A Serial Commutator Fast Fourier Transform Architecture for Real-Valued Signals

Mario Garrido, *Member, IEEE*, Nanda K. Unnikrishnan, and Keshab K. Parhi, *Fellow, IEEE*

***Abstract*—This brief presents a novel pipelined architecture to compute the fast Fourier transform of real input signals in a serial manner, i.e., one sample is processed per cycle. The proposed architecture, referred to as real-valued serial commutator, achieves full hardware utilization by mapping each stage of the fast Fourier transform (FFT) to a half-butterfly operation that operates on real input signals. Prior serial architectures to compute FFT of real signals only achieved 50% hardware utilization. Novel data-exchange and data-reordering circuits are also presented. The complete serial commutator architecture requires $2\log_2 N - 2$ real adders, $\log_2 N - 2$ real multipliers, and $N + 9\log_2 N - 19$ real delay elements, where N represents the size of the FFT.**

***Index Terms*—Fast Fourier transform (FFT), pipelined architecture, real-valued signals, serial commutator (SC).**

I. Introduction

The design of VLSI architectures for implementation of the fast Fourier transform (FFT) has been of great interest. Various types of pipelined FFT architectures have been presented. They include pipelined-serial [1]–[12] and pipelined-parallel [12]–[19] architectures. The inherent hardware inefficiency of the serial architectures is due to the fact that the input signal of an N-point FFT can be sampled in N cycles whereas the butterfly operations can be computed in $N/2$ cycles. Therefore, butterflies are only used half of the time, i.e., they have 50% hardware utilization. This inefficiency does not exist in parallel architectures where the N samples are processed in N/P clock cycles and the N/P butterflies are computed in N/P clock cycles, where P is the number of parallel samples. This achieves full hardware utilization of the butterflies, as they are used 100% of the time. Such parallel architectures have been presented in [14]–[17]. In recent work, an architecture called serial commutator (SC) FFT was presented in [10]. This architecture processes input samples in a serial manner with full hardware utilization where each sage of the FFT is mapped to a half-butterfly operation in hardware that requires 2 adders instead of the usual 4 adders of a butterfly for complex-valued data.

Manuscript received August 4, 2017; accepted September 4, 2017. Date of publication September 18, 2017; date of current version October 29, 2018. This work was supported by the Swedish ELLIIT Program. This brief was recommended by Associate Editor L.-P. Chau. *(Corresponding author: Mario Garrido.)*

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

N. K. Unnikrishnan and K. K. Parhi are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: unnik005@umn.edu; parhi@umn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2017.2753941

For real-valued input data, various architectures have been presented in the last years [11], [18], [19]. As for complex-valued FFTs, butterflies in parallel architectures achieve a hardware utilization of 100% [18]. As data are real, the full hardware utilization in parallel real-valued FFTs is achieved with 2 adders instead of 4 needed in a complex butterfly operation. Conversely, current serial real-valued FFT architectures suffer from hardware underutilization [11]. The reason is the same as in complex-valued serial FFTs: Butterflies only need to be used half of the time.

In this brief we present a novel architecture, referred to as *Real-valued Serial Commutator* (RSC) FFT, to compute the FFT of a real signal where the input is processed in a serial manner. Unlike prior architectures, the proposed architecture achieves full hardware utilization where each stage of the FFT is mapped to a half-butterfly that operates on real inputs. This corresponds to only 1 adder/subtractor per stage of the FFT. It may be noted that such an architecture has never been presented in past literature.

This brief is organized as follows. In Section II we review the real-valued FFT. In Section III we introduce the circuits for data permutation used in the proposed architecture. In Section IV we present the RSC FFT hardware architecture. In Section V we compare the proposed RSC FFT to previous serial FFTs in the literature. In Section VI we explain the data reordering circuit for the proposed architecture. Finally, in Section VII we summarize the main conclusions of the brief.

II. The Real-Valued FFT

Fig. 1 shows the flow graph of a 16-point real-valued FFT. The flow graph consists of $n = \log_2 N$ stages. Each stage includes butterflies and rotations. The rotations are represented by the numbers in between the stages, ϕ, where each value ϕ corresponds to a rotation by:

$$e^{-j\frac{2\pi}{N}\phi} \tag{1}$$

The numbers at the input of the flow graph are the time index of the signal $x[n]$. The numbers at the output represent the frequencies k of the output $X[k]$. Finally, boxed numbers are the index (Idx) that we use to refer to the data at different stages of the FFT.

The real-valued FFT differs from a complex-valued FFT in two important facts [18]: First, all the inputs are real. Therefore, the flow graph only includes operations with real data until the edges arrive to a rotation. This corresponds to the part of the flow graph that is over the dashed boxes. Second, the real-valued FFT has the property that the spectrum is symmetric, i.e., $X[N-k] = X^*[k]$. Based on this property half of the output frequencies do not need to be calculated. This allows for simplifying the flow graph, as is done in the figure.

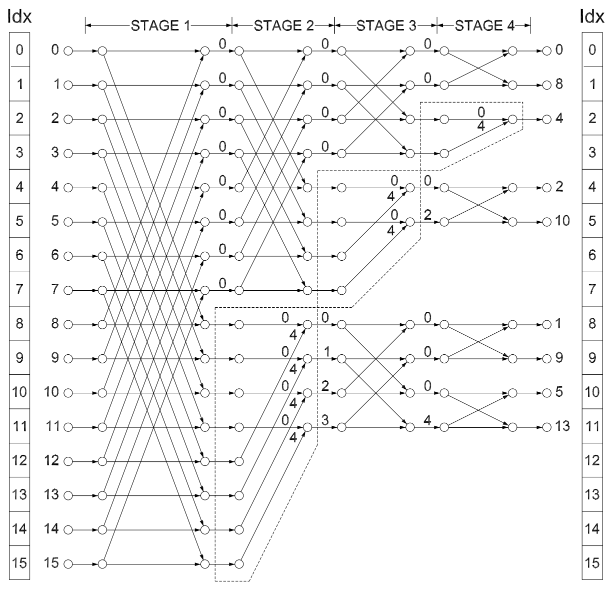


Fig. 1. Flow graph of a 16-point real-valued FFT.

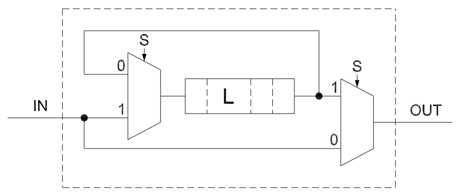


Fig. 2. Circuit for bit-dimension permutation of serial data.

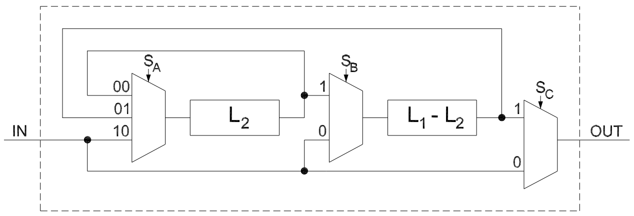


Fig. 3. Circuit for multiple-delay (L_1 and L_2) exchange.

III. Circuits for Data Permutation

The real-valued SC FFT uses two types of circuits for data permutation. First, Fig. 2 shows the basic circuit for bit-dimension permutation of serial data. This circuit has already been used in previous SC FFT architectures [10] and in circuits for bit reversal [20]. The circuit consists of a buffer of length L and two multiplexers controlled by the control signal S. The signal S is obtained directly from the bits of an n-bit counter $c_{n-1}, \dots, c_0$ that counts from 0 to $N-1$. For a buffer of length $L = 2^i - 2^j$, the control signal S is $S = \overline{c_i}$ OR c_j. From a data management point of view, this circuit simply interchanges two samples arriving in times t_0 and $t_1 = t_0 + L$ if $S = 0$. If no interchange is needed ($S = 1$), data just passes through the buffer. The circuit has a latency of L clock cycles.

Second, for the RSC FFT we also need to interchange samples with different separations between them. For this purpose, we use the circuit in Fig. 3, which is just an extension of the circuit of Fig. 2. If we need to interchange samples separated by L_1 clock cycles, we activate $S_A = 01$, $S_B = 1$ and $S_C = 0$. If we want to interchange samples separated by L_2 clock cycles, we activate $S_A = 00$, $S_B = 0$ and $S_C = 1$. Finally, if we want that samples pass through the buffers, then $S_A = 10$, $S_B = 1$ and $S_C = 1$. In the case of two different separations, L_1 and L_2, the circuit has a buffer of length L_2 and a buffer of length $L_1 - L_2$. Note that L_1 is the largest among both separations, i.e., $L_1 > L_2$.

IV. The Real-Valued SC FFT

A. The RSC FFT Architecture

Fig. 4 shows the proposed RSC FFT hardware architecture for $N = 16$. Each stage consists of butterflies, rotators and circuits for data management. Butterflies and rotators are marked with 1/4 as they only require 1/4 of the resources of a complex butterfly or rotator: 1 adder instead of 4 for the butterflies and 1 multiplier instead of 4 for the rotators. Note also that the first and last stages do not need any rotator. The circuits for data management are of the types explained in Section III. Thus, they are used to interchange pairs of data separated a number of clock cycles.

The structure of the butterflies is shown in Fig. 5. Fig. 5(a) shows the butterfly of the first stage of the architecture. It only consists of a real adder, two registers and a multiplexer. Following the idea in [10], the circuit operates on samples that arrive in consecutive clock cycles: First it calculates the addition of the butterfly, and then the subtraction in the next clock cycle. The difference in this case is that data are real instead of complex. The timing diagram of the circuit is shown in Table I. It can be observed that the butterfly has a latency of one clock cycle.

The butterfly in Fig. 5(b) is used in stages 2 to 4. The only difference with respect to Fig. 5(a) is that the butterfly in Fig. 5(b) allows to bypass data. It calculates a butterfly most of the times, but is some cases data are bypassed. The latency of this butterfly is also 1 clock cycle.

Fig. 6 shows the rotator used in the proposed RSC FFT hardware architecture. The rotator has three different configurations: To calculate a rotation of consecutive samples, to calculate a rotation of samples separated by two clock cycles and to bypass the input data. The bypass is the lower edge, whereas the rest of the circuit calculates the rotation.

A timing diagram for the rotator is shown in Table II. In this example, the rotator bypasses x_0, x_2, x_{8r} and x_{8i}, rotates x_4 together with x_6, which are separated one clock cycle, and rotates together x_{10r} and x_{10i}, which are separated two clock cycles. The values of the control signals are shown in the table. Note that for x_4 and x_6 what is calculated is $(x_4 - jx_6) \cdot (\cos(\alpha_1) + j\sin(\alpha_1))$. This is the operation that occurs inside the dashed boxes of Fig. 1. Finally, the latency of the rotator is 2 clock cycles.

For a general N, the hardware components of the architecture are calculated as follows. The architecture has one real multiplier per rotator and there are rotators in all stages except the first and the last one. Therefore, the total number of real multipliers is $\log_2 N - 2$. Regarding adders, one real adder per butterfly and one real adder per rotator are needed. Therefore, the total number of real adders in the architecture is $2\log_2 N - 2$. Finally, the memory for permutations is calculated as:

$$\mathrm{Mem}_{\mathrm{Perm}} = \log_2 N - 1 + 2 + \sum_{s=3}^{n-1} \left(2^s - 1\right) = N - 4 \quad (2)$$

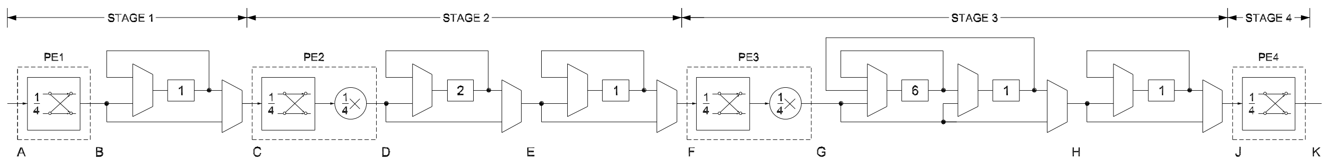


Fig. 4. Proposed 16-point real-valued SC FFT.

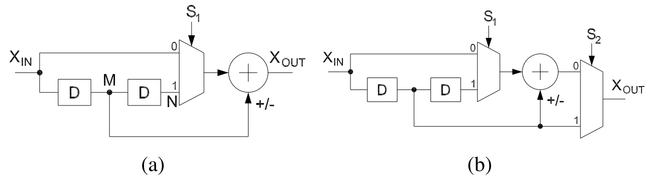


Fig. 5. Butterflies in the proposed real-valued SC FFT. (a) Basic butterfly (stage 1). (b) Butterfly with bypass (rest of stages).

TABLE I
TIMING DIAGRAM OF THE BUTTERFLY IN FIG. 5(A)

TIME	X_{IN}	M	N	S_1	X_{OUT}
0	x_0	-	-	-	-
1	x_8	x_0	-	0	x_0+x_8
2	x_4	x_8	x_0	1	x_0-x_8
3	x_{12}	x_4	x_8	0	x_4+x_{12}
4	x_2	x_{12}	x_4	1	x_4-x_{12}

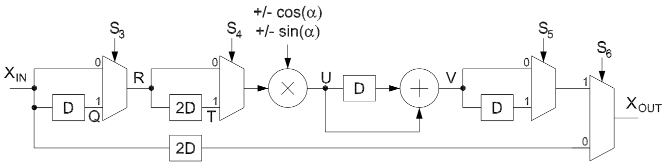


Fig. 6. Rotator in the proposed real-valued SC FFT. This rotator corresponds to stage 3 in Fig. 7. The rotator for stage 2 is similar, with the difference that $S_3 = 0$ all the time. Thus, the first multiplexer and the buffer can be omitted.

If we add the internal memory in butterflies and rotators, the total memory of the architecture is $N + 9\log_2 N - 19$ real values.

B. Data Management of the RSC FFT Architecture

Fig. 7 shows the data management for the RSC FFT architecture in Fig. 4. The stages and the letters *A* to *K* on the bottom of both figures match. They are used to indicate different points in the circuit. In vertical, Fig. 7 shows the order of arrival of the data at those points of the circuit. The first sample that arrives at a certain point of the circuit is in the upper edge, whereas the last one is in the lower edge. The exact time of arrival is shown in red under the letter *t*. The time of arrival of the first sample at each point of the circuit correspond to the total latency of the previous elements of the circuit, as shown at the bottom of the figure. For instance, the first sample that arrives at the input of stage 2, indicated by letter *C*), does it at $t = 2$, and this time is the sum of the latency of the butterfly and the shuffling circuit of the first stage, between points *A* and *B*, and *B* and *C*, respectively. Notice that the latency of the butterflies is 1 clock cycle, the latency of the rotators is 2 clock cycles, and the latency of the shuffling circuits corresponds to the lengths of the buffers in Fig. 4. Finally, the figure includes the data index (*Idx*) that corresponds to Fig. 1 in black color, and the output frequencies, *k*, in blue color. For instance, the sample with index $Idx = 6$ at the beginning of stage 3 (letter *F*) arrives at time $t = 11$ and is the fourth sample to arrive at that point of the circuit.

At stage 1, all pairs of data that must be processed together in the butterfly arrive in consecutive clock cycles. According to Fig. 1 these are the pairs with *Idx* 0 and 8, 1 and 9, and so on. Samples 0 and 8 arrive at times 0 and 1, and samples 1 and 9 arrive at times 8 and 9. To process them, the butterfly in Fig. 5(a) described before is used. The delay of the butterfly is 1 clock cycle. Therefore, the variable *t* at *B* is increased by 1 with respect to the values at *A*.

Between *B* and *C* there is a permutation circuit with buffer length $L = 1$. Therefore, it is used to exchange samples that arrive in consecutive clock cycles. This happens, for instance, to the data with indexes 8 and 4, which arrive at *B* at $t = 2$ and 3, respectively.

The butterfly and rotator computations of stage 2 happen between *C* and *D*. Some data are bypassed by the butterfly such as 8 and 12, which are rotated instead. Other data, such as 0 and 4 are mixed in the butterfly and then bypassed in the rotator. Notice that this corresponds to the computations for these samples in Fig. 1.

The shuffling circuits at stage 2 are between *D* and *E*, and *E* and *F*. They are used to interchange data separated 2 and 1 clock cycles, respectively.

At stage 3, most of the input samples require the calculation of the butterfly, some consecutive samples are rotated and some samples separated 2 clock cycles are rotated, as shown in Fig. 7. Therefore, all the configurations of the butterfly and the rotator explained in the previous section are used. Note that Table II describes the behavior of the rotator at stage 3.

The permutation circuit between *G* and *H* allows for interchanging data separated 6 or 7 clock cycles. This can be noticed in Fig. 7.

Finally, stage 4 uses a butterfly with bypass, where only samples with index 2 and 3 bypassed.

V. COMPARISON

Table III compares FFT architectures for processing serial data. The table is divided into architectures for complex-valued data and architectures for real-valued data. Most architectures are for complex-valued data, but there exist a couple of previous architectures that process serial real-valued data. The table includes the area in terms of real adders, real multipliers and real sample memory, as well as the performance in terms of latency and throughput. As all the architectures process serial data, their throughput is 1 sample per clock cycle.

To calculate the real adders and the real multipliers, we have considered that a complex rotator consists of four real multipliers and two real adders. In the table, the adders in the

TABLE II
TIMING DIAGRAM OF THE ROTATOR IN FIG. 6

TIME	X_{IN}	Q	S_3	R	T	S_4	U	V	S_5	S_6	X_{OUT}
0	x_0	-	-	-	-	-	-	-	-	0	-
1	x_2	x_0	-	-	-	-	-	-	0	1	-
2	x_4	x_2	0	x_4	-	0	$x_4\cos(\alpha_1)$	-	-	0	x_0
3	x_6	x_4	0	x_6	-	0	$x_6\sin(\alpha_1)$	$x_4\cos(\alpha_1)+x_6\sin(\alpha_1)$	-	0	x_2
4	x_{8r}	x_6	-	-	x_4	1	$x_4\sin(\alpha_1)$	-	1	1	$x_4\cos(\alpha_1)+x_6\sin(\alpha_1)$
5	x_{10r}	x_{8r}	-	-	x_6	1	$-x_6\cos(\alpha_1)$	$x_4\sin(\alpha_1)-x_6\cos(\alpha_1)$	0	1	$x_4\sin(\alpha_1)-x_6\cos(\alpha_1)$
6	x_{8i}	x_{10r}	1	x_{10r}	-	0	$x_{10r}\cos(\alpha_2)$	-	-	0	x_{8r}
7	x_{10i}	x_{8i}	0	x_{10i}	-	0	$-x_{10i}\sin(\alpha_2)$	$x_{10r}\cos(\alpha_2)-x_{10i}\sin(\alpha_2)$	0	1	$x_{10r}\cos(\alpha_2)-x_{10i}\sin(\alpha_2)$
8	x_1	x_{10i}	-	-	x_{10r}	1	$x_{10r}\sin(\alpha_2)$	-	-	0	x_{8i}
9	x_3	x_1	-	-	x_{10i}	1	$x_{10i}\cos(\alpha_2)$	$x_{10r}\sin(\alpha_2)+x_{10i}\cos(\alpha_2)$	0	1	$x_{10r}\sin(\alpha_2)+x_{10i}\cos(\alpha_2)$

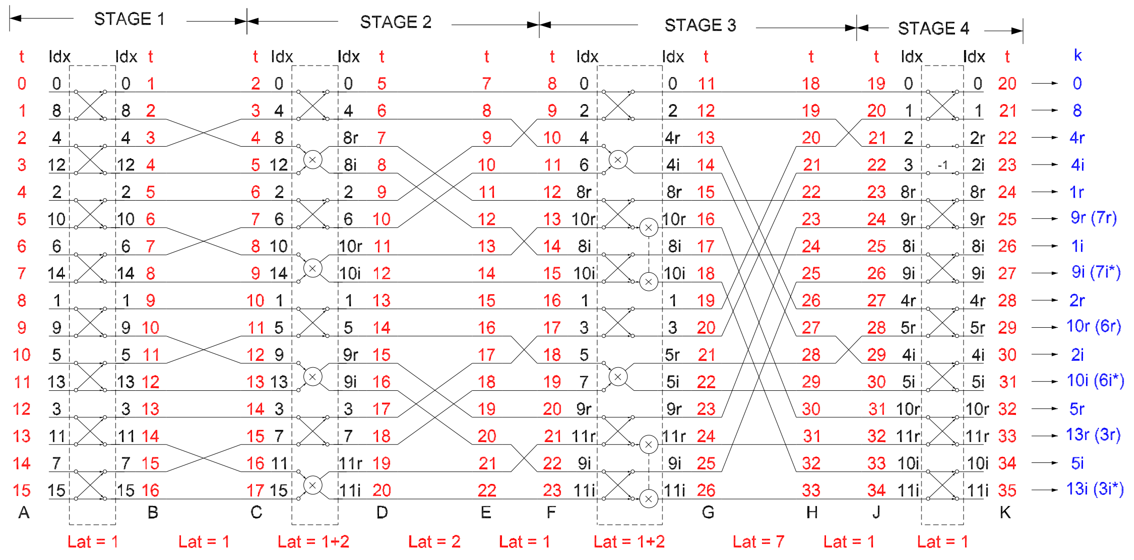


Fig. 7. Data management of the proposed 16-point SC FFT for real-valued signals.

TABLE III
COMPARISON OF PIPELINED HARDWARE ARCHITECTURES FOR THE COMPUTATION OF AN N-POINT FFT ON SERIAL DATA

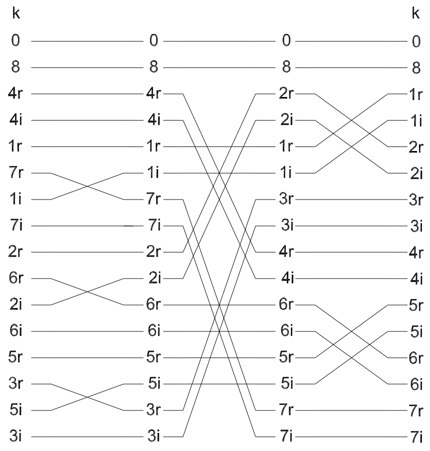
PIPELINED ARCHITECTURE	AREA			PERFORMANCE	
	Real Adders	Real Multipliers	Real Data Memory	Latency (cycles)	Throughput (samples/cycle)
ARCHITECTURES FOR COMPLEX-VALUED DATA					
SDF Radix-2, [12]	$6\log_2 N-4$	$4\log_2 N-8$	$2N$	N	1
SDF Radix-2, [1]	$3\log_2 N-2$	$2\log_2 N-4$	$8N/3$	$4N/3$	1
SDF Radix-4, [2], [3]	$9\log_2 N-2$	$2\log_2 N-4$	$2N$	N	1
SDF Radix-2^2, [12]	$5\log_2 N-2$	$2\log_2 N-4$	$2N$	N	1
SDF Split-radix, [4]	$5\log_2 N-2$	$2\log_2 N-4$	$2N$	N	1
SDC Radix-2, [5], [6]	$4\log_2 N-4$	$4\log_2 N-8$	$6N/2$	$3N/2$	1
SDC Radix-2, [8]	$4\log_2 N-4$	$4\log_2 N-8$	$6N/2$	$3N/2$	1
SDC Radix-4, [7]	$4\log_2 N-2$	$2\log_2 N-4$	$4N$	N	1
SDC-SDF Radix-2, [9]	$3\log_2 N$	$2\log_2 N-4$	$6N/2$	$3N/2$	1
SC Radix-2 [10]	$3\log_2 N-2$	$2\log_2 N-4$	$\approx 2N$	N	1
ARCHITECTURES FOR REAL-VALUED DATA					
SDF, Radix-2 Hybrid [11]	$6\log_2 N-10$	$4\log_2 N-12$	$5N/4-2$	N	1
SDF, Radix-2 Fully-real [11]	$4\log_2 N-6$	$4\log_2 N-12$	$3N/2-5$	N	1
RSC Radix-2, Proposed	$2\log_2 N-2$	$\log_2 N-2$	$N+9\log_2 N-19$	N	1

rotators are summed to the adders of the butterflies to obtain the total real adders.

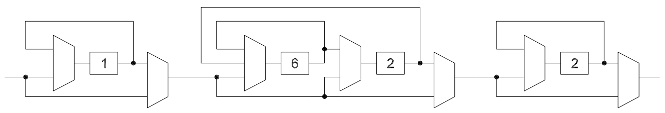
The table shows that the proposed RSC FFT architecture requires the least number of adders, multipliers and memory. Compared to most efficient FFTs for complex-valued data, the RSC FFT requires 2/3 adders, half of the multipliers and approximately half of the memory. Compared to previous architectures for real-valued data, the proposed RSC FFT almost halves the number of adders, divides the number of multipliers by four, and reduces the memory. Finally, the latency is similar as in previous FFT architectures. Note that in the proposed architecture, the data memory size is reduced

TABLE IV
COMPARISON OF 1024-POINT SERIAL PIPELINED REAL-VALUED FFT HARDWARE ARCHITECTURES

FFT	Tech.	Area (mm^2)		Power (mW)	
Architecture	(nm)	100 MHz	500 MHz	100 MHz	500 MHz
Hybrid FFT [11]	65	0.41	0.41	22.0	110.2
Fully Real FFT [11]	65	0.33	0.34	16.5	84.2
RSC FFT, Proposed	65	0.22	0.25	12.1	63.0



(a)



(b)

Fig. 8. Reordering of the output sequence. (a) Reordering procedure. (b) Reordering circuit.

by at least one-third with respect to previous works. For large N, these memory savings may reduce the area of the circuit significantly.

Table IV compares the proposed architecture to previous serial FFTs for real-valued data for $N = 1024$. The results in the table are obtained by using 65nm Low Power (LP) library at 1 V and 25 °C, and clock frequencies of 100 MHz and 500 MHz. The table shows improvements of 27% in area and 25% in power consumption with respect to previous approaches. The maximum clock frequency of the proposed architecture is $f_{CLK} = 500$MHz.

VI. Reordering the Output Data

The output data of the RSC FFT is provided in scrambled order. Fig. 8(a) shows the reordering procedure for $N = 16$ data. The output order before reordering is shown by the first column of numbers. The reordering consists of three stages of permutations. The first stage places together the real and imaginary components of each output frequency. The next two stages sort out the frequencies. The circuit that carries out the permutation is shown in Fig. 8(b). The first stage exchanges data separated by one clock cycle. The second stage interchanges data separated by 6 or 8 clock cycles and the third stage exchanges data separated by 2 clock cycles.

VII. Conclusion

This brief has presented the RSC FFT architecture. This architecture calculates the real-valued FFT in a continuous flow of one sample per clock cycle. The RSC FFT presents a novel data management scheme that allows for a high utilization of butterflies, rotators and memory. This allows to reduce the amount of these components with respect to previous serial FFT architectures in the literature.

References

[1] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 7, pp. 585–589, Jul. 2006.
[2] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 993–1001, Oct. 1974.
[3] M. A. Sánchez, M. Garrido, M. López-Vallejo, and J. Grajal, "Implementing FFT-based digital channelized receivers on FPGA platforms," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 4, pp. 1567–1585, Oct. 2008.
[4] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874, Mar. 2003.
[5] 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.
[6] Y.-N. Chang, "Design of an 8192-point sequential I/O FFT chip," in *Proc. World Congr. Eng. Comput. Sci.*, vol. 2. Oct. 2012, pp. 816–821.
[7] G. Bi and E. V. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 12, pp. 1982–1985, Dec. 1989.
[8] 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, Jan. 2011.
[9] Z. Wang, X. Liu, B. He, and F. Yu, "A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 5, pp. 973–977, May 2015.
[10] M. Garrido, S.-J. Huang, S.-G. Chen, and O. Gustafsson, "The serial commutator FFT," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 10, pp. 974–978, Oct. 2016.
[11] A. Chinnapalanichamy and K. K. Parhi, "Serial and interleaved architectures for computing real FFT," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Brisbane, QLD, Australia, Apr. 2015, pp. 1066–1070.
[12] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, Santa Clara, CA, USA, May 1998, pp. 131–134.
[13] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 6, pp. 451–455, Jun. 2010.
[14] M. Garrido, J. Grajal, M. A. Sánchez, 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.
[15] C. Cheng and K. K. Parhi, "High-throughput VLSI architecture for FFT computation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 10, pp. 863–867, Oct. 2007.
[16] M. Garrido, M. Acevedo, A. Ehliar, and O. Gustafsson, "Challenging the limits of FFT performance on FPGAs," in *Proc. Int. Symp. Integr. Circuits*, Singapore, Dec. 2014, pp. 172–175.
[17] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.
[18] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 12, pp. 2634–2643, Dec. 2009.
[19] S. A. Salehi, R. Amirfattahi, and K. K. Parhi, "Pipelined architectures for real-valued FFT and hermitian-symmetric IFFT with real datapaths," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 8, pp. 507–511, Aug. 2013.
[20] 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.