The Serial Commutator FFT

Mario Garrido Gálvez, Shen-Jui Huang, Sau-Gee Chen and Oscar Gustafsson

Journal Article

Mario Garrido Gálvez, Shen-Jui Huang, Sau-Gee Chen and Oscar Gustafsson, The Serial Commutator FFT, IEEE Transactions on Circuits and Systems - II - Express Briefs, 2016. 63(10), pp.974-978.
http://dx.doi.org/10.1109/TCSII.2016.2538119
Postprint available at: Linköping University Electronic Press

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-132529

The Serial Commutator (SC) FFT

Mario Garrido, *Member, IEEE*, Shen-Jui Huang, Sau-Gee Chen and Oscar Gustafsson, *Senior Member, IEEE*

***Abstract*—This paper presents a new type of FFT hardware architectures called serial commutator (SC) FFT. The SC FFT is characterized by the use of circuits for bit-dimension permutation of serial data. The proposed architectures are based on the observation that in the radix-2 FFT algorithm only half of the samples at each stage must be rotated. This fact, together with a proper data management makes it possible to allocate rotations only every other clock cycle. This allows for simplifying the rotator, halving the complexity with respect to conventional serial FFT architectures. Likewise, the proposed approach halves the number of adders in the butterflies with respect to previous architectures. As a result, the proposed architectures use the minimum number of adders, rotators and memory that are necessary for a pipelined FFT of serial data, with 100% utilization ratio.**

***Index Terms*—Serial Commutator (SC), FFT, Pipelined Architecture**

I. Introduction

THE fast Fourier transform (FFT) is one of the most important algorithms in signal processing. Many hardware FFT architectures have been proposed with the aims of speeding up the calculation of the FFT and reducing the amount of hardware resources.

Pipelined FFT architectures are the most common ones [1]–[9]. They process a continuous flow of data using a relatively small amount of resources. There are two main types of pipelined FFTs: Serial pipelined FFTs process one sample per clock cycle, whereas parallel pipelined FFTs process several samples in parallel per clock cycle.

Parallel FFT architectures have been widely developed. Nowadays, there exist multi-path delay commutator (MDC) FFT architectures that use the minimum amount of butterflies and memory, with 100% utilization ratio [1], as well as an efficient use of rotators.

Conversely, serial pipelined FFTs have not reached the efficiency of parallel ones yet. Typical radix-2 single-path delay feedback (SDF) FFTs [2] have a utilization ratio of 50% in butterflies and rotators. Other radices such as radix-4 [3], [4] and radix-2^2 [2] improve the use of rotators. However, they do not improve the efficiency of butterflies. Single-delay commutator (SDC) FFTs [5]–[8] improve the use of butterflies and rotators at the cost of larger memory. The same happens to the locally pipelined FFT [9]. Therefore, in all cases there is a trade-off among butterflies, rotators and memory.

M. Garrido and O. Gustafsson are with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, e-mail: mario.garrido.galvez@liu.se, oscar.gustafsson@liu.se

S.-J. Huang is with Novatek Corp., Hsinchu 300, Taiwan, e-mail: shenray.ee95g@g2.nctu.edu.tw

S.-G. Chen is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, e-mail: sgchen@cc.nctu.edu.tw

This work was supported by the Swedish ELLIIT Program.

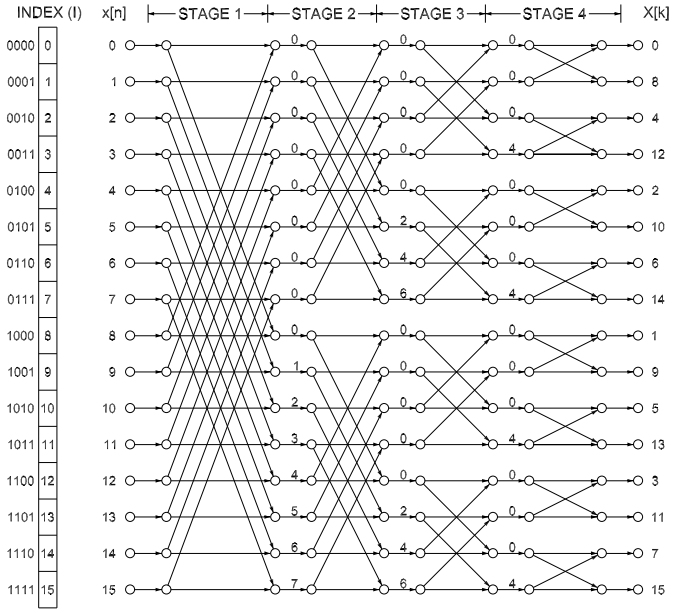

Fig. 1. Flow graph of a radix-2 DIF FFT.

This paper presents the serial commutator (SC) FFT. The SC FFT uses a novel data management based on circuits for bit-dimension permutation of serial data. The resulting SC FFT is the first one that requires the theoretical minimum amount of butterflies, rotators and memory with 100% utilization.

The paper is organized as follows. Section II reviews the FFT algorithm. Section III studies the theoretical boundaries of the hardware resources. Section IV presents in detail the SC FFT. Section V shows the case of natural I/O order. Section VI compares the proposed architectures to previous ones. Section VII presents experimental results. Finally, Section VIII summarizes the main conclusions of the paper.

II. The FFT Algorithm

The N-point DFT of an input sequence $x[n]$ is defined as:

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

where $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$.

In order to compute the DFT efficiently, the FFT based on the Cooley-Tukey algorithm [10] is most times used. The FFT reduces the number of operations from $O(N^2)$ for the DFT to $O(N \log N)$.

Figure 1 shows the flow graph of a 16-point radix-2 FFT decomposed according to decimation in frequency (DIF) [11]. The FFT is calculated in a series of $n = \log_\rho N$ stages, where ρ is the base of the radix, r, of the FFT, i.e. $r = \rho^\alpha$. In the figure, the numbers at the input represent the index of the input sequence, whereas those at the output are the frequencies, k.

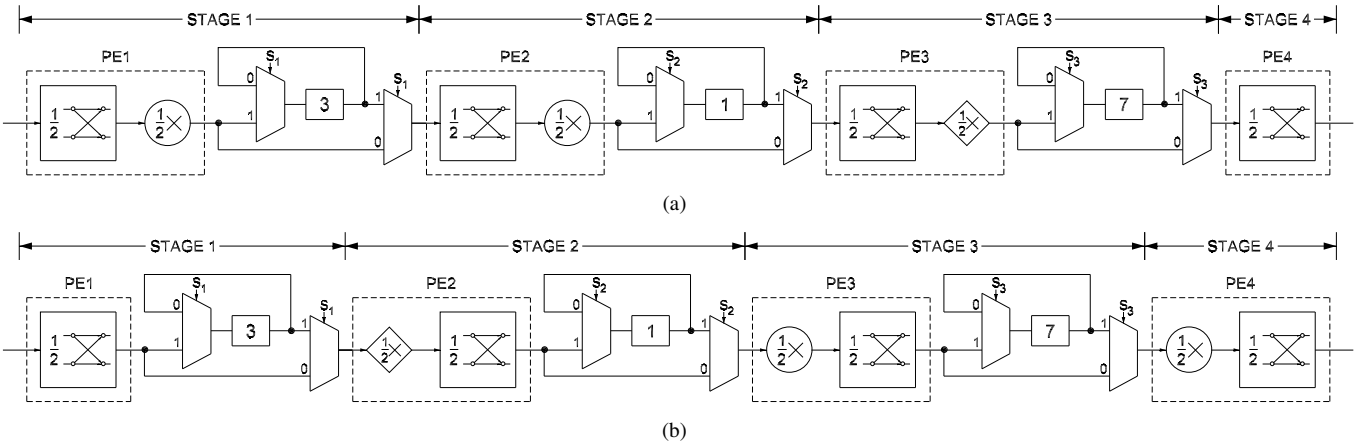

Fig. 2. Proposed serial feedforward architectures for the computation of a 16-point radix-2 FFT. (a) DIF. (b) DIT.

At each stage of the graph, $s \in \{1, \ldots, n\}$, butterflies and rotations are calculated. Specifically, each number, ϕ, in between the stages indicates a rotation by:

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

As a consequence, if $\phi = 0$ no rotation must be carried out. Likewise, rotations by $\phi \in [N/4, N/2, 3N/4]$ are trivial. This means that they can be carried out in hardware simply by interchanging the real and imaginary components and/or changing the sign of the data.

III. Theoretical Boundaries

The serial commutator (SC) FFT is based on a simple observation. In Figure 1 each stage calculates N complex additions and $N/2$ rotations. Therefore, any radix-2 FFT architecture that processes one sample per clock cycle only needs a complex adder and half a rotator per stage. This leads to $\log_2 N$ butterflies and $\log_4 N - 1$ rotators for the entire FFT, considering that the last stage does not have a rotator. These are the theoretical minimum number of resources for any radix-2 FFT that processes one sample per clock cycle.

Regarding memory, the theoretical minimum $N - P$ [1] for P-parallel data also holds for serial data where $P = 1$. Thus, the minimum memory for a serial FFT is $N - 1$.

IV. The Serial commutator FFT

Figures 2(a) and 2(b) show the proposed serial commutator FFT for $N = 16$ points and radix-2, respectively for DIF and DIT. The architectures consist of $n = \log_2 N = 4$ stages that include butterflies, rotators and circuits for data management. Rotators that carry out trivial rotations are diamond-shaped whereas general rotators are represented by a circle.

Both butterflies and rotators are marked with $1/2$. This means that they require half of the components in conventional butterflies and rotators: butterflies only use a real adder and a real subtracter instead of complex ones, and rotators use two real multipliers and one adder instead of four real multipliers and two adders. The half butterfly and half rotator form the processing element (PE) of the architecture, which is explained in detail in section IV-A.

The circuits for data permutation are circuits for elementary bit exchange. These circuits have already been used for the calculation of the bit reversal [12]. However, this is the first time that this type of circuits are used in an FFT architecture. The data management of the SC FFT is explained in section IV-B.

A. Processing Element

Figure 3 shows in detail the processing element (PE) used to calculate the butterflies and rotations of the radix-2 DIF SC FFT in Fig. 2(a). The processing element for the DIT SC FFT in Fig. 2(b) is analogous. The only difference is that the rotator is placed before the butterfly.

The PE is composed by the half butterfly and the half rotator. The PE does the calculation of a butterfly followed by a rotator:

$$\begin{aligned} Y_0 &= X_0 + X_1 \\ Y_1 &= (X_0 - X_1)e^{j\alpha} \end{aligned} \tag{3}$$

with the particularity that the inputs $X_0 = X_{R0} + jX_{I0}$ and $X_1 = X_{R1} + jX_{I1}$, and outputs $Y_0 = Y_{R0} + Y_{I0}$ and $Y_1 = Y_{R1} + Y_{I1}$, are provided in consecutive clock cycles.

Table I shows the timing diagram of the PE in Fig. 3. It can be observed that the butterfly operates first on the real part of the inputs and then on the imaginary part. The rotator also multiplexes the calculations in time. This allows for halving the adders and multipliers in the butterfly and rotator.

B. Data Management

The PE calculates a butterfly and a rotation on pairs of data that arrive in consecutive clock cycles. In order to fulfill this, the data management of the SC FFT places samples that must be operated together in consecutive clock cycles. This happens at all the stages of the FFT.

Figure 4 shows the data management of the SC FFTs in Fig. 2. The data management is the same for both DIF and DIF cases. Each column in Fig. 4 represents the input order to the corresponding stage. The order of arrival is from top to bottom. Therefore, $x[0]$ and $x[8]$ are the first and second inputs to the first stage, respectively. The figure shows that, at all the stages, consecutive samples are operated together in a butterfly. This allows for the use of the PE with half of the resources.

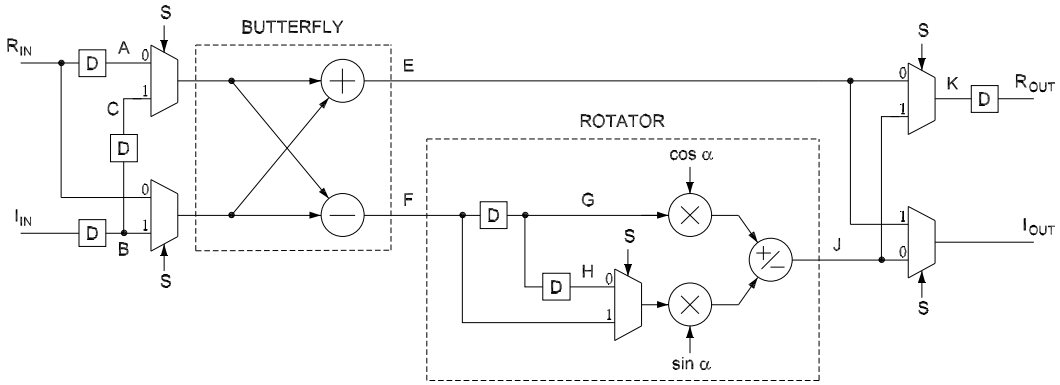

Fig. 3. Processing element for the calculation of butterflies and rotations of the radix-2 DIF SC FFT.

TABLE I
TIMING DIAGRAM OF THE PROCESSING ELEMENT.

TIME	R_{IN}	I_{IN}	S	E	F	J	K	R_{OUT}	I_{OUT}
0	X_{R0}	X_{I0}							
1	X_{R1}	X_{I1}	0	$Y_{R0} = X_{R0} + X_{R1}$	$Z_R = X_{R0} - X_{R1}$		Y_{R0}		
2			1	$Y_{I0} = X_{I0} + X_{I1}$	$Z_I = X_{I0} - X_{I1}$	$Y_{R1} = Z_R \cos\alpha - Z_I \sin\alpha$	Y_{R1}	Y_{R0}	Y_{I0}
3			0			$Y_{I1} = Z_R \sin\alpha + Z_I \cos\alpha$		Y_{R1}	Y_{I1}

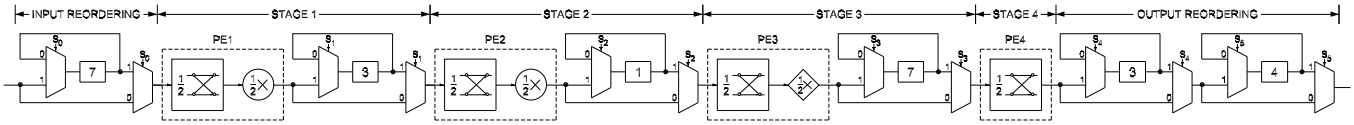

Fig. 6. Proposed 16-point DIF SC FFT with natural I/O order.

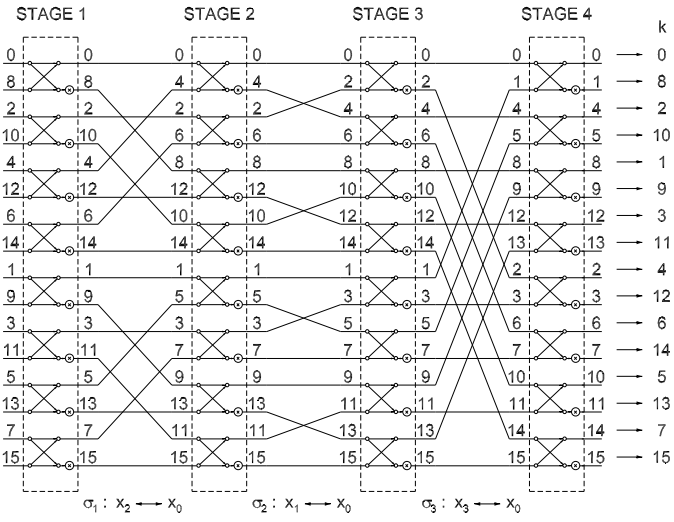

Fig. 4. Data management for the 16-point radix-2 DIF SC FFT.

In order to achieved the desired order, the SC FFT uses circuits for bit-dimension permutations of serial data, as shown in Fig. 5. These circuits interchange pairs of data delayed by L clock cycles. In Fig. 4, the first, second and third stages interchanges data separated 3, 1 and 7 clock cycles, respectively. These are equal to the lengths of the buffers of the three first stages in Fig. 2.

In a general case, for a SC FFT of length N, the length and

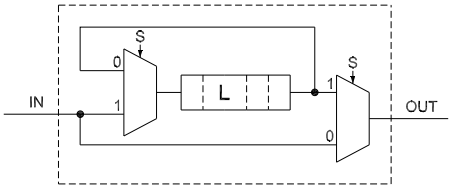

Fig. 5. Circuit for the elementary bit-exchange of serial dimensions.

delay of the buffers at stages $s = 1, \ldots, n-2$ is

$$L = 2^{n-s-1} - 2^0, \tag{4}$$

and $L = 2^{n-1}$ at stage $s = n-1$.

The control of the architecture is simple and obtained directly from the bits of an n-bit counter $c_{n-1}, \ldots, c_0$ that counts from 0 to $N-1$. For a buffer of length $L = 2^i - 1$, the control signal S_i is:

$$S_i = \overline{c_i} \text{ OR } c_0 \tag{5}$$

The control signals must be delayed according to the pipeline of the architecture, so that the count starts when the first sample arrives at the corresponding shuffling circuit.

The total amount of memory for the shuffling circuits can be obtained by adding the delays at all the stages. This leads to a total memory of

$$\sum_{i=1}^{n-1} 2^i - 2^0 = N - n - 1 \tag{6}$$

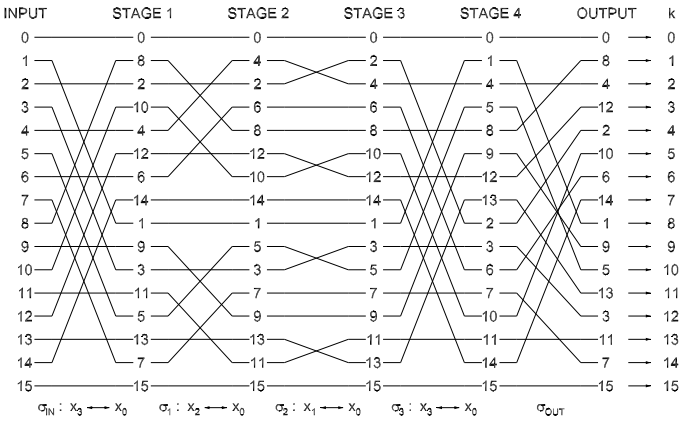

Fig. 7. Data management of a 16-point SC FFT for natural I/O order.

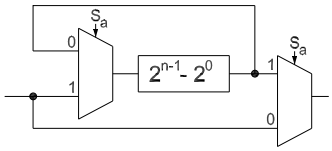

Fig. 8. Input reordering circuit for a natural I/O N-point SC FFT.

By adding the memory included in the processing elements, the total memory of the architecture is approximately N, which is the minimum for an N-point FFT.

As a result, the proposed SC FFT architectures use the minimum number of components for the butterflies, rotators and memory, with a utilization of 100%.

V. SC FFT Architectures for Natural I/O

The input and output orders of the SC FFT follow a sequence that is not in natural order, as shown in Fig. 4. In order to achieve natural I/O order, shuffling circuits can be added at the input and output. This is shown in Fig. 6 for a natural I/O 16-point SC FFT. The data management for the architecture in Fig. 6 is shown in Fig. 7.

In the general case of a natural I/O N-point SC FFT, the input reordering circuit only needs to calculate the elementary bit-exchange $\sigma : x_{n-1} \leftrightarrow x_0$. As explained in [12], this permutation requires a shuffling circuit with a buffer of length $L = 2^{n-1} - 2^0 = N/2 - 1$, as shown in Fig. 8. In our example in Fig. 6 for $N = 16$ the buffer length of the input reordering circuit is $L = 16/2 - 1 = 7$.

The output reordering circuit is more complex and requires $\lfloor (n+1)/2 \rfloor$ elementary bit-exchanges in series, as shown in Fig. 9. The first two elementary bit-exchanges have a buffer length of $L_a = 2^{n-2} - 2^0 = N/4 - 1$ and $L_b = 2^{n-1} - 2^{n-2} = N/4$, respectively. The following $\lfloor (n-3)/2 \rfloor$ elementary bit-exchanges have a buffer length $L_i = 2^{n-i-2} - 2^i$ for $i = 1, \ldots, \lfloor (n-3)/2 \rfloor$. In our example in Fig. 6 for $N = 16$ the number of elementary bit-exchanges in series is $\lfloor (n+1)/2 \rfloor = \lfloor (4+1)/2 \rfloor = 2$. Therefore it only includes the permutations with buffer lengths $L_a = N/4 - 1 = 16/4 - 1 = 3$ and $L_b = N/4 = 16/4 = 4$, as shown in Fig. 6.

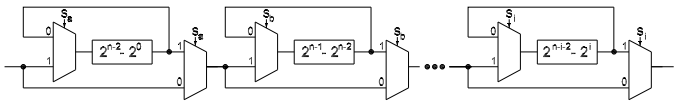

Fig. 9. Output reordering circuit for a natural I/O N-point SC FFT.

In a general case, the shuffling circuits for natural I/O add an overhead to the total memory of approximately $5N/4$, leading to a total memory of about $9N/4$.

VI. Comparison and Analysis

Tables II and III compare pipelined FFT architectures for serial data. Table II does not impose any specific order of inputs and outputs, whereas table III compares architectures for natural I/O order.

In table II, the first column shows the type of architecture. The second, third and fourth columns show the resources used by the architecture: rotators, adders and data memory. The last two columns compares the performance in terms of latency and throughput. As all the architectures that are compared process serial data, the throughput of all of them is 1 sample per clock cycle.

In table II, it can be observed that previous architecture require the minimum of some of the hardware resources, but not all of them. Various SDF FFT architecture [2]–[4], [13] use the minimum amount of rotators and memory. Previous SDC FFTs [5], [6], [8] achieve the minimum number of adders. And the locally pipelined FFT [9] achieves the minimum number of rotators and adders. Finally, the proposed SC FFT is the first architecture that achieves the minimum amount in all hardware resources.

For natural I/O order, table III compares previous SDC architectures to the proposed SC FFT. Compared to [5]–[7] the proposed architecture reduces the number of rotators by 50%. Furthermore, up to $N = 64$ points the memory of the proposed approach is also smaller than that in [5]–[7]. Compared to [8], the proposed architecture has less memory for $N \leq 64$, and more for larger N, being the differences small.

VII. Experimental results

The proposed SC FFT for $N = 1024$ points and word length 16 bits has been implemented on ASIC technology using the library UMC 55 nm process. Table IV compares the implementation with previous serial FFTs on ASICs. The proposed architecture improves the clock frequency of previous designs. At the same time it achieves less area than previous 2048-point [16] and 256-point [17] SDF FFTs, high SQNR and low power consumption. In the table, area and power are normalized to 55 nm and 0.9 V according to [18].

VIII. Conclusions

This paper has presented the serial commutator (SC) FFT architecture. This architecture is the first FFT to use circuits to calculate bit-dimension permutation on serial data. This creates a data management that allows for using the theoretical minimum amount of hardware resources for a serial FFT with 100% utilization. Compared to previous designs, the proposed

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

PIPELINED ARCHITECTURE	AREA			PERFORMANCE	
	Complex Rotators	Complex Adders	Complex Data Memory	Latency (cycles)	Throughput (samples/cycle)
SDF Radix-2, [2]	$2(\log_4 N - 1)$	$4(\log_4 N)$	N	N	1
SDF Radix-2, [9]	$\log_4 N - 1$	$2(\log_4 N)$	$4N/3$	$4N/3$	1
SDF Radix-4, [3], [4]	$\log_4 N - 1$	$8(\log_4 N)$	N	N	1
SDF Radix-2^2, [2]	$\log_4 N - 1$	$4(\log_4 N)$	N	N	1
SDF Split-radix, [13]	$\log_4 N - 1$	$4(\log_4 N)$	N	N	1
SDC Radix-2, [6], [7]	$2(\log_4 N - 1)$	$2(\log_4 N)$	$3N/2$	$3N/2$	1
SDC Radix-2, [5]	$2(\log_4 N - 1)$	$2(\log_4 N)$	$3N/2$	$3N/2$	1
SDC Radix-4, [14]	$\log_4 N - 1$	$3(\log_4 N)$	$2N$	N	1
SDC-SDF Radix-2, [8]	$\log_4 N - 1$	$2(\log_4 N) + 1$	$3N/2$	$3N/2$	1
Proposed SC Radix-2	$\log_4 N - 1$	$2(\log_4 N)$	N	N	1

TABLE III
COMPARISON OF PIPELINED HARDWARE ARCHITECTURES FOR THE COMPUTATION OF AN N-POINT FFT ON SERIAL DATA WITH NATURAL I/O.

PIPELINED ARCHITECTURE	AREA			PERFORMANCE	
	Complex Rotators	Complex Adders	Complex Data Memory	Latency (cycles)	Throughput (samples/cycle)
SDC Radix-2, [6], [7]	$2(\log_4 N - 1)$	$2(\log_4 N)$	$2N$	$2N$	1
SDC Radix-2, [5]	$2(\log_4 N - 1)$	$2(\log_4 N)$	$2N$	$2N$	1
SDC-SDF Radix-2, [8]	$\log_4 N - 1$	$2(\log_4 N) + 1$	$2N + 1.5\log_2 N - 1.5$	$2N$	1
Proposed SC Radix-2, even N	$\log_4 N - 1$	$2(\log_4 N)$	$9N/4 - 3\sqrt{N}/2 - 1$	$2N$	1
Proposed SC Radix-2, odd N	$\log_4 N - 1$	$2(\log_4 N)$	$9N/4 - \sqrt{2N} - 1$	$2N$	1

TABLE IV
COMPARISON OF SERIAL FFTS IMPLEMENTED ON ASICS.

	Proposed	[15]	[16]	[17]
FFT Size	1024	64	2048	256
Radix	2	2^3	2^2	2^4
Architecture	SC	SDF	SDF	SDF
Word length	16	16	10	-
Technology (nm)	55	180	350	180
Voltage (V)	0.9	-	2.7	1.8
Clk (MHz)	200	166	76	51.5
Area (mm^2)	0.15	0.47	7.58	-
Norm. Area	0.15	0.04	0.19	-
Gate Count	134066	-	-	173875
SQNR (dB)	55	-	45.3	-
Power (mW)	8.0	29.7	526	-
Norm. Power	8.0	-	9.18	-

SC FFT reduces either the number of rotator, or the number of adders or the memory of the design. A solution for natural I/O has also been presented, which offers comparable results to previous natural I/O FFTs. Finally, experimental results have been obtained to verify the architecture, leading to small area and low power consumption.

REFERENCES

[1] M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson, "'Pipelined Radix-2^k Feedforward FFT Architectures'," *IEEE Trans. VLSI Syst.*, vol. 21, pp. 23–32, Jan. 2013.

[2] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1998, pp. 131–134.

[3] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Comput.*, vol. C-23, pp. 993–1001, Oct. 1974.

[4] M. Sánchez, M. Garrido, M. López, 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.

[5] X. Liu, F. Yu, and Z. Wang, "'A pipelined architecture for normal I/O order FFT'," *Journal of Zhejiang University - Science C*, vol. 12, no. 1, pp. 76–82, Jan. 2011.

[6] Y.-N. Chang, "An efficient VLSI architecture for normal I/O order pipeline FFT design," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 12, pp. 1234–1238, Dec. 2008.

[7] ——, "Design of an 8192-point sequential I/O FFT chip," in *Proc. World Congress Eng.Comp. Science*, vol. II, Oct. 2012.

[8] Z. Wang, X. Liu, B. He, and F. Yu, "A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT," *IEEE Trans. VLSI Syst.*, vol. 23, no. 5, pp. 973–977, May 2015.

[9] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 7, pp. 585–589, Jul. 2006.

[10] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, 1965.

[11] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Prentice-Hall, 1989.

[12] 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.

[13] W.-C. Yeh and C.-W. Jen, "'High-speed and low-power split-radix FFT'," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 864–874, Mar. 2003.

[14] G. Bi and E. Jones, "'A pipelined FFT processor for world-sequential data'," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 1982–1985, Dec. 1989.

[15] T. Adiono, M. S. Irsyadi, Y. S. Hidayat, and A. Irawan, "'64-point fast efficient FFT architecture using radix-2^3 single path delay feedback'," *Proc. Int. Conf. Electrical Engineering Informatics*, vol. 02, pp. 654 – 658, Aug. 2009.

[16] T. Lenart and V. Öwall, "Architectures for dynamic data scaling in 2/4/8K pipeline FFT cores," *IEEE Trans. VLSI Syst.*, vol. 14, no. 11, pp. 1286–1290, Nov. 2006.

[17] C.-P. Fan, M.-S. Lee, and G.-A. Su, "A low multiplier and multiplication costs 256-point FFT implementation with simplified radix-2^4 SDF architecture," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, Dec. 2006, pp. 1935 – 1938.

[18] T. Ahmed, M. Garrido, and O. Gustafsson, "A 512-point 8-parallel pipelined feedforward FFT for WPAN," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Nov. 2011, pp. 981–984.