한국정보전자통신기술학회논문지(jkiiect)'18-10, Vol.11 No.5
논문 18-11-05-475 http://dx.doi.org/10.17661/jkiiect.2018.11.5.475

A Low-area and Low-power 512-point Pipelined FFT Design Using Radix-2^4-2^3 for OFDM Applications

Jian Yu*, Kyung-Ju Cho**

Abstract In OFDM-based systems, FFT is a critical component since it occupies large area and consumes more power. In this paper, we present a low hardware-cost and low power 512-point pipelined FFT design method for OFDM applications. To reduce the number of twiddle factors and to choose simple design architecture, the radix-2^4-2^3 algorithm are exploited. For twiddle factor multiplication, we propose a new canonical signed digit (CSD) complex multiplier design method to minimize the hardware-cost. In hardware implementation with Intel FPGA, the proposed FFT design achieves more than about 28% reduction in gate count and 18% reduction in power consumption compared to the previous approaches.

Key Words : low-power, pipelined, FFT, OFDM, constant complex multiplier

1. Introduction

FFT processor is one of the components with high complexity in the physical layer of OFDM-based applications such as IEEE 802.11a/g/n, WPAN, LTE systems, and so on. Thus, many FFT design approaches have been developed to reduce the computational complexity [1][2].

The radix-2 algorithm is popular due to simple butterfly for implementation. However it needs more complex multiplications. The radix-4 algorithm can reduce the number of complex multiplications following higher butterfly complexity for implementation. For achieving a simple butterfly and reducing the number of twiddle factor multiplication, radix-2^k (k=2~5) FFT algorithms have been proposed in [3-5].

Among the various FFT architectures, the pipelined architectures provide high throughputs at the cost of reasonable hardware overhead. There are two types in the pipelined FFT architecture: feedforward and feedback. Feedforward architectures can be classified into single path delay commutator and multi-path delay commutator. Feedback architectures can be classified into single path delay feedback (SDF) and multi-path delay feedback [6].

For twiddle factor multiplication in [2], CSD multipliers are adopted to efficiently design the complex multipliers composed of four multiplications and two additions.

In this paper, we present a low hardware-cost and low power 512-point FFT with radix-2^4-2^3 SDF architecture. To reduce the hardware-cost for twiddle factor multiplication, we propose new CSD complex multipliers which provide removal of ROM to store twiddle factors.

2. Design Issues of 512-point FFT

The discrete Fourier transform $X(k)$ of an N-point input signal $x(n)$ is defined as

This paper was supported by Wonkwang University in 2018.
*Department of Electronic Engineering Wonkwang University
**Corresponding Author: Department of Electronic Engineering Wonkwang University (kjcho@wku.ac.kr)
Received October 11, 2018 Revised October 12, 2018 Accepted October 19, 2018

$$X(k)=\sum_{n=0}^{N-1}x(n)\,W_N^{nk},\quad 0\le k\le N-1, \qquad (1)$$

where the twiddle factor $W_N^{nk}=e^{-j2\pi nk/N}$.

For the computation of (1), large hardware resources and computation time are required. To overcome the shortcoming, radix-2^k algorithms with a reduced number of complex multiplication have been presented.

The 512-point FFT computation with radix-2^k algorithms is composed of nine stages. Table 1 shows the sequence of twiddle factors at each stage for $N=512$ and radix-2^k algorithms, where $-j$ means trivial multiplication and '#CM' denotes the required number of complex multiplications excluding $-j$.

The radix-2^5 and radix-2^4-2^3 algorithms have less number of complex multiplications and simpler twiddle factors compared to the others. The radix-2^4-2^3 algorithm is simpler than radix-2^5 algorithm in butterfly control. Thus, the radix-2^4-2^3 algorithm is optimal candidate for the FFT design.

Among the various pipelined FFT architectures, we adopt SDF approach based on radix-2^k algorithm for its low cost and high efficiency [1].

Table 1. Base number of twiddle factor

Algorithms	Stages								#CM
	1	2	3	4	5	6	7	8	
radix-2^2	$-j$	W_{512}	$-j$	W_{128}	$-j$	W_{32}	$-j$	W_8	1196
radix-2^3	$-j$	W_8	W_{512}	$-j$	W_8	W_{64}	$-j$	W_8	1208
radix-2^4	$-j$	W_{16}	$-j$	W_{512}	$-j$	W_{16}	$-j$	W_{32}	1200
radix-2^5	$-j$	W_8	W_{32}	$-j$	W_{512}	$-j$	W_{16}	$-j$	1168
radix-2^4-2^3	$-j$	W_{16}	$-j$	W_{512}	$-j$	W_8	W_{32}	$-j$	1168

Fig. 1. 512-point radix-2^4-2^3 SDF FFT architecture.

Fig. 1 shows the proposed architecture of the radix-2^4-2^3 512-point SDF FFT. In order to obtain proper data at the butterfly input, two types of butterfly (BF1 and BF2 in [3]) and several delay buffers with different sizes are used for data shuffling in Fig. 1. Control signal (ctrl) is used for switching the butterfly types. Also, it provides a proper control for multiplication of twiddle factor.

3. Proposed FFT Design

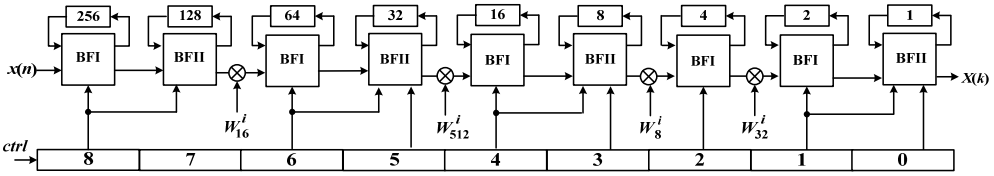
3.1 Proposed CSD complex multiplier for W_8^i, W_{16}^i and W_{32}^i

In order to design constant complex multiplier with twiddle factors W_8^i, W_{16}^i and W_{32}^i, we first find out the required constant values for these twiddle factors.

Twiddle factors W_8^i at stage 6 only need 4 factors (i=0~3). By using $W_N^{N/4}=-j$ and the symmetry property of complex sinusoidal function, only one twiddle factor W_8^1 is required. In the twiddle factor, a constant $Re\{W_8^1\}$ is needed since $Re\{W_8^1\}=Im\{W_8^1\}$, where $Re\{t\}$ and $Im\{t\}$ denotes the real part and imaginary part of t, respectively.

There are 7 twiddle factors for W_{16}^i (i= 0~4, 6, 9) at stage 2. Applying rules similar to W_8^i, these twiddle factors can be explained by using only three constant values $Re\{W_{16}^1\}$, $Re\{W_{16}^2\}$ and $Re\{W_{16}^3\}$.

Twiddle factors W_{32}^i at stage 7 need 16 factors (i=0~10, 12, 14, 15, 18, 21). By the same, the factors can be expressed by 7 values of $Re\{W_{32}^1\}$,



$Re\{W_{32}^2\}$, $Re\{W_{32}^3\}$, $Re\{W_{32}^4\}$, $Re\{W_{32}^5\}$, $Re\{W_{32}^6\}$, and $Re\{W_{32}^7\}$ as shown in Table 2.

In order to efficiently design constant multiplication, CSD representation and common sub-expression (CSE) sharing algorithm in [8] are adopted. Table 3 shows the CSD representations of 7 coefficients. The CSE '101' (or -10-1) lies in the red solid line. Also, CSE '10-1' (or -101) and '1000-1' (or -10001) lie in the blue dashed line and purple dotted line, respectively. The 7 CSD multipliers can be obtained by using 16 shifters and 13 additions as

$$
\begin{aligned}
&CSE1 = d + d \gg 2 \\
&CSE2 = d - d \gg 2 \\
&CSE3 = d - d \gg 4 \\
&d \times Re\{W_{32}^1\} = d - CSE \gg 6 \\
&d \times Re\{W_{32}^2\} = d - CSE \gg 4 + d \gg 9 \\
&d \times Re\{W_{32}^3\} = CSE2 + CSE \gg 4 + CSE2 \gg 8 \\
&d \times Re\{W_{32}^4\} = CSE2 - d \gg 4 + CSE1 \gg 6 \\
&d \times Re\{W_{32}^5\} = d \gg 1 + d \gg 4 - CSE3 \gg 7 \\
&d \times Re\{W_{32}^6\} = CSE2 \gg 1 + CSE3 \gg 7 \\
&d \times Re\{W_{32}^7\} = CSE2 \gg 2 + CSE3 \gg 7
\end{aligned}
\tag{2}
$$

where, d and $\gg t$ stands for the multiplicand for twiddle factors and the right-shift operation by t, respectively. Note that the CSD constant complex multiplier is only consist of adders, shifters and multiplexers with lower hardware resources compared to general complex multiplier.

Table 2. Representation of W_{32}^i

W_{32}^0	1	W_{16}^8	$-j$
W_{32}^1	$Re\{W_{32}^1\} - jRe\{W_{32}^7\}$	W_{16}^9	$-Re\{W_{32}^7\} - jRe\{W_{32}^1\}$
W_{12}^2	$Re\{W_{32}^2\} - jRe\{W_{32}^6\}$	W_{16}^{10}	$-Re\{W_{32}^6\} - jRe\{W_{32}^2\}$
W_{12}^3	$Re\{W_{32}^3\} - jRe\{W_{32}^5\}$	W_{16}^{12}	$-Re\{W_{32}^4\} - jRe\{W_{32}^4\}$
W_{12}^4	$Re\{W_{32}^4\} - jRe\{W_{32}^4\}$	W_{16}^{14}	$-Re\{W_{32}^2\} - jRe\{W_{32}^6\}$
W_{12}^5	$Re\{W_{32}^5\} - jRe\{W_{32}^3\}$	$W15$	$-Re\{W_{32}^1\} - jRe\{W_{32}^7\}$
W_{32}^6	$Re\{W_{32}^6\} - jRe\{W_{32}^2\}$	W_{16}^{18}	$-Re\{W_{32}^2\} + jRe\{W_{32}^6\}$
W_{32}^7	$Re\{W_{32}^7\} - jRe\{W_{32}^1\}$	W_{16}^{21}	$-Re\{W_{32}^5\} + jRe\{W_{32}^3\}$

Table 3. CSD representation for W_{32}^i with 12 bits

$Re(W_{32}^1)$	1	0	0	0	0	0	-1	0	-1	0	0	0
$Re(W_{32}^2)$	1	0	0	0	-1	0	-1	0	0	1	0	0
$Re(W_{32}^3)$	1	0	-1	0	1	0	1	0	1	0	-1	0
$Re(W_{32}^4)$	1	0	-1	0	-1	0	1	0	1	0	0	0
$Re(W_{32}^5)$	0	1	0	0	1	0	0	-1	0	0	0	1
$Re(W_{32}^6)$	0	1	0	-1	0	0	0	1	0	0	0	-1
$Re(W_{32}^7)$	0	0	1	0	-1	0	0	1	0	0	0	-1

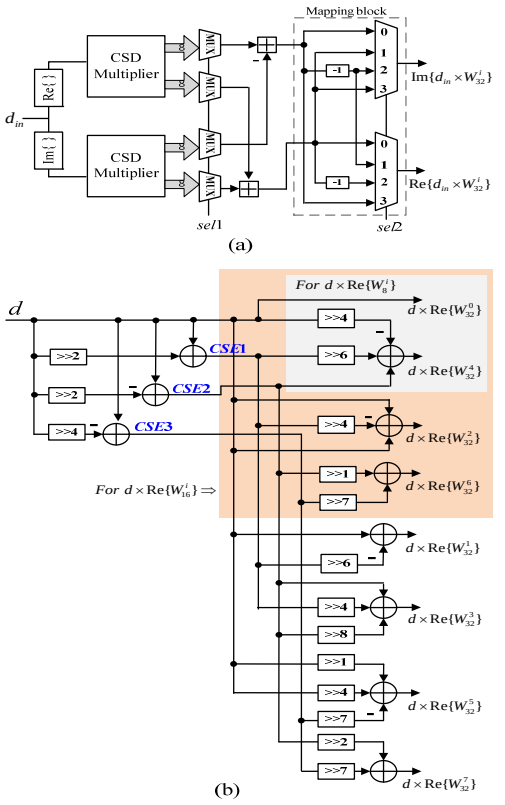


Fig. 2. Proposed CSD constant multipliers for W_{32}^i.

Fig. 2 (a) shows the proposed CSD complex multiplier structure for W_{32}^i. The detailed structure of CSD multipliers is shown in Fig. 2 (b). To select the proper twiddle factor multiplication result in Table 2, two type multiplexers are needed by two signals (sel1 and sel2). Note that we can design CSD complex multipliers for W_8^i and W_{16}^i using

$Re\{W_{32}^2\}$, $Re\{W_{32}^4\}$, and $Re\{W_{32}^6\}$.

3.2 Cascade CSD complex multiplier W_{512}^i

As shown in Fig. 1, the output signals at stage 4 are multiplied by proper twiddle factors W_{512}^i $(i=0\sim511)$. The complexity of CSD complex multiplier increases as the base number of twiddle factor increases. Thus, the approach described above is not practical for W_{512}^i. To reduce the required number of constants for W_{512}^i, we utilize 1/8 symmetry property and decomposition of twiddle factor [2]. The proposed CSD complex multiplier design procedure is as follows

1. Divide the i of W_{512}^i into 8 regions($k=0\sim64$) using 1/8 symmetry property.
2. Decompose the k into $8i_1(i_1=0\sim8)$ and i_2 $(i_2=0\sim7)$ as $W_{512}^k=W_{512}^{8i_1+i_2}=W_{512}^{8i_1}W_{512}^{i_2}$.
3. Make CSD coefficient table for $W_{512}^{8i_1}$ and $W_{512}^{i_2}$, and find the optimized CSE.

By applying the procedure, the required number of constant values can be reduced to 16. Table 4 shows CSD representation of the 16 different values. The CSE '101' (or -10-1) and '10-1' (or -101) are in the red solid line and blue dashed line, respectively. Also, the CSE '1001' (or -100-1), '100-1' (or -1001) and '1000-1' lie in the yellow dash-double dotted line, green dash-single dotted line and purple dotted line, respectively.

Table 4. CSD representation of 16 values

i_1	$Re\{W_{512}^{8i_1}\}$											
1	1	0	0	0	0	0	0	0	-1	0	-1	0
2	1	0	0	0	0	0	-1	0	-1	0	0	0
3	1	0	0	0	-1	0	1	0	1	0	0	0
4	1	0	0	0	-1	0	-1	0	0	1	0	0
5	1	0	0	-1	0	0	0	1	0	0	-1	0
6	1	0	-1	0	1	0	1	0	1	0	0	-1
7	1	0	-1	0	0	1	0	-1	0	0	0	-1
8	1	0	-1	0	-1	0	1	0	1	0	0	0
i_2	$Re\{W_{512}^{i_2}\}$											
0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	-1	0
4	1	0	0	0	0	0	0	0	0	0	-1	0
5	1	0	0	0	0	0	0	0	0	-1	0	0
6	1	0	0	0	0	0	0	0	-1	0	1	0
7	1	0	0	0	0	0	0	0	-1	0	0	0
i_1	$-Im\{W_{512}^{8i_1}\}$											
1	0	0	0	1	0	-1	0	0	1	0	0	1
2	0	0	1	0	-1	0	0	1	0	0	0	0
3	0	0	1	0	0	1	0	1	0	0	1	0
4	0	1	0	-1	0	0	0	1	0	0	0	0
5	0	1	0	0	0	-1	0	0	1	0	-1	0
6	0	1	0	0	1	0	0	-1	0	0	1	0
7	0	1	0	1	0	0	0	1	0	1	0	-1
8	1	0	-1	0	-1	0	1	0	1	0	0	0
i_2	$-Im\{W_{512}^{i_2}\}$											
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	-1	0	0	1
2	0	0	0	0	0	1	0	-1	0	0	1	0
3	0	0	0	0	0	1	0	1	0	-1	0	0
4	0	0	0	0	1	0	-1	0	0	1	0	0
5	0	0	0	0	1	0	0	0	0	0	-1	0
6	0	0	0	0	1	0	1	0	-1	0	-1	0
7	0	0	0	1	0	-1	0	-1	0	0	0	0

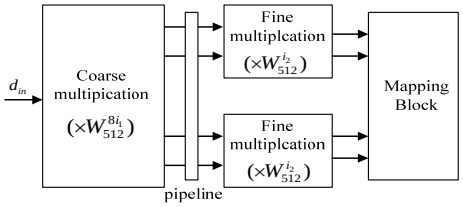


Fig. 3. Proposed CSD multiplier structure.

Fig. 3 shows the proposed cascade CSD complex multiplier for W_{512}^{i} which is composed of coarse and fine multiplication. The pipeline technique can be used to reduce the critical path. The detailed architecture of corse and fine multiplier is shown in Fig. 4.

4. Results and Comparison

The proposed and previous approaches for 512-point FFT were designed using Verilog HDL. These designs are synthesized based on Intel Cyclone 10LP FPGA using QUARTUS PRIME design tool. The input and output word-length are 12-bit and 25-bit, respectively.

Approach in [7] employs the CSD complex multiplier for W_{16}^{i} and, conventional complex multipliers (CCMs) for W_{32}^{i} and W_{512}^{i}. Also, approach in [5] employs CSD complex multipliers for W_{8}^{i}, W_{16}^{i} and W_{32}^{i}, and CCM for W_{512}^{i}. To implement CCM, 4 modified Booth multiplier and 2 ripple carry adder are used. In the proposed approach, only CSD complex multipliers are used to implement twiddle factor multiplication. It provides elimination of ROM to store the twiddle factors.

The performance comparison of the proposed approach and previous approaches is summarized in Table 5. Note that the proposed design approach achieves 28% gate count reduction and 34% memory reduction compared to radix-2^4approach. In addition, the proposed approach is 18% less than radix-2^4 approach in power consumption.

5. Conclusion

We proposed a hardware efficient and low-power 512-point pipelined FFT with radix-2^4-2^3 algorithm. To reduce the hardware-cost and power consumption, we proposed the CSD complex multi

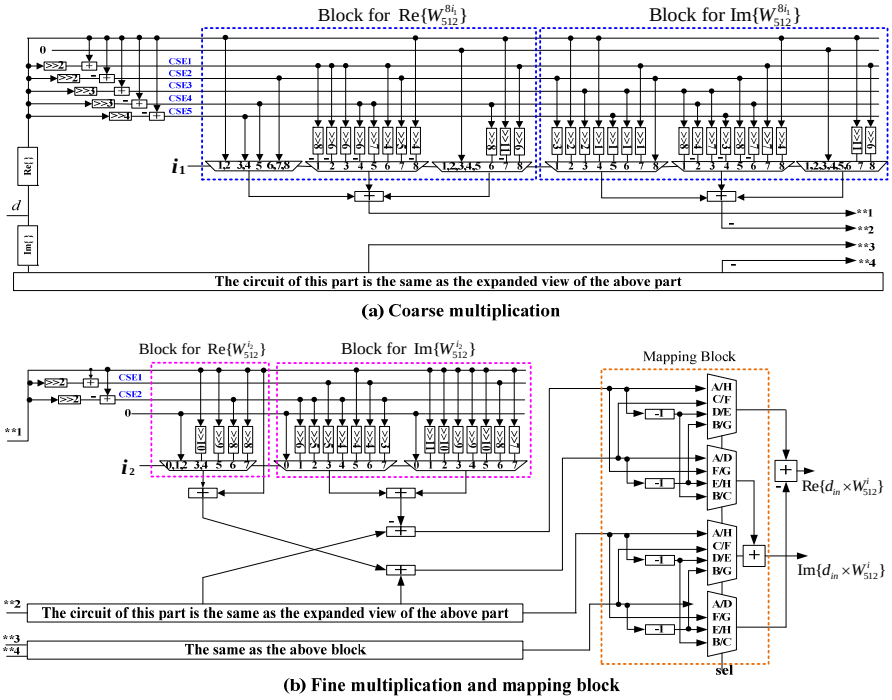


Fig. 4. Detailed structure of CSD complex multiplication for W_{512}^{i}.

Table 5. Hardware comparison results of 512-point FFT designs

Approaches	Multiplier types	Logic elements	Registers	Memory bits	Power (mW)
Radix-2^4 [7]	CSD and CCM	7,823 (1)	635 (1)	32,568 (1)	149.8 (1)
Radix-2^5 [5]	CSD and CCM	6,730 (0.86)	575 (0.91)	26,356 (0.81)	136.7 (0.92)
Proposed	CSD	5,669 (0.72)	523 (0.82)	21,608 (0.66)	122.2 (0.82)

pliers which replace conventional complex multi plier and remove ROM for storing twiddle factors. By simulation, the proposed FFT design achieves more than about 28% reduction in gate count and 18% reduction in power consumption compared t o the previous approaches.

REFERENCES

[1] C. Yu, M. H. Yen and S. J. Chen, "A Low-power 64-point pipeline FFT/IFFT processor for OFDM applications", IEEE Consum. Electron., vol. 5, pp. 40-45, 2011.

[2] J. Yu and K. J. Cho, "An area-efficient 256-point FFT design for WiMAX systems", KIIECT, vol. 11, no. 3, pp. 270-276, 2018.

[3] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation", Proc. URSI Int. Symp. Signals. Syst., Electron., 1998. pp. 257-262.

[4] J. Y. Oh and M. S. Lim, "New radix-2 to the 4th power pipeline FFT processor", IEICE trans. Electron., vol. E88-C, no. 8, pp.1740-1746, 2005.

[5] T. S. Cho and H. H. Lee, "A high-speed low-complexity modified radix-25 FFT processor for high rate WPAN applications", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no.1, pp. 187-191, 2013.

[6] M. Garrido, et al., "Pipelined radix-2k feedforward FFT architectures", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no.1, pp. 23-32, 2013.

[7] C. Yu and M. H. Yen, "Area-efficient 128- to 2048/1536-point pipeline FFT processor for LTE and mobile WiMAX systems", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no.9, pp. 1793-2015, 2015.

[8] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Willey-Interscience, 1999.

Author Biography

Jian Yu [Member]

•Jun. 2001: Hebei Normal Univ., Electronic Engr., BA
•Mar. 2008: Tianjin Polytechnic Univ., Electronic Engr., MS
•Mar. 2016 ~ current : Wonkwang Univ., Dept. of Electronic Engr., PhD course

<Research Interests> VLSI Design

Kyung-Ju Cho [Member]

•Aug. 2006 : Chonbuk National Univ., Info. & Comm. Engr., PhD
•Mar. 2012 ~ current : Wonkwang Univ., Dept. of Electronic Engr., Professor

<Research Interests> VLSI Design, SOC