Novel Bit-Reordering Circuit for Continuous-Flow Parallel FFT Architectures

Gyanendra, *Student Member, IEEE*, Balasubramanian Raman, *Member, IEEE*, and Brajesh Kumar Kaushik, *Senior Member, IEEE*

***Abstract*—This brief presents a parallel bit-reordering circuit for calculating bit reordering on a stream of parallel data. The circuit consists of buffers, multiplexers, a bit-reversal circuit for single path serial data, and control signal generators. A novel data reordering mechanism has been devised that has twofold benefits: firstly, it helps in attaining continuous flow of data with minimum memory and minimum latency; secondly, the area and power requirement of the circuit reduces by 15-20%. The proposed circuit provides a solution for reordering the data of different common radices such as radix-2^k, radix-4, radix-8, and other radices. The circuit generates natural order output for variable power-of-2 (2^n) fast Fourier transform (FFT) lengths.**

***Index Terms*—Bit-reversal, data reordering, fast Fourier transform (FFT), parallel pipelined FFT.**

I. Introduction

THE FAST Fourier transform (FFT) algorithm based on Cooley-Tukey decomposition is widely used for fast computation of discrete Fourier transform (DFT). This decomposition reduces the arithmetic complexity to the $O(N\ log_2 N)$, while the brute-force approach requires computations of $O(N^2)$. In the recent past, several hardware architectures had been proposed for the fast computation of FFT. These architectures are broadly classified as memory-based in-place FFT hardware architectures and pipelined FFT hardware architectures. Pipelined architectures consist of two sections, one for calculating FFT itself and the other one for performing bit reversal to generate natural-order FFT output. These pipelined architectures are preferred for their high-throughput rate. There are two broad classes of pipelined FFT architectures, e.g., serial-pipelined FFT and parallel-pipelined FFT.

Several efforts have been made by researchers for realizing bit-reversal and bit-reordering in the software, firmware, and hardware. The bit-reordering algorithm permutes a set of indexed bit by reversing the bits of indices [1]. Based on the famous Gold-Rader's algorithm, many algorithms have been proposed in the literature for the efficient realization of bit reversal. These algorithms are suitable for general-purpose computers and other microprocessor-based systems. However, this is not suitable for real-time systems due to long delays [2]–[5]. The firmware implementation requires a read-only memory (ROM) to store digit-reversed address sequence so that the desired data are fetched during the execution of the FFT algorithm. Moreover, a lookup table is stored in the ROM for the execution of FFT of different sizes and radices. This leads to a significant increase in memory size subject to the size or the radix of FFT [6]. In order to reduce the memory, bit-reversal circuits operating on a series of words have been proposed [7]–[9]. These circuits perform bit reversal using either double-buffering strategy or single memory, where memory addresses are generated either in natural or in a bit reversed order. The optimum bit reversal circuit operating on a series of data has been discussed in [10]. This circuit uses a minimum number of buffers and has minimum latency. Several bit-reversal circuits for parallel pipelined FFT architectures have also been discussed in [11]–[14]. Bit-reversal circuits proposed in [11] works for only 8-parallel, 128 points FFT and hence not scalable. The circuit discussed in [12] can support a limited number of parallel data. The circuit proposed in [13] requires huge memory; however, the memory required in [14] gets significantly reduced to the sample size N. This is achieved with the help of a commutator based switching mechanism and efficient address generations. The approach published in [15] is the first one to attain minimum latency and minimum memory simultaneously. Commutator is used in this circuit either for writing or reading memory banks efficiently. However, this circuit uses a large number of multiplexers. In addition to this, several circuits are used to generate addresses and signals for performing read/write operations. Thus, the hardware complexity of these designs is high. The effort made in [16] helps in reducing the number of multiplexers, however, at the expense of huge memory and latency.

In order to resolve the above-mentioned issues, a novel approach has been proposed in this brief that not only attains minimum memory and latency requirement of the circuit but also the circuit complexity is significantly reduced. An appropriate buffering strategy and data-flow along with concurrent data-switching are the key ideas that help in reducing the circuit complexity, thereby attaining minimum memory and latency.

The rest of this brief is organized as follows. Framework for continuous flow parallel bit reversal has been discussed in Section II. Section III describes the proposed circuit while the implementation and comparison with previous circuits are discussed in Section IV. Finally, the conclusions are made in Section V.

Manuscript received March 4, 2020; accepted March 29, 2020. Date of publication April 9, 2020; date of current version November 24, 2020. This brief was recommended by Associate Editor A. J. Acosta. *(Corresponding author: Brajesh Kumar Kaushik.)*

Gyanendra and Brajesh Kumar Kaushik are with the Microelectronics and VLSI Group, Department of Electronics and Communication Engineering, Indian Institute of Technology Roorkee, Roorkee 247667, India (e-mail: gyanendra@ec.iitr.ac.in; bkk23fec@iitr.ac.in).

Balasubramanian Raman is with the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee 247667, India (e-mail: balarfma@iitr.ac.in).

Digital Object Identifier 10.1109/TCSII.2020.2984892

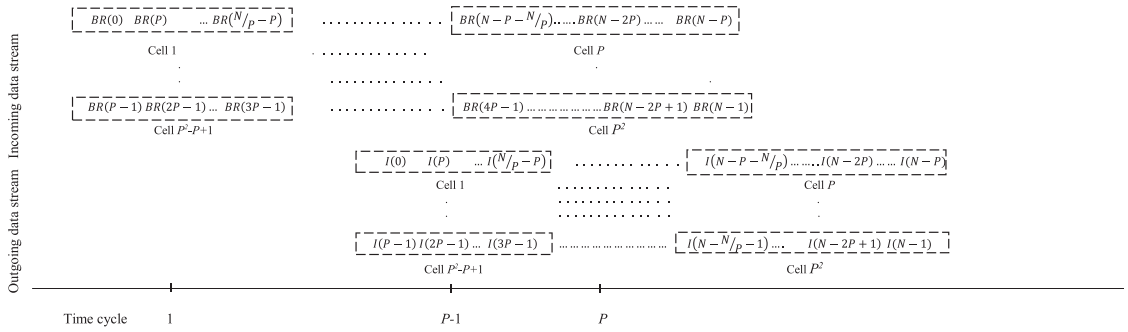

Fig. 1. Indices of incoming and outgoing data stream against time.

II. Framework for Continuous-Flow Parallel Bit Reordering

The computation of P-parallel N-point FFT in natural-order generates reordered output, and the order depends on FFT radices. The parallelism P is in power of radices and the size of indexed sample is N, where N is multiple of P^2 [13]. Bit- reordering is done to convert the reordered output to natural-order FFT output. As shown in Fig. 1, the incoming data stream represents the reordered indices of FFT output and the outgoing data stream represents the indices of natural-order FFT output. Each stream of incoming data is divided into P cells, and thus N samples arriving from P-parallel paths are stored in P^2 cells. Each cell consists of m number of data, where $m = N/P^2$.

The N-point data is presented by n bits, where $n = log_2 N$. These n bits, in the form of a 1-D array, present the indices of outgoing data as shown in equation (1) [10].

$$I = b_{n-1}b_{n-2}b_{n-3}\cdots\ \cdots\ \cdots\ \cdots b_2b_1b_0 \tag{1}$$

where, bit $b_i \in \{0,1\}$. In radix-2 and other radix-2^k FFT calculations, the indices BR_2 of output data are reordered as follows:

$$BR_2 : b_0b_1b_2\ \cdots\ \cdots\ \cdots\ \cdots b_{n-3}b_{n-2}b_{n-1} \tag{2}$$

Here, positions of bits in the array get reversed.

While performing bit reordering for radix-4 FFT, indices of output data BR_4 are considered in pairs and arranged as follows

$$BR_4 : b_1b_0\ b_3b_2\ \cdots\ \cdots\ \cdots b_{n-1}b_{n-2} \tag{3}$$

For re-ordering of radix-8 FFT, indices BR_8 are reordered as below:

$$BR_8 : b_{n/2-1,}\cdots\ \cdots b_1b_0\ b_{n-1}b_{n-2}\cdots b_{n/2} \tag{4}$$

Indices of data in cell array comprise path and cell indices. Out of n bits, q least significant bits (LSBs) represent path indices, q most significant bits (MSBs) represent cell indices and intermediate bits present the number of data in a cell as shown in equation (5).

$$I = \underbrace{b_{n-1}\ldots b_{n-q}}_{\text{Cell index}}\ \underbrace{b_{n-q-1}\cdots\ \cdots\ \cdots\ \cdots b_q}_{\text{Path index}}\ b_{q-1}\cdots b_1b_0 \tag{5}$$

Here, it is assumed that the order of parallelism or the number of cells in a path P is $P = 2^q$, where q is the number of bits required to represent a path as well as cell. All cells of a particular row have the same path index, whereas cells in a particular column have different cell index. The reordering is done in four phases: input path reordering, cell exchange, output path reordering, and bit reversal.

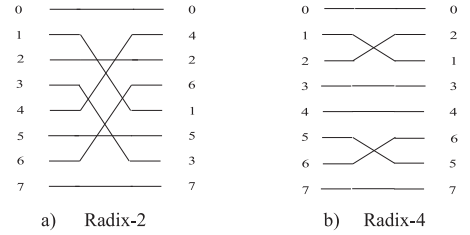

Fig. 2. Input and output path reordering for (a) Radix-2 (b) Radix-4.

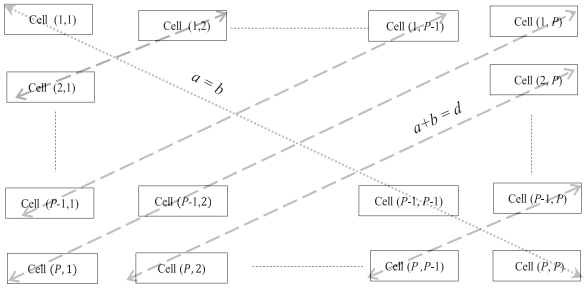

Fig. 3. Cell array.

A. Input Path Reordering

Out of n bits of indices, q least significant bits (LSBs) form indices of P parallel path and these indices are reordered according to the radices of FFT as described in (2), (3), and (4). On reordering the path, for instance radix-2 FFT, the index of data is given below.

$$I = b_{n-1}\cdots b_{n-q}b_{n-q-1}\cdots\ \cdots\ \cdots\ \cdots b_qb_0b_1\cdots b_{q-1} \tag{6}$$

B. Cell Exchange and Output Path Reordering

As shown in Fig. 3, a cell can be identified with the help of row a and column b as C (a, b), where $a \in \{1, \ldots, P\}$ and $b \in \{1, \ldots, P\}$. The cells above the line $a = b$ ($a < b$), and below it ($a > b$), are exchanged along the dotted line where $a + b = d$, and $d \in \{2, \ldots, 2P\}$. After the exchange, the value of row and column gets exchanged like C (a, b) to C (b, a) and vice-versa. Thus, the index of data is obtained below as.

$$I = b_0b_1\ldots b_{q-1}\ b_{n-q-1}\cdots\ \cdots\ \cdots\ \cdots b_qb_{n-1}\cdots b_{n-q} \tag{7}$$

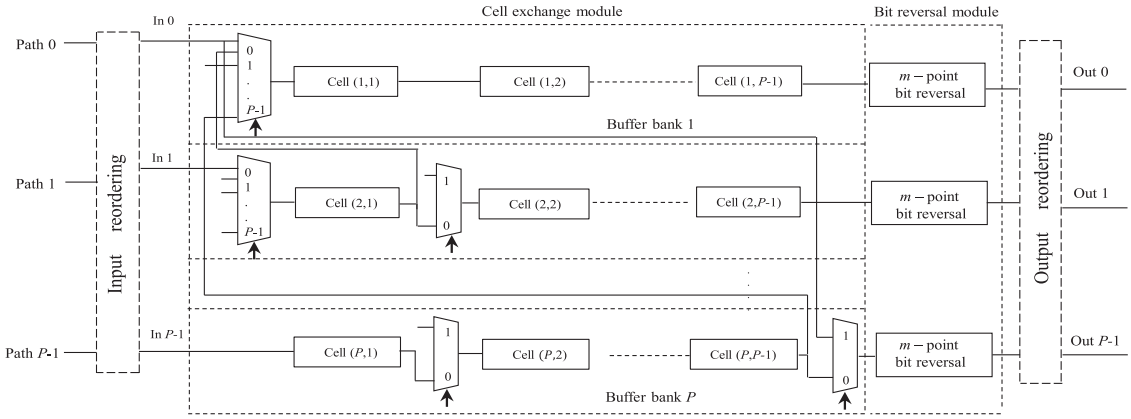

Fig. 4. Block diagram of proposed parallel bit reversal circuit.

Paths of output data are reordered and the corresponding indices are obtained as.

$$I = b_0 b_1 \dots b_{q-1}\ b_{n-q-1} \cdots \cdots \cdots \cdots .b_q b_{n-q} \cdots b_{n-1} \quad (8)$$

C. Bit-Reversal for Single Path Serial Data

Bit reordering, as discussed in Section A, is performed on each cell to get normal order output. A single cell consists of m number of data, and each one consists of n bits. After reordering the indices of data are obtained as follows.

$$I = \underbrace{b_0\ b_1 \cdots b_{q-1}}_{\text{Cell index}}\ \underbrace{b_q \cdots \cdots \cdots \cdots b_{n-q-1}\ b_{n-q} \cdots b_{n-1}}_{\text{Path index}} \quad (9)$$

Finally, the indices shown in (9) are reversed order indices. In a similar fashion, the algorithm can operate for other radices as well.

III. Proposed Parallel Bit Reordering Circuit

A novel circuit, based on the framework discussed in Section II, is proposed in this section for computing bit reordering of parallel data. As shown in Fig. 4, the circuit operates over P- parallel inputs and calculates bit reversal for the continuous flow of data. The circuit consists of buffers, multiplexers, a bit-reversal circuit for single path serial data, and control signal generators for data scheduling mechanisms. The data coming out of P- parallel paths pass through the input reordering module where the input path is reordered based on (2), (3), and 4. Next, the output of this module is made available at the input of the cell exchange module.

A. Cell Exchange Module

This module performs the cell exchange as discussed in Fig. 3. It consists of buffer banks and multiplexers. The module stores cells up to column number $P-1$ and the P^{th} column is forwarded for maintaining the continuity in the data flow. The number of parallel placed buffer banks are P and each one consists of P-1 cells. Each cell holds m number of samples and the content of cells is exchanged with the help of multiplexers.

For convenience, the position of a multiplexer can be identified with the help of row and column as $M\ (u,\ v)$, where $u \in \{1, \dots, P\}$ and $v \in \{1, \dots, P\}$. A cell lies in between two horizontal points. Thus, $P(P-1)$ number of buffer cells are arranged in P rows and $P-1$ columns. Data coming out of i^{th} path, where $i \in \{1, \dots, P\}$, is made available at the input of multiplexers for which $u - i + 1 = q$. Similarly, the position of a buffer cell is identified with the help of row and column as $B\ (j,\ k)$ where $j \in \{1, \dots, P\}$ and $k \in \{1, \dots, P-1\}$. For all those cells with $j > k$, the output is connected to the input of multiplexers with $u\ = j-1$, and $v\ = 1$.

TABLE I
Number of Registers in Bit-Exchange Circuit

Type of FFT	M_B	
	$\frac{N}{P^2}$ as even power of the radix	$\frac{N}{P^2}$ as odd power of the radix
Radix-2 and Radix-2^k	$\left(\sqrt{N/P^2}-1\right)^2$	$\left(\sqrt{2N/P^2}-1\right)\left(\sqrt{N/2P^2}-\right)$
Radix-4	$\left(\sqrt{N/P^2}-1\right)^2$	$\left(\sqrt{4N/P^2}-\right)\left(\sqrt{N/4P^2}-1\right)$
Radix-8	$\left(\sqrt{N/P^2}-1\right)^2$	$\left(\sqrt{8N/P^2}-1\right)\left(\sqrt{N/8P^2}-1\right)$

B. Bit Reversal Module

Bit reversal module consists of P-bit reversal circuits (Fig. 4) for computing bit reversal on a cell. These circuits are described in [10]. Table I shows total memory M_B needed in the module for different radices.

C. Memory and Latency Calculation

The number of registers in a cell exchange module is calculated as follows:

$$\begin{aligned} M_{cell}\ &= \textit{Numbe of cells}\ \times \textit{Number of registers in a cell} \\ &=\ P(P-1)\ \times N/P^2 \\ &= N - N/P \end{aligned}$$

Memory required for bit exchange is described in Table I. The memory needed for bit exchange module is calculated as

$$M_{BR}\ = P \times \left(\sqrt{N/P^2}-1\right)^2 =\ N/P + P - 2\sqrt{N}$$

Total memory required for calculating bit reversal is

$$\begin{aligned} M_{Total} &= M_{CELL} + M_{BR} \\ &= N - 2\sqrt{N} + P \end{aligned}$$

TABLE II
TIMING DIAGRAM OF 4-PARALLEL 64-POINT BIT REVERSAL

Time (Cycles)	Input data	Reordering at input port	Cell exchange module			Input data at Bit reordering	Output data
			Buffer cell a, d, g, j	Buffer cell b, e, h, k	Buffer cell c, f, i, l		
1	**12 4 8 0**	**12 4 8 0**	**12 4 8 0**				
	44 36 40 32	28 20 24 16	28 20 24 16				
	28 20 24 16	44 36 40 32	44 36 40 32				
	60 52 56 48	60 52 56 48	60 52 56 48				
2	**14 6 10 2**	**14 6 10 2**	28 20 24 16	**12 4 8 0**			
	46 38 42 34	30 22 26 18	30 22 26 18	**14 6 10 2**			
	30 22 26 18	46 38 42 34	46 38 42 34	44 36 40 32			
	62 54 58 50	62 54 58 50	62 54 58 50	60 52 56 48			
3	**13 5 9 1**	**13 5 9 1**	44 36 40 32	28 20 24 16	**12 4 8 0**		
	45 37 41 33	29 21 25 17	46 38 42 34	30 22 26 18	**14 6 10 2**		
	29 21 25 17	45 37 41 33	45 37 41 33	29 21 25 17	**13 5 9 1**		
	61 53 57 49	61 53 57 49	61 53 57 49	62 54 58 50	60 52 56 48		
4	**15 7 11 3**	**15 7 11 3**	60 52 56 48	44 36 40 32	28 20 24 16	**12 4 8 0**	**12 8 4 0**
	47 39 43 35	31 23 27 19	62 54 58 50	46 38 42 34	30 22 26 18	**14 6 10 2**	**13 9 5 1**
	31 23 27 19	47 39 43 35	61 53 57 49	45 37 41 33	29 21 25 17	**13 5 9 1**	**14 10 6 2**
	63 55 59 51	63 55 59 51	63 55 59 51	47 39 43 35	31 23 27 19	**15 7 11 3**	**15 11 7 3**

TABLE III
MULTIPLEXERS SIGNAL

Time (Cycles)	S_a	S_f	S_h	S_k	S_m	S_o	S_r	S_t	S_v
1	00	00	0	0	0	0	0	0	0
2	01	01	1	0	0	0	0	0	0
3	10	10	0	1	1	1	0	0	0
4	11	11	0	0	0	0	1	1	1

For radix-2 FFT, total memory needed is

$$M_{Tot.} = \begin{cases} N - 2\sqrt{N} + P \ m \ is \ even \ power \ of \ 2 \\ N/P - \sqrt{2N}/P - \sqrt{N/2}/P + 1 \ m \ is \ odd \ power \ of \ 2 \end{cases}$$

As shown in Fig. 3, all cells below the diagonal line are accompanied by a multiplexer that facilitates exchange across the diagonal line as explained in Section II-B. Therefore, the total number of multiplexers in a cell exchange module is

$$M_{cellmux.} = \sum_{i=1}^{P} i = \frac{P(P+1)}{2}$$

Number of multiplexers needed for bit exchange module is derived as

$$M_{bitmux.} = 2P\left\lfloor \frac{log_2 m}{2} \right\rfloor$$

Integer in the braces accounts for the number of exchanges done for reordering data in a cell. The elementary bit reversal circuit consists of two multiplexers [10]. Total multiplexers needed for calculating bit reversal is calculated as

$$\begin{aligned} Mmux &= M_{cellmux} + M_{bitmux} \\ &= \frac{P(P+1)}{2} + 2P\left\lfloor \frac{log_2 m}{2} \right\rfloor \end{aligned}$$

The number of memory elements in a circuit decides its latency. In this approach, latency (in cycles) introduced by the cell exchange module is calculated as $L_{BR} = M_{Total.}/P$ which includes the latency introduced by the bit reversal module also.

D. Signal Generation

The necessary signal for controlling multiplexers are generated with the help of l- bit counter where $l = log_2(N/P)$. Out of l bits, the MSBs l to $l-u+1$ are connected to multiplexers M $(u = 1, v)$. Signals for multiplexers, with a given value of u and $v > 1$, are generated by *AND* gate operation over l-bits. Each cell takes m number of clock pulses to store the data and thus, the signal remains constant for that period.

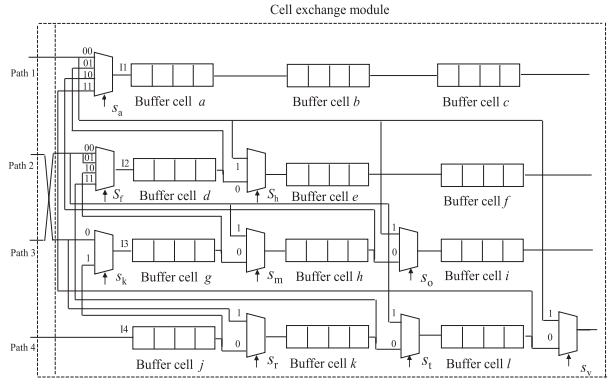

Fig. 5. Cell exchange module for 4- parallel 64-point bit reversal circuit.

E. 4-Parallel 64-Point Bit Reversal

Fig. 5 presents the circuit for performing bit reversal over 4-parallel 64-point data. The circuit gets data in bit reversed order. The path reordering is performed by changing the order of input and output wires. Table II explains the functioning of 4-parallel 64-point bit reversal circuit. The First column of the table is time cycle, the second column shows input data and the third column is reordered input data. The fourth to sixth columns are the content of buffer bank cells. The seventh column holds data arriving at the input of bit reordering circuit. Finally, the eighth column is the output. In Table III, the first column is time cycle and the second to tenth column are signals for driving multiplexers.

During first cycle, input data gets filled in buffer cells *a, d, g*, and *j*. In the second cycle, buffer cells *b, h,* and *k* get data from cells *a, g,* and *j*, respectively while buffer cell *e* gets data from input port I_1. In the next cycle, buffer cell *i* gets data from I_2 and data available at buffer cell *h* gets transferred to *d*. Similarly, during fourth cycle data from cell *j, k,* and *l* are made available respectively to cell *g, d,* and *a*. Data from *I* 2, *I*3, and *I*4 are filled in cells *l, j*, and *k*. At the same time, data from I_1 is directed to output of cell *l*. Output of cells *c, f, i,* and *l* are fed to bit reordering circuits and output of this circuit is in natural order.

IV. IMPLEMENTATION AND COMPARISON

Performance comparison of the proposed circuit with earlier circuits are shown in Table IV. These parallel circuits have the same number of throughput and parallel inputs.

TABLE IV
COMPARISON OF DESIGN FOR CALCULATING PARALLEL BIT REVERSAL

Design	Supported parallelism	Type	Memory	Latency (cycles)	Multiplexer	Cases
11	2^3	Memory	$P.N$	-	-	-
12	2^1/ 2^2/ 2^3/ 2^4	Memory	N-(N/P)	-	-	-
13	2^3	Memory	(9/8)N+192	-	-	-
14	2^1/ 2^2/ 2^3	Memory	N	N/P-$\sqrt{N}/P$+1	$2P^2 - P$	m is even
			N	N/P-$\sqrt{N/2}/P$+1		m is odd
15	2^n, n any integer	Delay	N-$2\sqrt{N} + P$	N/P-$2\sqrt{N}/P$+1	$4P^2 - 4P + 2P\lfloor(log_2 m)/2\rfloor$	m is even
			N-$\sqrt{2N}$-$\sqrt{N/2} + P$	N/P-$\sqrt{2N}/P - \sqrt{N/2}/P$+1		m is odd
16	2^n, n any integer	Memory	N	N/P	$Plog_2 P$	
This work	2^n, n any integer	Delay	N-$2\sqrt{N} + P$	N/P-$2\sqrt{N}/P$+1	$P/2\,(P+1)+2P\lfloor(log_2 m)/2\rfloor$	m is even
			N-$\sqrt{2N}$-$\sqrt{N/2} + P$	N/P-$\sqrt{2N}/P - \sqrt{N/2}/P$+1		m is odd

TABLE V
COMPARISON OF EXPERIMENTAL RESULTS FOR 4-PARALLEL 64-POINT FFT

	[15]	[16]	This work
Area in μm^2 (@90nm)	14,090	15,235	10,005
Gate count	2,548	2755	1809
Power consumption(mW)	7.42	6.6	5.92
Maximum frequency of operation (MHz)	378	558	826

Latency and memory have not been described for [11]–[13]. Memory-based designs [11]–[14] can support a limited number of parallel architecture. The Circuit described in [16] operates over any number of parallel architectures. As compared with the circuit discussed in [11], the circuit discussed in [14] and [16] are memory efficient. The Latency of [14] is the minimum among all of these. The circuit published in [15] is the first delay based design that attains minimum memory and minimum latency simultaneously. However, it requires a huge number of multiplexers. Commutator along with signal generators increase the number of circuit components also. All the above-discussed circuits find their application for calculating bit reversal in radix-2 FFT. The proposed approach is the first delay-based design that achieves minimum memory and latency with significantly less number of multiplexers in comparison to the previous designs. For a given parallelism, the number of registers in a cell increases with increase in the length of FFT. The proposed architecture can be scaled for different common radices such as radix-2, radix-4, radix-8, and other radices. For this purpose, input and output path reordering is done while keeping the cell exchange module intact. The size of a buffer in the bit exchange circuit is changed as per the requirement of different radices applications. Table V compares the synthesis and implementation results obtained for 4-parallel 64-point bit reversal circuits. The synthesis results is obtained with the help of Synopsys DC compiler. The area of the proposed circuit is found to be 15-20% lesser than that of [15] and [16]. The frequency of operation is also higher than that of [15] and [16]. There is a decrease in gate count by 15-20% and the reduced number of gate count reflects the reduction in circuit complexity. These circuits are implemented on Virtex-7 FPGA (field-programmable gate array) for analyzing power consumption at 250 MHz. As observed from the table, there is a 10-15% decrease in power consumption in the proposed design in comparison to [15] and [16]. While keeping the order of parallelism constant, circuit can be operated at optimum frequency for larger FFT length.

V. CONCLUSION

This brief has proposed an algorithm for reordering continuous flow parallel data of different radices and the corresponding circuit. The proposed circuit requires minimum memory and posses minimum latency. Simultaneously, the number of multiplexers and additional circuits is minimized. Necessary signals for controlling multiplexers are generated with the help of counter. There is a significant improvement in power, area, and frequency of operation in the proposed work. The circuit is simple and highly scalable for other radices. Moreover, the circuit supports higher-order parallelism and different radices that makes it highly useful for parallel pipelined FFT architectures.

REFERENCES

[1] B. Gold and C. M. Rader, *Digital Processing of Signals*. New York, NY, USA: McGraw-Hill, 1969.

[2] J. Rius and R. D. Porrata-Dòria, "New FFT bit-reversal algorithm," *IEEE Trans. Signal Process.*, vol. 43, no. 4, pp. 991–994, Apr. 1995.

[3] J. Prado, "A new fast bit-reversal permutation algorithm based on a symmetry," *IEEE Signal Process. Lett.*, vol. 11, no. 12, pp. 933–936, Dec. 2004.

[4] Z. Zhang and X. Zhang, "Cache-optimal methods for bit-reversals," in *Proc. ACM/IEEE Conf. Supercomput.*, Portland, OR, USA, 1999, pp. 1–18.

[5] K. S. Gatlin and L. Carter, "Memory hierarchy considerations for fast transpose and bit-reversals," in *Proc. Int. Symp. High Perform. Comput. Archit.*, Orlando, FL, USA, 1999, pp. 33–42.

[6] T. C. Choinski and T. T. Tylaska, "Generation of digit reversed address sequences for fast Fourier transforms," *IEEE Trans. Comput.*, vol. 40, no. 6, pp. 780–784, Jun. 1991.

[7] 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, Dec. 2008.

[8] F. Kristensen, P. Nilsson, and A. Olsson, "Flexible baseband transmitter for OFDM," in *Proc. IASTED Int. Conf. Circuits Signals Syst.*, Cancún, Mexico, 2003, pp. 356–361.

[9] W. Gao, S. Kwong, and H. Sang, "Low-cost memory data scheduling method for reconfigurable FFT bit-reversal circuits," *Electron. Lett.*, vol. 51, no. 3, pp. 217–219, 2015.

[10] 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, Oct. 2011.

[11] S. Yoshizawa, A. Orikasa, and Y. Miyanaga, "An area and power efficient pipeline FFT processor for 8×8 MIMO-OFDM systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, Rio de Janeiro, Brazil, 2011, pp. 2705–2708.

[12] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined radix2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[13] 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, Apr. 2013.

[14] 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.

[15] W. Li, F. Yu, and Z. Ma, "Efficient circuit for parallel bit reversal," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 4, pp. 381–385, Apr. 2016.

[16] M. Garrido, "Multiplexer and memory-efficient circuits for parallel bit reversal," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 4, pp. 657–661, Apr. 2019.