ISSN XXXX XXXX © 2017 IJESC

IJESC

Research Article Volume 7 Issue No.5

A Comparative Study on High Speed and Low Power Radix FFT

Krishan Kumar[1], Sonal Dahiya[2], Ved Prakash[3]
Assistant Professor[1, 2, 3]
Department of Computer Science & Engineering[1], Department of Electronics & Communication Engineering [2, 3]
ASET, Amity University Haryana, Gurgaon, India

Abstract:
This paper presents comparison between radix-2, radix-4, radix-8 and split radix FFT on the basis of signal flow graphs. Out of which split radix FFT is found to be best radix FFT in which pipeline is repartitioned to balance the latency between complex multiplication and butterfly operation. The number of complex multiplier is minimized via a bit inverse and bit reverse data scheduling scheme.

Keywords: speed, power, radix, FFT

I. INTRODUCTION

The Fast Fourier transforms (FFT) and its inverse (IIFT) is one of the fundamental operations in the field of digital signal processing. The FFT /IIFT are widely used in various areas as telecommunication, speech and image processing, medical electronics and seismic processing etc. Recently, the FFT/IIFT is used as one of the key component in OFDM (orthogonal frequency division multiplexing) wide band based communication system like XDSL modem and wireless mobile terminal according to European digital video (audio broadcasting (DVB-T/DAB standard) ,an OFDM system may require FFT length ranging 256 – 8192 point. Wireless local area network (WLAN) and hyper LAN /2 system require high speed and low power FFT /IIFT design. The fourth generation Cellular phone and forthcoming new LAN system may also incorporate OFDM system to deliver higher bandwidth .Hence, it is important to design high performance and low power FFT for these applications. For a static CMOS circuit the power consumption is usually determined by dynamic power which can be written as:

$P_{dynamic} = \alpha . V_{dd}^2 . F_{CLK}$ ……eq 1

Where α is switching probability, C_L is capacitance being charged or discharge when switching, V_{dd} is supply voltage and F_{CLK} is clock rate. At architecture level ά can be divided as operation count for specific module ,C_L is proportional to part of module that is being active e.g. for a multiplier if the operation count is MUL# within T clock cycles cycle then α is proportional to MUL# /T . As the α is usually determined by algorithm and its corresponding architecture, it is desirable that chosen FFT algorithm has the least computational complexity as well as the corresponding hardware complexity. In this paper we will compare various radix algorithm for example-radix-2, radix-4, split radix, radix2^2, radix2/4/8 and higher radix version, in general these entire algorithm decompose a length n (=2^{λ}) FFT into odd half and even half recursively and effectively reduce the number of complex multiplication by utilizing symmetric properties of the FFT kernel.

II. RADIX-2^N FFT ALGORITHM.

Basic formulation and low radix algorithm for a given input sequence x_n and N-point Fourier tans form is defined as:-

$A_k = \sum_{n=0}^{N-1} x_n W_N^{nk}$ where, k=0,1,2------N-1 -----eq 2

Where n is time index and k is frequency index. And coefficient w_N^{nk} is defined as:

$$W_N^{nk} = \cos\left(\frac{2n.nk}{N}\right) - j.\sin\left(\frac{2n.nk}{N}\right) \quad \text{-----eq3}$$

For Cooley-Tukey radix -2 decimation in frequency (DIF) decomposition is decomposed into even and odd frequency component

$$A_{2k} = \sum_{n=0}^{\left(\frac{n}{2}\right)-1} (x_n + x_{n+\left(\frac{N}{2}\right)}).W_{N/2}^{nk} \quad \text{----eq 4}$$

$$A_{2k+1} = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} (x_n - x_{n+\left(\frac{N}{2}\right)} W_N^n . W_{N/2}^{nk} \quad \text{----eq 5}$$

This is usually referred as twiddle factor. The split radix fast Fourier transform (SRFFT) algorithm further decomposes the odd frequency component into 4k+1 and 4k+3 frequency components as shown below….

$$A_{4k+1} = \sum_{n=0}^{\left(\frac{N}{4}\right)-1} (x_n + j.x_{n+\left(\frac{N}{4}\right)} - x_{n+\left(\frac{N}{2}\right)} - j.x_{n+\left(\frac{3N}{4}\right)} W_N^n . W_{\left(\frac{N}{4}\right)}^{nk} \quad \text{---------eq 6}$$

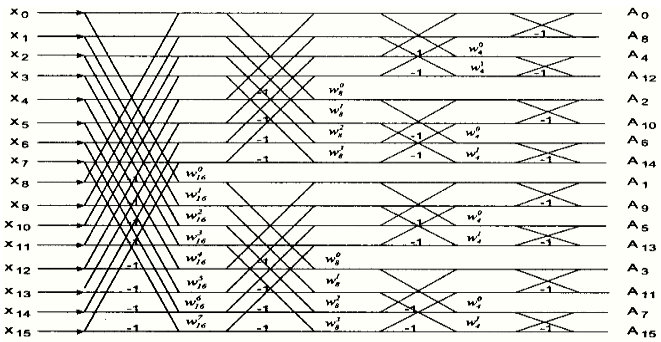
$$A_{4k+3} = \sum_{n=0}^{\left(\frac{N}{4}\right)-1} (x_n + j.x_{n+\left(\frac{N}{4}\right)} - x_{n+\left(\frac{N}{2}\right)} - j.x_{n+(3N+4)} . W_N^{3n} . W_{\left(\frac{N}{4}\right)}^{nk} \quad \text{-----eq 7}$$

By applying equation (4),(6) and(7) recursively the split radix FFT can be obtained. On the other hand radix-4 algorithm can be obtained by decomposing equation (4) and (5) into A_{uk}, A_{uk+2},A_{uk+1} and A_{uk+3} frequency components. Table-1 shows the multiplicative complexity or radix-2, radix_4 and split radix algorithm for non – 4^n length FFT, and additional radix 2 stages is used for the radix-4 algorithm.

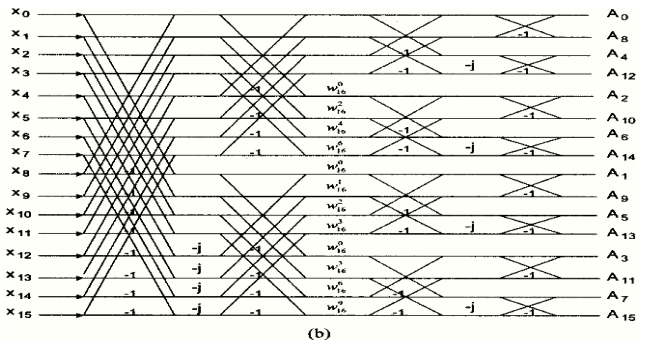
TABLE.I. MULTIPLICATIVE COMPLEXITIES

DFT SIZE	RADIX_2	RADIX-4	SPLIT RADIX
8	2	2	2
16	10	8	8
32	34	28	26
64	98	76	72
128	258	204	186
256	642	492	456
512	1538	1196	1082
1024	3586	2732	2504
2048	8194	6316	5690

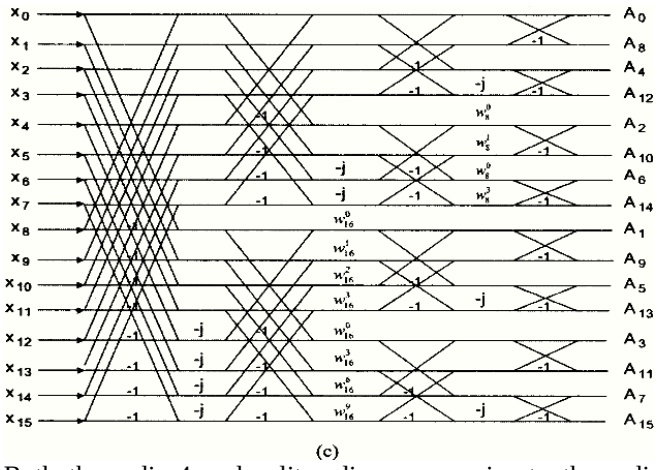
This shows the non-trivial complex multiplication required for 2^n- point FFT algorithm, from which it is clear that split radix algorithm shows a clear advantage over the other algorithm. To understand the relationship among the three algorithms intuitively we can examine their signal flow graph of 16 point. (a) radix-2 FFT (b) radix-4 FFT (c) split radix FFT, as shown below



(a)

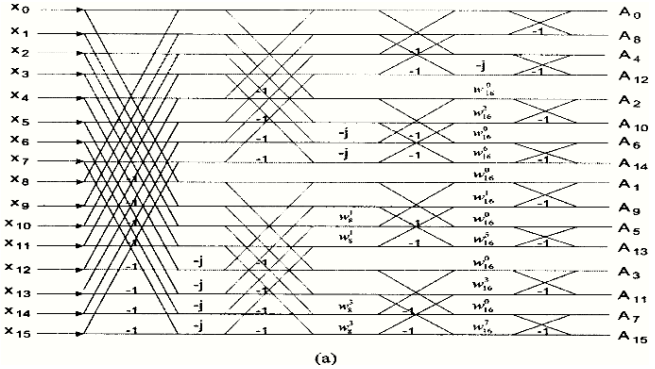


(b)

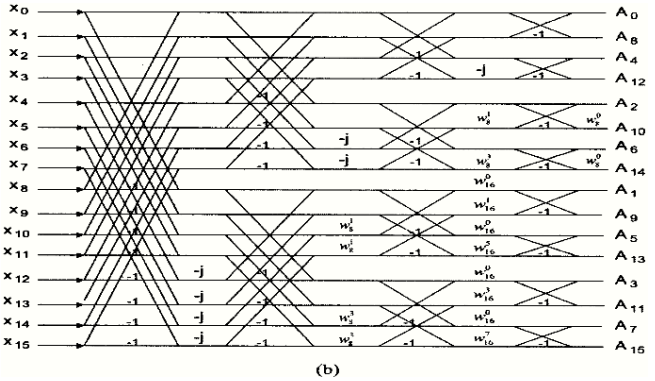


(c)

Both the radix-4 and split radix are superior to the radix-2 algorithm because "-j" term are extracted. The complex multiplication with –j are accomplished by exchanging the real and imaginary parts of the incoming data and then inverting the sign of imaginary part. The split radix algorithm is superior to radix4 algorithm because more "j" terms are extracted in the SFG. An important point to be rooted here is that four twiddle factors are moved from end of third butterfly stage and two of them become trivial multiplication. If multiplicative complexity lower than split-radix algorithm is desirable, higher radix FFT algorithm should be used. However high-radix FFT algorithm often increases the circuit complexity and is not easy to implement. To discuss and to compare the efficiency of high radix algorithm Radix-8 and radix- 2/8 algorithm are examined here. The radix 2/8 algorithm can be considered to be as an extension to split radix (radix 2/4) algorithm by decomposing (6) and (7) and one more radix 2-stage. The SFG (signal flow Graph) of radix 8 and radix 2/8 algorithm is shown below



(a)



(b)

Thus a complex multiplication with one of the two coefficients can be computed using a constant multiplier and additions. The number of complex multiplications and number of constant multiplications and summarized in table below.

TABLE.II. NONTRIVIAL MULTIPLICATIONS REQUIRED FOR RADIX-8 & RADIX 2/8 FFT ALGORITHMS

DFT size	Radix-8		Radix 2/8	
	Complex	Constant	Complex	Constant
8	0	2	0	2
16	6	4	4	6
32	20	8	16	14
64	48	32	44	38
128	152	64	120	94
256	376	128	308	214
512	824	384	736	494
1024	2104	768	1724	1126

Table shown here tells us about the nontrivial multiplications required for Radix-8 & Radix 2/8 FFT algorithms. Obviously the radix 2/8 algorithm has lower multiplicative complexity than radix 8 algorithm because it extracts more w^1_8 and w^3_8 coefficients in signal flow graph. As far as hardcore implementation, is concern, we found that the area of constant multiplier is one and half time of a real multiplier. Hence one constant multiplier is approximately equivalent to 0.4 complex multiplications. To compare the multiplicative complexity between low radix algorithm and high radix algorithms, one can multiply the number of constants multiplications by 0.4 and calculate the equivalent number of complex multiplication.

III. CONCLUSIONS

Based upon the above discussion we can see that if the multiplications with ± 1 and +-j are removed and multiplications with +-(√2/2)(1+-j) can be realized using

constant multiplication radix 2/8 algorithm will have the lowest multiplicative complexity amongst all the discussed algorithm. On the other hand the constant radix algorithm has more regular signal flow graphs than mixed radix algorithm. However radix -4 and radix 89 algorithms can be applied to 4^n-point and 8^n point FFT'S only unless a radix 2 stage is also employed in the pipeline. Such a limitation also exists in other fixed radix FFT algorithm but does not in the split radix or radix2/8 algorithm.

IV. REFERENCES

[1]. N.Weste and D.J. Skellen "VLSI for OFDM", IEEE Communication magazine, Vol.36,pp.127-131,OCT.1998.

[2]. P.Dhumael," Algorithms meeting the lower bounds on the multiplicative complexity of length-2 "DFT and their connection with practical algorithms", IEEE Trans. Acoust. Speech, Signal Processing, Vol.38, Sep.1990.

[3]. S.He and M.Torkelson ," Designing pipeline FFT processor for OFDM (dc) Modulation ", in Proc. IEEE URSI Int. Symp. Signals system, Electron, pp-257-262, 1998.

[4]. B.W.Y.Wei,H.Du and H.Chen " a complex number multiplier using Radix-4 digits", IEEE 12^{th} symp. Pp.84-90, 1995.

[5]. C.S. Wallace,"A suggestion for a fast multiplier", IEEE trans. Comput. Vol C-13,pp. 14-17,Feb. 1964.

[6]. Shousheng He and Mats Torkelson "Designing pipeline FFt processor for OFDM modulation", ISSSE Vol.2 pp.945-950, 1998.

[7]. A.V.Oppenhein and R.W.schafer "Digital Signal Processing", Prentice Hall, 1975.