] C. Chen and L.Wang, “A new efficient systolic architecture for the 2-D
discrete Fourier transform,” in *Proc. IEEE Int. Symp. Circuits and Systems*,
vol. 6, ch. 732, 1992, pp. 689–692.

[6] M. Ghouse, “2D grid architectures for the DFT and the 2D DFT,” *J. VLSI Signal Process.*, vol. 5, pp. 57–74, 1993.

[7] H. Lim and E. Swartzlander, “A systolic array for 2-D DFT and 2-D
DCT,” in *Proc. Int. Conf. Application Specific Array Processors*, 1994,
pp. 123–131.

[8] N. Zhang, X. Zhang, and C. Zhang, “Two kinds of array architecture for
FFT processor,” in *Proc. Int. Conf. Signal Processing*, vol. 2, ch. 355,
1990, pp. 1161–1162.

[9] J G Prokis “ Digital Signal Processing’’ 4th Editoin 2008.

[10] THOMAS KELLER AND LAJOS HANZO ‘‘Adaptive Multicarrier Modulation: A Convenient
Framework for Time-Frequency Processing in
Wireless Communications’’. IEEE Proceding of the IEEE, VOL. 88, NO. 5, MAY 2000.