# Energy-efficient fast Fourier transform for real-valued applications

Eleftheriadis, C., & Karakonstantis, G. (2022). Energy-efficient fast Fourier transform for real-valued applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *69*(5), 2458-2462. https://doi.org/10.1109/TCSII.2022.3163280

**Published in:**
IEEE Transactions on Circuits and Systems II: Express Briefs

**Document Version:**
Peer reviewed version

# Energy-Efficient Fast Fourier Transform for Real-Valued Applications

Charalampos Eleftheriadis and Georgios Karakonstantis, *Senior Member, IEEE*

***Abstract*—This brief presents a new energy efficient Fast-Fourier Transform (FFT) architecture for real-valued applications. The proposed architecture decimates the FFT in time domain with bit-reversed inputs which allows to avoid the use of all costly complex FFTs operations required by the existing schemes. This leads to the reduction of the required memory by a factor of 2 while processing two inputs in parallel, thus doubling the throughput and improving the energy efficiency compared to the current real-valued FFT designs. Furthermore, the output frequencies are computed at their natural order by using a novel memory management technique, without requiring any reordering circuit unlike existing works. In summary for a $N$ point FFT the proposed architecture leads to an increased throughput of 2 samples per clock cycle, requiring $N-2$ memory cells, $8logN-8$ real adders and $3logN-4$ real multipliers. Our results show that we can achieve up to 46.86% energy savings when compared with recent real-valued FFT architectures.**

***Index Terms*—Energy efficient FFT, pipelined FFT, real valued FFT, fast Fourier transform architecture, single path delay feedback FFT.**

## I. Introduction

THE FAST Fourier Transform (FFT) [1] is considered one of the most important signal processing algorithms and it is currently used in many applications [2], most of which are executed on resource and power constrained devices [3]. In general, the input (time domain) and output (frequency domain) data of the FFT are complex and various pipelined complex-valued FFT architectures were proposed in the past [4]. However, in most modern applications such as bio-signal processing [5] or telecommunications [6], the inputs are real-valued, which urged many recent works to try to simplify the FFT architecture [7], [8], [9] by exploiting the conjugate symmetry of the output frequencies. The pipelined architectures [4], [7], [8], [9], process the data either in a serial or in a parallel manner [10], [11] and each one uses the same hardware elements; a butterfly unit, a rotator that is used for complex multiplication, and various memory elements. The most resource consuming parts are the rotator and the memory which add combinational and sequential logic overhead respectively. Therefore, most of the current architectures aim at minimizing the number of these elements and maximizing their usage at every clock cycle [8], [9].

While very interesting the most previously introduced real-valued architectures [7], [8], [9], [10], [11] utilized a radix-2, radix-$2^3$ or radix-$2^4$ decimation in frequency (DIF) algorithm with a bit-reversed or natural order input and output sequence. In contrast by using a decimation in time (DIT) algorithm with a bit-reversed ordered inputs and natural ordered outputs as stated in [12] we are allowed to expose the conjugate symmetry of each FFT stage in a better way, thus reusing more efficiently the same hardware units while avoiding extra circuitry at the output.

In this brief we are proposing a Real-valued Single-Path Delay Feedback Decimation in Time (RSDF-DIT) FFT architecture that exploits the odd conjugate symmetry at each stage by utilizing a specific DIT method with bit-reversed inputs and natural outputs [12]. Therefore, combining this DIT algorithm with a "pseudo complex" representation for the real outputs, we can enable parallel processing of two inputs, thus improving the throughput and energy efficiency unlike any existing real-valued FFT (RFFT) architectures [7], [8], [9], [10], [11]. In summary, the contributions of this brief can be summarized as follows:

1) Develop a decimation in time FFT architecture that fully exploits the odd conjugate symmetry and eliminates the need for complex operations utilizing the real-value nature of the processed data.
2) Double the throughput compared to existing, state-of-the art RFFT architectures [8], [9] by processing two real-valued inputs in parallel, which is allowed by the adoption of the DIT topology and utilization of the pseudo-complex operations.
3) Introduce a novel low-cost reordering scheme for the output data, which is integrated within the main architecture, eliminating the need for any additional circuitry as opposed to existing RFFT architectures [8], [9], [10], [11]. This helps to circumvent the extra cost due to the additional units needed for realizing the proposed parallel processing.
4) Implement the proposed architecture in 45nm process technology and evaluate the energy savings. Our results show up to 46.86% energy savings compared to the recently introduced architectures in [8], [9], at a cost of only 11.49% area (due to the additional multiplier units).

Manuscript received February 11, 2022; accepted March 14, 2022. Date of publication March 30, 2022; date of current version May 3, 2022. This work was supported by the European Union's Horizon 2020 Programme through the Marie Sklodowska-Curie Grant under Agreement 945231. This brief was recommended by Associate Editor Y.-P. Lin. *(Corresponding author: Charalampos Eleftheriadis.)*

The authors are with the Institute of Electronics, Communications and Information Technology, Queen's University Belfast, Belfast BT3 9DT, U.K. (e-mail: celeftheriadis01@qub.ac.uk; g.karakonstantis@qub.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2022.3163280.

Digital Object Identifier 10.1109/TCSII.2022.3163280

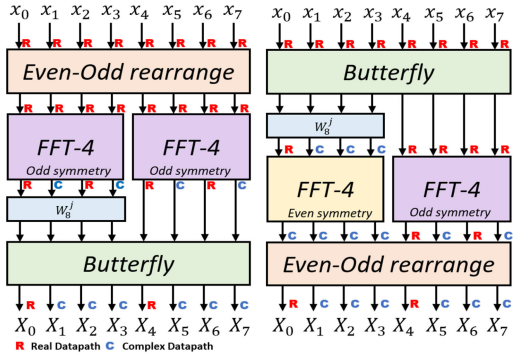


Fig. 1. a) DIT FFT flow graph b) DIF FFT.

The rest of this brief is organized as follows. Section II describes the background of the RFFT and highlights the challenges of the previous works. Section III presents our proposed approach; the RSDF-DIT FFT. Section IV compares our work with the previous ones in terms of energy savings and number of hardware units. Section V draws the final conclusions.

## II. Background & State of the Art

In general the $N$ point DFT can be generalized using the triangular matrices introduced in [12] where a vast amount of FFT algorithms can be generated by moving the FFT twiddle factors among various stages and the reordering circuits of the input and output data. The most commonly used FFT configurations are the radix-2 DIF (Eq. (1)) with natural ordered inputs and bit-reversed outputs and the radix-2 DIT (Eq. (2)) with bit-reversed inputs and natural ordered outputs; with the twiddle factors $W_N^k$ given by Eq. (3):

$$X(k) = \begin{cases} DFT_{N/2}\big[x(n) + x(n+N/2)\big], & \text{if } k = even \\ DFT_{N/2}\big[(x(n) - x(n+N/2))W_N^n\big], & \text{otherwise} \end{cases} \quad (1)$$

$$X(k) = DFT_{N/2}\Big[x(2n) + W_N^k x(2n+1)\Big] \quad (2)$$

$$W_N^k = cos(2\pi k/N) - isin(2\pi k/N). \quad (3)$$

### A. State of the Art: Real Valued FFTs

Fig. 1. depicts the data flow diagrams of both DIT and DIF methods in case of an 8-point input set. As it can be observed a DIF FFT processes (Fig. 1b.) the input data at their natural order by the so-called butterfly stage. After that the left half intermediate values are multiplied by the complex twiddle factors $W_N^i$ whereas the right half remains unchanged. Finally, both parts are passed to two $N/2$ sized FFTs which output the final frequencies in a bit-reversed order, necessitating the use of specialized reordering circuitry [8] at its output. In case of real valued inputs, the right FFT of the DIF is real, thus odd conjugate symmetric and the left one is even conjugate symmetric as the twiddle factor multiplication is conducted before it. On the other side, the DIT algorithm is the mirrored version of the DIF as shown in Fig. 1a. While examining the DIT algorithm for real valued inputs, we observe that the $N/2$ sized FFTs have also real inputs, thus they are both odd conjugate symmetric, unlike the DIF architecture which exhibits both even and odd symmetry. Eventually, utilizing the odd symmetry in both the FFTs of the DIT method, we can apply the same principle recursively and break down each of the $N/2$ FFT into smaller FFTs and reuse the same hardware units without any modification; a property that we exploit as we explain later in Section III-A.

Given the increased use of applications that require real-valued FFTs, new works based on the Single Path Delay Feedback (SDF) [13] architecture were proposed [7], [10], [11] which most of them utilize the reordering circuits at the their output. For an input set of size $N$ they require more than $N$ memory cells which allow to correctly time the twiddle factor multiplications of each stage. To further improve and optimize the utilization of the hardware units, and reduce the delay elements; recently in 2018 and in 2020 the RSC architecture [8] and its improved version mRSC [9] was introduced. All these architectures are composed of 3 different hardware units: the butterfly which performs an addition and a subtraction, the rotator that executes the complex multiplication of the twiddle factors $W_N^j$ and the data management circuits which are responsible for the correct timing of the design. Each of them requires the use of some combinational (adders, multipliers, multiplexers) and sequential elements (registers). The mRSC, that modifies the RSC, reduces the memory requirements from $N + 9logN - 19$ to $N + 5logN - 9$ while both use $2logN - 2$ and $logN - 2$ real adders and multipliers respectively. Their latency is greater than $N$ clock cycles, their throughput is 1 real sample per cycle while the total computation time of the total computation time of the FFT is $N$ clock cycles as they process one input a time. It is important to note that both architectures operate on bit-reversed inputs and their outputs are also scrambled, so the reordering circuit that is proposed in [8] adds further overhead in terms of hardware. In Table I we summarize the existing RFFT architectures.

## III. Proposed Approach

To address the memory overhead in [10], [11] and the additional reordering circuits mention in [8], [9], we are introducing an approach that exploits the odd conjugate symmetry of the DIT-FFT. By doing so we can efficiently reduce in half the operations at each stage, allowing us to process two inputs at a time, which leads to increased throughput. Lastly, we apply a unique memory management technique which enables the reduction of the required memory and the calculation of the output frequencies at their natural order by avoiding any reordering circuit at the output.

### A. Real Valued SDF-DIT Architecture

In this section we elaborate on how we have exploited the odd conjugate symmetry and we explain in detail our proposed architecture. Fig. 2 shows the data flow graph of a 16-point DIT-FFT with real inputs that are in bit-reversed order. The white circle nodes represent real data and the colored cross nodes represent complex data. As we can observe after stage 1 the odd symmetry of the bit-reversed input DIT FFT algorithm is revealed exposing some same-colored cross nodes which are conjugate, meaning that we only have to compute

TABLE I
COMPARISON OF THE STATE OF THE ART PIPELINED ARCHITECTURES FOR N-POINT RFFT

| HARDWARE COMPLEXITY | | | | PERFORMANCE | | | |
|---|---|---|---|---|---|---|---|
| Name | Adders | Multipliers | Data Memory | Latency (cycles) | Inputs (samples/cycle | Reordering | |
| | | | | | | IN | Out |
| SC Rdx -2 [14] | $3logN$–2 | $2logN$–4 | $2N$ | $N$ | 1 | Yes | Yes |
| SDF Rdx - 2 Hybrid [7] | $6logN$–10 | $4logN$–12 | $5N/4-2$ | $N$ | 1 | No | Yes |
| SDF Rdx- 2 Fully real [7] | $4logN$–6 | $4logN$–12 | $3N/2-5$ | $N$ | 1 | No | Yes |
| Radix -$2^3$ [10] | $2logN$ | $3log_8N$–6 | $3N/2-6$ | $\approx 3N/2-6$ | 2 | No | Yes |
| Radix -$2^3$ [11] | $2logN$ | $6log_8N$–6 | $3N/2-6$ | $\approx 3N/2-6$ | 2 | No | Yes |
| RSC Rdx - 2 [8] | $2logN$–2 | $logN$–2 | $N+9logN-19$ | $N+3logN-8$ | 1 | Yes | Yes |
| mRSC Rdx -2 [9] | $2logN$–2 | $logN$–2 | $N+5logN-9$ | $N+3logN-8$ | 1 | Yes | Yes |
| RSDF-DIT Rdx -4 Proposed | $8logN$–8 | $3logN-4$ | $N-2$ | $N/2$ | 2 | Yes | No |

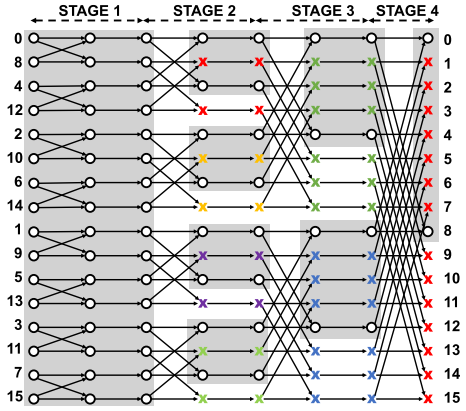


Fig. 2. DIT FFT stages; The shaded areas acquire all the information.

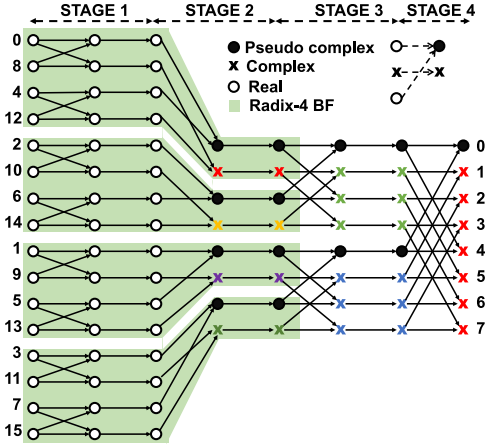


Fig. 3. Proposed truncated DIT FFT using a complex and "pseudo complex" data.

the gray shaded areas at each stage. In Fig. 3 we present the truncated 16-point real-valued DIT-FFT. At the output of each stage the 2 real values (white nodes) are merged into one "pseudo complex" value (black node) for better memory management. Eventually, we can use a radix-4 butterfly to merge stage 1 and 2 and after that we employ a half-sized complex FFT for stages 3 and 4. Specifically, in Fig. 6 we describe the timing of the RSDF-DIT FFT for $N = 16$ with real data colored as red and complex data colored as blue. The outputs of stage 1 are two per clock cycle (cc) as they do not require any delay element. The outputs of the stage 2 are available from cc1 to cc8 and as there is 1 complex delay element. The outputs of stage 3 are available from cc3 to cc10 as there are 2 additional complex delay elements while the outputs of stage 4 are available from cc7 to cc15 considering that we add another 4 complex delay elements. Finally, for an general $N$ sized FFT we only have to add the extra stages that are required by doubling the size of the memory and the number of twiddle factors of the rotators after each stage.

Fig. 4 depicts the components of the proposed architecture for $N = 16$. Our architecture is based on the conventional SDF FFT with a few modifications. Initially we have merged stage 1-2 into a single one by performing a radix-4 butterfly which enables the processing of 2 real inputs at time. At stage 3 and 4 we have used the SDF architecture with a modified rotator/butterfly unit as some of the inputs are "pseudo complex" and they require a different manipulation. Also, each stage has a shift register memory to store the complex values. For $N = 16$ the total complex memory elements are $N/2-1$ which equal $N-2$ real memory elements. Finally, we need a radix-4 butterfly and $logN-2$ rotator/butterfly units.

### B. Rotator/Butterfly Unit

In the proposed architecture we use a similar rotator and butterfly units as the ones used by the conventional SDF FFT [13] with some modifications to allow us to handle the "pseudo complex" data. While in [8], [9] the complex multiplication is conducted by using a single multiplier (Fig. 5b), in our approach we applied a 3 multiplier and 4 adder complex multiplication scheme by dividing in half the set of twiddle factors as shown in Fig. 5a. Essentially at each stage, due to the fewer twiddle factors per multiplier, our units end up being smaller and simpler, helping to keep the area overheads low. An example in case of $N = 16$ is shown in Fig 5a; the bottom two multipliers use the twiddle factors $w_1 = 0.19509$ and $w_2 = 0.98078$ and the top one their sum $w_3 = 1.17588$, to conduct the complex multiplication in one cycle, while in Fig. 5b both $w_1$ and $w_2$ are used in the same multiplier at different clock cycles. To optimize and reduce even further the multiplication cost there are many Multiple Constant Multiplication (MCM) heuristics based on either Directed Acyclic Graphs (DAGs) [15] or by utilizing Sum of

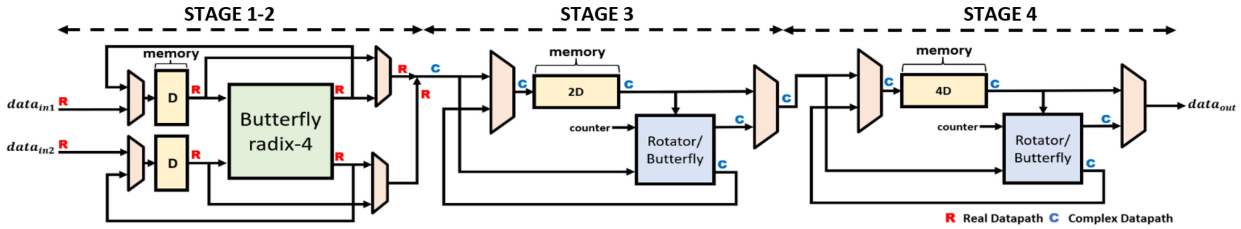


Fig. 4. Structure of the proposed RSDF-DIT architecture for $N = 16$.

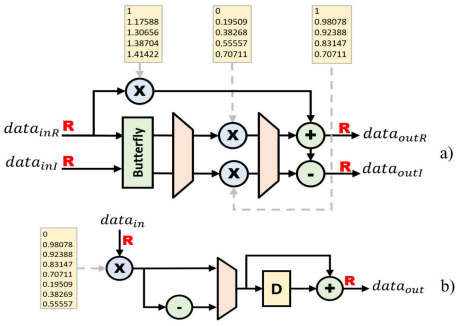


Fig. 5. Multipliers and constants in a) the proposed architecture and b) the existing RSC and mRSC schemes [8], [9].

| Input | Stage 1 | Stage 2 | Stage 3 | Stage 4 |
|---|---|---|---|---|
| $x_{0,0}$ cc0 | $x_{0,1}$ cc0 | $x_{0,2}$ cc1 | $x_{0,3}$ cc3 | $x_{0,4}$ cc7 |
| $x_{1,0}$ cc0 | $x_{1,1}$ cc0 | $x_{1,2}$ cc2 | $x_{1,3}$ cc4 | $x_{1,4}$ cc8 |
| $x_{2,0}$ cc1 | $x_{2,1}$ cc1 | $x_{2,2}$ cc1 | $x_{2,3}$ cc5 | $x_{2,4}$ cc9 |
| $x_{3,0}$ cc1 | $x_{3,1}$ cc1 | $\overline{x}_{1,2}$ | $x_{3,3}$ cc6 | $x_{3,4}$ cc10 |
| $x_{4,0}$ cc2 | $x_{4,1}$ cc2 | $x_{4,2}$ cc3 | $x_{4,3}$ cc3 | $x_{4,4}$ cc11 |
| $x_{5,0}$ cc2 | $x_{5,1}$ cc2 | $x_{5,2}$ cc4 | $\overline{x}_{3,3}$ | $x_{5,4}$ cc12 |
| $x_{6,0}$ cc3 | $x_{6,1}$ cc3 | $x_{6,2}$ cc3 | $\overline{x}_{2,3}$ | $x_{6,4}$ cc13 |
| $x_{7,0}$ cc3 | $x_{7,1}$ cc3 | $\overline{x}_{5,2}$ | $\overline{x}_{1,3}$ | $x_{7,4}$ cc14 |
| $x_{8,0}$ cc4 | $x_{8,1}$ cc4 | $x_{8,2}$ cc5 | $x_{8,3}$ cc7 | $x_{8,4}$ cc15 |
| $x_{9,0}$ cc4 | $x_{9,1}$ cc4 | $x_{9,2}$ cc6 | $x_{9,3}$ cc8 | $\overline{x}_{7,4}$ |
| $x_{10,0}$ cc5 | $x_{10,1}$ cc5 | $x_{10,2}$ cc5 | $x_{10,3}$ cc9 | $\overline{x}_{6,4}$ |
| $x_{11,0}$ cc5 | $x_{11,1}$ cc5 | $\overline{x}_{9,2}$ | $x_{11,3}$ cc10 | $\overline{x}_{5,4}$ |
| $x_{12,0}$ cc6 | $x_{12,1}$ cc6 | $x_{12,2}$ cc7 | $x_{12,3}$ cc7 | $\overline{x}_{4,4}$ |
| $x_{13,0}$ cc6 | $x_{13,1}$ cc6 | $x_{13,2}$ cc8 | $\overline{x}_{9,3}$ | $\overline{x}_{3,4}$ |
| $x_{14,0}$ cc7 | $x_{14,1}$ cc7 | $x_{14,2}$ cc7 | $\overline{x}_{10,3}$ | $\overline{x}_{2,4}$ |
| $x_{15,0}$ cc7 | $x_{15,1}$ cc7 | $\overline{x}_{13,2}$ | $\overline{x}_{11,3}$ | $\overline{x}_{1,4}$ |

Fig. 6. Timing of the proposed RSDF-DIT FFT architecture.

Products (SOPs) [16], [17]. In a similar direction our architecture can be further benefited in terms of accuracy by properly scaling the twiddle factors which is addressed by [18] or by applying Computation Sharing Multiplier (CSHM) techniques mentioned in [19], [20].

### C. Memory Management

Here we elaborate on the memory management technique that we employed in order to unscramble the output data. We achieved that by replacing the shift registers of each stage after stage 2 with a dual state memory. In Fig. 7 we describe the functionality of such a memory for stage 4 of our proposed in Fig. 6. The leftmost column (cc) represents the clock cycles, while columns A and B (complex datapath) represent the ports of the memory that interchangeably can be used as inputs or

| CC | A | Data flow | | | | B |
|---|---|---|---|---|---|---|
| | | Reg0 | Reg1 | Reg2 | Reg3 | |
| 0 | $x_0$ | | | | | |
| 1 | $x_1$ | $x_0$ | | | | |
| 2 | $x_2$ | $x_1$ | $x_0$ | | | |
| 3 | $x_3$ | $x_2$ | $x_1$ | $x_0$ | | |
| 4 | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $x_0$ | |
| 5 | $x_5$ | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $x_0$ |
| 6 | $x_6$ | $x_5$ | $x_4$ | $x_3$ | $x_2$ | $x_1$ |
| 7 | $x_7$ | $x_6$ | $x_5$ | $x_4$ | $x_3$ | $x_2$ |
| 8 | $x_8$ | $x_7$ | $x_6$ | $x_5$ | $x_4$ | $x_3$ |
| 9 | $x_7$ | $x_6$ | $x_5$ | $x_4$ | $x_8$ | $x_9$ |
| 10 | $x_6$ | $x_5$ | $x_4$ | $x_8$ | $x_9$ | $x_{10}$ |
| 11 | $x_5$ | $x_4$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ |
| 12 | $x_4$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ |
| 13 | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ |
| 14 | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ |
| 15 | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ |
| 16 | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ | $x_{16}$ |

Fig. 7. Timing of the proposed dual state memory.

outputs. Finally, we show the content of each register from columns Reg0 to Reg3. At clock cycle 0, A is the input port and B is the output port, while the data flows from right to left. Later at clock cycle 9 the state of the memory changes and now port B is the input and port A is the output. As we can see the data exits the memory in reverse order (port A) as its flow is changed. Finally, after clock cycle 16 the memory is at its initial state with port A being the input and port B being the output. Therefore, by utilizing this memory at each stage after stage 3, we compute the output frequencies in the correct order within the architecture as opposed to [8], [9].

## IV. Results

To evaluate the performance gains of our architecture we have compared it with state-of-the-art works (that utilize the least amount for delay elements), in terms of the required hardware units and the achievable latency and throughput. In comparison with previous complex valued architectures [13], [14] our architecture has much less memory cells and multipliers due to the applied optimization for real valued signals. Furthermore, the real-valued architecture in [7], [10], [11] may have reduced the memory elements compared with the complex ones, but they are outperformed in terms of both latency and memory by our approach. Considering, the most recent real-valued architectures, RSC [8] and mRSC [9] our approach outperforms them by doubling the throughput and requiring half the latency while using the least number of memory elements. As we showed, this is achieved by processing in parallel two real inputs at a time, which required the addition of more multipliers. In fact, an input set of size $N$ requires $3logN - 4$ multipliers as opposed to $logN - 2$ multipliers

TABLE II
COMPARISON OF HARDWARE 128-POINT PIPELINED FFT ARCHITECTURES

| Architecture; $N = 128, W = 16$ $@1.1V, 45nm, 100MHz$ | Latency (Cycles) | Area($\mu m^2$) | | Energy ($\mu W/MHz$) |
|---|---|---|---|---|
| | | Total | Seq | |
| mRSC FFT | 141 | 58060 | 36588 | 1667 |
| RSC FFT | 137 | 56489 | 35017 | 1610 |
| RSDF DIT | 64 | 63820 | 18494 | 886 |

required by the schemes in [8], [9]. However, as we highlight in Section III-B our rotators may require more multipliers, but they are much simpler. Furthermore, the overhead of the additional multipliers is circumvented by the proposed memory management technique. This helps to limit the memory cells to $N-2$, which are 26.16% and 17.53% less than the ones required by [8], [9] due to the novel utilization of the odd symmetry in the DIT scheme. Finally, our scheme allows to eliminate the need for a reordering circuit at the output stage by moving it at the input stage thus avoiding the associated overheads as opposed to almost all existing works.

### A. Measurements

Table II compares our proposed architecture with the most recently introduced real-valued ones; the RSC FFT [8] and the mRSC FFT [9] in case of $N = 128$ with word length $W = 16$ bits. For our results we synthesized the circuits using the 45nm NangateOpenCell library at 1.1V and clock frequency of 100 MHz. Note that, for the energy measurements we accounted for the power consumption $P(\mu W)$ of the circuit multiplied by the total execution cycles $N_c$ divided by the clock frequency $f_{clk}(MHz)$ (Eq. (4)).

$$E(\mu W/MHz) = N_c \times P(\mu W)/f_{clk}(MHz). \quad (4)$$

So, we observe that the RSC and the mRSC schemes require 141 and 137 cycles to compute their first output as their latency in greater than $N = 128$, while the proposed RSDF-DIT FFT requires only 64 cycles. In terms of the sequential area (registers) our proposed architecture is 49.45% and 47.18% more efficient compared with the RSC and mRSC respectively, as their intermediate pipeline registers and their reordering circuits at their outputs add substantial overhead. So, the total area cost of our approach is only 9.02% and 11.49% more than the RSC and the mRSC, respectively as our memory optimization compensate for the additional but simpler multipliers that we employ. Consequently, our increased throughput combined with the reduced number of memory cells, the integrated reordering and the smaller total execution time lead to significant 46.86% and 44.99% energy savings compared to the recent RSC and mRSC architectures, respectively.

## V. CONCLUSION

In this brief, we presented the RSDF-DIT FFT architecture which allows the substantial energy improvements in case of processing real-valued data. This is achieved by exposing and fully exploiting the odd conjugate symmetry, which is enabled by applying the DIT method as opposed to the ones used by existing schemes. The proposed architecture efficiently reuses the same hardware units for each computation and processes 2 real inputs at a time using more but much simpler multipliers. In combination with a novel memory management method, the extra memory cells are reduced and any output reordering circuit is eliminated, thus allowing the doubling of the throughput with minor area overhead. Overall, the proposed architecture leads up-to 46.86% energy savings in 45nm technology when compared with the latest real valued FFT schemes.

## REFERENCES


[1] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90,pp. 297–301, 1965.

[2] G. G. Kumar, S. K. Sahoo, and P. K. Meher, "50 years of FFT algorithms and applications," *Circuits Syst. Signal Process.*, vol. 38, no. 12, pp. 5665–5698, 2019.

[3] G. Karakonstantis, A. Sankaranarayanan, M. M. Sabry, D. Atienza, and A. Burg, "A quality-scalable and energy-efficient approach for spectral analysis of heart rate variability," in *Proc. DATE*, 2014, pp. 1–6.

[4] M. Garrido, "A survey on pipelined FFT hardware architectures," *J. Signal Process. Syst.*, no. 6, pp. 849–863, Jul. 2021.

[5] M. Estévez *et al.*, "Spectral analysis of heart rate variability," *Int. J. Disabil. Human Develop.*, vol. 15, no. 1, pp. 5–17, 2016.

[6] O. W. Ibraheem and N. N. Khamiss, "Design and simulation of asymmetric digital subscriber line (ADSL) modem," in *Proc. Int. Conf. Inf. Commun. Technol. Theory Appl.*, 2008, pp. 1–6.

[7] A. Chinnapalanichamy and K. K. Parhi, "Serial and interleaved architectures for computing real FFT," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2015, pp. 1066–1070.

[8] M. Garrido, N. K. Unnikrishnan, and K. K. Parhi, "A serial commutator fast Fourier transform architecture for real-valued signals," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 11, pp. 1693–1697, Nov. 2018.

[9] S. Park and D. Jeon, "A modified serial commutator architecture for real-valued fast Fourier transform," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2020, pp. 1–6.

[10] M. Ayinala and K. K. Parhi, "FFT architectures for real-valued signals based on radix-$2^3$ and radix-$2^4$ algorithms," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2422–2430, Sep. 2013.

[11] 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. II, Exp. Briefs*, vol. 60, no. 8, pp. 507–511, Aug. 2013.

[12] M. Garrido, "A new representation of FFT algorithms using triangular matrices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 10, pp. 1737–1745, Oct. 2016.

[13] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *Proc. URSI Int. Symp. Signal Syst. Electron.*, 1998, pp. 257–262.

[14] 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.

[15] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, no. 2, pp. 11–48, 2007.

[16] V. Karkala, J. Wanstrath, T. Lacour, and S. P. Khatri, "Efficient arithmetic sum-of-product (SOP) based multiple constant multiplication (MCM) for FFT," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2010, pp. 735–738.

[17] R. Kumar, A. Mandal, and S. P. Khatri, "An efficient arithmetic sum-of-product (SOP) based multiplication approach for FIR filters and DFT," in *Proc. IEEE 30th Int. Conf. Comput. Des. (ICCD)*, 2012, pp. 195–200.

[18] M. Garrido, F. Qureshi, and O. Gustafsson, "Low-complexity multiplierless constant rotators based on combined coefficient selection and shift-and-add implementation (CCSSI)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2002–2012, Jul. 2014.

[19] G. Karakonstantis and K. Roy, "An optimal algorithm for low power multiplierless FIR filter design using Chebychev criterion," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, vol. 2, 2007, pp. 49–52.

[20] G. Karakonstantis, N. Banerjee, and K. Roy, "Process-variation resilient and voltage-scalable DCT architecture for robust low-power computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 10, pp. 1461–1470, Oct. 2010.