An Optimum Architecture for Continuous-Flow Parallel Bit Reversal

Chen Cheng and Feng Yu, *Member, IEEE*

***Abstract*—With the aim of minimizing memory and latency, this letter presents a novel bit-reversal architecture for continuous-flow parallel pipelined FFT processors. It harnesses the theory that any permutation can be decomposed to a series of elementary bit-exchanges. The main contribution of this letter are twofold. First, it achieves continuous-flow bit reversal in parallel with the minimum memory and minimum latency. Second, the architecture, composed of memory and 2-to-1 multiplexers, are simple and regular for general power-of-2 parallelism. Furthermore, it supports different common radices, including radix-2,radix-4, and radix-8.**

***Index Terms*—Bit-reversal architecture, continuous-flow parallel architecture, fast fourier transform(FFT).**

I. Introduction

FAST FOURIER TRANSFORM (FFT) has become ubiquitous in signal processing applications, such as speech [1], audio, radar [2] and image signal processing [3]. For the consideration of high-throughput and real-time demands, parallel pipelined FFT designs [4]–[6] have been frequently adopted over the past decades. To achieve these designs, the bit-reversal arhitecture is a major function block that should be taken into consideration as well as FFT architecture itself [7], [8]. It reorders a set of indexed data according to a reversing of the bits of the index, which is necessary in many applications [9].

However, only a few works are dedicated to design bit-reversal circuits up to present time. Several works in the literature [6], [10], [11] investigate bit reversal for single-path pipelined FFT architecture [12]–[14]. [6], [10], [11] performs pipelined bit reversal with a single memory of size N by generating the memory address in nature and bit-reversed order, alternatively for even and odd sequences. The control of this design is simple but the usage of memory for bit reversal can be further reduced. [11] theoretically formulates the minimum memory and latency that is necessary for bit reversal on data arriving in series. [15] proposes optimum circuits for continuous-flow bit reversal with lowest memory and lowest latency, which is very suitable for hardware implementation. For parallel pipelined FFT architecture, it becomes more challenging to permutate multiple paths in parallel and maintain continuity of the data. [4] supports 8-parallel bit reversal but it consumes $P \cdot N$ memory and its hardware complexity is high. [16] requires the input data to be bit-reversed by row in advance albeit consuming $N-N/P$ memory. The circuits proposed in [5] can only accept specific parallel order. The proposed circuits in [17] consist of two flexible commutators and memory of size N, which is regular and flexible for hardware implementation. Nevertheless, according to the method of theoretical analysis on minimum memory and latency [11], theoretical memory and latency that are necessary for parallel bit reversal in series can be derived, which is less than N.

This letter presents a novel architecture to achieve parallel bit-reversing with minimum memory and latency. The rest of this letter is organized as such. Section II reviews existing circuits for single-path bit reversal in [15]. Section III derives the implementation of parallel bit-dimension permutation. In Section IV, the optimum circuits for parallel bit-reversing are presented and extended to common radices. Section V makes comparisons with existing bit-reversal circuits. Finally, some conclusions are drawn in Section VI.

II. Review of Single-Path Bit Reversal

According to [18], any index transformation can be decomposed to a series of elementary bit-exchanges. This means that bit-reversing can be implemented by a cascade of basic circuits for bit-exchanges.

Let us consider $N= 2^n$ indexed data and denote its index as $I \equiv b_{n-1}, \ldots, b_1, b_0$, herein n is a non-negative integer and the symbol ($\equiv$) is the binary representation of index. The bit-reversing function block is to move the data with index $I \equiv b_{n-1}, \ldots, b_1, b_0$ to the place $BR(I) \equiv b_0, b_1, \ldots, b_{n-1}$. This phase can be decomposed to a series of bit-exchanges between b_j and b_k, where $j > k$ and $j = n-1-k$. Therefore, it is a key point to efficiently reorder data based on indexed bit-exchanges between b_j and b_k.

$$I_A \equiv b_{n-1}, \ldots, b_{j+1}, 0, b_{j-1}, \ldots, b_{k+1}, 1, b_{k-1}, \ldots, b_0 \tag{1}$$

$$I_B \equiv b_{n-1}, \ldots, b_{j+1}, 1, b_{j-1}, \ldots, b_{k+1}, 0, b_{k-1}, \ldots, b_0. \tag{2}$$

Note that only samples for which $b_j \neq b_k$ should be exchanged their positions, namely that sample with index I_A should be swapped position with one with index I_B. To maintain the continuity of data, memory is necessary for swapping data and the depth of memory L for bit-exchange is $2^j - 2^k$.

Manuscript received June 18, 2015; accepted August 10, 2015. Date of publication August 19, 2015; date of current version September 28, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Philippe Coussy.

The authors are with the Department of Instrument Science & Technology, Zhejiang University, Hangzhou 310027, China (e-mail: cindycheng89@zju.edu.cn; osfengyu@zju.edu.cn).

Digital Object Identifier 10.1109/LSP.2015.2470519

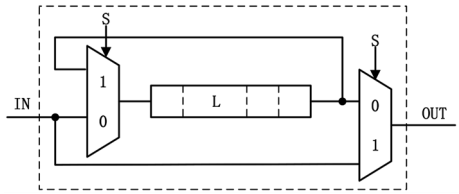

Fig. 1. Basic single-path circuit for bit-exchange permutation.

According to this, [15] proposed the basic single-path circuit for bit-exchange, as shown in Fig. 1. The control signal S can be generated by only AND operations as follows

$$S = b_j \& \overline{b_k}. \tag{3}$$

By cascading the basic circuit for bit-exchange, bit-reversing circuits can be derived and the total memory M_{total} used in circuits, equivalent to latency L_{total}, can also be calculated as follows.

$$M_{total} = L_{total} = \begin{cases} \left(\sqrt{N}-1\right)^2 & \text{for even n,} \\ (\sqrt{2N}-1)(\sqrt{\frac{N}{2}}-1) & \text{for odd n.} \end{cases} \tag{4}$$

In summation, [15] has proposed a novel circuit to efficiently implement bit-reversal permutation whereas it is dedicated for single-path design. For high throughput optimum circuits for parallel bit reversal should be designed to simultaneously permutate the multiple outputs of parallel pipelined FFT.

III. Optimum Approach for Parallel Bit-Exchange Permutation

For parallel pipelined bit-reversing, it is challenging to permutate among multi-path data flows and generate multiple outputs continuously [17]. Based on the algorithms in [18], once the basic circuits for parallel pipelined bit-exchanges are proposed, the whole architecture can be obtained by simply cascading these basic circuits. In this section, the circuits for parallel continuous-flow bit-exchanges are designed.

Consider the size of indexed data as $N = 2^n$ and the parallelism of data flows as $P = 2^q$. The range of P is limited to no more than $\frac{2\sqrt{N}}{r}$, where r is the radix, i.e. $r = 2$ for radix-2. If $P > \frac{2\sqrt{N}}{r}$, the parallelism is so high that most bit-exchanges are performed by simply swapping data between different paths. This situation rarely occurs in most applications and thereby is not discussed in this letter. The index of input data and output data is respectively represented as (5) and (6). And the index of data path is represented by the least low q bits.

$$I \equiv b_{n-1}, \ldots, b_{n-q}, b_{n-q-1}, \ldots, b_q, \underbrace{b_{q-1}, \ldots, b_0}_{\text{Path index}} \tag{5}$$

$$BR(I) \equiv b_0, \ldots, b_{q-1}, b_q, \ldots, b_{n-q-1}, \underbrace{b_{n-q}, \ldots, b_{n-1}}_{\text{Path index}}. \tag{6}$$

Let σ_i represents the bit-exchange between b_i and b_{n-i-1} as follows

$$\begin{aligned} &\sigma_i(b_{n-1}, \ldots, b_{n-i}, b_{n-1-i}, \ldots, b_{i+1}, b_i, \ldots, b_0) \\ &= b_{n-1}, \ldots, b_{n-i}, b_i, \ldots, b_{i+1}, b_{n-1-i}, \ldots, b_0 \end{aligned} \tag{7}$$

which can be also represented as follows

$$\sigma_i : b_i \leftrightarrow b_{n-1-i} \tag{8}$$

Observes that bit reversal from (5) to (6) is composed of $\lfloor n/2 \rfloor$ bit-exchanges σ_i, where $i \in [0, \lfloor n/2 \rfloor - 1]$. Let us define $k = i$ and $j = n-1-i$. The bit-exchanges can be discussed on two cases: bit-exchange σ_i within each path where $i \in [q, \lfloor n/2 \rfloor - 1]$ and bit-exchange σ_i cross the paths where $i \in [0, q-1]$.

A. Bit-Exchange within Each Path

For one parallel bit-exchange σ_i when $i \in [q, \lfloor n/2 \rfloor - 1]$, it can be performed by just swapping position between the data with index I'_A and the data with index I'_B. It is obvious that $P'_A = P'_B$ and thereby each path just permutates data inner the path.

$$\begin{aligned} I'_A \equiv\ & b_{n-1}, \ldots, b_{j+1}, 0, b_{j-1}, \ldots, b_{k+1}, 1, \\ & b_{k-1}, \ldots, b_q, \underbrace{b_{q-1}, \ldots, b_0}_{P'_A} \\ I'_B \equiv\ & b_{n-1}, \ldots, b_{j+1}, 1, b_{j-1}, \ldots, b_{k+1}, 0, \\ & b_{k-1}, \ldots, b_q, \underbrace{b_{q-1}, \ldots, b_0}_{P'_B}. \end{aligned}$$

Consequently, P number of single-path circuits in Fig. 1 are respectively adopted for each path. But the depth of memory for each circuit is $(2^{n-1-i} - 2^i)/2^q$ instead of $2^{n-1-i} - 2^i$. As a result, the summation of memory for P paths stays $2^{n-1-i} - 2^i$. Correspondingly, the control signals S of P paths can be generated by a single counter C, where $C \equiv c_{n-q-1}, \ldots, c_1, c_0$ and $S = c_{n-1-i-q} \& \overline{c_{i-q}}$. When $S = 0$, the positions of data are not changed. On the other hand, when $S = 1$, it swaps data as the rule mentioned above.

B. Bit-exchange Cross the Paths

As for one bit-exchange σ_i where $i \in [0, q-1]$, it swaps position between the data with index I''_A and the data with index I''_B. Note that $P''_A \neq P''_B$ and data in path P''_A are only swapped with data in path P''_B if necessary.

$$\begin{aligned} I''_A \equiv\ & b_{n-1}, \ldots, b_{j+1}, 0, b_{j-1}, \ldots, b_q, \\ & \underbrace{b_{q-1}, \ldots, b_{k+1}, 1, b_{k-1}, \ldots, b_0}_{P''_A} \\ I''_B \equiv\ & b_{n-1}, \ldots, b_{j+1}, 1, b_{j-1}, \ldots, b_q, \\ & \underbrace{b_{q-1}, \ldots, b_{k+1}, 0, b_{k-1}, \ldots, b_0}_{P''_B}. \end{aligned}$$

This means that P paths can be divided into $P/2$ pairs in which data interchange their positions and thereby the basic two-parallel circuit for bit-exchange when $i \in [0, q-1]$ is the key point of the design.

We propose a simple but efficient circuit to perform one bit-exchange σ_i between P''_A path and P''_B path as shown in Fig. 2.

The depth of memory L in Fig. 2 is $2^{n-1-i}/2^q$ and thereby the total memory for this bit-exchange permutation is 2^{n-1-i}, which is 2^i more than that for single-path implementation. In addition, the control signals S of all $P/2$ pairs are also controlled by a single counter C, where $S = c_{n-1-i-q}$.

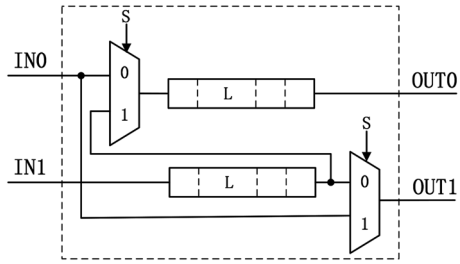

Fig. 2. Basic two-path circuit for bit-exchange permutation.

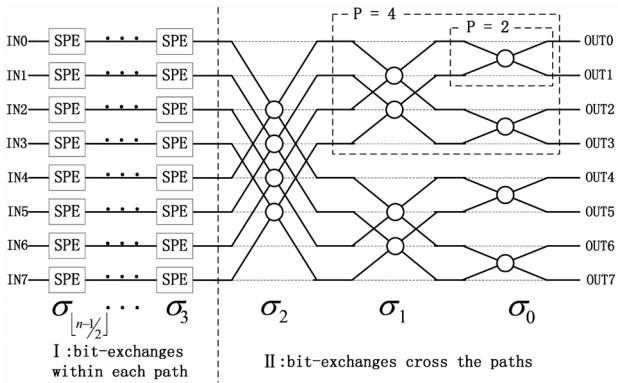

Fig. 3. Proposed architecture of 8-parallel bit reversal.

IV. The Optimum Architecture for Parallel Bit Reversal

This section cascades basic bit-exchange circuits to achieve bit reversal and extends this architecture to common radices.

A. Bit-Reversal Architecture

The bit reversal consists of $\lfloor n/2 \rfloor$ bit-exchanges and the order of bit-exchanges has no influence on the results. In terms of the regularity and simplicity, we do bit-exchanges σ_i as the sequence $i = \lfloor n/2 \rfloor - 1, \ldots, 1, 0$. Without loss of generality, the architecture of 8-parallel pipelined bit reversal is illustrated as an example in Fig. 3.

Fig. 3 shows that the architecture is divided into two parts, corresponding to the algorithm in Section III. In part I, one single-path element(SPE) stands for the single-path circuit in Fig. 1. Here one bit-exchange σ_i is accomplished by P number of SPEs that are distributed in P paths and the control signals of them are generated by one and the same $\log_2(N/P)$-bits counter. In part II, the symbol $\bigcirc$ in Fig. 3 stands for the two-path circuit in Fig. 2. For one bit-exchange, it contains $P/2$ number of two-path circuits and all control signals of them are controlled by one bit of a $\log_2(N/P)$-bits counter as Section III discussed.

From the Fig. 3, we can find that the switch scheme of dataflows for bit-exchanges in part II is very similar to that of FFT architecture. It is not surprising at all for the reason that the bit-reversing order is generated just by FFT architecture. In addition, the architecture of $P = 2$ and $P = 4$ for part II is also illustrated in Fig. 3. Through this regularity, higher parallelism can also be obtained. As for the case of single-path [15], i.e. $P = 1$, no bit-exchanges in part II occur and the architecture only contains one path of SPEs, which is also generalized in the proposed architecture.

To clarity the parallel processing of bit reversal, Table I shows the timing diagram of each bit-exchange output for $P = 4$ and $N = 64$. The input data arrives in bit-reversed order and are listed below the column of input data. Firstly, the bit-exchange σ_2 permutates input data within each path and outputs them as the sequence below the column of σ_2. Subsequently, the bit-exchange σ_1 permutates the output of σ_2. It divides the four paths into two pairs, i.e. $\{p_0, p_2\}$ and $\{p_1, p_3\}$, and each pair interchanges its data by the circuit in Fig. 2. The output of σ_1 is listed below σ_1 column. Finally, the bit-exchange σ_0 permutes data like σ_1 but the difference is that the paths of data are divided to $\{p_0, p_1\}$ and $\{p_2, p_3\}$. After three pairs of bit-exchanges, namely σ_2, σ_1 and σ_0, the output in nature order is generated. In Table I, the memory depth L of each path and the control signal S for each bit-exchange are shown as well.

B. Memory and Latency

In this subsection, the total memory and latency are calculated for the proposed architecture. Compared with single-path bit reversal, additional 2^i memory is consumed for each bit-exchange σ_i where $i \in [0, q-1]$. Consequently, the summation of additional memory is $\sum_{i=0}^{q-1} 2^i = P - 1$.

Based on the analysis for single-path bit reversing in (4), the final memory of parallel bit reversal M_{total} is

$$\begin{cases} (\sqrt{N}-1)^2 + P - 1 = N - 2\sqrt{N} + P \\ \text{for even n,} \\ (\sqrt{2N}-1)(\sqrt{\frac{N}{2}}-1) + P - 1 = N - \sqrt{2N} - \sqrt{\frac{N}{2}} + P \\ \text{for odd n.} \end{cases} \tag{9}$$

And the latency $L_{total} = M_{total}/P$ is

$$\begin{cases} ((N - 2\sqrt{N} + P)/P & \text{for even n,} \\ ((N - \sqrt{2N} - \sqrt{\frac{N}{2}} + P)/P & \text{for odd n.} \end{cases} \tag{10}$$

The results in (9) and (10) are just the theoretical minimum memory and minimum latency of parallel continuous-flow bit reversal. They can be easily derived by the method in [11], namely calculating the max time interval between the moment of input sample and that of output sample.

C. Extension to Other Radices

The bit reversal mentioned above are all about radix-2 FFT architecture. In many applications, radix-4 and radix-8 FFT architectures are adopted for the saving of multiplications. The order of the output data has some differences between different radices.

In general, the bit reversal of any radix-$r(BR_r)$ groups $\log r$ bits of the index and performs a reversal of these groups. All of them are essentially composed of $\lfloor n/2 \rfloor$ bit-exchanges and the only difference is the mapping of bit-exchanges σ_i.

For radix-r, it can be generally represented as follows

$$\sigma_i : b_i \leftrightarrow b_{n-i-\log r+mod(i,\log r)}$$

Analogously, these bit-exchanges are discussed on two cases: bit-exchange σ_i within each path where $i \in [q, \lfloor n/2 \rfloor - 1]$ and bit-exchange σ_i cross the paths where $i \in [0, q-1]$. The circuits

TABLE I
TIMING DIAGRAM OF EACH BIT-EXCHANGE OUTPUT FOR PARALLEL BIT REVERSAL

Time (cycles)	Input Data				Part I					Part II									
	No Operations				$\sigma_2 : \mathbf{b_3} \leftrightarrow \mathbf{b_2}(\mathbf{L}=\mathbf{1})$					$\sigma_1 : \mathbf{b_4} \leftrightarrow \mathbf{b_1}(\mathbf{L}=\mathbf{4})$					$\sigma_0 : \mathbf{b_5} \leftrightarrow \mathbf{b_0}(\mathbf{L}=\mathbf{8})$				
	p0	p1	p2	p3	S	p0	p1	p2	p3	S	p0	p1	p2	p3	S	p0	p1	p2	p3
0	0	32	16	48	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	8	40	24	56	0	0	32	16	48	0	-	-	-	-	-	-	-	-	-
2	4	36	20	52	1	4	36	20	52	0	-	-	-	-	-	-	-	-	-
3	12	44	28	60	0	8	40	24	56	0	-	-	-	-	-	-	-	-	-
4	2	34	18	50	0	12	44	28	60	0	-	-	-	-	-	-	-	-	-
5	10	42	26	58	0	2	34	18	50	1	**0**	32	**2**	34	0	-	-	-	-
6	6	38	22	54	1	6	38	22	54	1	**4**	36	**6**	38	0	-	-	-	-
7	14	46	30	62	0	10	42	26	58	1	**8**	40	**10**	42	0	-	-	-	-
8	1	33	17	49	0	14	46	30	62	1	**12**	44	**14**	46	0	-	-	-	-
9	9	41	25	57	0	1	33	17	49	0	**16**	48	**18**	50	0	-	-	-	-
10	5	37	21	53	1	5	37	21	53	0	**20**	52	**22**	54	0	-	-	-	-
11	13	45	29	61	0	9	41	25	57	0	**24**	56	**26**	58	0	-	-	-	-
12	3	35	19	51	0	13	45	29	61	0	**28**	60	**30**	62	0	-	-	-	-
13	11	43	27	59	0	3	35	19	51	1	**1**	33	**3**	35	1	**0**	**1**	2	3
14	7	39	23	55	1	7	39	23	55	1	**5**	37	**7**	39	1	**4**	**5**	6	7
15	15	47	31	63	0	11	43	27	59	1	**9**	41	**11**	43	1	**8**	**9**	10	11
16	-	-	-	-	0	15	47	31	63	1	**13**	45	**15**	47	1	**12**	**13**	14	15
17	-	-	-	-	-	-	-	-	-	0	**17**	49	**19**	51	1	**16**	**17**	18	19
18	-	-	-	-	-	-	-	-	-	0	**21**	53	**23**	55	1	**20**	**21**	22	23
19	-	-	-	-	-	-	-	-	-	0	**25**	57	**27**	59	1	**24**	**25**	26	27
20	-	-	-	-	-	-	-	-	-	0	**29**	61	**31**	63	1	**28**	**29**	30	31
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**32**	**33**	34	35
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**36**	**37**	38	39
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**40**	**41**	42	43
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**44**	**45**	46	47
25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**48**	**49**	50	51
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**52**	**53**	54	55
27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**56**	**57**	58	59
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	**60**	**61**	62	63

in Fig. 1 and Fig. 2 are respectively applied for two cases and the control signal S are generated by the data counter C, where $S = c_{n-i-\log r+mod(i,\log r)-q}\&\overline{c_{i-q}}$ for the circuits in Fig. 1 and $S = c_{n-i-\log r+mod(i,\log r)-q}$ for the circuits in Fig. 2. From this we find that the control of S for radix-2 is also generalized. Accordingly, the proposed architecture can support bit reversal of any radix-r by slightly changing the control signal.

V. COMPARISON

This section presents some comparison results of our proposed architecture with other bit-reversing designs.

Table II compares several bit-reversing architecture from four aspects: supported parallelism, memory size, throughput and latency. [15] presents optimum circuits for achieving bit reversal with lowest memory and lowest latency but it can only support single-path data. Among the parallel bit-reversing circuits, the circuits in [17] are most general and cost-efficient with memory of size N and lower latency. But [17] can not improve its architecture to further reduce memory size and latency. The proposed architecture in this letter successfully achieve parallel bit reversal with lowest memory and lowest latency. The architecture consists of only memory and 2-to-1 multiplexers, which are suitable and regular for hardware implementation. Besides, it can support common radices including radix-4 and radix-8.

TABLE II
COMPARISONS OF SEVERAL BIT-REVERSAL CIRCUITS

	Supported Parallelism	Memory Size (words)	Throughput	Latency (cycles)
[4]	8	PN	P	Not shown
[17]	2/4/8	N	P	$L\_[17]$ (1)
[15]	1	$M\_[15]$ (2)	1	$L\_[15]$ (2)
This letter	2^n (3)	$M\_[19]$ (4)	P	$L\_[19]$ (4)

[1] $L\_[17] = \frac{N}{P} - \sqrt{\frac{N}{2P^2}} + 1$ for even α and $\frac{N}{P} - \sqrt{\frac{N}{P^2}} + 1$ for odd α, where $\alpha = \log_2 \frac{N}{2P^2}$.

[2] $M\_[15] = N - 2\sqrt{N} + 1$ for even n and $N - \sqrt{\frac{N}{2}} - \sqrt{2N} + 1$ for odd n. And $L\_[15] = M\_[15]$.

[3] $n \in [0, \log_2 \frac{2\sqrt{N}}{r}]$

[4] $M\_[19] = N - 2\sqrt{N} + P$ for even n and $N - \sqrt{\frac{N}{2}} - \sqrt{2N} + P$ for odd n. $L\_[19] = \frac{M\_[19]}{P}$, that is $\frac{N}{P} - 2\sqrt{\frac{N}{P^2}} + 1$ for even n and $\frac{N}{P} - \sqrt{\frac{N}{2P^2}} - \sqrt{\frac{2N}{p^2}} + 1$ for odd n.

VI. CONCLUSION

In this letter, a new parallel bit-reversing architecture is proposed for parallel pipelined FFT with the aim of minimizing memory and latency. The proposed architecture is efficient because it requires lowest memory and achieves lowest latency in theory. Furthermore, this architecture is simple, regular and scalable for higher parallelism and common radices.

This paper previously published in IEEE Signal Processing Letters

REFERENCES

[1] A. Matsunaga, K. Koga, and M. Ohkawa, "An analog speech scrambling system using the fft technique with high-level security," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 540–547, 1989.

[2] H. Liu, H. Su, P. Shui, and Z. Bao, "Multipath signal resolving and time delay estimation for high range resolution radar," in *IEEE Int. Radar Conf. 2005*, 2005, pp. 497–502, IEEE.

[3] T. Bose, F. Meyer, and M.-Q. Chen, *Digital Signal and Image Processing*. New York, NY, USA: Wiley, 2004.

[4] S. Yoshizawa, A. Orikasa, and Y. Miyanaga, "An area and power efficient pipeline fft processor for 8×8 mimo-ofdm systems," in *IEEE Int. Symp. Circuits and Systems(ISCAS), 2011*, 2011, pp. 2705–2708, IEEE.

[5] K.-J. Yang, S.-H. Tsai, and G. C. Chuang, "Mdc fft/ifft processor with variable length for mimo-ofdm systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 4, pp. 720–731, 2013.

[6] M. Garrido, "Efficient hardware architectures for the computation of the fft and other related signal processing algorithms in real time," Ph.D. dissertation, Univ. Politécnica de Madrid, Madrid, Spain, 2009.

[7] D. Sundararajan, M. O. Ahmad, and M. Swamy, "A fast fft bit-reversal algorithm," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 41, no. 10, pp. 701–703, 1994.

[8] M. A. Jaber and D. Massicotte, "A novel approach for fft data reordering," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS) 2010*, 2010, pp. 1615–1618, IEEE.

[9] B. Gold, T. G. Stockham, A. V. Oppenheim, and C. M. Rader, *Digital Processing of Signals*. New York, NY, USA: McGraw-Hill, 1969.

[10] Y.-N. Chang, "An efficient vlsi architecture for normal i/o order pipeline fft design," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 55, no. 12, pp. 1234–1238, 2008.

[11] T. S. Chakraborty and S. Chakrabarti, "On output reorder buffer design of bit reversed pipelined continuous data fft architecture," in *IEEE Asia Pacific Conf. Circuits and Systems, 2008 APCCAS*, 2008, pp. 1132–1135, IEEE.

[12] X. Liu, F. Yu, and Z.-k. Wang, "A pipelined architecture for normal i/o order fft," *J. Zhejiang Univ. Sci C.*, vol. 12, no. 1, pp. 76–82, 2011.

[13] S. Lee and S.-C. Park, "Modified sdf architecture for mixed dif/dit fft," in *IEEE Int. Symp. Circuits and Systems, 2007 ISCAS 2007.*, 2007, pp. 2590–2593.

[14] S. He and M. Torkelson, "Designing pipeline fft processor for ofdm(de) modulation," in *URSI Int. Symp. Signals, Systems, and Electronics, ISSSE 1998*, 1998, pp. 257–262, IEEE.

[15] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum circuits for bit reversal," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 58, no. 10, pp. 657–661, 2011.

[16] M. Garrido, J. Grajal, M. Sanchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[17] S.-G. Chen, S.-J. Huang, M. Garrido, and S.-J. Jou, "Continuous-flow parallel bit-reversal circuit for mdf and mdc fft architectures," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 61, no. 10, pp. 2869–2877, Oct. 2014.

[18] A. Edelman, S. Heller, and S. Lennart Johnsson, "Index transformation algorithms in a linear algebra framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 12, pp. 1302–1309, 1994.

This paper previously published in IEEE Signal Processing Letters