Multiplexer and Memory-Efficient Circuits for Parallel Bit Reversal

Mario Garrido

Multiplexer and Memory-Efficient Circuits for Parallel Bit Reversal

Mario Garrido, *Member, IEEE*

***Abstract*—This paper presents novel circuits for calculating the bit reversal on parallel data. The circuits consist of delays/memories and multiplexers, and have the advantage that they requires the minimum number of multiplexers among circuits for parallel bit reversal so far, as well as a small total memory.**

***Index Terms*—Bit Reversal, FFT, Pipelined Architecture**

I. Introduction

BIT REVERSAL [1] is a type of bit-dimension permutation [2], [3] that permutes a set of indexed data according to a reversing of the bits of the index. Its main use is to sort out the outputs of the fast Fourier transform (FFT), which are generally provided in the so called bit-reversed order.

In the last years, researchers have provided efficient circuits to calculate the bit reversal of a continuous data flow arriving in series [4] or in parallel [5]–[11]. The calculation of parallel bit reversal has become popular due to the increasing number of high throughput FFT hardware architectures that have been proposed in the last years. These architectures process a continuous flow of data arriving in parallel and demand a bit reversal circuit with the same parallelization in order to keep the continuous flow.

There are numerous strategies to implement the parallel bit reversal. They depend on the type of elements used to store the data, i.e., delays or memories, and on how the bit reversal permutation is split in other sub-permutations. Previous works have focused on reducing the amount of delays/memory or the number of multiplexers. However, the minimization of the delays/memory results in a larger number of multiplexers, whereas the minimization of the multiplexers results in a larger memories.

In this work, we present memory and multiplexer-efficient circuits for parallel bit-reversal. Thus, the paper explores how to achieve a small memory and a small number of multiplexers simultaneously. This is done by studying the permutation that the bit reversal of parallel data carries out, as well as the alternatives of using delays and memories. As a result, the brief proposes two efficient circuits for $N > P^2$ and $N \leq P^2$, respectively, where N is the number of data involved in the permutation and P is the number of parallel paths. These circuits achieve both small memory usage and small multiplexer usage.

This brief is organized as follows. In section II, we review the concepts related to bit-dimension permutations that are needed for this brief. In section III, we review and classify the previous approach for parallel bit reversal. In section IV, we develop the proposed parallel bit reversal circuits. In section V, we compare the proposed designs to previous approaches. Finally, in section VI, we collect the main conclusions of the paper.

M. Garrido is with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, e-mail: mario.garrido.galvez@liu.se

This work was supported by the Swedish ELLIIT Program.

II. Review of bit-dimension permutations

This section reviews the main ideas related to bit-dimension permutations, which are required for understanding this paper. For a detailed description of bit-dimension permutations, the reader is encouraged to read [2].

Bit-dimension permutations apply to a set of $N = 2^n$ data, $n \in \mathbb{N}$, in an n-dimension space with dimensions $x_{n-1}x_{n-2}\dots x_0$, where the only possible coordinates for each dimension are 0 or 1, i.e., $x_i \in \{0, 1\}$. In this context, a bit-dimension permutation defines a reordering of the data according to a permutation of the n bits of the dimensions [12]. This allows for defining a permutation operation on a set of n bits instead of defining it for 2^n values, which is most times mathematically inaccessible [3].

The dimensions of the space can be serial or parallel. $P = 2^p$ is the number of samples that flow in parallel. As the total amount of data is N, those data are provided in N/P clock cycles in series.

In this context, a bit-dimension permutation, σ, is a function represented by

$$\sigma(u_{n-1}\dots u_p|u_{p-1}\dots u_0) = u'_{n-1}\dots u'_p|u'_{p-1}\dots u'_0, \quad (1)$$

which transforms the vector $u_{n-1}u_{n-2}\dots u_p|u_{p-1}\dots u_0$ into a new vector $u'_{n-1}u'_{n-2}\dots u'_p|u'_{p-1}\dots u'_0$. Here, the $n-p$ rightmost dimensions are serial and p leftmost dimensions are parallel. The vertical bar (|) is used to separate them.

A. Elementary bit-exchange

An elementary bit-exchange (EBE) [3] is a bit-dimension permutation that only exchanges two dimensions. For example, the permutation defined by $\sigma(u_2u_1u_0) = u_2u_0u_1$ is an elementary bit-exchange of dimensions x_1 and x_0. Alternatively, an elementary bit-exchange of dimensions x_j and x_k can be represented as [4]

$$\sigma : x_j \leftrightarrow x_k. \quad (2)$$

The number of delays (D), the number of multiplexers (M) and the latency (Lat) of circuits that calculate an elementary bit exchange are summarized in Table I. This table is brought from [2] and the description of the circuits can be found in [2], [13]. The circuits for elementary bit-exchange are classified into circuits that exchange two serial dimensions, i.e., serial-serial (ss), two parallel dimensions, i.e., parallel-parallel (pp)

TABLE I
COSTS OF ELEMENTARY BIT-EXCHANGES

	Elementary Bit-Exchange		
Cost	Serial-serial (ss)	Parallel-parallel (pp)	Serial-parallel (sp)
D	$2^j - 2^k$	0	2^j
Lat	$(2^j - 2^k)/2^p$	0	$2^j/2^p$
M	2^{p+1}	$0^\star$	2^p

*: Some pp permutations may require multiplexers.

or a serial and a parallel dimensions, i.e., serial-parallel (sp). Note that the costs in Table I depend on this classification.

B. Bit reversal

Bit reversal is a specific bit-dimension permutation that flips the dimensions. For serial data, i.e., when there is no parallel dimension, the bit reversal corresponds to the permutation

$$\sigma(u_{n-1}u_{n-2}\dots u_0) = u_0 \dots u_{n-2}u_{n-1} \tag{3}$$

For parallel data, the bit reversal is represented as

$$\sigma(u_{n-1}\dots u_p|u_{p-1}\dots u_0) = u_0 \dots u_{n-p-1}|u_{n-p}\dots u_{n-1} \tag{4}$$

III. PREVIOUS APPROACHES FOR PARALLEL BIT REVERSAL

Previous approaches for parallel bit reversal are characterized by the type of permutations that they carry out and by the storage elements that they use, i.e., memories or delays.

Fig. 1 shows the different approaches that have been proposed depending on the type of permutations. The permutation blocks in the figure have several stages with ss, or pp or sp permutations. The first approach consists of the composition of the permutations ss-pp-ss [7], the second approach is pp-ss-pp [5]–[9], the third approach is ss-sp [11] and the forth one is ss-pp-ss-pp [10].

Regarding the storage elements, each ss permutation can be carried out either by a memory or by a circuit for serial-serial permutation [2], [4]. The latter consists of a buffer and two multiplexers. Each pp permutation admits two possibilities: Either it only requires to interconnect each input to the corresponding output by a wire [2], which does not require any hardware, or uses a set of multiplexers in parallel to route the input data [5]–[9]. The former case occurs when all the values coming from the same input follow the same path, whereas the latter case occurs when they may follow different paths. Finally, each sp permutation consists of a set of buffers and multiplexers [2]. The buffers may be implemented using delays or memory [14].

Previous ss-pp-ss and pp-ss-pp approaches use a set of buffers for the pp permutation and a memory for the ss permutation [5]–[9]. Conversely, previous ss-sp and ss-pp-ss-pp approaches use delays for ss and sp permutations [10], [11].

IV. PROPOSED BIT REVERSAL CIRCUITS

The proposed approach is based on carrying out elementary bit-exchanges of pairs of dimensions. These pairs are x_{n-1} and x_0, x_{n-2} and x_1, x_{n-3} and x_2, etc. This results in flipping the order of the bits, which is the essence of bit reversal.

Fig. 2 shows some cases on how the parallel bit reversal permutation may look like. In all cases the permutation consist of flipping the order of the bits. However, the type of permutations that must be carried out are different depending on the relation between N and P.

When $N = P^2$, half of the dimensions are serial and the other half are parallel. In this case, all the pairs of dimensions to be exchanged include a serial dimension and a parallel one. For instance, x_{n-1} is serial as it is to the left of the bar (|), whereas x_0 is parallel.

When $N < P^2$, there are more parallel dimensions than serial dimensions. Thus, all the serial dimensions are exchanged with parallel ones and, additionally, there are some parallel dimensions that are exchanged with other parallel dimensions. This results in sp and pp permutations.

When $N > P^2$, there are more serial dimensions than parallel dimensions. Therefore, the p upper serial dimensions are exchanged with parallel dimensions, and there are also some serial dimensions that are exchanged with other serial dimensions. This results in sp and ss permutations.

These cases are studied in the next sections.

A. Bit reversal circuits for $N \leq P^2$

If $N \leq P^2$, the bit reversal of parallel data is described as

$$\begin{aligned}\sigma(u_{n-1}\dots u_p|u_{p-1}\dots u_{n-p}u_{n-p-1}\dots u_0) = \\ u_0\dots u_{n-p-1}|u_{n-p}\dots u_{p-1}u_p\dots u_{n-1}\end{aligned} \tag{5}$$

If we split this permutation into the sp and the pp permutations, we obtain

$$\begin{aligned}\sigma_1(u_{n-1}\dots u_p|u_{p-1}\dots u_{n-p}u_{n-p-1}\dots u_0) = \\ u_{n-1}\dots u_p|\underline{u_{n-p}\dots u_{p-1}}u_{n-p-1}\dots u_0\end{aligned} \tag{6}$$

$$\begin{aligned}\sigma_2(u_{n-1}\dots u_p|u_{p-1}\dots u_{n-p}u_{n-p-1}\dots u_0) = \\ \underline{u_0\dots u_{n-p-1}}|u_{p-1}\dots u_{n-p}\underline{u_p\dots u_{n-1}}\end{aligned} \tag{7}$$

where σ_1 is pp and σ_2 is sp. Note that both permutation deal with different dimensions. Therefore, we can calculate first any of the permutations and then the other one, i.e.,

$$\sigma = \sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1 \tag{8}$$

The permutation σ_1 is a pp permutation with no hardware cost. How to implement it is described in [2]. The permutation σ_2 consists of p sp elementary bit exchanges. Its total number of delays according to Table I is

$$D = \sum_{j=p}^{n-1} 2^j = \frac{2^p - 2^n}{1-2} = N - P \tag{9}$$

Likewise, the number of multiplexers is

$$M = \sum_{j=p}^{n-1} 2^p = (n-p)\cdot 2^p = P \cdot \log_2\left(\frac{N}{P}\right) \tag{10}$$

and the latency is

$$\text{Lat} = D/P = \frac{N}{P} - 1 \tag{11}$$

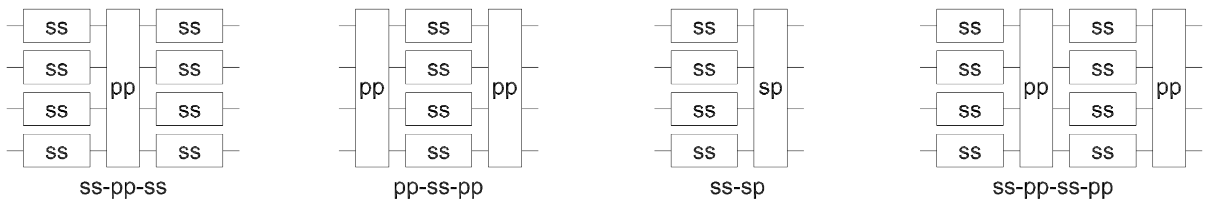

Fig. 1. Approaches for parallel bit reversal.

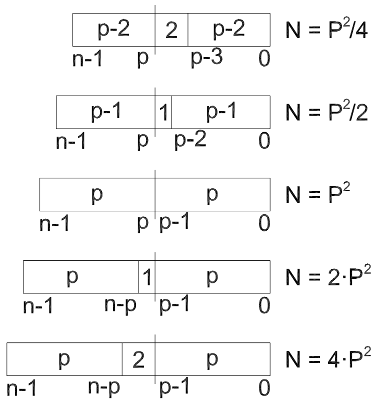

Fig. 2. Cases for the parallel bit reversal permutation.

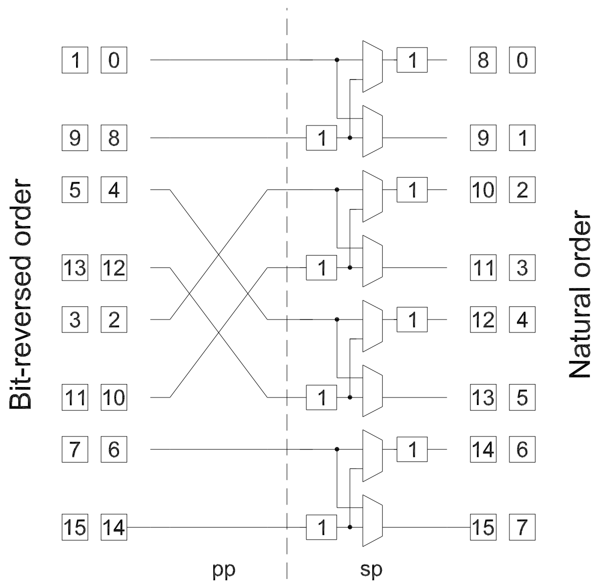

Fig. 3. Parallel bit reversal circuit for $N = 16$ and $P = 8$.

B. Implementation of the bit reversal circuits for $N \leq P^2$

The circuit in Fig. 3 calculates the parallel bit reversal for $N = 16$ and $P = 8$. The proposed circuit includes a pp permutation and a sp one. It requires 8 delays, 8 multiplexers and has a latency of 1 clock cycle, which matches equations (9), (10) and (11).

For other values of N and P, once the permutations have been identified, the paper [2] describes how to implement the elementary bit exchanges.

Note also that the buffers can alternatively be grouped in memories [14]. However, as $N \leq P^2$, the length of the buffers is small, so it is generally preferred to use delays.

C. Bit reversal circuits for $N > P^2$

If $N > P^2$, the bit reversal for parallel data is described as

$$\begin{aligned}\sigma(u_{n-1}\dots u_{n-p}u_{n-p-1}\dots u_p|u_{p-1}\dots u_0) =\\ u_0\dots u_{p-1}u_p\dots u_{n-p-1}|u_{n-p}\dots u_{n-1}\end{aligned} \tag{12}$$

If we split this permutation into the ss and the sp permutations, we obtain

$$\begin{aligned}\sigma_1(u_{n-1}\dots u_{n-p}u_{n-p-1}\dots u_p|u_{p-1}\dots u_0) =\\ u_{n-1}\dots u_{n-p}\underline{u_p\dots u_{n-p-1}}|u_{p-1}\dots u_0\end{aligned} \tag{13}$$

$$\begin{aligned}\sigma_2(u_{n-1}\dots u_{n-p}u_{n-p-1}\dots u_p|u_{p-1}\dots u_0) =\\ \underline{u_0\dots u_{p-1}}u_{n-p-1}\dots u_p|\underline{u_{n-p}\dots u_{n-1}}\end{aligned} \tag{14}$$

where σ_1 is ss and σ_2 is sp. If this case, the order of the permutations is also arbitrary, i.e., $\sigma = \sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1$.

The number of delays for the sp permutation is

$$D_{sp} = \sum_{j=n-p}^{n-1} 2^j = \frac{2^{n-p} - 2^n}{1-2} = N - \frac{N}{P} \tag{15}$$

Likewise, the number of multiplexers is

$$M_{sp} = \sum_{j=n-p}^{n-1} 2^p = p \cdot P = P \cdot \log_2 P \tag{16}$$

and the latency is

$$\text{Lat}_{sp} = D/P = \frac{N}{P} - \frac{N}{P^2} \tag{17}$$

The ss permutation σ_1 is calculated by a bank of memories. As σ_1 does not affect neither depends on the parallel dimensions, all the memories carry out the same permutation. This permutation flips the order of the $n - 2p$ lower serial dimensions. Therefore, the operation that each memory carries out is the serial bit reversal of 2^{n-2p} data. When data is written in memory in natural order, the bit reversal is calculated by reading the data in bit reversed order. Likewise, if data is stored in bit-reversed order, then the bit reversal of the data is achieved by reading the data from memory in natural order. In this way, a memory whose address alternates between natural and bit reversed order calculates the bit reversal of the input data. The size of each memory is equal to

$$\text{Mem Size}_{ss} = 2^{n-2p} = \frac{N}{P^2} \tag{18}$$

As there is one memory per parallel branch, the number of memories is

$$\#\text{Mem}_{ss} = P \tag{19}$$

and the total memory for the ss permutation is

$$\text{Mem}_{ss} = \text{Mem Size}_{ss} \cdot \#\text{Mem}_{ss} = \frac{N}{P} \tag{20}$$

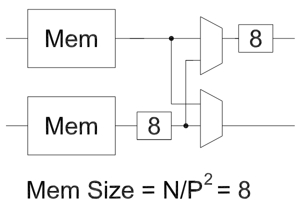

Fig. 4. Parallel bit reversal circuit for $N = 32$ and $P = 2$.

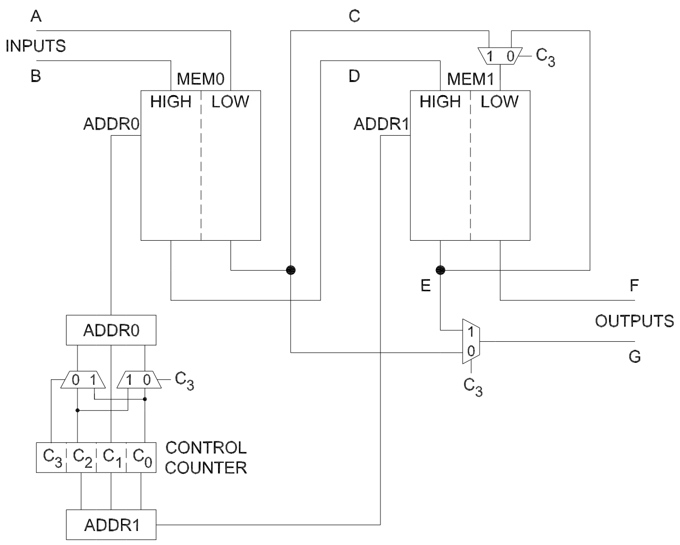

Fig. 5. Implementation of the parallel bit reversal circuit for $N = 32$ and $P = 2$.

Regarding latency, as the read and write addresses are the same, the latency of each memory is equal to its size, i.e.,

$$\text{Lat}_{ss} = \frac{N}{P^2} \tag{21}$$

Finally, the memories do not include any multiplexer, as the permutation is done by modifying the read and write addresses of the memories.

By combining the cost of the sp and the ss permutation, we obtain the total memory cost of the proposed approach, which results in

$$\text{Mem} = D_{sp} + \text{Mem}_{ss} = N \tag{22}$$

The multiplexers come only from the sp permutation and the total latency is

$$\text{Lat} = \text{Lat}_{sp} + \text{Lat}_{ss} = \frac{N}{P} \tag{23}$$

D. Implementation of the bit reversal circuits for $N > P^2$

Fig. 4 shows the parallel bit reversal circuit for $N = 32$ and $P = 2$. It includes two memories and a circuit for sp permutation. As $N > P^2$, the buffers of the sp permutation are always large. Therefore, it is preferable to implement them by using memories instead of delays. Likewise, the addresses for the two memories in the figure are always the same. This means that they can be grouped in a single memory.

As a result, Fig. 5 shows the detailed implementation of the circuit in Fig. 4. The circuit only includes two memories, a control counter and two multiplexers, considering that the area of the multiplexers coming from the control counter is negligible, because they are 1-bit multiplexers. In the circuit, MEM0 groups the two memories in Fig. 4. Its address ADDR0 is used for reading and writing and it is generated from the bits of the counter. Note that it consists of the counter bits $C_2C_1C_0$ during 8 clock cycles and $C_0C_1C_2$ during the following 8 clock cycles. This allows for calculating the bit reversal of groups of 8 inputs. The second memory, MEM1, calculates the sp permutation. This memory acts as a buffer of length 8 and combines the two buffers in Fig. 4. Note also that the control of the circuit is simple, as it is easily obtained from the bits of a counter.

TABLE II
TIMING DIAGRAM OF THE PARALLEL BIT REVERSAL CIRCUIT FOR $N = 32$ AND $P = 2$ IN FIG. 5

Counter	Addresses		Inputs		Internal signals			Outputs	
$C_3C_2C_1C_0$	ADDR0	ADDR1	A	B	C	D	E	F	G
0000	000	000	0	16	-	-	-	-	-
0001	001	001	8	24	-	-	-	-	-
0010	010	010	4	20	-	-	-	-	-
0011	011	011	12	28	-	-	-	-	-
0100	100	100	2	18	-	-	-	-	-
0101	101	101	10	26	-	-	-	-	-
0110	110	110	6	22	-	-	-	-	-
0111	111	111	14	30	-	-	-	-	-
1000	000	000	1	17	0	16	-	-	-
1001	100	001	9	25	2	18	-	-	-
1010	010	010	5	21	4	20	-	-	-
1011	110	011	13	29	6	22	-	-	-
1100	001	100	3	19	8	24	-	-	-
1101	101	101	11	27	10	26	-	-	-
1110	011	110	7	23	12	28	-	-	-
1111	111	111	15	31	14	30	-	-	-
0000	000	000	-	-	1	17	16	0	1
0001	001	001	-	-	3	19	18	2	3
0010	010	010	-	-	5	21	20	4	5
0011	011	011	-	-	7	23	22	6	7
0100	100	100	-	-	9	25	24	8	9
0101	101	101	-	-	11	27	26	10	11
0110	110	110	-	-	13	29	28	12	13
0111	111	111	-	-	15	31	30	14	15
1000	000	000	-	-	-	-	17	16	17
1001	100	001	-	-	-	-	19	18	19
1010	010	010	-	-	-	-	21	20	21
1011	110	011	-	-	-	-	23	22	23
1100	001	100	-	-	-	-	25	24	25
1101	101	101	-	-	-	-	27	26	27
1110	011	110	-	-	-	-	29	28	29
1111	111	111	-	-	-	-	31	30	31

The timing diagram for the circuit in Fig. 5 is shown in Table II. Apart from the control signals, inputs and outputs, it includes some internal signals to keep track of the data. Note that the input is received in bit-reversed order and the output is provided in natural order.

V. COMPARISON

Table III compares proposed and previous approaches to calculate the parallel bit reversal. Previous approaches that use memories are pp-ss-pp [5]–[9] or ss-pp-ss [7], whereas

TABLE III

COMPARISON OF APPROACHES FOR CALCULATING THE PARALLEL BIT REVERSAL.

Approach	Type	Perm.	Delays/Memory	Multiplexers	Latency	Th.	Note
Yoshizawa [5]	Memory	pp-ss-pp	PN	$2P^2-2P$	N/P	P	
Püschel [6]	Memory	pp-ss-pp	$2N$	$2P\log_2 P$	N/P	P	
Serre [7]	Memory	pp-ss-pp	$2N$	$P\log_2 P$	N/P	P	RAM/SNW/RAM
Chen [8]	Memory	pp-ss-pp	N	$2P^2-2P$	$\lesssim N/P$	P	
Serre [7]	Memory	ss-pp-ss	N	$2P\log_2 P$	N/P	P	SNW/RAM/SNW
Chen [9]	Memory	pp-ss-pp	N	$2P\log_2 P$	N/P	P	
Li [10]	Delays	ss-pp-ss-pp	$N-2\sqrt{N}+P$	$2P^2$	$N/P-2\sqrt{\frac{N}{P^2}}+1$	P	For even N
Li [10]	Delays	ss-pp-ss-pp	$N-\sqrt{2N}-\sqrt{\frac{N}{2}}+P$	$2P^2$	$N/P-\sqrt{\frac{2N}{P^2}}-\sqrt{\frac{N}{2P^2}}+1$	P	For odd N
Cheng [11]	Delays	ss-sp	$N-2\sqrt{N}+P$	$P\log_2(\frac{N}{P})$	$N/P-2\sqrt{\frac{N}{P^2}}+1$	P	For $N>P^2$ & even N
Cheng [11]	Delays	ss-sp	$N-\sqrt{2N}-\sqrt{\frac{N}{2}}+P$	$P\log_2(\frac{N}{2P})$	$N/P-\sqrt{\frac{2N}{P^2}}-\sqrt{\frac{N}{2P^2}}+1$	P	For $N>P^2$ & odd N
Proposed	Any	pp-sp	$N-P$	$P\log_2(\frac{N}{P})$	$N/P-1$	P	For $N\leq P^2$
Proposed	Memory	ss-sp	N	$P\log_2 P$	N/P	P	For $N>P^2$

TABLE IV

BIT REVERSAL IMPLEMENTATION FOR $N=4096$, $P=4$, AND $WL=32$.

Approach	LUTs	FFs	Slices	BRAM	f_{CLK} (MHz)
[8]*	4806	190	-	0	250
[10]	4776	78	-	0	250
Proposed	739	156	245	8	280

* : According to the implementation provided in [10].

previous approaches based on delays use ss-pp-ss-pp [10] or ss-sp [11]. By contrast, the proposed approach uses memories and ss-pp for $N > P^2$, and pp-sp for $N \leq P^2$.

Regarding delays/memory, previous approaches based on delays [10], [11] achieve the theoretical minimum, and some approaches based on memories [7]–[9] achieve a reasonable and slightly larger memory size of N. The proposed approach also requires a memory size of N.

Regarding multiplexers, some approaches [5], [8], [10] require a number of multiplexers proportional to P^2. Other approaches [6], [7], [9] reduce the complexity to $2P\log_2 P$. Finally, only [7] and the proposed approach reduce the complexity to $P\log_2 P$, whereas the complexity of [11] is in most cases larger, as it depends on N and $N > P^2$.

By considering delays/memories and multiplexers, the advantage of the proposed approach for $N > P^2$ is that it is the only approach that reduces the memory to N addresses and the number of multiplexers to $P\log_2 P$. Thus, it minimizes memory and multiplexers simultaneously. For $N \leq P^2$ the proposed approach reduces the amount of memory and multiplexers even further.

Finally, the latency is similar in all the approaches with small variations, and the throughput (Th.) of all the approaches is the same, as all of them process P data in parallel in a continuous flow.

Regarding experimental results, Table IV compares implementations of bit reversal circuits on a Virtex-7 XC7VX330T-3-FFG1157 FPGA. The results are obtained for $N = 4096$, $P = 4$ and 32 bits of word length (WL). Compared to previous approaches, the proposed solution is more compact, as it makes use of BRAM memory instead of a large amount of distributed logic.

VI. CONCLUSIONS

This brief proposes multiplexer and memory-efficient circuits for parallel bit reversal. They are the result of a good selection of the sub-permutations used to calculate the bit reversal and an efficient use of the delays/memories and multiplexers. For $N > P^2$, the proposed circuits use an ss-sp permutation and require a memory of N addresses and $P\log_2 P$ multiplexers. For $N \leq P^2$, the circuits carry out a pp-sp permutation and require delays/memories of a total size of $N - P$, and $P\log_2(N/P)$ multiplexers. This represents small memory and small number of multiplexers simultaneously.

REFERENCES

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

[2] M. Garrido and J. Grajal, "Optimum circuits for bit-dimension permutations," *IEEE Trans. VLSI Syst.*, Under review.

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

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

[5] S. Yoshizawa, A. Orikasa, and Y. Miyanaga, "An area and power efficient pipeline FFT processor for 88 MIMO-OFDM systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2011, pp. 2705–2708.

[6] M. Püschel, P. A. Milder, and J. C. Hoe, "Permuting streaming data using RAMs," *J. ACM*, vol. 56, no. 2, pp. 10:1–10:34, Apr. 2009.

[7] F. Serre, T. Holenstein, and M. Püschel, "Optimal circuits for streamed linear permutations using RAM," in *Proc. ACM/SIGDA Int. Symp. FPGAs*. ACM, Feb. 2016, pp. 215–223.

[8] S.-J. Huang, S.-G. Chen, M. Garrido, and S.-J. Jou, "Continuous-flow parallel bit-reversal circuit for MDF and MDC FFT architectures," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 10, pp. 2869–2877, Oct. 2014.

[9] R. Chen and V. K. Prasanna, "Optimal circuits for parallel bit reversal," in *Proc. IEEE Design Automation Conf.*, Jun. 2017, pp. 1–6.

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

[11] C. Cheng and F. Yu, "An optimum architecture for continuous-flow parallel bit reversal," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2334–2338, Dec 2015.

[12] D. Fraser, "Array permutation by index-digit permutation," *J. Assoc. Comp. Machinery (ACM)*, vol. 23, no. 2, pp. 298–309, Apr. 1976.

[13] M. Garrido, "Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time," Ph.D. dissertation, Universidad Politécnica de Madrid, Dec. 2009.

[14] M. Garrido, M. Acevedo, A. Ehliar, and O. Gustafsson, "Challenging the limits of FFT performance on FPGAs," in *Int. Symp. Integrated Circuits*, Dec. 2014, pp. 172–175.