ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Radix 2^5 FFT Architecture Implementation in Xilinx FPGA

[#1]S.Selvakumar, [#2]L.Stephy jasmine rani, [#3]G.Vijayalakshmi, [#4]N.Vishnudevi,[#5]N.Janakiraman

Department of Electronics and Communication Engineering, K.L.N College of Engineering ,Tamlinadu,India

ABSTRACT- FFT is a highly efficient procedure to reduce computation time and also improves the performance. The Radix 2^2, 2^3 and 2^4 FFT architectures are not efficient because of its low utilization of components. Our proposed design will provide high data throughput and low complexity VLSI structure for Radix 2^5 FFT architecture. Most of previous architectures were designed using the complex booth multipliers, but our proposed architecture uses canonical signed digit (CSD) multiplier circuit. This entire proposed architecture simulated in Xilinx 12.2 system edition software and implemented in Xilinx Virtex-5 XUP FPGA kit. To optimize the power, area and speed of the signal process, pipelining and parallel processing techniques have to be used in this proposal. In future this Radix 2^5 FFT architecture will be incorporated in MIMO-OFDMA based software defined radio (SDR) architecture.

KEY WORDS :Fast fourier transform, pipelining, discrete fourier transform, radix

I.INTRODUCTION:

Today's technology based on hardware and power efficiency for high performance. Application such as digital signal processing , communications etc are based on digital function which requires complex functionalities .For this fast fourier transform is one of the efficient method to implement discrete fourier transform due to its reduced computations. Our proposed design "RADIX 2^5" will provide high data throughput and low complexity VLSI structure .Power , area and speed of the signal processing can be optimized using pipelining techniques.

II.FAST FOURIER TRANSFORM:

The Fourier Transform decomposes a wave form basically any real world wave form into sinusoids. It is possible to generalize the Fourier transform on discrete structures such as Finite Groups. The Efficient Computation Of such structures, by fast Fourier transform, is essential for high speed computing. FFT algorithms are commonly employed to compute DFTs, but there is a clear distinction is that "DFT" refers to a Mathematical transformation, regardless of how it is computed, whereas "FFT" refers to a specific families of a algorithms for computing DFTs. Fast Fourier Transform (FFT) is developed by **Cooley and Tukey** in **1965.**

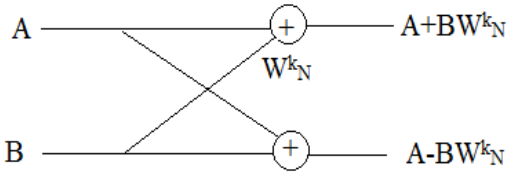
Highly efficient procedure for computing the DFT of a finite series and requires less number of computations than that of direct evaluation of DFT. Fast Fourier transform (FFT) is based on decomposition and breaking the sequence into smaller sequences and combining them to get total sequence. The FFT time domain decomposition is usually carried out by a bit reversal sorting algorithm .There are various pipeline structures using radix-2, radix-4 and split radix algorithm. In 1998, HE and TORKESON suggested radix 2^2 and 2^3 FFT algorithm .These algorithms are characterized by the trait that reduces the number of non –trivial multiplications in the radix-2 algorithm architectures .It has same number of non –trivial multiplications at the same positions in the single flow graph as the same butter fly structure as that of the radix-4 algorithm but has the same butterfly structure as that of the radix-2 algorithm.Radix-2 algorithm is characterized according to the merit that it has some multiplicative complexity as the radix-4 algorithm

but still retains simple structures of the radix-2 butterfly[1].

III.RADIX-2:

The Radix indicates the size of FFT decomposition. In this paper Radix is 2 which is single-Radix FFT. For single Radix FFTs, transform size must be choose according to the power of Radix. Here we use 32and64 sizes, which is 2^5 and 2^6.The Radix-2 Decimation-in-Time FFT (DIT-FFT) is applied to the two Points N/2 DFT's. To find the number of butterfly stages required to compute N length, sequence can be $M=\log_2 N$, and N/2 butterfly operations are computed in each stage. In this paper there are 5 butterfly stages and 16 butterfly operations are computed to produce 32 Point FFT [2].

In DIT-FFT the given input sequence is in shuffled order and the output sequence is in natural order. By using Bit-Reversal input sequence gets shuffled. The Radix-2 decimation in time FFT is the basic form of Cooley-Tukey implementation algorithms [1]. Radix-2 first computes the DFT of the even index inputs and the odd index inputs and then combines the two results to produce the entire DFT sequence. The basic computation block in the FFT is butterfly in which the two inputs are combined to give two outputs The FFT operation of butterfly diagram is shown in the below figure, and the powers of the twiddle factors associated in butterflies are in natural order.



The twiddle factor exponent k of each stage is calculated by using below equation;

K = N t/2m where t=0, 1, 2 ,..... 2m-1

IV.RADIX 2^2 ALGORITHM :

Radix 2^2 algorithm derivation was derived by considering the first 2 steps of Cascade Decomposition in the radix 2 DIF – FFT together .when a 3 – Dimensional index map[3] was applied ,

$n=N/2\ n_1+N/4\ n_2+ n_3\{n_1,n_2=0,1\ n_3=0\sim N/4-1\}$

$n=k_1+2k_2+4k_3\ \{k_1, k_2=0,1\ k_3=0\sim N/4-1\}$ (1)

Decomposing the composite Twiddle Factor ,it can be rewritten as

$W_N^{(N/4\ n2+n3)(k1+2k2+4k3)(N/4\ n2+n3)(k1+2k2+4k3)}$

$=(-j)^{n_2(k_1=2k_2)}W_N^{n_3(k_1+2k_2)}W_{N/4}^{n_3k_3}$ (2)

After Substitution in (2)

$X^{(k_1+2k_2+4k_3)}=$

$$\sum_{n3=0}^{\frac{N}{4}}\left[\frac{HN}{4^{k1k2(n3)}\,WN^{n3(k1+2k2)}}\right]WN^{n3k3}$$

(3)

Then, full Multiplications are used to apply the decomposed Twiddle Factor $WN^{n3(k1+2k2)}$ in (3).After this Cascade decomposition recursively to the remaining DFT's of length N/4 in (4) the complete Radix-2^2 FFT algorithm was obtained.

Radix-2^2 algorithm was obtained based on the merit that it contains same Multiplicative Complexity.

V.RADIX 2^3 ALGORITHM :

Radix 2^3 Algorithm was derived by considering the first 3 steps of Cascade decomposition .The linear index mapping converted into 4-dimensional linear maps[3].

$n=(N/2n_1+N/4n_2+N/8n_3+n_4)N$

$k=(k_1+2k_2+4k_3+8k_4)N$ (4)

As previously explained, with cascade decomposition Twiddle Factor can be defined as

$W_N^{(N/2n_1+N/4n_2+N/8n_3+n_4)(k_1+2k_2+4k_3+8k_4)}$

$=(-1)^{n_1k_1}\ (-j)^{n_2(k_1+2k_2)}\ W_8^{n_3(k_1+2k_2+4k_3)}\ W_N^{n_4(k_1+2k_2+8k_3)}$

$W_{N/8}^{n_4k_4}$ (5)

After substitution of eqn (5),

$X(k_1+2k_2+4k_3+8k_4)=$

$$\sum_{n4=0}^{\frac{N}{8}-1}\left[\frac{TN}{8^{k1k2k3(n4)}W^{n4(k1+2k2+4k3)}}\right]WN/8\hat{}n4k4 \quad (6)$$

Hence a Booth Multiplier called a Programmable Multiplier is used instead of a Constant Multiplier.

VI.RADIX 2^4 ALGORITHM :

Radix 2^4 algorithm was derived by taking 1st cascade decomposition. The linear index mapping transformed in the form of 5-dimensional linear index map [3].

$n=(N/2n_1+N/4n_2+N/8n_3+N/16n_4+n_5)N$

$k=(k_1+2k_2+4k_3+8k_4+16k_5)N$ (7)

With cascade decomposition Twiddle Factor can be expressed in the form,

$W_N^{(N/2n_1+N/4n_2+N/8n_3+N/16n_4+n_5)(k_1+2k_2+4k_3+8k_4+16k_5)}=\{(-1)^{n1k1}(-j)^{n2(k1+2k2)}\ W_8^{n3(k1+2k2+4k3)}\ W_{16}^{n4(k1+2k2+4k3+8k4)}\}$

$W_N^{n5(k1+2k2+4k3+8k4)}\ W_{N/16}^{n4k4}$ (8)

After substitution of eqn (8)

$X(k_1+2k_2+4k_3+8k_4+16k_5)=$

$$\sum_{n5=0}^{\frac{N}{16}-1}[GN/16^{k1k2k3k4(n5)}$$

$W_N^{n5(k1+2k2+4k3+8k4)}]\ W_{N/16}^{n5k5}$ (8)

Hence the Complex Multiplication for the Twiddle Factors, $W_{16}^{i(k_1+2k_2)}$,can be reduced to a Constant Multiplier with some control logics.

VII.RADIX 2^5ALGORITHM:

This paper Proposes and concentrate on the design of 32 point FFT and its performance analysis. By using VHDL as a design entity the synthesis and stimulation is done on Xilinx ISE Design Suite12.2.A DFT Decomposes a sequence of values into components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly: Computing a DFT of N points, takes $O(N^2)$Arithmetical operations, while an FFT can compute the same DFT in only O(N log N) operations. FFTs can decomposed using DFTs of even and odd points ,which is called a decimation in-time (DIT) FFT, or they can decomposed using another approach which is called a Decimation-infrequency(DIF) FFT.

Computation of the end point DFT corresponds of computation of N samples of Fourier transform at N equally spaced frequencies. Consider the input x(n) of length N is a complex data sequence, its DFT X(k) is also complex data sequence of length N which is defined as map[3]

$$X(K)=\sum_{n=0}^{N-1} x(n)\, W_N^{nK},\ K=0, 1,.....N-1.$$

Where W_N denotes, exp{-j2π/N}, the Nth primitive root of unity with its exponent being evaluated modulo N the 'n' is the time index and the 'K' is the frequency index. Twiddle Factor coefficients are used to combine the results from the previous stage to form inputs to the next stage.

Twiddle Factor Computation based on Radix 2^5 algorithm[1]

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Radix 2^2	-j	W_{512}	-j	W_{128}	-j	W_{32}	-j	W_8
Radix 2^3	-j	W_8	-j	W_{512}	-j	W_{64}	-j	W_8
Radix 2^4	-j	W_{16}	-j	W_{512}	-j	W_{16}	-j	W_{32}
Radix 2^5	-j	W_8	W_{32}	-j	W_{512}	-j	W_{16}	-j

VIII.CANONIC SIGNED DIGIT MULTIPLIER

In this paper we are using canonic Signed Digit (CSD) Multiplier instead of Fixed Width Multiplier used in previous architecture. In Fixed Width Multiplier, in the resulting product a significant error will be introduced and it is undesirable for many DSP applications. To reduce the error of the Fixed Width Multiplier, a constant bias is added to the retained cells. However its, product is still large [10]. To overcome this, CSD is used in this architecture. CSD Multiplier is used for encoding the floating point numbers in two's complement representation [4].

The CSD Multiplier has the function to multiply successive input data values by one or more predetermined constant values, when the input data values are in binary format and finally result will make rounded to p-number of bits.

The constant value is in CSD format. Input data values are in the form d0,d1,d2,.....dM-1for each di for i=0,1,2.....M-1 takes one of the values 0 & +1 and which the constant values are in the form b0,b1,b2.....bN-1 for each bi for i=0,1,2,.....N-1 takes one of the values 0,+1,-1 and where no consecutive bi are non-zero. The CSD Multiplier result was obtained addition and shift operations[8].

In normal way the multiplication operation involves two major steps, Partial product generation and Accumulation. The speed of multiplication can be improved by reducing number of partial product. Number of partial products depends on the number of non-zero digits. Number of non-zero digits is proportional to number of partial products.

BLOCK DIAGRAM:

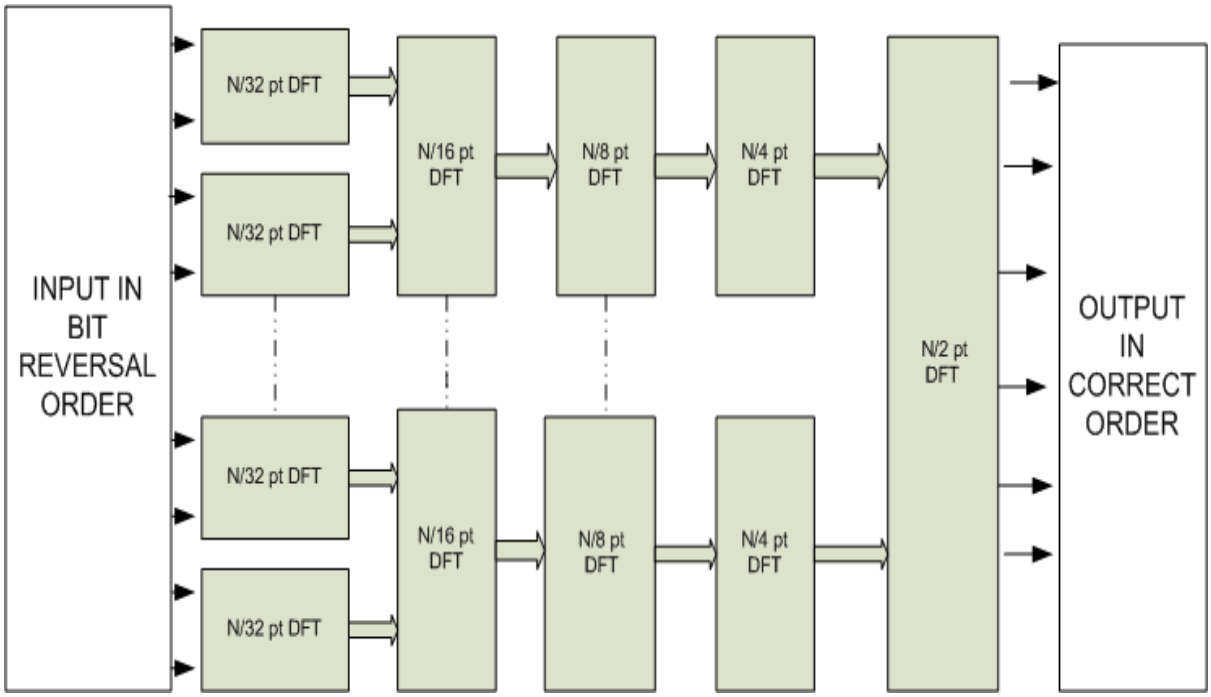


Fig 1:Block of 32 point radix-2 algorithm

The method of CSD is used for multiply the floating point. The representation of floating point contains a sign, mantissa and exponent. The floating point multiplication involves three steps:[11]

- First, Multiplication process involves in two mantissa numbers**:** Floating point stores in signed form but multiplier need with unsigned form. Mantissa have p-number of bits, the product will be 2p number of bits and finally result will make rounded to p-number of bits.
- Second, to compute the exponen**t:** the exponent is represented as bias. It differs from various number (i.e) for single the bias is 127 and for double precision the bias is 1023.
- Third, To compute sign bit**:** by using Ex-or operation for 2 sign bits.

This method has the advantage of decreasing the number of additions/subtractions, needed, as well as handling negative multipliers. Results are obtained by expressing the multiplier in Canonic Signed Digit (CSD) form. CSD representation is useful for the design and implementation of digital filters such as the area-efficient programmable FIR digital filter architecture. It enables the reduction of the number of partial products that must be calculated fast and also low-power consumption and low area structure of a multiplier for DSP applications.

CSD representation is unique and has two main properties:

- the number of nonzero digits is minimal, and
- No two consecutive digits are both nonzero, that is, two nonzero digits are not adjacent.

The CSD representation of an integer number is a signed and unique digit representation that contains no adjacent nonzero digits. Given an n-digit binary unsigned number

$X=\{x0,x1,\ldots\ldots x_{n-1}\}$

and then (n+1) digit representation

$Y=\{y0,\ldots\ldots\ldots,y_n\}$.

There are two encoding method used in binary representation of CSD.The encoding method have two variables y_i^s is sign bit and y_i^d is data bit[12].

	Encoding 1	**Encoding 2**
yi	$y_i^s\ y_i^d$	$y_i^s\ y_i^d$
0	00	00
1	01	01
-1	11	10

To convert binary representation to CSD representation is based on

$$2^{i+j-1}+2^{i+j-2}+\ldots\ldots\ldots+2^i=2^{i+j}-2^i$$

IX.CSD Coding:

Xi+1	Xi	Ci-1	Yi	Ci	Description
0	0	0	0	0	Last two bits are zero
0	0	1	1	0	Last one bit is one
0	1	0	1	0	Any of the bit is one
0	1	1	0	1	Last two bits are one
1	0	0	0	0	Last two bits are zero
1	0	1	1	1	Any one of the bit is zero
1	1	0	1	1	Beginning bit is one
1	1	1	0	1	Last two bits are one

The CSD has 2 advantages features:

- Reduces the critical path by computing it in parallel and it simplifies the algebraic expressions which minimizes the overall hardware implementations
- Simulations are performed with the proposed circuits and it shows high efficiency in speed and area terms in comparison with other previous counterpart architecture[6].

By using CSD multiplier with common sub expression sharing technique , to reduce the area. If area is reduced delay is reduced and also power consumption is also reduced.

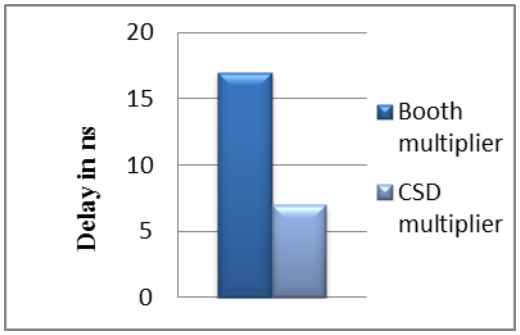


Fig: Delay comparison versus Booth multiplier &CSD multiplier.

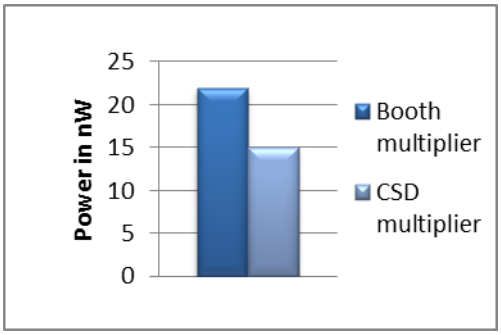


Fig: Power comparison versus Booth multiplier& CSD multiplier.

X.CONCLUSION :

In this paper we have proposed 32point FFT design using Radix 2 algorithm. It was done by using XILINX synthesis tool on vertex kit. Here XILINX ISE design suite is used and also Vertex kit is named as V5XUPLX110t hence the CCM is implemented by CSD Multiplier and CSS technique. In addition, the hardware complexity of the proposed CSD Multiplier used for the reduction of area and the power consumption by approximately 33%.The proposed architecture of expected to be incorporated in SDR.

REFERENCES :

[1] Taesang cho & Hanho Lee, "*A High –speed low-complexity modified radix 25 FFT processor for gigabit WPAN applications*", IEEE Journal vol.11,2011.

[2] Jung- yeol OH and Myoung-seob LIM, "*New Radix 2 to the 4th power pipeline FFT processor*",IEEETrans,Electron,vol E88-C,No.8 August 2005.

[3] S.He and .Torkelson, "*Designing pipeline FFT processor for OFDM (de)modulation*",proc.IEEE URSIInt. Symp. sig. syst, Electron, PP.257-262,1998.

[4] J.Y.oh,J.S.cha,S.K.Kim,and M.S Lim, "*Implementation of orthogonal frequency division Multiplexing using Radix-N Pipeline Fast Fourier Transform Processor*", Jpn,J.Appl.Plug.,Vol.42,Part 1,no.4B,PP.1-6,2003.

[5] K.K.Parthi, "*VLSI Digital Signal Processing Systems*", John Wiley & sons,USA,1999.

[6] K.Sowjanya,B.Leele Kumari, "*Design & Performance Analysis of 32 & 64 point FFT using Radix 2 Algorithm*", AECE-IRAJ international conference,14th July 2013.

[7] S.M.Kim.J.G.Chung,and K.K.Parthi, "*Low error fixed-width CSD multiplier with efficient sign extension*", IEEE Trans,Circuits Syst.I,Vol.50,no.12,PP.984-993,Dec 2003.

[8] G.Zhong et al, "*An Energy-efficient Reconfigurable Angle-Rotator Architecture*", Proc.IEEE ISACS,Vol.3.May 2004,PP.661-664.

[9] S.J.Jou and H.H.Wang, "*Fixed-Width Multiplier for Digital Signal processing Application*", in Proc 2000 Int.Conf.Computer Design (ICCD), Austin,Tx,sept 2000,PP.318-322.

[10]S.Cho and K.Kang, *"A Low Complexity 128-pointmixed Radix FFT Processor for MB-OFDM UWB Systems",* Journal, Vol.32, no.1, PP.1-10, Feb 2010.

[11] D. Harini Sharma and Addanki Purna Ramesh,*" Floating point multiplier using Canonical Signed Digit",* International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Vol. 2, Issue 11, Nov 2013.
[12] Gustavo A.Ruiz , MercedesGranda," *Efficient canonic signed digit recoding*", Microelectronics Journal 42 (2011) 1090–1097.