FPGA Realization of the Observer-Based Sliding Discrete Fourier Transform

PETER PLESZNIK1, ZSOLT KOLLAR2, (Member, IEEE), AND DAEJIN PARK3, (Member, IEEE)

1School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, South Korea
2Department of Measurement and Information Systems, Budapest University of Technology and Economics, 1111 Budapest, Hungary
Corresponding author: Daejin Park (boltanut@knu.ac.kr)

This work was supported in part by the Budapest University of Technology and Economics through the National Research, Development, and Innovation Fund of Hungary under Grant TKP2021-EGA-02; in part by the BK21 FOUR Project funded by the Ministry of Education, South Korea, under Grant 419990013966; in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant NRF-2019R1A2C2005099; in part by the Ministry of Education under Grant NRF-2018R1A6A1A03025109 and Grant NRF-2020R1I1A1A01072343; and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korean Government (MSIT) (Metamorphic Approach of Unstructured Validation/Verification for Analyzing Binary Code, 70%) under Grant 2021-0-00944.

ABSTRACT Discrete Fourier transform (DFT) is a widely used method of signal analysis in digital signal processing. The DFT converts a signal from time domain to frequency domain for further processing. For fixed-size sliding window applications of the DFT, the observer-based sliding DFT (oSDFT) algorithm has been shown to be stable, accurate, and theoretically faster, than the well-known block-oriented fast Fourier transforms (FFT). However, no hardware implementation of the oSDFT has been proposed yet. In this paper, a hardware optimized implementation of two variants of the algorithm for FPGA is presented. Such implementation is compared with the Xilinx FFT Intellectual Property in terms of processing speed and hardware requirements. The structure is implemented in Verilog HDL using Vivado IDE, with the aim of maximizing the processing speed and minimizing the required hardware resources. The analysis of the FPGA-based oSDFT and FFT circuits in a sample-by-sample processing scenario, reveals that the latency and energy usage of the oSDFT are smaller relative to the FFT. The latency and energy usage per sample processed of the implemented structures are up to 9 and 10 times lower than those of FFT respectively. The required resources for these methods are also presented and analyzed.

INDEX TERMS Sliding DFT, oSDFT, FFT, FPGA, low latency.

I. INTRODUCTION The discrete Fourier transform (DFT) with its inverse transform (IDFT) is one of the most important digital signal analysis tools. Several, highly efficient algorithms have been developed for the evaluation of the DFT making it applicable in a wide variety of use cases and applications. Such well known methods form the family of fast Fourier transform (FFT) algorithms. However, the FFT algorithms calculate their output by using blocks of samples, which make them less suitable for sliding window signal processing, because in sliding DFT (SDFT) applications the spectra must be evaluated over the input data stream in an overlapping window approach – in the most demanding scenarios continuously, sample by sample – however, FFT algorithms do not share and utilize calculation results between subsequent calculations. For this reason recursive formulas and methods have been developed for SDFT purposes which can evaluate the DFT in a continuous and filter-like manner with relatively high precision and low latency.

Several applications have been published recently where such algorithms can be employed efficiently. An efficient method using SDFT was proposed for frequency estimation [1] and tone detection [2]. For synchronization purposes, such an application was given in [3]. Methods for spectrum estimation and spectrum sensing in wireless communication systems were reported in [4], [5]. Furthermore, a method for decoding data transmission using SDFT was presented in [6]. In mechanical systems, the SDFT can be efficiently applied for tuning vibration neutralizers [7].

The general implementation for SDFT is derived directly from the recursive form of the DFT equation. At first glance, it looks promising, as only the frequencies of interest are calculated against the DFT’s full spectra computation,
but it applies feed-forward and resonator structures. This simplest approach is the well-known Goertzel algorithm combined with a comb filter, but its hardware implementation is structurally unstable [8] due to the limited arithmetical precision in the resonator pole. An improved solution is the modulated sliding DFT (mSDFT), which resolves the resonator’s stability problem by replacing it with an integrator via exploiting the modulation theorem of the Fourier transform, but its accuracy and long-term stability is still highly affected by the computational resolution [9]. In the past few years, numerous improvements have been presented to improve the stability and to reduce the calculation requirements of sliding window DFT algorithms [10]–[12].

Instead of optimizing the structure of the recursive DFT, the fundamentally different, but lesser known implementation of the SDFT is considered which is based on the observer theory approach taken from control theory [13], [14]. This structure will be referred to hereafter as observer SDFT (oSDFT). This structure is highly stable and insensitive to numerical precision errors due to its control feedback loop [14]. Moreover, its added benefit is that it can be used efficiently for various applications, such as frequency estimation [15], [16], system identification [17] or digital filtering [13]. The disadvantage of this structure is that all the frequency bins – not just the ones of interest – must be calculated, as an observer first deconstructs the signal then reconstructs an estimation which is also used for the global control feedback loop. This structure has been investigated mainly through simulations, but no hardware implementation has yet been reported, only software-based solutions have been proposed [18]. The aim of this paper is to verify the oSDFT’s practical advantage in speed by providing a hardware implementation for the theoretical structures as a proof of concept. Using minimal modifications of the original structures and dedicated FPGA resources, these implementations are compared with the optimized FFT hardware provided by the manufacturer of the FPGA.

The current work focuses on two known variants of the observer based SDFT (oSDFT) method given in [14], namely the resonator- and the modulator-based structures, offering FPGA implementations for them which are also investigated in terms of speed, accuracy, latency, and the number of required hardware resources.

The outline of the paper is as follows. First, the SDFT and mSDFT structures are briefly described, then an overview of the oSDFT and its two variants are presented. In Section III, a computationally efficient FPGA implementation of two oSDFT structures are proposed and analyzed. Then, in Section IV, the FFT IP provided by Xilinx is described. In section V the oSDFT and the FFT IP are compared in terms of required hardware resources, latency, and energy usage. The quantization errors of the investigated structures are also analyzed. In the last section, conclusions are drawn.

II. THEORETICAL BACKGROUND OF SDFT AND oSDFT

The sliding DFT, processing with an N sample long window size for an input sequence \( x[n] \) can be given as

\[
X[k+1] = \sum_{n=0}^{N-1} x[n + n - N + 1] e^{-j \frac{2\pi}{N} nk},
\]

where \( X[k] \) represents the \( k \)th frequency component in the DFT of the samples \( x[n - N + 1], \ldots, x[n] \) at the time index \( n \). This equation can be reformulated recursively [14] as

\[
X[k+1] = (x[n] - x[n - N] + X[k]) W_N^k, \tag{2}
\]

where

\[
W_N^k \triangleq e^{j \frac{2\pi}{N} k} \tag{3}
\]

is called the twiddle factor. This recursive equation can be implemented directly using a comb filter and resonator structure [8] as depicted in Fig. 1. However this structure lacks the long term stability [9] because resonator poles implemented in real hardware cannot be placed perfectly on the unit circle resulting in non-unity gain, and in worst case a continuously growing value.

The mSDFT suggested in [9] mitigates this stability issue by replacing the resonator with an integrator utilizing the DFT’s modulation property preserving the original transfer function. The structure of the mSDFT is presented in Fig. 2. Similar to the SDFT structure, first, a comb filter is applied, then the signal is demodulated to the frequency value of 0 (DC) using a rotating phasor of \( W_N^{-kn} \). In the next step an integration is performed which tracks the amplitude of the \( k \)-th DFT component, then finally a modulation is applied by a rotating phasor of \( W_N^{nk} \) supplying the current phase for the DFT value of the \( k \)th frequency component at the time instant \( n \). The transfer function of the \( k \)th realization of both SDFT and mSDFT using the Z-transform can be expressed as:

\[
H_{k,SDFT}(z) = \frac{X_k(z)}{X(z)} = \left( 1 - z^{-N} \right) H_k(z), \tag{4}
\]

where

\[
H_k(z) = \frac{W_N^k z^{-1}}{1 - W_N^k z^{-1}}. \tag{5}
\]

The benefit of these SDFT implementations over the FFT is that not all DFT bins need to be calculated, but only the frequency of interest needs evaluation. Each resonator branch can be evaluated independently from each other using a common comb filter’s output as their input.

In the case of the oSDFT, all frequency bins – still in separate branches – must be implemented due to the observer system, but instead of a comb filter a global feedback signal \( y[n] \) is formed, from which an error signal \( e[n] \) is calculated against \( x[n] \) and used by all the branches. The structure is shown in Fig. 3. This is a prediction-correction algorithm via signal decomposition into branches and recombination from those. The structure observes the input signal’s – as a system’s – state variables, where the state variables are the \( N \) complex
DFT components $X_k[n]$. At each cycle, the observer creates a signal prediction, then corrects itself using the global negative feedback to update the state variables.

The oSDFT is equivalent to the SDFT – and the mSDFT as well – in such a way that the transfer functions of the $k$th branches are equal for both. The transfer function of the $k$th branch of the oSDFT using the equivalence of feedback control theory can be expressed as

$$H_{k,\text{osDFT}}(z) = \frac{X_k(z)}{X(z)} = \frac{1}{1 + \frac{1}{N} \sum_{k=0}^{N-1} H_k(z)}.$$  \hspace{1cm} (6)

In order to prove that equations (4) and (6) are equal the following equations has to be proven:

$$1 - z^{-N} = \frac{1}{1 + \frac{1}{N} \sum_{k=0}^{N-1} H_k(z)}.$$  \hspace{1cm} (7)

The mathematical proof of the equality presented in (7) is given in the Appendix. Another advantage of this structure over SDFT and mSDFT is the positive effect of the control feedback system on the overall stability which can compensate for the quantization errors.

Two variants exist for the implementation of the oSDFT branches as well: the resonator-based and the modulator-based. The resonator-based implementation is similar to SDFT consisting of a delay element and a multiplication by the twiddle factor, as can be seen in Fig. 4. The modulator-based implementation is similar to the mSDFT requiring two multiplications by complex rotating exponents and a delay element, as is depicted in Fig. 5. The equivalence of the resonator- and modulator-based oSDFT structures is also given in [14].

### III. FPGA IMPLEMENTATION OF oSDFT

#### A. GENERAL oSDFT HARDWARE IMPLEMENTATION

A practical hardware realization of the oSDFT structure presented in Fig. 3 is shown in Fig. 6. The real and imaginary parts of the signals are processed separately. First, the real $y_R[n]$ and imaginary $y_I[n]$ parts of the feedback signal $y[n]$ are subtracted from the corresponding real and imaginary parts of the input signal $x[n]$ to form the error signal $e[n]$. Then, each branch requires the real $e_R[n]$ and the imaginary $e_I[n]$ parts of the error signal as input. The real $W_R$ and imaginary $W_I$ parts of the complex exponent corresponding to the chosen implementation of the branch must be made available as well. The resonator-based structure needs one coefficient, whereas the modulator-based structure needs two of these. The detailed implementation of the resonator- and the modulator-based branches is discussed in the next sections. The output signals of each branch – real and imaginary parts separately – are required to be summed using adders with $N$ inputs. These results are then divided by $N$, and the feedback signals are formed. It is possible to implement the structures using resource sharing with time multiplexing. In this case only $M$ of $N$ branches are implemented physically, and the $M$ input adders produce only partial results, which are accumulated and then divided by $N$ to produce the feedback signals. In this case, the optional accumulator blocks shown in Fig. 6 are needed.

#### B. FPGA REALIZATION

The design presented in this paper was simulated, synthesized, and implemented in Xilinx Vivado 2019.2 edition, on a Basys 3 evaluation board, equipped with the Xilinx XC7A35TCPG236-1 FPGA, in Verilog HDL language.
For the analog-to-digital conversion, the integrated 12-bit analog-to-digital converter (ADC) was used.

The Xilinx FPGAs incorporate DSP 'hard macros', dedicated circuits for high-speed signal processing, supporting accelerated arithmetic and logical functions. The data format is selected so that it suits the best the properties of these resources. The DSP blocks have one 25-by-18-bit wide multiplier, which can optionally receive its 25-bit wide input from a pre-adder, and produces a 43-bit result, sign extended to 48 bits. The 25-bit wide DSP input is chosen for the input signal, because the input from the ADC is 12-bit wide, and the values of the frequency components in the Fourier transform can reach the maximal value of $N$, which needs an additional $\log_2 N$ number of bits to be stored during the calculation. Also, this way the built-in pre-adder can be used, enhancing performance. The 18-bit wide input is used for the coefficients. The pre-adder in the DSP is also 25-bit wide, with no overflow protection implemented; therefore, it can only receive a 24-bit wide input safely. Considering these properties, the input signal is chosen to be 12-bit wide with 11-bit fractional part, and one sign bit. The internal values are determined to be represented as 24-bit wide with 11 bit integer part, 12 bit fractional part and one sign bit. The 11 bit integer part accommodates the $\log_2 N$ bit growth, enabling a maximal $N$ of 2048. The coefficients are determined to be 18-bit wide with a 17-bit fractional part and one sign bit. The ratio of the integer and fractional parts of the internal values can be changed to enable longer transforms or higher input resolution. Since the real and imaginary parts of the complex input are processed separately, the statements above apply to the real and imaginary parts separately.

The FPGA also has hard macro random access memories (RAMs) called block RAMs (BRAMs) and lookup tables (LUTs). These two components can be used as memory resources. The BRAMs are 36Kb dual-port RAMs and are made up of two subblocks, each of which can store 18 Kb of data. These memories can support two parallel read operations from arbitrary addresses, and their width and depth can be configured within a given range. The LUTs can be used as 64 by 1 memories, storing up to 64 individual bits of information. These LUTs can be cascaded into wider and/or deeper memories, and are referred to as LUT-RAMs in this case. For small amounts of data, which is distributed in the circuit, LUT-RAMs are faster, use less energy and resources, whereas, for large blocks of data, BRAMs are more efficient, and have higher performance.

C. IMPLEMENTATION OF THE STRUCTURES

The highly parallel property of the FPGA allows the parallel implementation of multiple blocks for improved throughput. We aimed to set the level of parallelization to stay within a level comparable to the FFT IP’s resource usage. In both the resonator- and modulator-based structures, eight branches are implemented in parallel based on the resource usage of each branch design and the FFT IP presented later in Section V in detail. This solution processes eight spectral components, and calculates the sum of these components in one clock cycle. The entire $N$ point calculation is performed in multiple cycles by sharing the resources of the implemented branches. For the purpose of avoiding the use of a division circuit with high complexity, only lengths of $N = 2^a$ are considered, as in this case the division in the global feedback can be performed with simple truncation and optional rounding for lower error. The global feedback requires two adders—one for the real and imaginary parts each—to calculate the error signal. These global adders are implemented in LUTs, as they showed no performance disadvantages compared with DSP implementation.

1) RESONATOR-BASED oSDFT BRANCH

The structure of the resonator-based oSDFT branch implemented in FPGA is shown in Fig. 7. Each of these branches are built up from adders, multipliers and a register pair (implemented in LUT RAM) for storing the complex state variables.

The blocks highlighted in gray are the Xilinx DSP hard macro resources, which are responsible for calculating the complex operations. A complex multiplication requires four real multiplications and two real additions:

$$\text{Re}(A \ast B) = \text{Re}(A) \ast \text{Re}(B) - \text{Im}(A) \ast \text{Im}(B),$$
$$\text{Im}(A \ast B) = \text{Re}(A) \ast \text{Im}(B) + \text{Im}(A) \ast \text{Re}(B).$$

(8)

This can be entirely done in DSP blocks, using four of them in a branch. The first two DSP blocks – positioned at the center of the figure – using the pre-adders, perform the addition of the feedback of the delay element and the error signal seen in Fig. 4. Then, the real multiplications...
are performed to produce the partial results of the complex multiplication. These partial results are forwarded to the next two DSP blocks. These blocks perform the same operation as the first two, but also use the partial results to produce the output of the complex multiplication. The results of the complex multiplications can overflow; therefore, a saturation is applied to the output of the second two multipliers producing the final results to avoid errors. For every branch, the state variables of the resonator must be stored. For this, LUT resources are chosen, as they have been proved to be faster and also, they occupy a smaller area compared with BRAMs. Every branch needs two LUT-RAMs – consisting of 24 LUTs each – to store the 24-bit wide internal values of the real and imaginary parts that are stored separately. These are the real and imaginary parts of the delay element in Fig. 4. The 18-bit coefficients are stored in BRAMs, taking up 36 bits -real and imaginary parts combined- each. This choice was made, because for large blocks of data, the BRAMs are faster, compared with the LUT-RAMs. In the case of the resonator-based structure, to reduce the number of BRAMs used, it is possible to take advantage of the fact that the BRAMs in the FPGA are implemented with dual-ports. Therefore, two of the branches can share one BRAM if the data are stored alternately, and each branch uses one port. This can reduce the number of BRAMs used by half.

In order to carry out the time multiplexing, the theoretical structure in Fig. 4 was modified. The results of the 43-bit complex multiplications - saturated to 24 bits - are directly fed to an 8-input adder, and are summed, then saved in an accumulator. After all the $M = N/8$ additions are finished, the value of the accumulator is divided by $N$, and the global feedback register is updated with this result in a separate clock cycle. This is then fed back to the input.

The longest signal path in the circuit between two clocked registers is called the critical path, and determines the maximal possible operating frequency of the design. The critical path of this structure starts at the output of the LUT-RAMs storing the internal values in each branch, goes through the DSP blocks, and ends at the input of the accumulator that processes the results of the partial summations of the branch outputs. Pipelining could shorten the critical path, by dividing it with additional registers. However, the feedback structure prevents the pipelining of the branches. The output of the resonator must be delayed by exactly a single time index before the signal is fed back to the input to form the error signal. However, the pipeline would introduce additional delays, which would alter the transfer function of the structure, by changing and ultimately violating the time relations of Fig 4. For this reason, the critical path is relatively long and limits the achievable operating frequency.

The BRAMs storing the coefficients can hold 1024 entries of 36-bit wide data, and each BRAM is shared between two branches. Therefore, in this implementation, the maximal number of coefficients is 512 using four BRAMs. Each LUT-RAM can hold values of 64 different branches, therefore the eight branch parallel implementation can store the complex state variables of 512 long transformations.

2) MODULATOR-BASED oSDFT BRANCH
The modulator-based branch shown in Fig. 8 is similar to the resonator-based one. As a result, similar design considerations can be applied.

Each branch can be separated into three parts: the demodulator, the integrator and the modulator. The demodulator is the complex multiplier at the input of the branch in Fig. 5, while the modulator is the complex multiplier at the output. The delay element, and the complex adder together form the integrator circuit. The modulator and demodulator circuits can be implemented in DSP blocks, using eight blocks in a branch, whereas the integrator can be implemented in LUT resources, using 48 LUTs. The complex multiplications and overflow protections are performed in the same manner as in the resonator-based structure. The four DSP blocks on the left side of Fig. 8 form the demodulator circuit performing the first complex multiplication. The results are saturated to 24 bits for further processing. The two adders...
and LUT-RAMs together form the integrator. The complex multiplication of the modulator is calculated by the four DSP blocks on the right side of Fig. 8. The results are saturated to 24 bits once again, and are fed to the global adder. To reduce the number of BRAMs used for storing the coefficients, the addition of the partial results in the modulator circuit is modified. Since the modulating and demodulating coefficients are complex conjugates of each other, they do not need to be stored separately. Instead, the coefficients for the demodulating circuit are stored, and the modulating circuit calculates the complex conjugated results. Because the sequence of the coefficients repeats after \( N \) values, only these \( N \) values are stored in a dual-port RAM, and this RAM is used within one branch, therefore the number of required BRAMs is reduced by half.

This implementation also uses the time multiplexing approach, in which eight components are calculated in parallel, and the sum of these components are calculated in the same clock cycle. This requires the modification of the original structure in Fig. 5 as well. The demodulation circuit must use the result directly from the modulator circuit, and calculate the feedback for the next sample in advance. This also means that both the coefficients of the current and the next time indices are required, using both output ports of a BRAM for one branch.

The critical path in this structure is longer than that in the resonator-based one. It starts from the register at the signal input, goes through the modulator and demodulator circuits, and ends at the accumulator processing the partial summations of the branch outputs. This structure cannot be pipelined for the same reason as that for the resonator-based, therefore the maximal operating frequency is limited as well.

Each branch uses one 36 Kb BRAM block by itself, therefore it could store 1024 different coefficients, enabling transforms of such length, however, when using the minimal amount of LUTs, the maximal transform length is limited to 512.

### D. SCALABILITY AND PORTABILITY

The implemented HDL designs use Xilinx specific components (DSP and BRAM blocks), which are identical in all Xilinx 7 series FPGAs.\(^4\) Hence, the designs can be moved between different Xilinx chips without modifying the code. Also, since the implementation of the DSP and BRAM macros, and also the LUT structures are the same, the migration between these would have no significant impact on the maximal achievable frequency. Xilinx Ultrascale devices use macros with enhanced performance, but similar structures. Therefore, migration to these devices also does not require the modification of the presented design.

By keeping the constraint of supporting transforms only with a length of \( N = 2^m, N < 2048 \), the level of parallelization can be arbitrarily altered, only the \( M \) input adder, and the time multiplexing control circuit need to be modified.

The maximal transform length, under 2048 points, can be extended by adding more memory resources to the designs. For longer transforms additional DSP cores also need to be added in cascade with the existing cores. This is discussed in detail in Section V-A.

### IV. FFT IP

For the basis of comparison, the Fast Fourier Transform v9.1 IP core provided by Xilinx\(^5\) is used. The core applies the Cooley-Tukey algorithm and is designed for communicating on an Advanced eXtensible Interface (AXI). It has various implementation options, including the following:

- **Architecture**: radix-2, radix-4, or pipelined streaming
- **Transform length**: \( 2^3 - 2^{16} \)
- **Data representation**: fixed or floating point
- **Data handling**: scaling and rounding
- **Output ordering**
- **Resource usage**

The goal of the applied FFT design is to aim for the highest achievable processing speed. This way the best comparison can be made with the processing speeds of the oSDFT structures. Furthermore, the implementation options are chosen so that the data representation and handling would be as similar to that of the implemented oSDFT structure as possible.

For the architecture the pipelined streaming structure is chosen, as this has the highest processing speed among all the available options. This architecture pipelines radix-2 based processing engines to be able to simultaneously load, process and unload the data. Each of these engines has its own memory to store the input and intermediate data. Blocks of \( N \) samples are loaded into the core, and after the calculation latency, the \( N \) spectral components are unloaded. As mentioned earlier, the settings are applied so that the highest processing speed possible is achieved. Therefore data and phase factors of the FFT are stored in BRAMs, complex multiplications are realized with the four-multiplier structure, and the arithmetic of the butterflies also use DSP slices instead of configurable logic resources.

For a better comparison with the oSDFT structure, the data format was set to fixed point, without scaling, and the rounding mode was set to truncation. Similarly to the oSDFT, the input data is 12-bit wide, and the phase factor width is 18 bit, both are two’s complement signed numbers. The output is reordered to the natural order, which is also done in BRAM for optimal speed.

The core can operate only on a block basis, and is not capable of calculating sliding window FFT by itself. For this to be done, a first in first out structure must be placed on its input. This is updated with the sampling frequency, the oldest sample is discarded and the new sample is put into the buffer.

---

\(^4\)Series FPGA Data Sheet: Overview,” 2020, available online.

\(^5\)XILINX, “Fast Fourier Transform v9.1,” 2017, Available online.
This can be realized with a circular buffer implemented in BRAM.

V. COMPARISON STUDY

In this section, the required hardware resources, speed, area, and power consumption for the two oSDFT structures are compared with the Xilinx FFT IP. For this analysis, the built-in synthesizer and design analyzer of Vivado is used. The power consumption is also measured with the Microchip Power Debugger tool. For the purpose of inspecting the numerical accuracy of the implemented hardware, simulation models are created in MATLAB for the two oSDFT structures to compare them with the FFT IP provided by Xilinx. The comparing metrics of the different properties are calculated assuming both the oSDFT and FFT structures are calculating a sliding-window Fourier transform.

A. RESOURCE USAGE OF THE STRUCTURES

The two main resources used by the transformation structures are the DSP blocks and the memory resources. The memory resources consist of two solutions: BRAMs and LUT-RAMs. The FFT IP utilizes BRAMs exclusively, while the data of the implemented oSDFT structures are partially stored in LUT-RAM as described in Section III. The Xilinx FFT design uses minimal parallelization, depending on the length of the transform, to optimize the utilization of the reserved resources. The level of parallelization is fixed and cannot be altered. On the other hand, the oSDFT structures use a fixed level of parallelization therefore, their resource usage is constant over the examined transformation window lengths.

As shown in Fig. 9, the FFT IP has an advantage of low resource usage for smaller transform lengths, but becomes comparable to the oSDFT solutions for larger lengths, and even surpasses the number of DSPs used in the resonator-based structure. The resonator- and modulator-based structures use four and eight DSP blocks per branch, respectively. This gives the result of 32 and 64 DSP blocks used by the two designs.

In Fig. 10 similar trends can be observed in the case of the number of BRAMs used by the three architectures. The FFT IP loses its initial advantage in the case of longer transform lengths. As described before, the oSDFT structures store their coefficients in BRAMs, which are shared between two branches for the resonator-based oSDFT, and one 36Kb BRAM block is used per branch in the modulator-based oSDFT, resulting in 8 and 16 blocks used respectively. In the case of the LUT-RAM, only the oSDFT architectures utilize this type of resource. Both structures store their state variables as two 24-bit values per branch, in 6-input, 1-output LUTs. This requires 48 LUTs per branch, and 384 LUTs in total for both structures and a transform length up to 512 points can be used.

Up to 2048 points, the maximal length is limited only by the memory resources. In the case of the resonator-based structure, both the BRAMs and LUT-RAMs limit the length at 512 points, and doubling the length requires both of these resources to be doubled. The BRAMs in the modulator-based structure can hold coefficients up to 1024 points, but the LUT-RAMs are also limited at 512 points. Therefore, for 1024 points, only the number of LUTs used needs to be doubled, and for greater lengths, the numbers of both resources need to be doubled when the transform length is doubled. Over 2048 points, the DSP cores also limit the maximal length. By keeping the input resolution, each DSP needs to be cascaded with an additional DSP. This doubles the width to 48 bits, enabling the handling of up to $N = 2^{35}$ long transformations. This however doubles the number of DSP blocks used.

The results considering the number of hardware resources are summarized in Table 1.

B. LATENCY, POWER AND ENERGY USAGE

After the circuits are synthesized, a preliminary maximal clock frequency of the implemented FPGA circuits can be calculated using the Vivado timing tool. As expected, the
TABLE 1. Comparison of the FFT IP and oSDFT structures in terms of the required number of hardware resources: DSP, 18Kb BRAM and LUT.

| N    | FFT IP | Res. oSDFT | Mod. oSDFT | FFT IP | Res. oSDFT | Mod. oSDFT | FFT IP | Res. oSDFT | Mod. oSDFT |
|------|--------|------------|------------|--------|------------|------------|--------|------------|------------|
| 8    | 10     | 32         | 64         | 1      | 8          | 16         | 0      | 384        | 384        |
| 16   | 12     | 32         | 64         | 1      | 8          | 16         | 0      | 384        | 384        |
| 32   | 18     | 32         | 64         | 2      | 8          | 16         | 0      | 384        | 384        |
| 64   | 20     | 32         | 64         | 3      | 8          | 16         | 0      | 384        | 384        |
| 128  | 32     | 32         | 64         | 6      | 8          | 16         | 0      | 384        | 384        |
| 256  | 34     | 32         | 64         | 7      | 8          | 16         | 0      | 384        | 384        |
| 512  | 46     | 32         | 64         | 10     | 8          | 16         | 0      | 384        | 384        |

pipelined FFT can operate on a high, 250 MHz frequency, while the resonator-based structure can reach 50 MHz, and the modulator-based structure can reach 35 MHz maximum. By using these frequencies and the number of steps needed to process a new sample, the latency of the three investigated structures can be calculated. In Fig. 11 these values can be observed. For the oSDFT structures, the latency grows linearly with the length of the transformation, since the eight branches are used in time multiplexing. The latency of the FFT IP core depends not only on the length of the transform, but also on the level of parallelization changing with the number of samples. The results show that both oSDFT structures have lower latency, despite working on a fraction of the clock frequency of the FFT IP and using only low level parallelization.

The power consumption is estimated by the Vivado synthesized design analyzer power report function, using default values, and excluding input and output power for a better comparison. The implemented designs are also measured, and compared to an FPGA programmed up with an empty design on the Basys 3 board. The dynamic power consumption of the three designs is calculated by subtracting the power consumption of the empty FPGA and the board from the measured values. The estimation of the design analyzer showed only a 10% deviation for the resonator-based structure and the FFT IP compared to the measurements. For the modulator-based structure, the estimation is very close to the resonator-based structure’s power consumption, however the measured values are twice as high. Because the modulator-based structure uses twice as many resources, while running on only a 30% slower clock frequency, the higher value was determined to be the reasonable result. Based on these results, the resonator-based structure has the advantage over the Xilinx FFT IP, while the modulator-based structure has relatively high power consumption, as it is shown in Fig. 12. The resonator-based structure has the lowest power consumption for every transform length, even when the FFT IP uses fewer resources. This is probably caused by the fact that the FFT IP operates on a significantly higher frequency, which causes the dynamic switching power to be dominant in the circuit. The modulator-based structure’s power consumption is surpassed by the FFT IP at the transform length of 128.

In Fig. 13, the energy required to process a new sample in an N length transform is shown. It can be seen that the conversion requires significantly more energy for the FFT IP, and the difference grows larger with the increase of the transform length. These results show that there is no need to fully parallelize the computation of the oSDFT branches to achieve lower latency compared with the FFT IP. Therefore, the level of parallelization we chose resulted in
higher performance, even as the resource usage remained at a level comparable to that of the FFT IP.

The results considering the latency, power and energy usage are summarized in Table 2.

### TABLE 2. Comparison of the FFT IP and oSDFT structures in terms of latency, power and energy consumption.

| N   | FFT IP (µs) | Res. oSDFT | Mod. oSDFT | FFT IP (W) | Res. oSDFT | Mod. oSDFT | FFT IP (nJ) | Res. oSDFT | Mod. oSDFT |
|-----|-------------|------------|------------|------------|------------|------------|-------------|------------|------------|
| 8   | 0.18        | 0.02       | 0.03       | 0.13       | 0.11       | 0.19       | 23.37       | 2.16       | 5.43       |
| 16  | 0.27        | 0.06       | 0.09       | 0.14       | 0.11       | 0.19       | 37.25       | 6.48       | 16.29      |
| 32  | 0.43        | 0.10       | 0.14       | 0.16       | 0.11       | 0.19       | 68.48       | 10.80      | 27.14      |
| 64  | 0.58        | 0.18       | 0.26       | 0.18       | 0.11       | 0.19       | 105.12      | 19.44      | 48.86      |
| 128 | 0.88        | 0.34       | 0.49       | 0.23       | 0.11       | 0.19       | 199.78      | 36.72      | 92.29      |
| 256 | 1.42        | 0.66       | 0.94       | 0.25       | 0.11       | 0.19       | 358.85      | 71.28      | 179.14     |
| 512 | 2.49        | 1.30       | 1.86       | 0.3        | 0.11       | 0.19       | 742.62      | 140.4      | 352.86     |

### FIGURE 13. Comparison of the FFT IP and oSDFT structures in terms of energy required for N point transformation.

As can be seen the errors of the investigated structures differ both in magnitude – or offset in other words – and in variance. The FFT IP has the lowest error as it requires the smallest number of operations due to the applied butterfly structure, which can reuse multiplication results even amongst DFT bins thanks to the divide-and-conquer method. The two oSDFT structures have higher errors, especially the resonator-based structure, which has an order of magnitude higher error compared with the FFT IP. The error of the two oSDFT structures were found via simulations to be increasing with the number of samples. This can
be explained by the higher number of operations and the inaccuracies of the twiddle factor. The increase has shown a linear pattern in function of the length of the observations window, however the exact mathematical formulation was not investigated in this paper.

The relatively large offset error is caused by the numerical error from storing the constant $W_N^k$ pole with finite precision. This error will not be averaged out to any extent over the period of $N$ in contrast to the modulator based oSDFT, where not only a single time position of the twiddle factor but a whole cycle is stored and used. This attribute of the resonator structure leads to a constant frequency offset in the center frequency of the resonators, manifesting in a measurement offset. The resonator based oSDFT has the highest variance due to the fact that the finite precision multiplication by the twiddle factor – which can be considered as an additional noise source – is located within the resonator’s loop in contrast to the modulator version. This noise will dominate the variance due to the integrator’s accumulative nature, thus it leads to randomly displaced poles over the complex plane.

By enlarging the fractional part, both the magnitude and the variance of the error can be effectively reduced for all of the presented architectures [14].

VI. CONCLUSION

In this work, high-speed and stable recursive architectures of the resonator- and modulator-based oSDFT structures implemented on an FPGA platform are presented. The results of the comparison indicate, that both oSDFT structures could outperform the Xilinx FFT IP in terms of speed and energy efficiency, while using a comparable amount of resources. Therefore, the oSDFT structures can process signals with a higher sampling rate, or use fewer resources than the IP at the same sampling rate, all while using less energy for the calculations. This makes these structures preferable candidates for high-speed DSP applications, in which sliding window DFT calculations are required. The simulations show the disadvantages of the two structures. Although both have long-term stability, given the constraints, the FFT IP proves to have higher accuracy. In this regard, a possible area of research could be the investigation of the implemented structures, whether their accuracy can be improved while keeping the original constraints. Based on the results, it can be concluded that the oSDFT structures could be very promising candidates if fast and accurate sample-by-sample re-evaluation of the DFT is required. Also, given these results, a VLSI implementation of the structures, exploring additional optimization possibilities could be a topic of a further research.

APPENDIX

In order to prove that:

$$1 - z^{-N} = \frac{1}{1 + \frac{N}{N} \sum_{k=0}^{N-1} H_k(z)}, \quad \text{(A-1)}$$

the following equality must hold:

$$\sum_{k=0}^{N-1} H_k(z) = \sum_{k=0}^{N-1} \frac{W_N^k z^{-1}}{1 - W_N^k z^{-1}} = N \frac{z^{-N}}{1 - z^{-N}}. \quad \text{(A-2)}$$

We can unfold equation (A-2) as a geometric series. As a result, the previous equation can be reformulated as

$$\sum_{k=0}^{N-1} \frac{W_N^k z^{-1}}{1 - W_N^k z^{-1}} = \sum_{p=0}^{N-1} (z^{-1})^p \sum_{k=0}^{N-1} W_N^{kp}. \quad \text{(A-3)}$$

Furthermore, if we take in consideration that

$$\sum_{k=0}^{N-1} W_N^{kp} z^{-1} = \begin{cases} N, & \text{if } p = 0, N, 2N, \ldots \\ 0, & \text{otherwise} \end{cases} \quad \text{(A-4)}$$

equation (A-3) can be simplified using the formula for the sum of a geometric series as

$$\sum_{k=0}^{N-1} W_N^{kp} z^{-1} = z^{-N}N + z^{-2N}N + \ldots \Rightarrow N - \frac{z^{-N}}{1 - z^{-N}}. \quad \text{(A-5)}$$

As a result, we have shown that (A-2) and (A-1) are valid.

REFERENCES

[1] T. Tyagi and P. Sumathi, “Comprehensive performance evaluation of computationally efficient discrete Fourier transforms for frequency estimation,” IEEE Trans. Instrum. Meas., vol. 69, no. 5, pp. 2155–2163, May 2020.
[2] S.-C. Lai, S.-F. Lei, W.-H. Juang, and C.-H. Luo, “A low-cost, low-complexity, and memory-free architecture of novel recursive DFT and IDFT algorithms for DTMF application,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 9, pp. 711–715, Sep. 2010.
[3] T. Xia, X. Zhang, G. Tan, and Y. Liu, “Synchronous reference frame single-phase phase-locked loop (PLL) algorithm based on half-cycle DFT,” IET Power Electron., vol. 13, no. 9, pp. 1893–1900, Jul. 2020.
[4] T. Schlechter, C. Juritsch, and M. Huemer, “Spectral estimation for long-term evolution transceivers using low-complex filter banks,” J. Eng., vol. 2014, no. 4, pp. 265–274, Jun. 2014.
[5] Z. Chen, Y.-Q. Xu, H. Wang, and D. Guo, “Deep STFT-CNN for spectrum sensing in cognitive radio,” IEEE Commun. Lett., vol. 25, no. 3, pp. 864–868, Mar. 2021.
[6] L. M. Wolff and S. Badri-Hoeheier, “Convolutionally coded hopping pattern for MFSK modulation in underwater acoustic communication,” IEEE Access, vol. 7, pp. 95569–95579, 2019.
[7] C. Q. Howard, “A sliding Goertzel algorithm for adaptive passive neutralizers,” J. Sound Vib., vol. 331, no. 9, pp. 1985–1993, Apr. 2012.
[8] E. Jacobsen and R. Lyons, “The sliding DFT,” IEEE Signal Process. Mag., vol. 20, no. 2, pp. 74–80, Mar. 2003.
[9] K. Duda, “Accurate, guaranteed stable, sliding discrete Fourier transform,” IEEE Signal Process. Mag., vol. 27, no. 6, pp. 124–127, Nov. 2010.
[10] S. C. Lai, S. F. Lei, C. L. Chang, C. C. Lin, and C. H. Luo, “Low computational complexity, low power, and low area design for the implementation of recursive DFT and IDFT algorithms,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 12, pp. 921–925, Dec. 2009.
[11] W.-H. Juang, S.-C. Lai, C.-H. Luo, and S.-Y. Lee, “VLSI architecture for novel hopping discrete Fourier transform computation,” IEEE Access, vol. 6, pp. 30491–30500, 2018.
[12] C.-S. Park, “Guaranteed-stable sliding DFT algorithm with minimal computational requirements,” IEEE Trans. Signal Process., vol. 65, no. 20, pp. 5281–5288, Oct. 2017.
[13] G. Picelli, “Resonator-based digital filters,” IEEE Trans. Circuits Syst., vol. 36, no. 1, pp. 156–159, Jan. 1989.
[14] Z. Kollar, F. Plesznik, and S. Trumpf, “Observer-based recursive sliding discrete Fourier transform [tips & tricks],” IEEE Signal Process. Mag., vol. 35, no. 6, pp. 100–106, Nov. 2018.
[15] G. Simon and G. Péceli, “Convergence properties of an adaptive Fourier analyzer,” IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 46, no. 2, pp. 223–227, Feb. 1999.
[16] L. Sujbert, G. Simon, and G. Péceli, “An observer-based adaptive Fourier analysis,” IEEE Signal Process. Mag., vol. 37, no. 4, pp. 134–143, Jun. 2020.
[17] L. Sujbert, G. Péceli, and G. Simon, “Resonator-based nonparametric identification of linear systems,” IEEE Trans. Instrum. Meas., vol. 54, no. 1, pp. 386–390, Feb. 2005.
[18] M. Kovács and Z. Kollár, “Software implementation of the recursive discrete Fourier transform,” in Proc. 27th Int. Conf. Radioelektronika (RADIOELEKTRONIKA), Apr. 2017, pp. 1–5.

PETER PLESZNIK received the B.S. degree in electrical engineering from the Budapest University of Technology and Economics, Hungary, in 2017, and the M.S. degree in electronics engineering from Kyungpook National University, Daegu, South Korea, in 2021. He is currently with Silicon Labs, Hungary, still pursuing low-power high performance DSP research. The synthesizable logics implemented by this study is being extended and integrated to the low-power microcontrollers with active noise canceller unit.

ZSOLT KOLLAR (Member, IEEE) was born in 1983. He received the Diploma and Ph.D. degrees in electrical engineering from the Budapest University of Technology and Economics (BME), in 2008 and 2013, respectively. Since 2016, he has been the Head of the Numerical Optimization (MATLAB) Laboratory. He is currently an Associate Professor at the Department of Measurement and Information Systems, BME. He is responsible for the maintenance and development of the Fdident (Frequency Domain Identification Toolbox) MATLAB Toolbox. He has participated in numerous research and industrial projects. His research interests include digital signal processing, system identification, quantization effects, and wireless communications.

DAEJIN PARK (Member, IEEE) received the B.S. degree in electronics engineering from Kyungpook National University, Daegu, South Korea, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2003 and 2014, respectively. From 2003 to 2014, he was a Research Engineer with SK Hynix Semiconductor, Samsung Electronics, for a period of over 12 years. He worked on designing low-power embedded processor architectures and implementing fully AI-integrated system-on-chip with intelligent embedded software on a custom-designed hardware accelerator, especially for hardware/software tightly coupled applications, such as smart mobile devices and industrial electronics. He was nominated as one of the Presidential Research Fellows 21, Republic of Korea, in 2014. He has been a full-time Associate Professor with the School of Electronics and Electrical Engineering and the School of Electronics Engineering, Kyungpook National University, since 2014. He has published over 200 technical papers and 40 patents.