The Nucleus **51**, No. 3 (2014) 345-353

www.thenucleuspak.org.pk

The Nucleus

ISSN 0029-5698 (Print)

ISSN 2306-6539 (Online)

Area Efficient Radix 4^2 64 Point Pipeline FFT Architecture Using Modified CSD Multiplier

F. Siddiq[1*], H. Jamal[2], T. Muhammad[1] and M. Iqbal[1]

[1]Electrical Engineering Department, University of Engineering and Technology, Taxila, Pakistan

[2]Ghulam Ishaq Khan Institute of Engineering Science and Technology, Topi KPK, Pakistan

(Received March 04, 2014 and accepted in revised form August 18, 2014)

A modified Fast Fourier Transform (FFT) based radix 4^2 algorithm for Orthogonal Frequency Division Multiplexing (OFDM) systems is presented. When compared with similar schemes like Canonic signed digit (CSD) Constant Multiplier, the modified CSD multiplier can provide a improvement of more than 36% in terms of multiplicative complexity. In Comparison of area being occupied the amount of full adders is reduced by 32% and amount of half adders is reduced by 42%. The modified CSD multiplier scheme is implemented on Xilinx ISE 10.1 using Spartan-III XC3S1000 FPGA as a target device. The synthesis results of modified CSD Multiplier on Xilinx show efficient Twiddle Factor ROM Design and effective area reduction in comparison to CSD constant multiplier.

Keywords: FFT, Radix 4^2 , OFDM, CSD multiplier, Xilinx, Twiddle factor, FPGA

1. Introduction

The FFT based processor is commonly used in cellular communication for image and signal processing. The continuous processing makes pipelined FFT the major mechanism for low power applications [1]. In the frequency domain the filtering and correlation can be executed with lesser number of processes [2]. Pipelined mechanism is efficient for lower latency and low power consumption [3]. FFT is used to compute efficient DFT, and find its applications in digital spectrum analysis, ultra wide band and multi-rate filters [4, 5]. The FFT is a set of algorithms which are more computationally efficient than the DFT [6]. Table 1 represents the computational requirements of different FFT architectures as depicted in [7]. In radix 2^4 algorithm complex multiplication is minimized [8]. Novel coefficient ordering based low power radix 4 FFT diminishes the switching activity between consecutive coefficients was presented in [9].

By choosing the input and output carefully, it leads to significant memory and latency savings [10]. In pipeline mechanism radix 4 algorithms is advantageous as compared to radix 2 algorithms [11]. Computational and circuit complexity can be balanced by using radix 4 [12]. FFT processor is categorized as the pipeline mechanism, parallel mechanism and perfect systolic range [13]. Pipeline mechanism gives a trade of hardware complexity and dissension rate. Figure 1 depicts the pipeline architecture for FFT [14]. Figure 2 comprises the quantity of computational units scattered with delay commutator for inter stage data reallocation.

Pipeline FFT Architectures enlisted in Table 2 shows different structures alongwith computational complexities [15].

2. FFT Algorithm

The N pt. FFT algorithm derived from DFT framework is:

$$X(K)=\sum_{n=0}^{N-1} x(n)W_N^{nk} \qquad \text{For k=0, 1……N-1} \qquad (1)$$

The N pt. IFFT is:

$$X(K)=\frac{1}{N}\sum_{n=0}^{N-1} x(n)W_N^{-nk} \qquad \text{For k=0, 1……N-1} \qquad (2)$$

Table 1. Operations required for 64-point FFT.

Operations	Radix-2	Radix-4	Radix-8	Split Radix	Wino grad
Real Additions	1032	976	972	964	1394
Real Multiples	264	208	204	196	198
Total	1296	1174	1176	1160	1592

[*] Corresponding author : faisal.siddiq@uettaxila.edu.pk

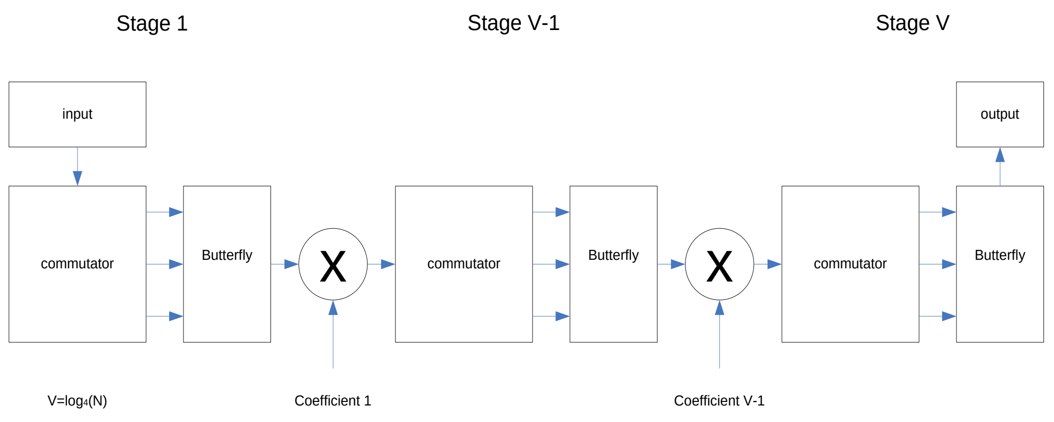


Figure 1. N point radix 4 pipelined FFT processor.

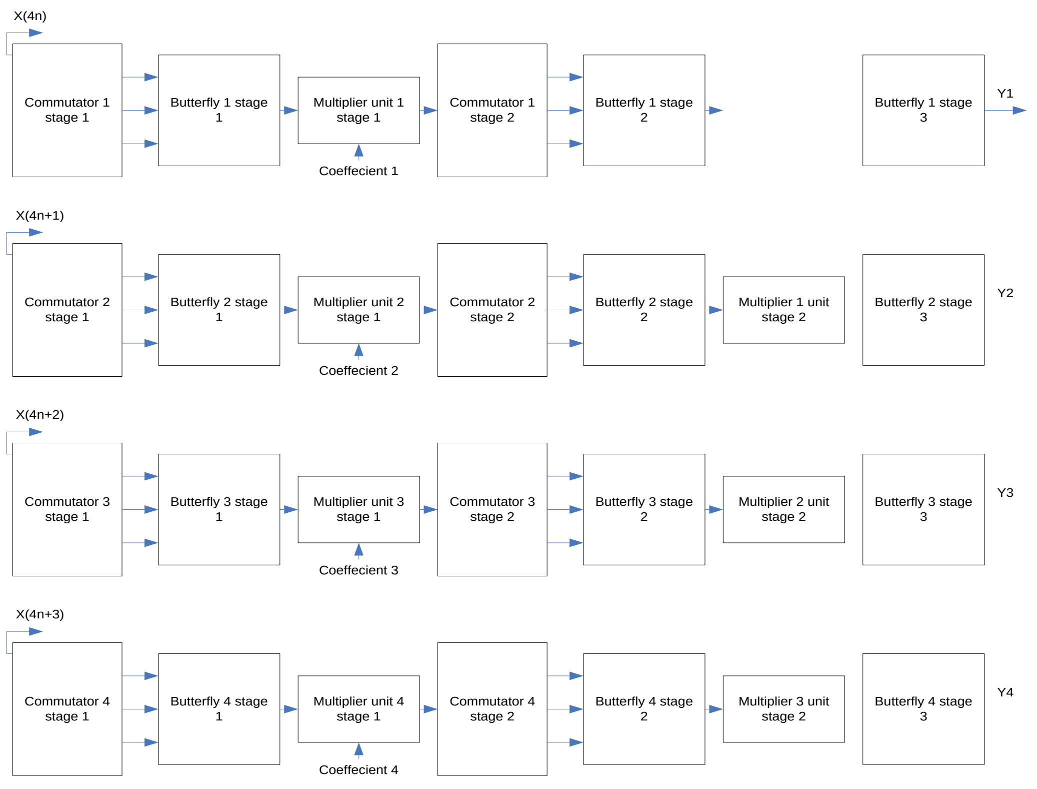


Figure 2. 64 pt. 4 parallel pipeline mechanism.

Table 2. Relationship of the pipeline FFT architectures.

	Complex Multipliers	Complex Adders	Memory	Control
R2MDC	$2\log_4 N - 1$	$4\log_4 N$	$3\frac{N}{2} - 2$	Simple
R2SDF	$2\log_4 N - 1$	$4\log_4 N$	$N - 1$	Simple
R4SDF	$\log_4 N - 1$	$8\log_4 N$	$N - 1$	Medium
R4MDC	$3(\log_4 N - 1)$	$8\log_4 N$	$5\frac{N}{2} - 4$	Simple
R4SDC	$\log_4 N - 1$	$3\log_4 N$	$2N - 2$	Complex
R2^2SDF	$\log_4 N - 1$	$4\log_4 N$	$N - 1$	Simple

Where $$W_N^{nk} = e^{-j(\frac{2\pi}{N})kn} \qquad (3)$$

FFT and IFFT processors have two alterations. FFT has inverted direction of the multiplier. IFFT has different standardization feature of $\frac{1}{N}$ in Eqn. (2). N samples FFT/IFFT computation needs $O(N_2)$ arithmetic operations so lop of chip area is required. By using radix-r fast Fourier formulation, it reduces to $O(\log_2(N))$arithmetic procedures in $\log_r(N)$phases. The ‘r’ can have be 2, 4 or 8 value [15]. There are two decimation approaches, DIF (Decimation in frequency) and DIT (Decimation in Time). Radix-4 DIF based FFT represents the DFT equation as four additions, and also splits it into 4 equalities, and every equality computes every 4th output sample. Sub-sequent equalities shows DIF for radix-4.

$$X(K)=\sum_{n=0}^{N/4-1} x(n)W_N^{nk} \qquad (4)$$

$$X(K)=\sum_{n=0}^{N/4-1} x(n)W_N^{nk} + \sum_{n=\frac{N}{4}}^{\frac{N}{2}-1} x(n)\ W_N^{nk} + \sum_{n=\frac{N}{2}}^{\frac{3N}{4}-1} x(n)\ W_N^{nk} + \sum_{n=\frac{3N}{4}}^{N-1} x(n)\ W_N^{nk} \qquad (5)$$

$$=\sum_{n=0}^{N/4-1} [x(n)+ W_N^{K\left(\frac{n}{4}\right)} x\left(n+\frac{N}{4}\right) + W_N^{K\left(\frac{n}{2}\right)} x\left(n+\frac{N}{2}\right) + W_N^{K(3n/4)}\ x(n+3N/4)]W_N^{nk} \qquad (6)$$

The three twiddle factors are shown:

$$W_N^{K(N/4)}=e^{-j\left(\frac{2\pi}{N}\right)K(N/4)}=e^{-j\left(\frac{\pi}{2}\right)}=(-j)^K \qquad (7)$$

$$W_N^{K(N/2)}=e^{-j\left(\frac{2\pi}{N}\right)K(N/2)}=e^{-j(\pi)}=(-1)^K \qquad (8)$$

$$W_N^{K(3N/4)}=e^{-j\left(\frac{2\pi}{N}\right)K(3N/4)}=e^{-j(3\pi/2)}=(j)^K \qquad (9)$$

Equation (4) can thus be expressed as:

$$X(K)=\sum_{n=0}^{N/4-1}\begin{bmatrix} x(n)+(-j)^K x\left(n+\frac{N}{4}\right)+(-1)^K x \\ \left(n+\frac{N}{2}\right)+(j)^K x\left(n+\frac{3N}{4}\right)\end{bmatrix} W_N^{nk} \qquad (10)$$

Four sub- output sequences can be generating by putting ‘k=4r, k=4r+1, k=4r+2 and k=4r+3’:

$$X(K)=\sum_{n=0}^{\frac{N}{4}-1}\begin{bmatrix} x(n)+x\left(n+\frac{N}{4}\right)+x\left(n+\frac{N}{2}\right)+ \\ x\left(n+\frac{3N}{4}\right) W_N^0\end{bmatrix} W_{\frac{N}{4}}^{nr} \qquad (11)$$

$$X(K)=\sum_{n=0}^{\frac{N}{4}-1}\begin{bmatrix} x(n)-jx\left(n+\frac{N}{4}\right)-x\left(n+\frac{N}{2}\right)+ \\ jx\left(n+\frac{3N}{4}\right) W_N^n\end{bmatrix} W_{\frac{N}{4}}^{nr} \qquad (12)$$

$$X(K)=\sum_{n=0}^{N/4-1} [x(n)-x\left(n+\frac{N}{4}\right)+x\left(n+\frac{N}{2}\right)- x(n+3N/4)W_N^{2n}]W_{N/4}^{nr} \qquad (13)$$

$$X(K)=\sum_{n=0}^{\frac{N}{4}-1}\begin{bmatrix} x(n)+jx\left(n+\frac{N}{4}\right)-x \\ \left(n+\frac{N}{2}\right)_{jx\left(n+\frac{3N}{4}\right)W_N^{3n}}\end{bmatrix} W_{\frac{N}{4}}^{nr} \qquad (14)$$

By putting from r = ‘0’ to ‘N/4–1’:

X (4r), X (4r+1), X (4r+2), and X (4r+3) are N/4-point DFTs. Every N/4 output is a sum of four input samples (x(n) $x\left(n+\frac{N}{4}\right)$, $x\left(n+\frac{N}{2}\right)$ and $x\left(n+\frac{3N}{4}\right)$), all multiplied by -1, j, or +1 –j. Twiddle Factor (W_N^0, W_N^n, W_N^{2n} or W_N^{3n}) is multiplied by above sum. Every N/4-sample DFTs is divided into four N/16-sample DFTs. Every N/16 DFT is distributed further in four N/64-Pt.and so on.. A basic radix-4 Processing element (Butterfly) is represented in Figure 3.

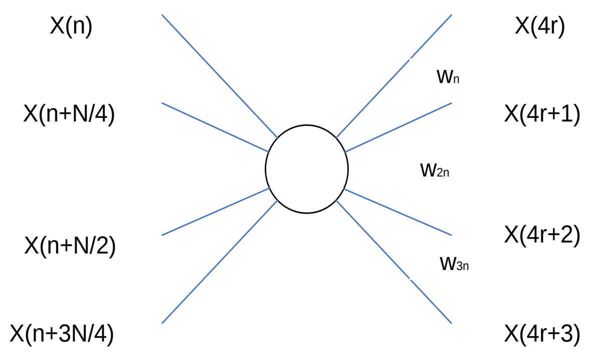


Figure 3. Radix-4 DIF FFT butterfly.

Every sample is complex in processing element (Butterfly unit). A basic Processing Element graph between inputs and outputs is depicting in Figure 4. Real and imaginary parts of twiddle factor can be calculated as follows:

$$W_N= e^{-j\left(\frac{2\pi}{N}\right)}=\cos\left(\frac{2\pi}{N}\right)-j\sin\left(\frac{2\pi}{N}\right).$$

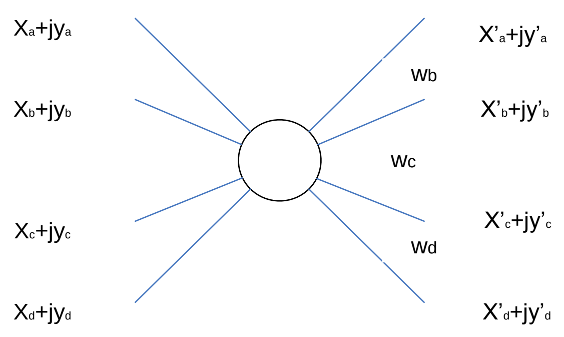


Figure 4. Radix-4 DIF FFT Butterfly, Complex data.

Table 3. The second stage twiddle factors.

k1,k2 n3,n4	0,0	0,1	1,0	1,1
0,0	1	1	1	1
0,1	1	W_{16}^{2}	W_{16}^{1}	W_{16}^{3}
1,0	1	$W_{16}^{2} = -j$	W_{16}^{2}	$W_{16}^{6} = -jW_{16}^{2}$
1,1	1	$W_{16}^{6} = -jW_{16}^{2}$	W_{16}^{3}	$W_{16}^{9} = -W_{16}^{1}$

Table 4. Modified CSD binary illustration ($\bar{1}$ means -1).

Coefficients		Dec.	2's Complement	CSD	Modified CSD
mo	Cos(pi/8)	0.9239	011101100100	100010100100	(0.3827 + 0.0793)*2
m1	Sin(pi/8)	0.3827	001100001111	010100010001	(0.5-0.1173) = (010000000000) − (000100010000)

3. Radix-4^{2} Algorithm

Using a 3-D linear index mapping of radix-4 FFT, the first two stages in cascade decomposition can be equated as:

$$\text{n}=\frac{N}{4}\text{n1}+\frac{N}{16}\text{n2}+\text{n3} \quad \{0\leq \text{n1, n2}\leq 3,\ \text{n3}=0\sim\frac{N}{16}-1\} \qquad (15)$$

$$\text{k1}+4\text{k2}+16\text{k3} \quad \{0\leq \text{k1, k2}\leq 3,\ \text{k3}=0\sim\frac{N}{16}-1\}$$

The DFT equations are:

$$x(\text{k1}+4\text{k2}+16\text{k3}) \qquad (16)$$

$$=\sum_{\text{n3}=0}^{\frac{N}{16}-1}\sum_{\text{n2}=0}^{3}.\sum_{\text{n1}=0}^{3}.\,x(\frac{N}{4}\text{n1}+\frac{N}{16}\text{n2}+\text{n3})$$

$$W_{N}^{\left(\frac{N}{4}\text{n1}+\frac{N}{16}\text{n2}+\text{n3}\right)(\text{k1}+4\text{k2}+16\text{k3})}$$

$$=\sum_{\text{n3}=0}^{\frac{N}{16}-1}.\ \sum_{\text{n2}=0}^{3} B_{\frac{N}{4}}^{\text{k1}}\ \frac{N}{16}\text{n2}+\text{n3})\ W_{16}^{(\text{n2})(\text{k1}+4\text{k2})}\}$$

$$W_{N}^{(\text{n3})(\text{k1}+4\text{k2})}\ W_{\frac{N}{16}}^{(\text{n3})(\text{k3})}$$

$$=\sum_{\text{n3}=0}^{\frac{N}{16}-1} H_{\frac{N}{16}}^{\text{k1k2}}(\text{n3})\ (\ W_{16}^{(\text{n3})(\text{k1}+4\text{k2})}\}\ W_{\frac{N}{16}}^{(\text{n3})(\text{k3})}$$

The first butterfly stage is:

$$\Delta=\frac{N}{16}\text{n2}+\text{n3} \qquad (17)$$

$$B_{\frac{N}{4}}^{\text{k1}}(\Delta)=x(\Delta)+(-j)^{\text{k1}}x\left(\Delta+\frac{N}{4}\right)+(-1)^{\text{k1}}x\left(\Delta+\frac{N}{2}\right)+$$

$$(j)^{\text{k1}}x\left(\Delta+\frac{3N}{4}\right)$$

The second butterfly has the structure, $H_{\frac{N}{16}}^{\text{k1k2}}(\text{n3})$ is expressed as

$$H_{\frac{N}{16}}^{\text{k1k2}}(\text{n3})=B_{\frac{N}{4}}^{\text{k1}}(\text{n3})+(-j)^{\text{k2}}W_{16}^{\text{k1}}\ B_{\frac{N}{4}}^{\text{k1}}\left(\text{n3}+\frac{N}{16}\right)+$$

$$(-1)^{\text{k2}}W_{16}^{2\text{k1}}\ B_{\frac{N}{4}}^{\text{k1}}\left(\text{n3}+\frac{2N}{16}\right)+(j)^{\text{k2}}W_{16}^{3\text{k1}}B_{\frac{N}{4}}^{\text{k1}}\left(\text{n3}+\frac{3N}{16}\right) \qquad (18)$$

Twiddle factor $W_{16}^{\text{n3}(\text{k1}+4\text{k2})}$ decomposition is performed by fully dedicated multiplications. In Eq. (16). The Eq. (18) depicts the first processing element structure with twiddle factor $W_{16}^{\text{ik1}}(\text{i}=0,1,2,3)$ multiplication.

W_{16}^{0} show the value of "1". W_{16}^{1} have value of cos (pi/8-jsin (pi/8). This demands a composite multiplier which can be realized with shift and add operations. W_{16}^{2} has value of Cos (pi/4-jsin (pi/4). Twiddle factor can be obtained by using given trigonometric function:

Cos (pi/4) =sin (pi/4) =2sin (pi/8) cos (pi/8).

W_{16}^{3} can be written as sin (pi/8)-j cos (pi/8) and executed by inverting real and imaginary parts of composite Multiplier W_{16}^{1}. CSD representation is used for area and power consumption to about 33% [17-18]. Twiddle factor W_{16}^{1} in Table 4 showing the 12-bit

Table 5. Constant multiplier results by control signals.

S(1)	S(0)	O_1	O_2	O_3	O_4	$O_{re} + jO_{Im}$
0	0	xm_0	xm_1	ym_0	ym_1	$(x+jy)W_{16}^1$
0	1	$2xm_0m_1$	$2xm_0m_1$	$2ym_0m_1$	$2ym_0m_1$	$(x+jy)W_{16}^2$
1	0	xm_0	xm_1	ym_0	ym_1	$(x+jy)W_{16}^3$
1	1			Bypass mode		$(x+jy)W_{16}^0$

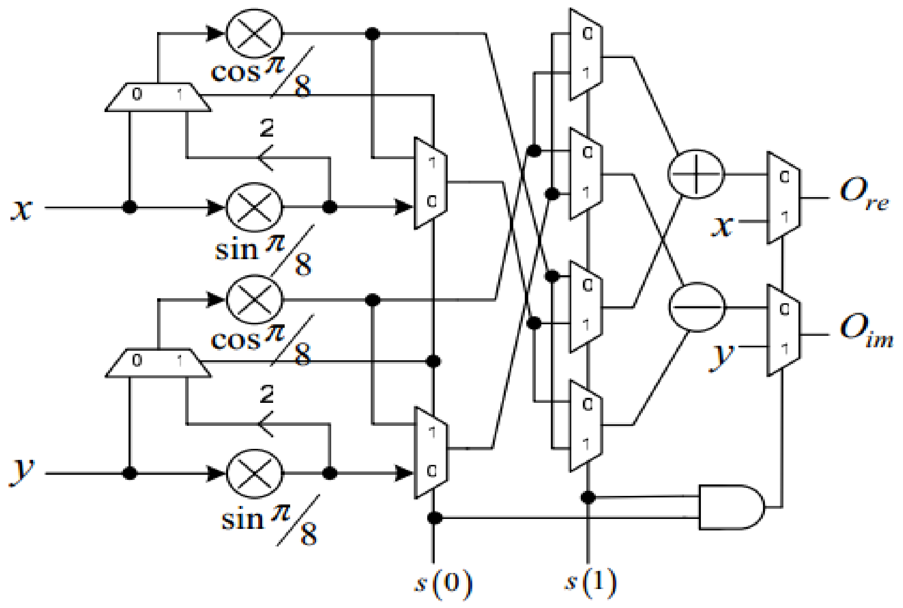


Figure 5. Proposed Modified CSD multiplier.

coefficients for both Modified CSD format and 2's complement. Here x+jy is input, the Modified CSD multiplier for the twiddle factors W_{16}^i is represented in Figure 5. Table 5 representing output product for the Modified CSD multiplier which can be produced by control signals (S).

4. R2^{i}SDF and R4^{i}SDC Pipeline Architecture

Figures 6 and 7 represent the R2^{i}SDF pipeline approach and R4^{i}SDC pipeline approach when samples are 256. The sign, $\otimes$, characterizes the programmable multiplier and the mark, $\odot$, characterizes the Modified CSD multiplier. Figure 6(a) shows R2^{2}SDF where the programmable memory for twiddle factor is allotted. Figure 6(b) shows R2^{3}SDF in which 2 programmable multipliers along with two fix valued multipliers are allotted. One fix valued multiplier is almost equal to 0.4 programmable multipliers [18-20]. Figure 6(c) depicts the Modified R2^{4}SDF architecture. Two modified CSD multipliers replace the programmable multiplier that is used in radix 2^{2}SDF. Figure 7(a) depicts the R4SDC in which the programmable multiplier and memory for the twiddle factor is allotted for every column. Figure 7(b) depicts the Modified R4^{2}SDC in which two of the proposed CSD multipliers in the place of programmable multipliers.

5. Implementation Example

Both real as well as imaginary parts of 12 bit multipliers are implemented. Out of these multipliers is the CSD complex one having lesser number of partial products [17]. The second one is modified CSD multiplier. For the compensation of quantization error, less error constant width CSD multiplier [21] together with reduced error constant width Booth multiplier [22] is presented. When the input signal is 'X', the designed modified CSD multipliers have m_0 with m_1 CSD coefficients as presented in Table 4. The implemented modified CSD multiplier is shown in Figure 8. Figure 8(a) depicts sin (pi/8); Figure 8 (b) depicts the model of 0.0793 that is used in the implementation of cos (pi/8) and similarly figure 8 (c) depicts cos (pi/8).

The comparison of full adders and half adders of modified Constant complex Multiplier and Constant Complex Multiplier is shown in Figure 9. When compared with similar schemes like CSD Multiplier [17], the modified CSD multiplier can provide improvement of more than 36% in terms of multiplicative complexity. In terms of area occupied amount of Full adders is reduced approximately by 32% and amount of half adders is reduced by 42% when compared with [17]. The synthesis of Modified CSD Multiplier on Xilinx shows the efficient Twiddle Factor ROM Design in Table 6.

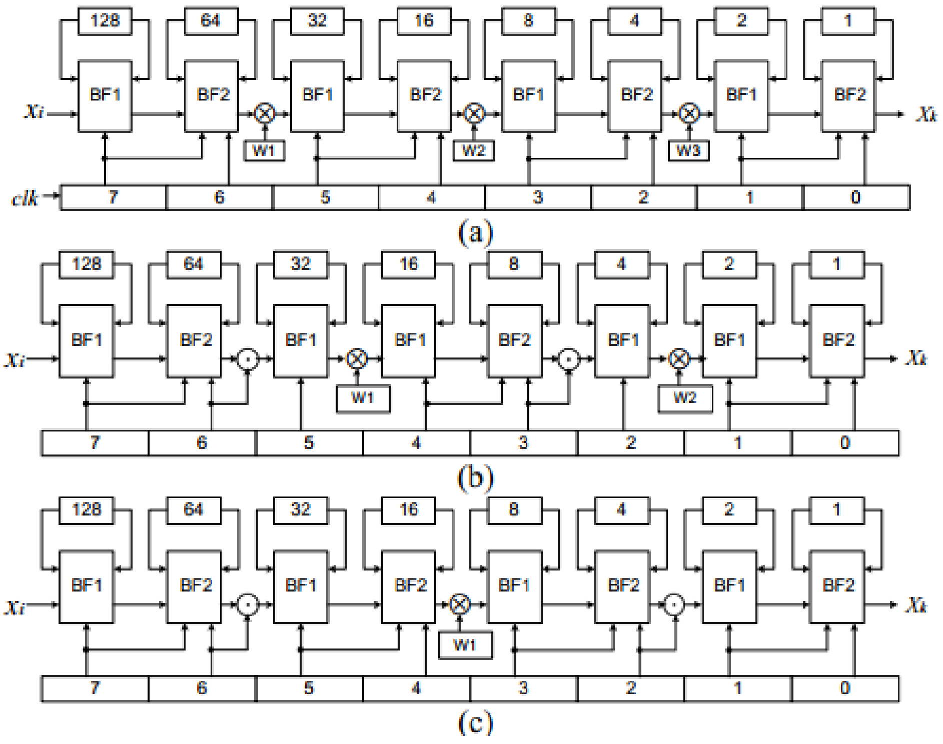


Figure 6. Radix-2i SDF 256 point pipeline FFT (a) R2^{2}SDF (b) R23SDF(c) R24SDF.

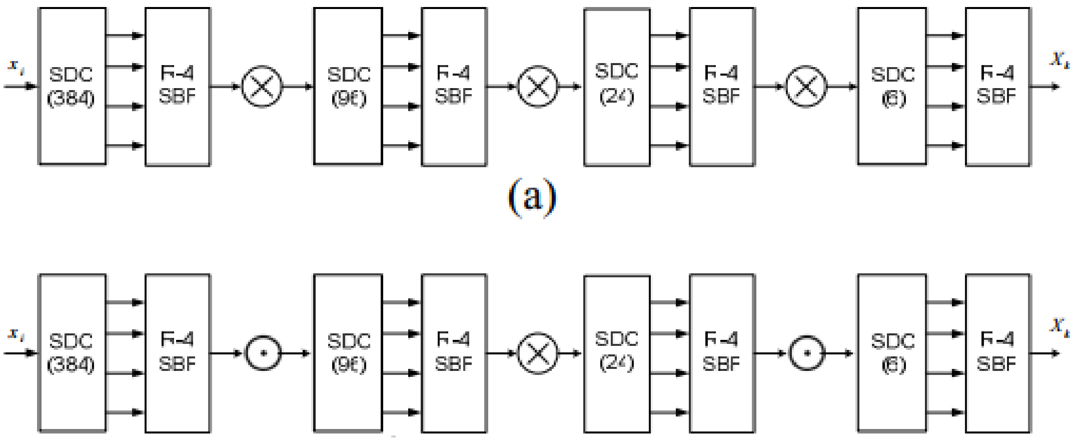
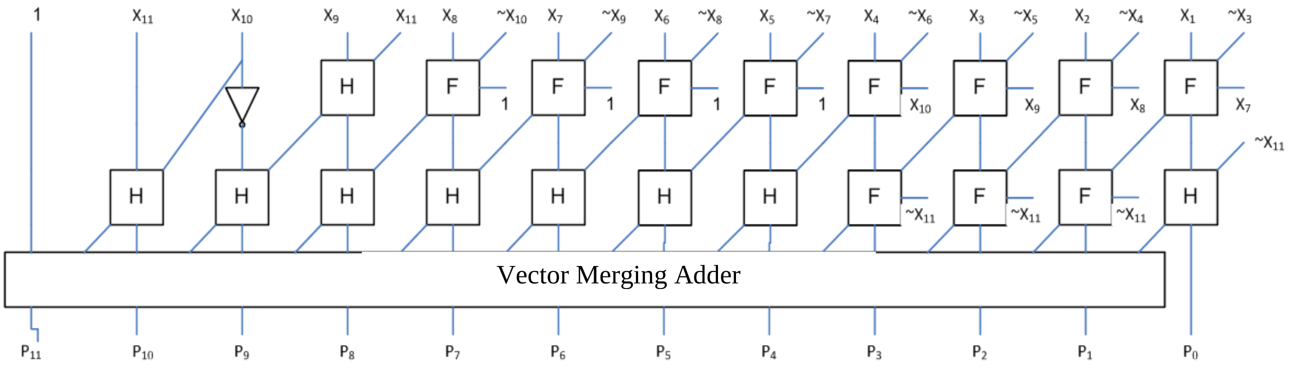
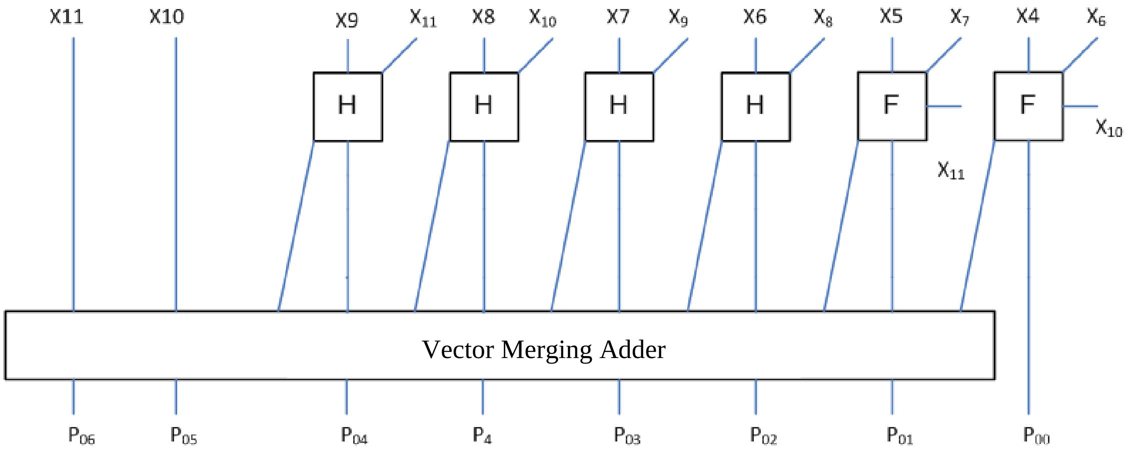


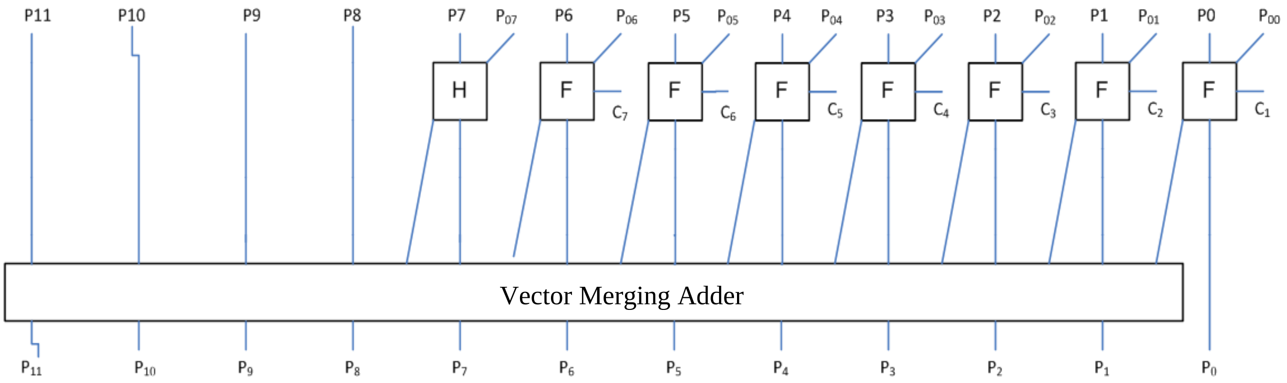
Figure 7. Radix-4^{i}SDC 256 point pipeline FFT (a) R4SDC (b) R42SDC.



(a)

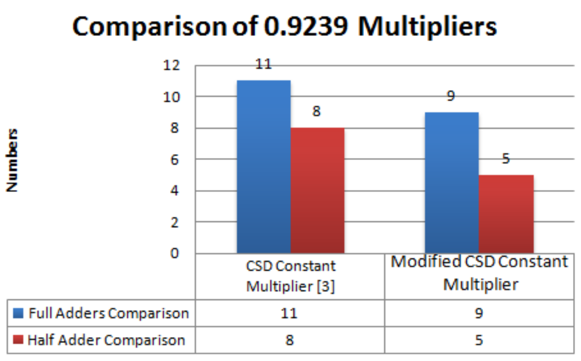


(b)



(c)

Figure 8 (a) sin (pi/8) architecture, (b) 0.0793 architecture and (c) cos (pi/8) architecture.



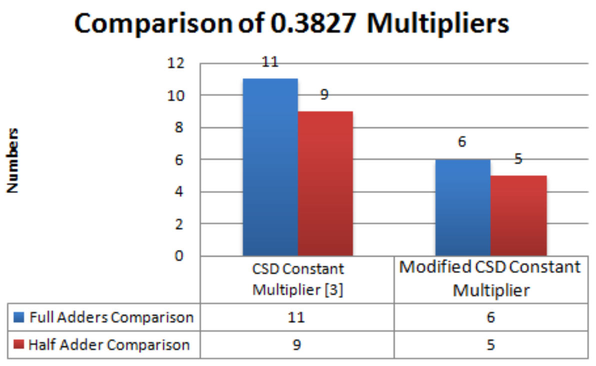


Figure 9. Comparison of full adders and half adders.

Table 6. Synthesis results of proposed modified CSD constant complex multiplier.

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
No. of Slices	7	7680	0%
No. of 4 inputs LUTs	13	15360	0%
No. of bounded IOBs	24	173	13%

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
No. of Slices	6	7680	0%
No. of 4 inputs LUTs	11	15360	0%
No. of bounded IOBs	24	173	13%

6. Conclusion

The proposed pipeline architecture allows 50 percent of total multipliers to be swapped by CSD multipliers. Synthesis Results of modified CSD Multiplier shows the Efficient Twiddle Factor Design. The modified CSD multiplier scheme is implemented on Xilinx ISE 10.1 using Spartan-III XC3S1000 FPGA as a target device. When compared with similar schemes like CSD Multiplier, the modified CSD multiplier can provide a reduction of more than 36% in terms of multiplicative complexity. In terms of area occupied amount of Full adders is reduced approximately by 32% and amount of half adders is reduced by 42%. Using this technique, long length FFT processor can be developed for wireless Applications.

References

[1] W. Han, A. T. Erdogan, T. Arslan and M. Hasan, ETRI Journal **30** (2008) 451.

[2] J. G. Proakis and D. G. Manolakis, Digital Signal Processing, Prentice Hall of India Private Limited (2003).

[3] A. Saeed, M.Elbably, G. Abdelfadeel and M.I. Eladawy, Int. J. of Circuits, Systems and Signal Processing **3** (2009) 103.

[4] K. Harikrishna, T.R. Rao and. V.A. Labay, An Efficient FFT Architecture for OFDM Communication Systems, Asia Pacific Microwave Conference (APMC), Singapore, 7-10 Dec. (2009) 449.

[5] U. Rashid, F. Siddiq, T. Muhammad and H. Jamal, The Nucleus **50** (2013) 301.

[6] J.W. Cooley and J.W. Tukey, Math. Computation **19** (1965) 297.

[7] Chi-hau Chen, Signal Processing Handbook, CRC Press (1988).

[8] L. Jia, Y. Gao, J. Isoaho and H. Tenhunen, A New VLSI Oriented FFT Algorithm and Implementation, Proceedings of

Eleventh Annual IEEE International ASIC Conference (1998) p. 337.

[9] M. Hasan, T. Arslan and J.S. Thompson, IEEE Transaction on Consumer Electronics **49** (2003) 128.

[10] User Guide “FFT MegaCore Function,” Version 8.1, Altera Corporation. Available: http://www. Altera.com, Nov. (2008).

[11] E.E. Swartzlander, VLSI Signal Processing Systems, Kluwer Academic Publishers (1998).

[12] Amphion.CS246064-Point Pipelined FFT/IFFT; Available from: http://www.datasheetarchive.com/ 64-Point-datasheet. html (2002).

[13] Y. Jung; H. Yoon and J. Kim, IEEE Transactions on Consumer Electronics **49** (2003) 14.

[14] W. Han, T. Arslan, A. T. Erdogan and M. Hasan, Proc. IEEE Int. Conf. on Acoustics Speech and Signal Processing **5** (2005) 45.

[15] S. He and M. Torkelson, Designing Pipeline FFT Processor for OFDM (de) Modulation, Proc. IEEE URSI Int. Symp. Sig. Syst. Electron (1998) 257.

[16] L. Jia, Y. Gao, Jouni and H. Tenhunen, A New VLSI-oriented FFT Algorithm and Implementation, IEEE International ASIC Conf.(1998) 337.

[17] Jung-yeol Oh and Myoung-Seob Lim, Area and Power Efficient Pipeline FFT Algorithm, IEEE Workshop on Signal Processing Systems Design and Implementation (2005) 520.

[18] 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 (FFT) Processor, Jpn. J. Appl. Phys. **42** (2003) 1.

[19] K.K. Parhi, VLSI Digital Signal Processing Systems, John Wiley & Sons, Inc., USA (1999).

[20] W. C. Yey and C. W. Jen, IEEE Trans. Sig. Proc. **51** (2003) 864.

[21] S. M. Kim, J. G. Chung and K. K. Parhi, IEEE Int. Symp. Cir. Syst. (2002) 69.

[22] K.J. Cho, K.C. Lee, J.G. Chung and K.K. Parhi, IEEE Trans. VLSI Syst.**12** (2004) 90.