Journal of
Low Power Electronics and Applications

MDPI

Article

Field-Programmable Gate Array Architecture for the Discrete Orthonormal Stockwell Transform (DOST) Hardware Implementation

Martin Valtierra-Rodriguez [1], Jose-Luis Contreras-Hernandez [2], David Granados-Lieberman [3], Jesus Rooney Rivera-Guillen [1], Juan Pablo Amezquita-Sanchez [1] and David Camarena-Martinez [2,*]

1 ENAP-RG, CA-Sistemas Dinámicos y Control, Facultad de Ingeniería, Campus San Juan del Río, Universidad Autónoma de Querétaro, San Juan del Río 76807, Mexico; martin.valtierra@enap-rg.org (M.V.-R.); jesus.rooney.rivera@uaq.mx (J.R.R.-G.); juan.amezquita@enap-rg.org (J.P.A.-S.)
2 ENAP-RG, Departamento de Ingeniería Electrónica, División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato (UG), Carretera Salamanca-Valle de Santiago km 3.5 + 1.8 km, Comunidad de Palo Blanco, Salamanca 36885, Mexico; jose.contreras@ugto.mx
3 ENAP-Research Group, CA-Fuentes Alternas y Calidad de la Energía Eléctrica, Departamento de Ingeniería Electromecánica, Tecnológico Nacional de Mexico, ITS Irapuato (ITESI), Carr. Irapuato-Silao km 12.5, Colonia El Copal, Irapuato 36821, Mexico; david.granados@enap-rg.org
* Correspondence: david.camarena@enap-rg.org

Abstract: Time–frequency analysis is critical in studying linear and non-linear signals that exhibit variations across both time and frequency domains. Such analysis not only facilitates the identification of transient events and extraction of key features but also aids in displaying signal properties and pattern recognition. Recently, the Discrete Orthonormal Stockwell Transform (DOST) has become increasingly utilized in many specialized signal processing tasks. Given its growing importance, this work proposes a reconfigurable field-programmable gate array (FPGA) architecture designed to efficiently implement the DOST algorithm on cost-effective FPGA chips. An accompanying MATLAB app enables the automatic configuration of the DOST method for varying sizes (64, 128, 256, 512, and 1024 points). For the implementation, a Cyclone V series FPGA device from Intel Altera, featuring the 5CSEMA5F31C6N chip, is used. To provide a complete hardware solution, the proposed DOST core has been integrated into a hybrid ARM-HPS (Advanced RISC Machine–Hard Processor System) control unit, which allows the control of different peripherals, such as communication protocols and VGA-based displays. Results show that less than 5% of the chip's resources are occupied, indicating a low-cost architecture that can be easily integrated into other FPGA structures or hardware systems for diverse applications. Moreover, the accuracy of the proposed FPGA-based implementation is underscored by a root mean squared error of 6.0155×10^{-3} when compared with results from floating-point processors, highlighting its accuracy.

Keywords: discrete orthonormal Stockwell transform; FPGA; reconfigurable hardware architecture; time–frequency representation

check for updates

Citation: Valtierra-Rodriguez, M.; Contreras-Hernandez, J.-L.; Granados-Lieberman, D.; Rivera-Guillen, J.R.; Amezquita-Sanchez, J.P.; Camarena-Martinez, D. Field-Programmable Gate Array Architecture for the Discrete Orthonormal Stockwell Transform (DOST) Hardware Implementation. *J. Low Power Electron. Appl.* **2024**, *14*, 42. https://doi.org/10.3390/jlpea14030042

Academic Editor: Stefania Perri

Received: 17 June 2024
Revised: 2 August 2024
Accepted: 5 August 2024
Published: 7 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Time–frequency (TF) analysis has increasingly become a crucial tool in the study of linear and non-linear systems, which exhibit both stationary and non-stationary signals with variations in time and frequency domains [1,2]; through the use of time–frequency representations, the identification of transient events, extraction of key features, and pattern recognition from the signal properties are possible. These capabilities provide a comprehensive view of a signal's time-varying frequency content, facilitating the extraction of meaningful information. This, in turn, supports informed decision-making in a wide range of applications, from biomedical diagnostics [3] to machine condition monitoring [4].

Over time, many techniques have emerged to generate time–frequency representations. The short-time Fourier transform (STFT) is the simplest and most widely used tool for this purpose [5]; however, its main drawback is its limited time–frequency resolution due to the fixed size of its analysis window. The STFT uses a constant window length throughout the analysis, which forces a trade-off between time resolution and frequency resolution, which can be an inconvenience in many applications. Another widely adopted approach for obtaining the frequency information over time is the wavelet transform (WT) [6], where the underlying principle is that any time series can be decomposed into a self-similar series of dilations and translations of a signal called a mother wavelet. Although the WT inherently provides a representation in the state space of dilations and translations, it can effectively measure the power spectrum in a localized manner when the appropriate wavelet is chosen; however, this requirement is one of its main disadvantages: both the decomposition level and the mother wavelet must be carefully selected to obtain a correct decomposition, which is not always possible. A more recent technique to obtain the time–frequency representation of a signal is the Stockwell transform, also known as S-transform (ST) [7], which has found many applications across various fields. It shares similarities with the WT in terms of progressive time–frequency resolution, but its main significant advantage is that the ST preserves phase information. While the WT uses a variable-length window that scales with frequency, the ST uses a fixed-length Gaussian window. This configuration results in a more evenly distributed frequency resolution across different frequencies. The preservation of phase by the ST is particularly crucial for certain applications, such as the analysis of audio and biomedical signals [8], as it provides critical insights into the signal characteristics. The absolute referencing of the ST means that the phase information corresponds to the argument of the sinusoid at zero time, which aligns with the definition of phase in the Fourier transform. This characteristic enhances its utility in precise phase analysis, making it a preferred choice in fields where phase fidelity is essential. However, a drawback of the ST lies in the redundancy of its calculation for performing a time–frequency representation. To address this issue, the Discrete Orthonormal Stockwell Transform (DOST) was proposed [9]. This method involves dividing the time–frequency domain into regions, each represented by a single coefficient. DOST is based on a series of orthonormal basis functions that effectively localize the Fourier spectrum of the signal, thus retaining the valuable property of the ST in preserving phase information. To improve the computational efficiency of DOST, Wang and Orchard [10] proposed decomposing the DOST matrix. Subsequently, Battistia and Ribaa [11] extended the algorithm presented in [12] to compute the Stockwell coefficients using an admissible window. Both proposed methods achieve a computational complexity of O(Nlog$N$), aligning them with the efficiency of the fast Fourier transform (FFT) algorithm. These advancements make DOST a compelling alternative for time–frequency analysis, offering both speed and accuracy. This has been demonstrated by a number of studies ranging from the analysis of cardiovascular diseases [13] and bearing fault diagnosis [14] to the classification of power quality disturbances [15].

Although time–frequency representations, including the STFT, are computationally intensive and time-consuming—thereby limiting their suitability for continuous signal monitoring—hardware implementations can effectively address these challenges. A hardware implementation involves integrating analytical techniques directly onto physical computing devices, such as field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs), rather than running them through software on general-purpose computers. This approach not only mitigates the computational load but also accelerates data processing speeds. These hardware solutions, therefore, enable the application of time–frequency analysis across a broad range of applications. By taking advantage of the inherent properties of specific hardware, computational complexity can be reduced, leading to faster computations and facilitating real-time analysis [16]. Implementing a time–frequency analysis (TFA) technique in hardware always presents a challenge. However, FPGA systems are particularly suitable for signal analysis due to their ability to perform high-speed calcu-

lations and low power consumption compared to software implementations performed on CPUs or GPUs [17].

Various works have proposed different TFA implementations on FPGA; for example, in [18], a system based on the STFT was developed to extract features in the frequency domain. Another study in [19] proposed a new architecture for the harmonic WT based on the discrete cosine transform. Furthermore, in [20], various parallel architectures for different time–frequency representations based on the FFT are proposed. Among these, the ST is implemented; however, this implementation is based on the classical ST algorithm, i.e., the issue of operational redundancy is not addressed, which can affect its practicality and efficiency. From this point of view, it is crucial to develop an FPGA architecture that not only implements the efficient DOST algorithm but also ensures that it can be deployed on low-cost FPGA chips.

This study introduces an FPGA architecture to implement the DOST algorithm, which can be adapted to the number of processing points required by a specific application by using a developed MATLAB-based app (64, 128, 256, 512, and 1024 points). This flexibility enables the architecture to address a wide range of applications, as reported in the literature, enhancing its applicability and efficiency in practical implementations. Although the proposed architecture is implemented on a Cyclone V series FPGA device from Intel Altera, featuring the 5CSEMA5F31C6N chip, other FPGA boards can also be utilized as the proposed cores are not vendor-dependent. The obtained results demonstrate a low resource usage (<5%) of the chip and high accuracy (root mean square error (RMSE) of 6.0155×10^{-3}) when compared with results from floating-point processors. Additionally, to provide a complete hardware solution, the proposed DOST core has been integrated into a hybrid ARM-HPS (Advanced RISC Machine–Hard Processor System) control unit, which allows the control of different peripherals, such as communication protocols and VGA-based display.

This paper is organized as follows. Section 2 will provide an overview of the S-transform, its different variants, and how its algorithm has evolved to create a more computationally efficient implementation. Section 3 will explain the hardware design of the implementation based on the DOST algorithm. In Section 4, the results of the proposed implementation are presented, and Section 5 offers our conclusions.

2. S-Transform and Its Variants

The ST method is characterized by providing a multi-resolution time–frequency representation of one-dimensional or time signals, which allows for illustrating the behavior of frequency or spectral components over time while preserving the phase information of the signals. To obtain the ST of a function h(t), it can be calculated by convolving a Gaussian window with the function h(t), Equation (1), as it is performed by the FT method [7]:

$$\mathbf{S}(\tau, f) = \int h(t) \frac{|f|}{\sqrt{2\pi}} e^{\frac{(\tau - t)^2 f^2}{2}} e^{-i2\pi f t} dt, \tag{1}$$

where t and τ are a time variable and time translation, while f denotes a frequency variable. Essentially, the ST operates as a windowed FT known as STFT, where the window width, centered at τ, varies inversely with the frequency f.

Employing the integral characteristics of the Gaussian function, it is possible to establish the relation or connection between the FT of h(t), represented as $\mathbf{H}(f)$, and $\mathbf{S}(\tau,f)$, in the following manner:

$$\mathbf{H}(f) = \int_{-\infty}^{\infty} \mathbf{S}(\tau, f) d\tau, \tag{2}$$

Finally, the original or initial function h(t) is calculated by computing the ST as follows:

$$\mathbf{h}(t) = \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} \mathbf{S}(\tau, f) d\tau \right\} e^{i2\pi f t} df, \tag{3}$$

Conversely, the discrete version of ST (DST) is obtained as follows [7]:

$$\mathbf{S}[j, n] = \sum_{m=0}^{N-1} \mathbf{H}[m+n] e^{\frac{-2\pi^2 m^2}{n^2}} e^{\frac{i2\pi mj}{N}}, \tag{4}$$

where j represents a time translation index, n signifies the frequency shift, $e^{\frac{-2\pi^2 m^2}{n^2}}$ denotes the Gaussian function or window employed in the frequency domain, and **H**[.] represents the discrete FT of the discrete signal **h**[l] = h(lT) for l ranging from 0 to $N-1$, with a sampling interval given by T. For the n = 0 voice, define

$$\mathbf{H}[m] = \sum_{m=0}^{N-1} \mathbf{h}[l] e^{\frac{-i2\pi ml}{N}} \quad m = 0, 1, \cdots, N-1, \tag{5}$$

Similar to ST, the inverse DST is determined as follows:

$$\boldsymbol{h}[l] = \frac{1}{N} \sum_{n=0}^{N-1} \left\{ \sum_{j=0}^{N-1} S[j, n] \right\} e^{\frac{i2\pi nl}{N}}, \tag{6}$$

for N size signal.

Observing Equation (6), it is crucial to notice that for a time signal with a length N, N^2 Stockwell coefficients are computed, requiring O(N) computation time for each one. Consequently, the computation of all N^2 coefficients of the ST requires a computational burden of O(N^3). Therefore, considerable computational complexity is produced by the substantial redundancy inherent in its estimated time–frequency plane. This imposes significant computational resources, thereby limiting its effectiveness in processing large datasets.

With the aim of lessening the computational complexity of the DST method, Stockwell [9] introduced an enhanced ST method called the discrete orthonormal ST (DOST). The enhanced method relies on a set of orthonormal basis functions aimed at localizing the Fourier spectrum of the one-dimensional or time signal (1DTS) by calculating the TF plane determined by the ST method without introducing redundancy into the calculated information and preserving the phase attributes of the ST. Stockwell [9] defines a basis vector constructed by a sum of N orthogonal unit length, where each one corresponds to a distinct region in the calculated TF representation. The obtained regions are characterized by the following three parameters: τ indicates the location in time, β represents the width of each frequency band, and ν denotes the center of that band (voice). Employing these three parameters, a kth basis vector is obtained as follows:

$$\mathbf{D}[k]_{[\nu,\beta,\tau]} = \frac{e^{i\pi\tau}}{\sqrt{\beta}} \sum_{\nu-\beta/2}^{\nu+\beta/2-1} e^{-i\frac{2\pi}{N}kf} e^{i\frac{2\pi}{\beta}\tau f}, \tag{7}$$

Later, the inner product of the function **h**[k] (analyzed 1DTS) and **D**[K]$_{[\upsilon,\beta,\tau]}$ is considered, along with the DOST coefficient, denoted by **S**, for the corresponding region with the three parameters [υ,β,τ], which are determined as follows:

$$\begin{aligned} \mathbf{S}_{[\nu,\beta,\tau]} &= \left\langle \mathbf{D}[k]_{[\nu,\beta,\tau]}, \mathbf{h}[k] \right\rangle \\ &= \frac{e^{i\pi\tau}}{\sqrt{\beta}} \sum_{k=0}^{N-1} \sum_{\nu-\beta/2}^{\nu+\beta/2-1} e^{-i\frac{2\pi}{N}kf} e^{i\frac{2\pi}{\beta}\tau f} \mathbf{h}[k] \end{aligned}, \tag{8}$$

To create the set of orthogonal basis vectors from Equation (8) for k = 0, . . ., N − 1, the following equation is carried out:

$$\mathbf{D}[k]_{[\nu,\beta,\tau]} = \frac{ie^{-i\pi\tau}\left(e^{-i2a(\nu-\beta/2-1/2)} - e^{-i2a(\nu+\beta/2-1/2)}\right)}{2\sqrt{\beta}\sin(\pi k/N - \tau\pi/\beta)} \tag{9}$$

where the parameters ν, β, and τ need to be selected appropriately. By letting the variable p index the frequency bands, Stockwell defines the DOST basis vectors for the positive frequency for each p in [9] using the following.

- If p = 0, D[k][ν,β,τ] = 1 (only one basis vector);
- If p = 1, D[k][ν,β,τ] = exp(−i2kπ/N) (only one basis vector);
- For p = 2, 3. . ., log2 N − 1, pick

$$\left.\begin{array}{c} \nu = 2^{(p-1)} + 2^{(p-2)} \\ \beta = 2^{(p-1)} \\ \tau = 0, \ldots, \beta - 1 \end{array}\right\} \text{defines } 2^{p-1} \text{ basis vectors for each frequency band} \tag{10}$$

By combining these basis vectors with the basis vectors for the negative frequencies, it can be demonstrated that these parameter selections yield a set of N orthogonal unit vectors, resulting in N DOST coefficients.

Reordering the summation presented above, Equation (8) yields:

$$\begin{aligned} \mathbf{S}_{[\nu,\beta,\tau]} &= \frac{e^{i\pi\tau}}{\sqrt{\beta}} \sum_{\nu-\beta/2}^{\nu+\beta/2-1} e^{i\frac{2\pi}{\beta}\tau f} \sum_{k=0}^{N-1} e^{-i\frac{2\pi}{N}kf} \mathbf{h}[k] \\ &= \frac{e^{i\pi\tau}}{\sqrt{\beta}} \sum_{\nu-\beta/2}^{\nu+\beta/2-1} e^{i\frac{2\pi}{\beta}\tau f} \mathbf{H}[f] \end{aligned}, \tag{11}$$

where the f value summation is limited to a specific band, which is determined by two parameters, β and ν. Consequently, this summation can be depicted as the inner product between the **H** vector, calculated Fourier coefficients, and a row in a sparse matrix.

Finally, an inverse Fourier transform, denoted by F^{-1}, is applied to the obtained subband of the FT for 1DTS or function $\mathbf{h}[k]$. It is essential to correctly shift the three indices as follows:

$$\mathbf{S}[k] = \sqrt{\beta_k} e^{i\pi\tau_k} F_{\Omega_k}^{-1} F(\mathbf{h}[k]), \tag{12}$$

where $\tau_{k,}$, β_k, and Ω_k indicate a time index, the bandwidth, and the band for the k-th basis vector, respectively.

The DOST is distinguished by its similarity to the general Fourier family transform described in [5], with the key difference being the use of a window with rectangular properties instead of a truncated window with Gaussian properties. In particular, the fast algorithm developed in [7] can be adjusted to generate the conjugate-symmetric DOST.

Other important features are the sampling frequency, Fs, and the number of points, N, used during its computation. Although they are related to each other, it is possible to indicate that Fs determines the bandwidth, η, in Equation (13), and N determines the frequency resolution, Δf, in Equation (14). For instance, if Fs is equal to N, the bandwidth will be given by Fs/2 or $N/2$. Thus, for this particular case, the larger the value of N, the larger the bandwidth. The value of N can be changed to increase or decrease the bandwidth according to the application, with values that are powers of two being the most recommendable.

$$\eta = {}^{F_s}/_{2} \tag{13}$$

$$\Delta f = {}^{F_s}/_{N} \tag{14}$$

On the other hand, from an implementation point of view, the DOST algorithm consists of three main steps: first, computing the fast FT (FFT) of the input signal; second, calculating the inverse FFT (IFFT) for each region; and finally, merging each region to produce the final DOST output. This process is illustrated in Figure 1, where the different colors, as merely an explanatory example, represent the ordered time–frequency information of each region of the signal.

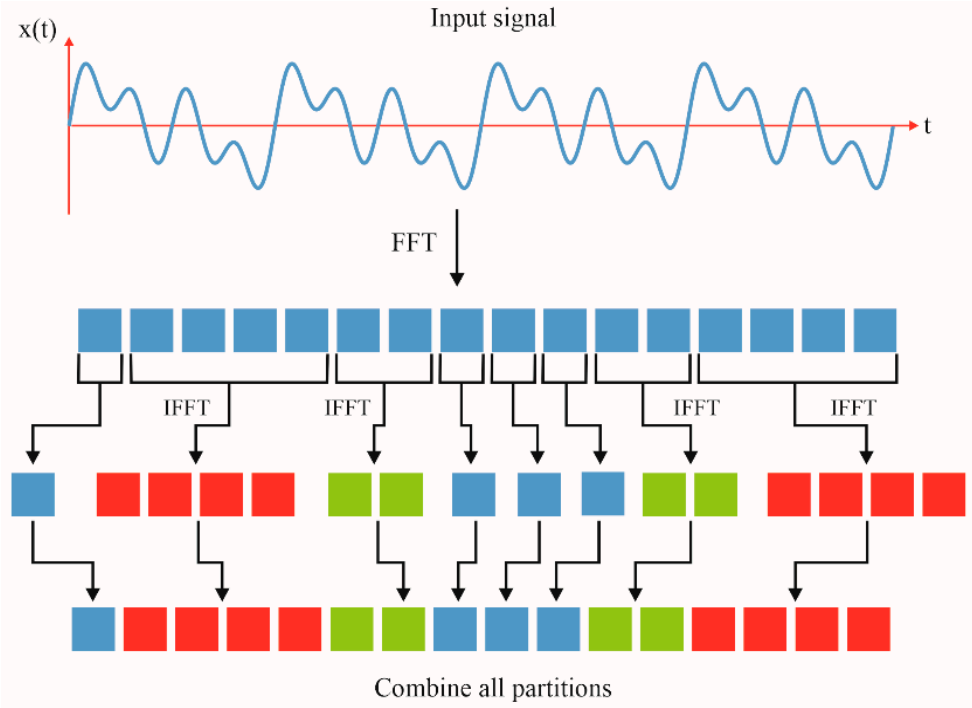


Figure 1. Algorithm for the DOST computation.

3. Proposed Hardware Architecture for the DOST

3.1. Flowchart for the Automatic Generator of the DOST Architecture

In order to provide a hardware solution for a wider range of applications, this work presents a configurable architecture for the DOST method based on the required N points. To achieve this, the flowchart shown in Figure 2 must be followed. Firstly, the number of points, N, for the DOST core must be selected according to the application, keeping in mind that this number is related to the bandwidth. The available sizes are 64, 128, 512, and 1024 points. Next, on the one hand (left path), the VHDL-HDL (Very-High-Speed Integrated Circuits–Hardware Description Language)-based reconfigurable DOST cores are automatically generated by means of a developed MATLAB app; then, the synthesis of the DOST core to fit the desired FPGA platform is performed. On the other hand (right path), the codes are designed to program the ARM_HSP processor to generate a DOST time–frequency representation. Finally, the DOST core is integrated into the ARM-HPS to provide access to different peripherals, such as VGA-based display and communication protocols. Thus, users can apply the developed hardware solution to their specific applications.

3.2. ARM-FPGA Solution

The overall structure of the proposed hybrid ARM-FPGA solution is shown in Figure 3. This solution combines the flexibility of the ARM-HPS control unit with the reconfiguration power of the FPGA. The design mainly consists of five modules: the reconfigurable FPGA unit, the ARM-HPS control unit, the on-chip memory, the SDRAM (synchronous dynamic random-access memory), and the input/output interface. In this regard, the FPGA unit primarily handles the hardware implementation of the DOST algorithm and the VGA controller. The ARM processor is responsible for reordering the results obtained from the DOST to create the DOST spectrum and sending it to graph the results via the VGA

protocol. On the other hand, the SDRAM memory stores the data to be graphed provided by the ARM-HPS, while the on-chip memory stores the values of all the variables within the processor software and the ASCII characters to be graphed on the screen. Finally, the I/O interface unit manages the input and output of data using the UART protocol to the PC. The entire system is interconnected through a 32-bit AXI bus.

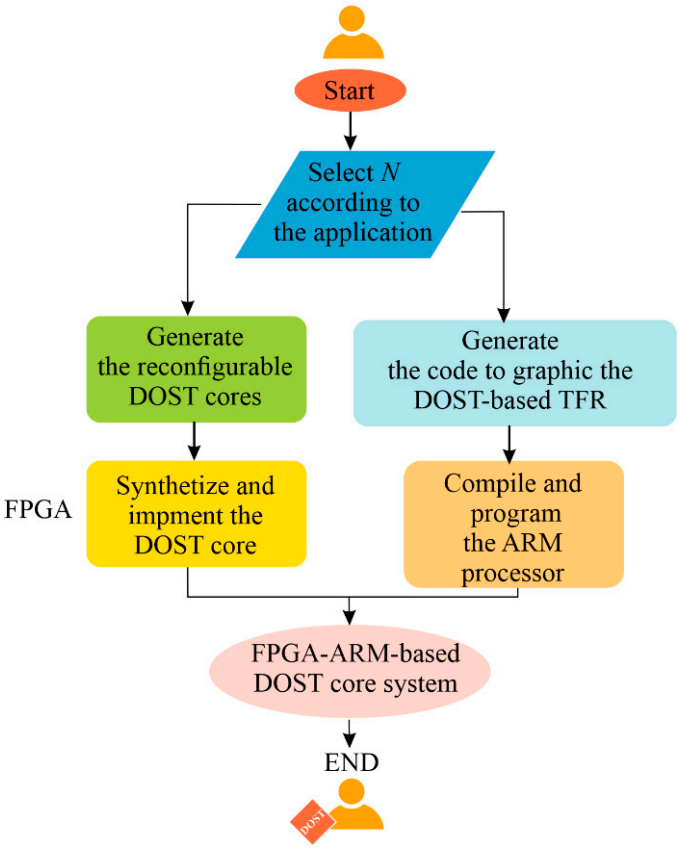


Figure 2. Flowchart to generate the DOST hardware solution.

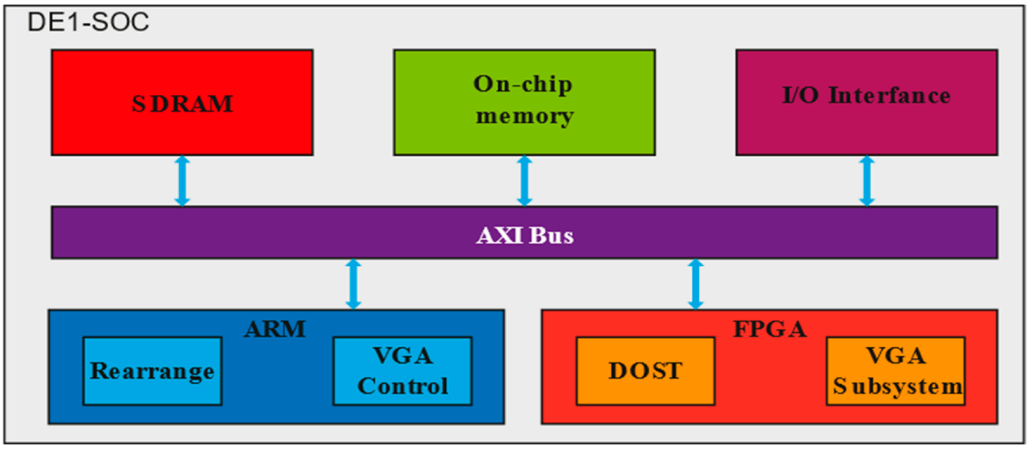


Figure 3. Basic internal modules of ARM-FPGA hybrid architecture.

In the next subsection, the FPGA-based DOST architecture, i.e., the main contribution of this work, is described in detail.

3.3. FPGA-Based DOST Architecture

Figure 4 presents the proposed top-level block diagram for the DOST processor. The Ctrl_DOST block is a finite state machine (FSM) that controls the process of obtaining

the DOST coefficients. Its main function is to provide the necessary parameters for the calculation of the FFT in different regions of the signal, according to the DOST algorithm described in Figure 1.

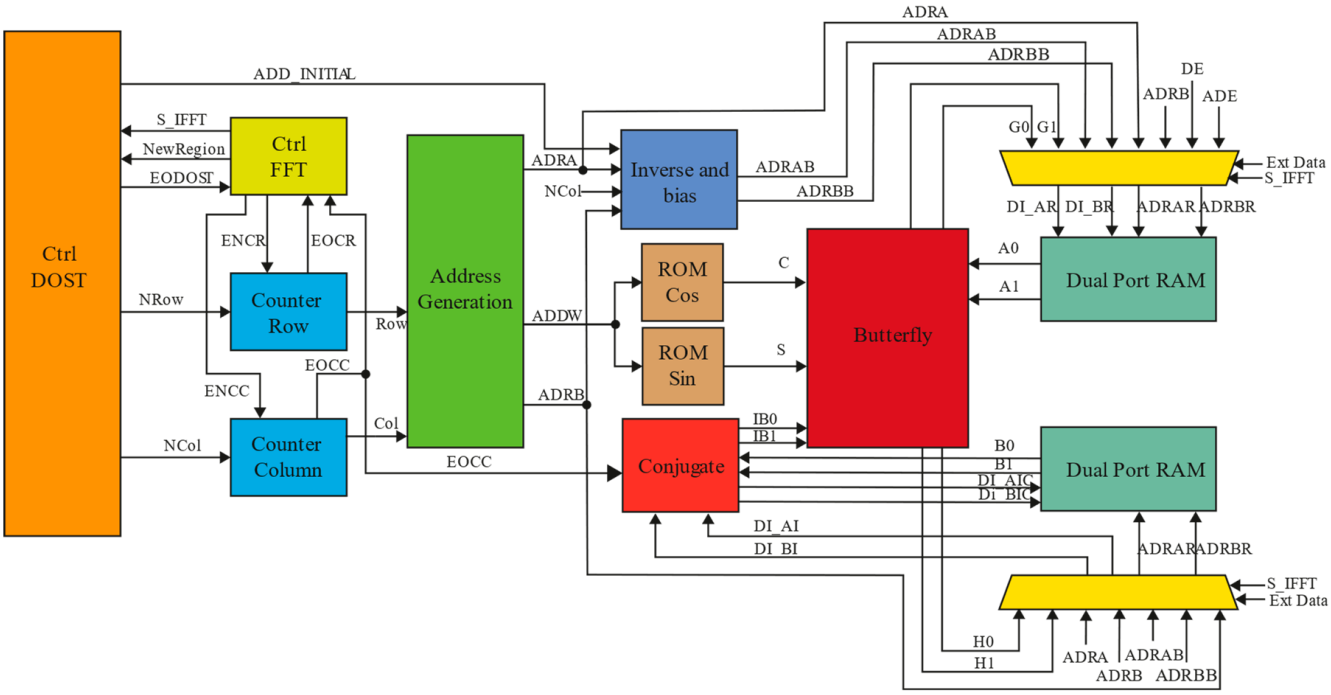


Figure 4. Top-level block diagram for the proposed hardware DOST processor.

This process begins by performing the FFT of the entire signal. Subsequently, several FFTs are calculated by dividing the signal FFT coefficients into different regions. For example, to calculate the 64-point DOST, the signal is divided into regions of 16, 8, 4, 2, 1, 1, 1, 2, 4, 8, and 16 points. Each region will generate an FFT matrix (butterfly diagram) with a variable number of columns and rows; therefore, the Ctrl_DOST block provides the number of columns and rows values through the NCol and NRow signals, and the ADD_INITIAL signal that indicates the start memory address of the region where the FFT coefficients calculated in the Dual-Port RAM will be stored.

The Ctrl_DOST block receives the NewRegion signal to indicate the start of the FFT calculation in a new region and the S_IFFT signal to signal the end of the FFT calculation of the entire input signal. Finally, using the EODOST signal, Ctrl_DOST indicates the completion of the calculation of the DOST coefficients.

The next block is Ctrl_FFT, an FSM that controls the internal process of calculating the FFT. The main function of this block is to manage the CounterRow and CounterColumn counters, which are responsible for counting the rows and columns of the FFT matrix. The ENCR and ENCC signals are used to enable these counters, determining when they will increment the count. An FFT matrix will have N/2 rows and Log2N columns, where N is the number of points. Every time N/2 rows are completed, the Ctrl_FFT block receives the EOCR signal and starts a new column. At the end of all columns, the Ctrl_FFT block receives the EOCC signal, thus indicating the end of the calculation of the FFT of the current region.

The Address_Generation block generates the correct addresses, ADRA and ADRB, where the results returned by the butterfly operation will be read and written. These addresses are determined by the values of the Row and Col signals coming from the CounterRow and CounterColumn counters. In turn, address generation will also generate the addresses ADDW to read from the ROM_Sin and ROM_Cos lookup tables to select the correct values of the twiddle factor depending on the position in the FFT matrix indicated by the Row and Col signals.

The two multiplexers in Figure 4 select between different input data sources and their corresponding addresses. Data can come from outside through the AXI bus via the DE signal and be stored at the ADRE address. During the calculation of the FFT matrix, the input data can also come from the butterfly results G0 and G1 for the real part and H0 and H1 for the imaginary part of the result. As for addresses, these can be generated directly by the Address_Generation block, or, after calculating the FFT of the entire signal, they can come from the Inverse and Bias block. In this case, the mowed directions are represented by the signals ADRAB and ADRBB and depend on the region of the FFT coefficient signal where the FFT is being recalculated for the subsequent calculation of the IFFT. The upper multiplexer provides the real part of the two data computed by the butterfly block, DI_AR and DI_BR, as well as their respective addresses, ADRAR and ADRBR, to be stored in the upper Dual-Port RAM. On the other hand, the lower multiplexer sends the two data representing the imaginary part, DI_AI and DI_BI, to the Conjugate block. This block will perform the 2's complement of the data if the IFFT is being calculated. Otherwise, the data will be sent directly to the lower Dual-Port RAM. The Ext_Data signals indicate whether the data source is external, and the S_IFFT signal indicates whether the process is in the process of calculating the IFFT of the regions.

The Inverse and Bias block, illustrated in Figure 5, is used to adapt the memory addresses, ADRA and ADRB, according to the region in which the process is located. At the beginning of each new region, it performs bit inversion on the addresses. Subsequently, as each new region starts at an offset address, the addresses are adjusted by adding the value of ADD_INITIAL, thus obtaining the addresses ADRAB and ADRBB.

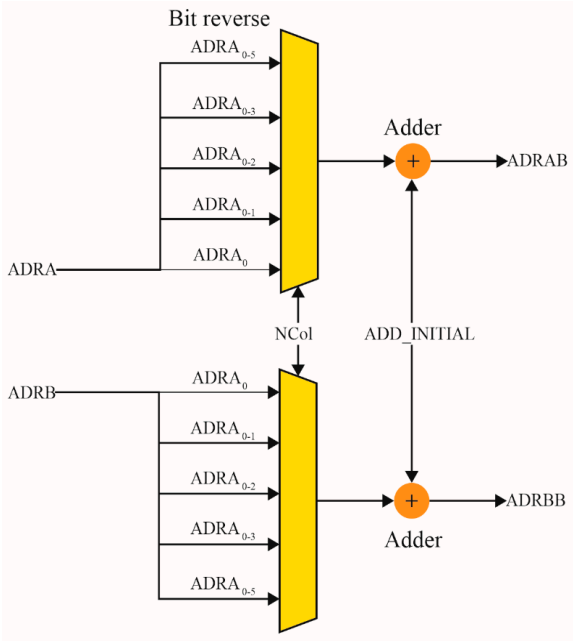


Figure 5. Inverse and bias block.

The ROM Cos and ROM Sin blocks are lookup tables that store the N/2 cosine and sine values corresponding to the twist factor used in the butterfly operation. These values have a fixed-point precision of Q15 bits (15 bits of fractional part and 3 bits of integer part). This bit width selection is because the multiplication and DSP blocks are optimized for 18 bits on Cyclone V devices.

One of the most used methods to calculate the IFFT is through the calculation of the direct FFT (Forward FFT), performing the conjugate of the input and output data, as shown in Figure 6. The conjugate block is responsible for this operation, allowing the IFFT to be calculated using the FFT according to Equation (13). For the division operation with a bit width of $N = 2^i$, where i is a natural number, it is performed by shifting $\log_2 N$ places.

$$\mathbf{IFFT}(X) = \frac{1}{N} conj(\mathbf{FFT}(conj(X))) \quad (15)$$

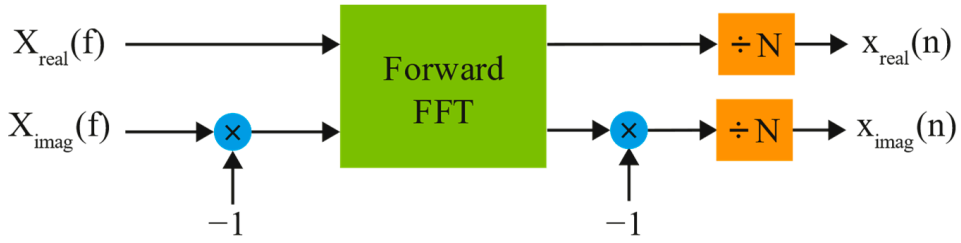


Figure 6. Calculation of IFFT through FFT.

The butterfly processing unit is a combinational block and is the heart of any FFT-based algorithm. Its function is to take the data from memory and calculate the simple two-point FFT. This operation is schematically shown in Figure 7, where A0 and A1 are the real part and B0 and B1 are the imaginary part of the inputs of the previous level; at its output, two complex numbers, Y0 and Y1, made up of G0 and G1, which represent the real part, and H0 and H1, which represent the imaginary part of said numbers, will be obtained. W is composed of C and S, which are the cosine and sine values from the lookup tables ROMCos and ROMSin, respectively.

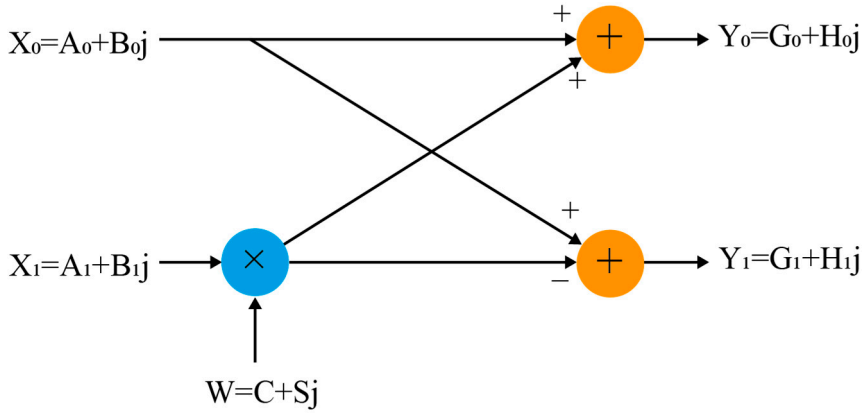


Figure 7. Butterfly operation.

Butterfly block contains four 18-bit multipliers, adders, and subtractors. To avoid the possibility of overflow, point to an additional 5 bits to accommodate the "bit growth" that occurs as the FFT processor goes through the butterfly levels. This is critical to preserving precision since it performs all the calculations using signed integer arithmetic. Finally, for each stage, the FFT is reduced by a factor of 2 in the radix-2 algorithms so that the final output of the FFT still maintains the same bit size. The results are written back to the same memory locations as an algorithm is used in place.

Finally, two blocks of Dual_Port_RAM are used to store the results of each butterfly operation. Since the data being read and written are complex in nature, with a real and an imaginary part, an upper RAM block is used to store the real part, and another lower RAM block is used to store the imaginary part. The input data and their respective addresses are managed by the multiplexers, depending on the source of the data. The outputs of the upper Dual-Port RAM, A0 and A1, represent the real part of the data, while the outputs of the lower Dual-Port RAM, B0 and B1, represent the imaginary part.

4. Experimentation and Results

4.1. Experimental Setup

To validate the proposal, the experimental setup shown in Figure 8a is used. In Figure 8a, the connections of the FPGA board are depicted. The JTAG interface is used

for programming the FPGA chip, while a UART-USB cable is utilized to configure the Linux operating system on the SD card, facilitating communication between the card and a computer. An Ethernet cable is employed to establish an SSH (Secure Shell) connection and transfer the .c file for ARM programming. Finally, a VGA cable is used to transmit information to the monitor. In this case, the time–frequency representation (TFR) results for the analyzed signal are displayed as shown in Figure 8c. The VGA displays the DOST-based TFR for a linear chirp signal, which will be detailed in the next subsection and shown in more detail in a MATLAB environment.

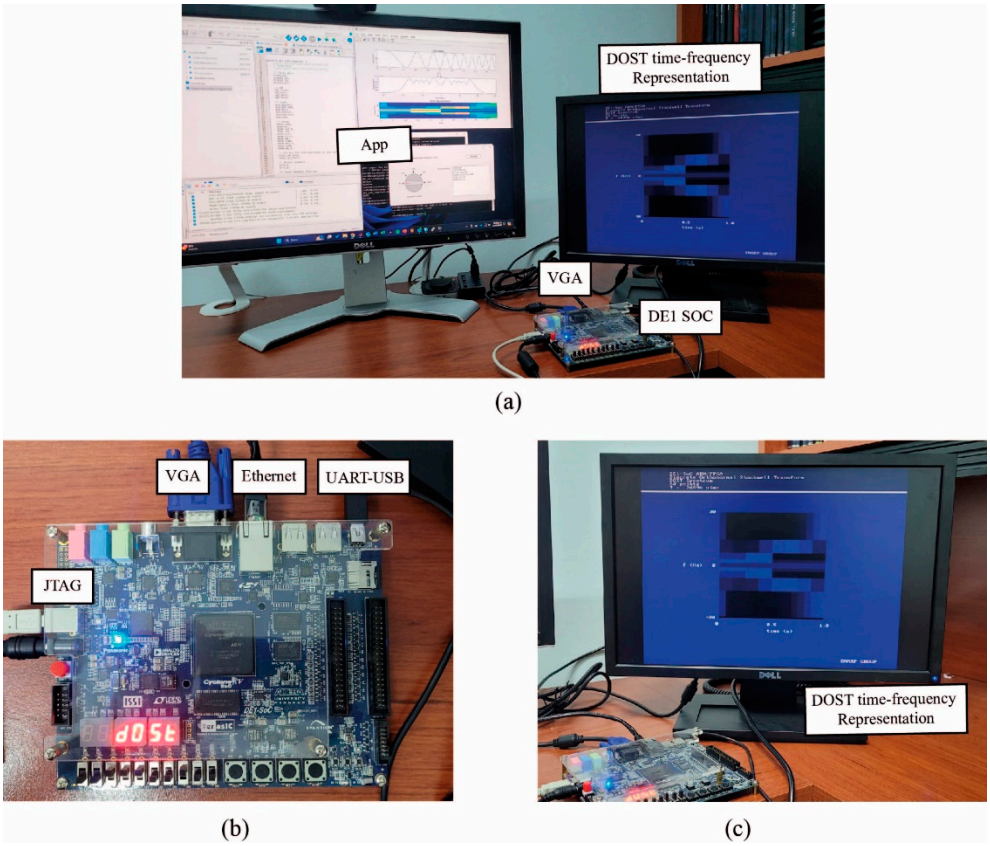


Figure 8. Experimental setup: (**a**) DE1-SOC connections, (**b**) FPGA board, and (**c**) TFR results.

4.2. Results for the DOST Method of 64 Points

This section presents the results corresponding to the configuration of the DOST architecture. It is important to highlight that the proposed architecture does not vary according to the number of points used; what changes is the storage size of both the sine and cosine components of the gyration factor, as well as the number of DOST coefficients to store. On the other hand, in order to compare with the FPGA implementation of the ST presented in [20], the results for the 64-point configuration are detailed first in this section. Then, the results for the other N-point configurations are presented summarily in the next subsection.

In Table 1, the resources used by the DOST implementation are shown. The maximum resources of the FPGA chip used are 32,075 ALMs (Adaptive Logic Modules), 128,300 registers, 3970 M10K memory blocks (4,065,280 bits), 87 DSP (digital signal processing) blocks, and 174 multipliers. As can be observed, in all the elements (DSP/multipliers, BRAMs, logic elements/ALMs, and registers), the resource usage is less than 5%, indicating efficient structures and the possibility of integrating other cores for specific applications, even when low-end devices are used.

Table 1. Resources utilization comparison for 64-bit ST.

Implementation	DSP/Multipliers	BRAMs	Logic Elements/ALMs	Registers	Slices
DOST TRANSFORM (DE1-SOC)	4 (4.6%)	2304 bits (0.0005%)	192 (0.0005%)	22 (0.0002%)	-
DOST TRANSFORM (Basys 3)	4 (4.44%)	1 block RAM (2%)	390 (0.01%)	20 (0.05%)	62 (0.01%)
S TRANSFORM [20]	0	0	652	68	182

Also, in order to make a comparison with a similar implementation reported in the literature [20], the DOST architecture is also implemented on the Basys 3 board (XC7A35T-1CPG236C) with the following characteristics: 33,280 logic cells in 5200 slices, 1 Block RAM of 1800 kbits, and 90 DSP slices. However, it is important to emphasize that comparing resources between different FPGA chips of different models and companies is not a trivial task due to the different architectures of the logic blocks. Additionally, in Table 1, a comparison with an ST architecture in FPGA presented in [20] can be observed, where the FPGA implementation was made using the classic discrete ST. In this architecture, designs with a fully parallel approach are proposed to minimize latency at the expense of extensive use of LUTs (area). This architecture is based on the utilization of a combinational CORDIC, which, when handling data with many bits, results in significant resource consumption [21,22]. Furthermore, the architecture proposed in [20] is based on DST and not on the more efficient DOST algorithm, employing a highly redundant algorithm with high computational complexity, making it impractical for implementations with large datasets. Table 1 compares the data for a 64-point FFT. The proposed architecture demonstrates efficient resource usage, utilizing only half of the logic elements and a third of the registers compared to the architecture presented in [20].

From Table 2, it can be observed that the ST exhibits a latency factor of 384, with a maximum clock frequency of 119 MHz. This means that if the common 50 MHz clock of the DE1-SOC card is used, calculating the DOST coefficients would take 7.68 microseconds, whereas using the maximum frequency would reduce this time to 3.22 microseconds. The power consumption is only 68 nW. Compared to the other reported implementations, the proposed architecture consumes less than half as much power, yet the energy consumed by a hardware design will always depend on various factors, including the algorithm used and, importantly, the design methodology implemented. Another important point is that being a parallel structure, if you want to maintain the same architecture for handling more data based on a combinational CORDIC and also a combinational FFT, it will lead to extensive use of resources and energy consumption, potentially lowering the maximum frequency. Only the CORDIC block, as reported in [12,13], would produce such an effect on the architecture. Furthermore, it is important to mention that low latency such as those reported in Table 2, i.e., values less than 500 clock cycles at a frequency higher than 100 MHz, is required for high-speed applications such as real-time biomedical diagnosis, protection of power systems, instantaneous seismic attributes estimation, and so on.

Table 2. Comparison of latency and power consumption.

Implementation	Latency (No of Clock Cycles)	Power Consumption (nW)	Maximum CLK Frequency (MHz)
DOST TRANSFORM (DE1-SOC)	384	68	119
DOST TRANSFORM (Basys 3)	384	71	121
S TRANSFORM [20]	284	154.2	198

Another crucial parameter is the quality of the results. The fixed-point DOST coefficients generated by the FPGA have been compared with those obtained from floating-point results calculated in MATLAB, as illustrated in Figure 9a,b. To quantify the quality of the results, the absolute error is presented as the value of the difference between the processor results and the FPGA results, as shown in Figure 9c. Also, the RMSE was found to be 6.0155×10^{-3}. This low value indicates that the FPGA implementation closely matches the floating-point results from MATLAB, confirming the precision of the fixed-point approach. Additionally, in the context of the overall system resolution, this error is negligible for the analysis of signals sampled even using 16-bit ADCs (analog-to-digital converters), thereby not impacting the performance or reliability of the application.

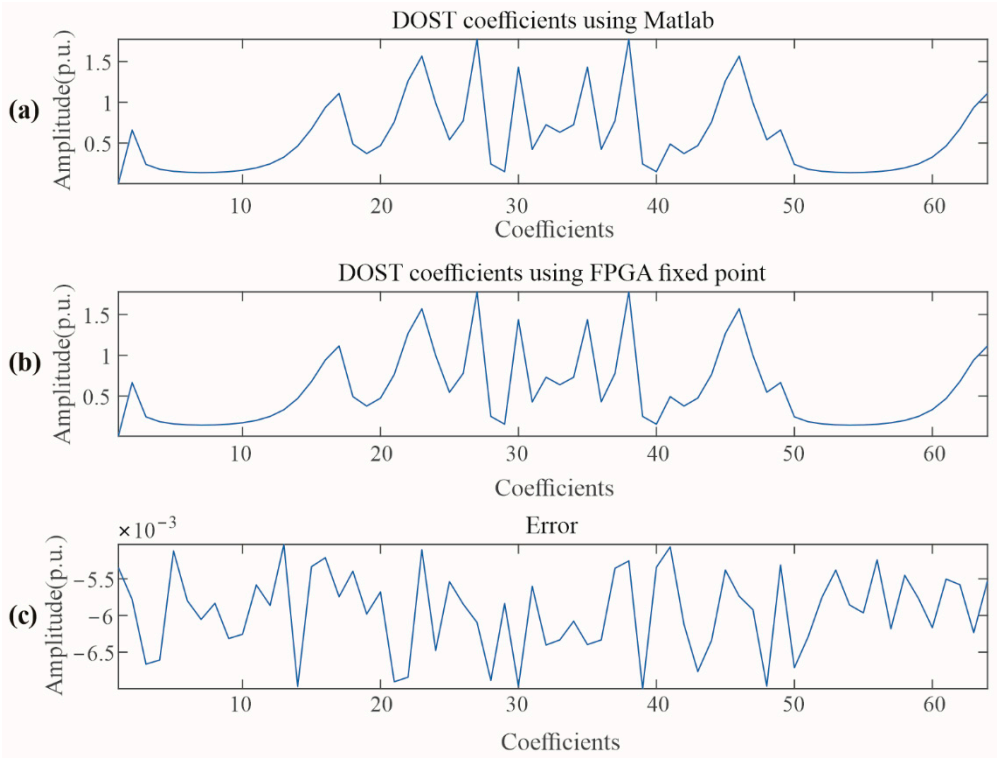


Figure 9. DOST coefficients of a chirp function. (**a**) Absolute DOST coefficients obtained from MATLAB. (**b**) Absolute DOST coefficients obtained from HDL simulation. (**c**) Absolute error of simulation results compared with MATLAB results.

Figure 10 shows the simulation of the calculation of the DOST, specifically the stage that follows the general calculation of the FFT of the signal. The "Newregion" signal indicates the start of the IFFT calculation through the calculation of the FFT for a new region, as indicated in Figure 1. To do this, the number of columns is adjusted (NCOL signal) since each new region requires a different number of points. Once the IFFT of each region has been calculated, the results are joined to complete the calculation of the DOST coefficients, represented by DR and DI, which are the real and imaginary parts, respectively. Figure 11 shows the time–frequency representation of the calculated DOST coefficients. Finally, Figure 12 shows the register transfer level (RTL) view of the DOST architecture, where the connections shown in Figure 4 can be seen. In addition, Figure 13 shows a zoomed-in version of the main blocks to better assess the corresponding inputs and outputs, where counters, ROMs, and RAMs are generic blocks.

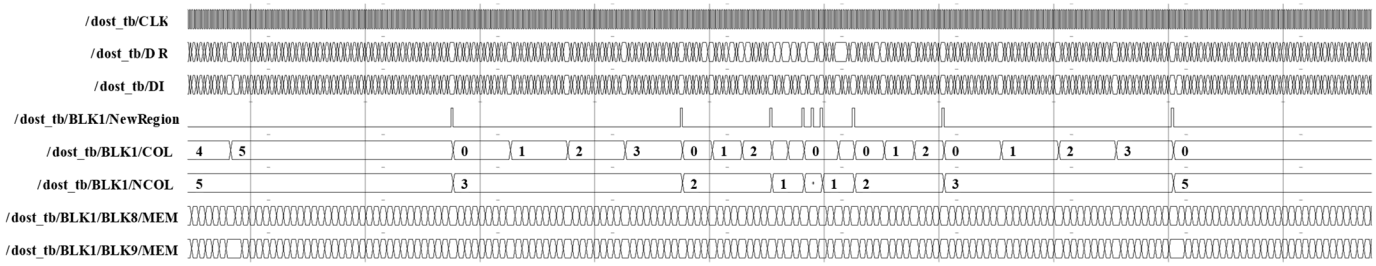


Figure 10. Simulation of DOST implementation.

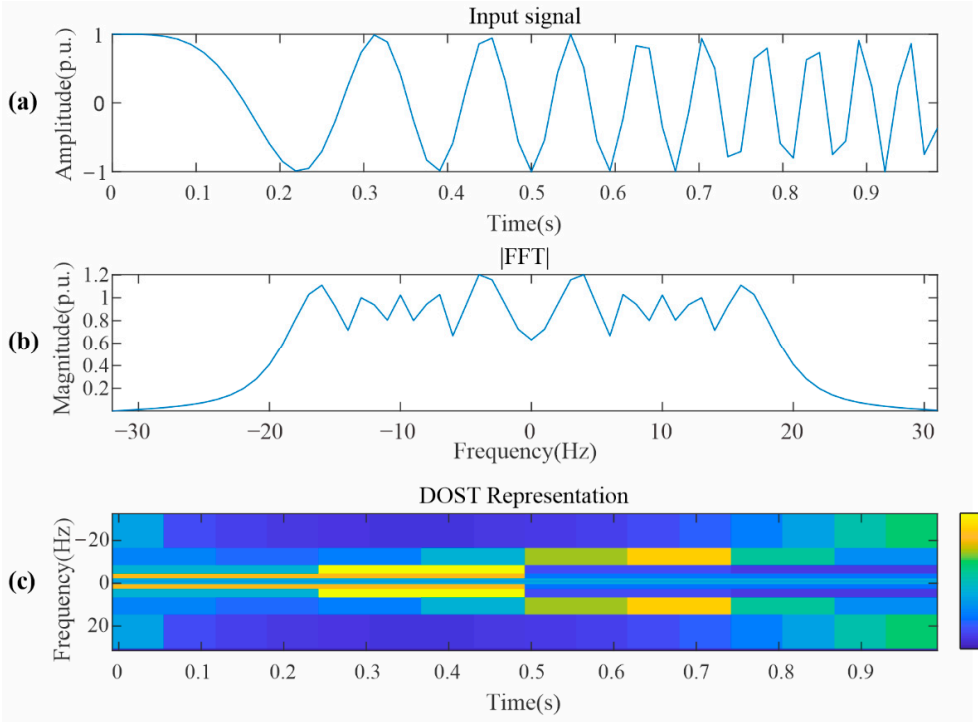


Figure 11. Results obtained from hardware implementation: (**a**) Input chirp signal, (**b**) absolute FFT coefficients, and (**c**) DOST TFR.

4.3. Results for the DOST Method of N Points

One of the main advantages of the proposed architecture is its easy adaptability to use a greater number of points in the FFT and process a larger amount of data. This is achieved through small modifications that can be automated to generate VHDL code. These adjustments include expanding the data in the sine and cosine ROMs to generate the twiddle factor, increasing the generation of addresses, expanding the number of addresses in the RAMs, and increasing the number of columns and rows that control the FFT process. With these considerations in mind, a simple interface shown in Figure 14 has been developed to create the DOST architecture for 64, 128, 256, 512, and 1024 points and generate the files that require changes. Table 3 details the resources used for the proposed architecture. The most notable change is the increased number of RAM blocks, attributed to the expanded sine and cosine ROMs, while other resources have remained largely consistent.

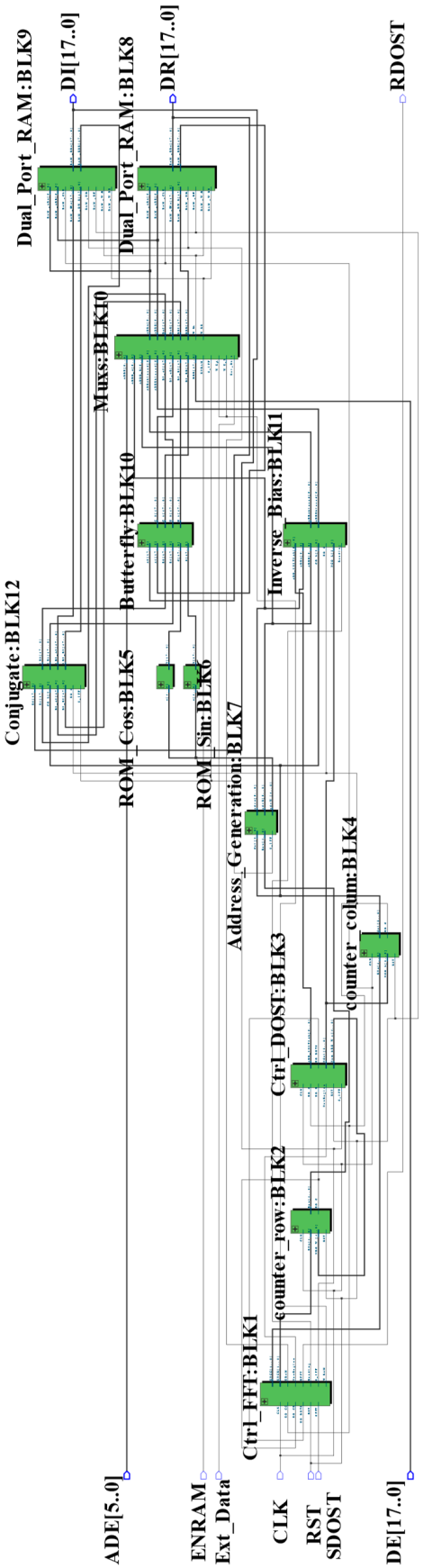


Figure 12. RTL diagram for the DOST processor.

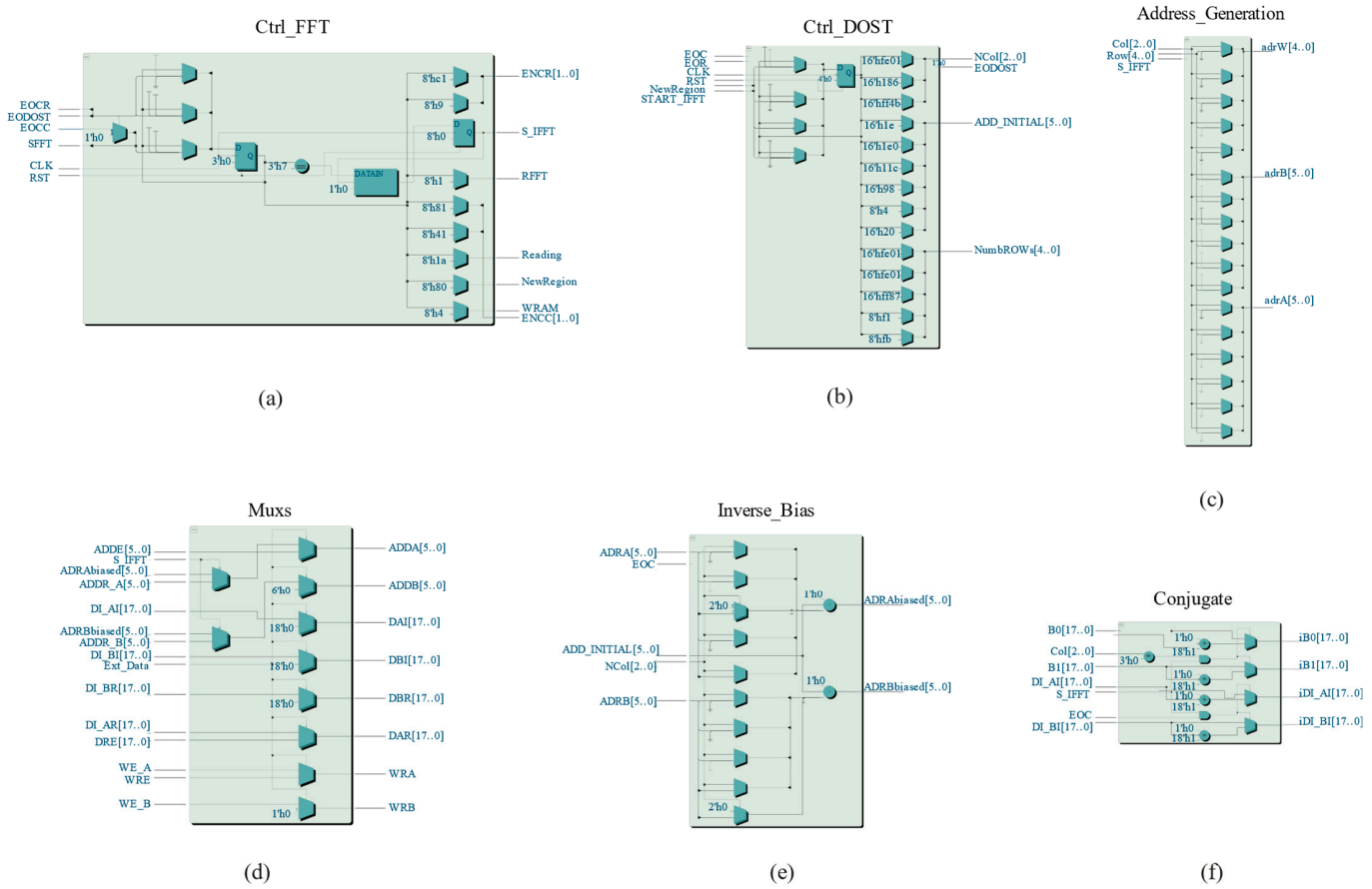


Figure 13. Blocks for the RTL diagram: (**a**) Ctrl_FFT, (**b**) Ctrl_DOST, (**c**) Address_Generation, (**d**) Muxs, (**e**) Inverse_Bias, and (**f**) Conjugate.

Table 3. Resources used by DOST implementation using different numbers of points in the FFT.

FFT Points	DSP Blocks	BRAMs	ALMs	Registers
64	4	2304	192	22
128	4	4608	220	22
256	4	9216	265	23
512	4	18,432	369	23
1024	4	36,864	505	29

Figure 15 shows the DOST results for all the N-configurations: 64, 128, 256, 512, and 1024 points. In this case, a linear chirp signal from 0 to 20 Hz is used as the input signal. The characteristics of this signal allow for computing its time–frequency representation across all the bandwidths possible for the proposed N-configurations. As observed in Figure 15a, the FFT coefficients range symmetrically from 0 to 20 Hz (middle figure), and their evolution over time is displayed using their DOST representation (right figure). In Figure 15b–e, the same behavior is observed; however, as the number of points increases, the time resolution of the input signal improves, and consequently, the bandwidth obtained for the DOST time–frequency representation increases. This increase in bandwidth allows for the observation of frequency components in broader frequency ranges, thereby expanding the range of signal-processing applications.

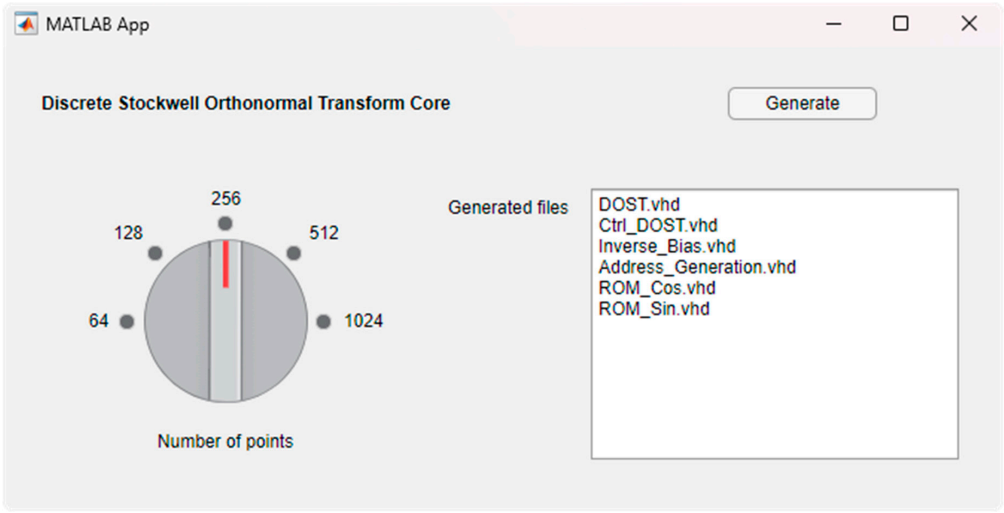


Figure 14. Interface for generating VHDL files.

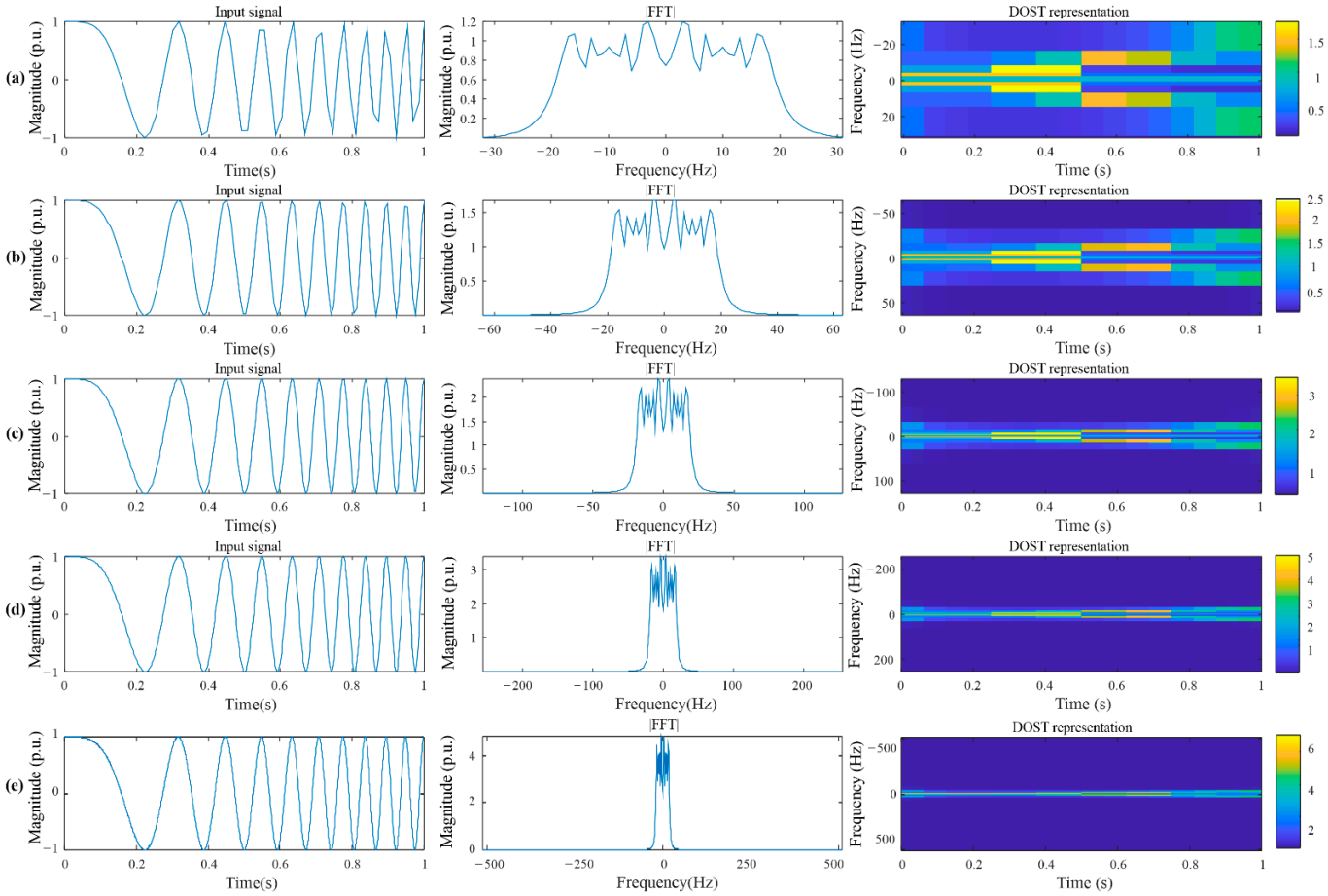


Figure 15. Results for a chirp input signal with (**a**) 64, (**b**) 128, (**c**) 256, (**d**) 512, and (**e**) 1024 points.

4.4. Discussions

The hardware implementation of the proposed DOST algorithm has several advantages. Firstly, the proposed architecture consumes few resources: only 5% DSP blocks and less than 1% ALM blocks, registers, and RAM blocks for the 64-point configuration. This low resource usage is because the proposed DOST architecture maintains practically the same structure regardless of the number of points used for calculating the DOST coefficients. However, the biggest change lies in the increased use of RAM blocks due to the

LUT, where the sine and cosine coefficients are stored to obtain the twiddle factor used in the FFT calculation within the DOST algorithm. On the other hand, the RMSE obtained from 6.0155×10^{-3} in the calculation of the DOST coefficients, due to the use of fixed-point format, does not affect the TFRs obtained.

Moreover, the proposed hardware architecture is easy to reconfigure, i.e., no major changes are required to modify the number of points, which facilitates the creation of a metaprogram to generate HDL code automatically. On the downside, the circuit has considerable latency compared to a fully combinational design; however, a combinational proposal would significantly affect resource usage and the maximum frequency at which the circuit can operate. Other works also propose to calculate the rotation factor using the CORDIC algorithm, which can be implemented as a sequential or combinational circuit. Sequential implementation would affect the total latency for each twiddle factor calculated, while a fully combinational CORDIC design would be resource-intensive.

5. Conclusions

Time–frequency methods are imperative for analyzing non-stationary signals, with swift and proficient computation being key. Given the growing significance of the DOST as a TFR in research, this work proposes an efficient FPGA architecture for its implementation. The proposal can automatically generate VHDL code for the implementation of 64-to-1024-point DOST with 15-bit fixed-point precision. The results indicate that the proposed DOST implementation features reduced power consumption without compromising the latency and the maximum clock frequency. Moreover, simulations demonstrated that the RMSE is constrained to 6.0155×10^{-3} when compared to the one implemented in a 64-bit floating-point processor. In addition, the proposed interface allows the user to select between 64, 128, 256, and 1024 points for DOST implementation. The more points of the DOST are required, the more RAM blocks are used without significantly affecting the FPGA logic elements block usage. This feature could allow designers to integrate more dedicated cores in combination with the proposed DOST core for efficient time–frequency analysis. Finally, the proposed DOST core has been integrated into a hybrid ARM-HPS control unit, which allows the control of different peripherals, such as communication protocols and VGA-based displays, thus facilitating the transfer and visualization of results.

In future research, the application of the proposed method in diagnosing faults in electrical machines will be pursued by analyzing current and vibration signals. Thanks to the reconfigurability of the proposed FPGA-ARM-based DOST hardware implementation, the development of online condition monitoring systems is feasible.

Author Contributions: Conceptualization, D.C.-M.; methodology, M.V.-R., J.-L.C.-H., J.R.R.-G., J.P.A.-S. and D.C.-M.; software and validation, M.V.-R., D.G.-L. and D.C.-M., writing—original draft preparation, J.-L.C.-H., J.R.R.-G. and J.P.A.-S.; writing—review and editing, M.V.-R., D.G.-L. and D.C.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are not publicly available because they are part of an ongoing study.

Acknowledgments: The authors would like to thank the "Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT)—México" and the "Sistema Nacional de Investigadoras e Investigadores (SNII)–CONAHCYT–México" for their support in this research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Boashash, B. *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*; Academic Press: Cambridge, MA, USA, 2015.
2. Yang, Y.; Peng, Z.; Zhang, W.; Meng, G. Parameterised time-frequency analysis methods and their engineering applications: A review of recent advances. *Mech. Syst. Signal Process.* **2019**, *119*, 182–221. [CrossRef]
3. Wacker, M.; Witte, H. Time-frequency techniques in biomedical signal analysis. *Methods Inf. Med.* **2013**, *52*, 279–296. [PubMed]

4. Feng, Z.; Liang, M.; Chu, F. Recent advances in time–frequency analysis methods for machinery fault diagnosis: A review with application examples. *Mech. Syst. Signal Process.* **2013**, *38*, 165–205. [CrossRef]
5. Rivera-Guillen, J.R.; De Santiago-Perez, J.J.; Amezquita-Sanchez, J.P.; Valtierra-Rodriguez, M.; Romero-Troncoso, R.J. Enhanced FFT-based method for incipient broken rotor bar detection in induction motors during the startup transient. *Measurement* **2018**, *124*, 277–285. [CrossRef]
6. Brown, R.A.; Lauzon, M.L.; Frayne, R. A general description of linear time-frequency transforms and formulation of a fast, invertible transform that samples the continuous s-transform spectrum nonredundantly. *IEEE Trans. Signal Process.* **2010**, *58*, 281–290. [CrossRef]
7. Stockwell, R.G. Why use the S-transform? In *Pseudo-Differential Operators: Partial Differential Equations and Time–Frequency Analysis*; American Mathematical Society: Providence, RI, USA, 2007; pp. 279–309.
8. Hutníková, M. On the range of Stockwell transforms. *Appl. Math. Comput.* **2013**, *219*, 8904–8909. [CrossRef]
9. Stockwell, R.G. A basis for efficient representation of the S-transform. *Digit. Signal Process.* **2007**, *17*, 371–393. [CrossRef]
10. Wang, Y.; Orchard, J. Fast discrete orthonormal Stockwell transform. *SIAM J. Sci. Comput.* **2009**, *31*, 4000–4012. [CrossRef]
11. Battistia, U.; Ribaa, L. Window-dependent bases for efficient representations of the Stockwell transform. *Appl. Comput. Harmon. Anal.* **2016**, *40*, 292–320. [CrossRef]
12. Wang, Y.; Orchard, J. Symmetric discrete orthonormal Stockwell transform. *Numer. Anal. Appl. Math.* **2008**, *1048*, 585–588.
13. Raj, S.; Ray, K.C.; Shankar, O. Cardiac arrhythmia beat classification using DOST and PSO tuned SVM. *Comput. Methods Programs Biomed.* **2016**, *136*, 163–177. [CrossRef] [PubMed]
14. Hasan, M.J.; Kim, J.M. Bearing fault diagnosis under variable rotational speeds using stockwell transform-based vibration imaging and transfer learning. *Appl. Sci.* **2018**, *8*, 2357. [CrossRef]
15. Abubakar, M.; Nagra, A.A.; Faheem, M.; Mudassar, M.; Sohail, M. High-Precision Identification of Power Quality Disturbances Based on Discrete Orthogonal S-Transforms and Compressed Neural Network Methods. *IEEE Access* **2023**, *11*, 85571–85588. [CrossRef]
16. Ladan, J.; Vrscay, E.R. *The Discrete Orthonormal Stockwell Transform and Variations, with Applications to Image Compression*; Springer: Berlin/Heidelberg, Germany, 2013.
17. Boutros, A.; Betz, V. FPGA architecture: Principles and progression. *IEEE Circuits Syst. Mag.* **2021**, *21*, 4–29. [CrossRef]
18. Bahoura, M.; Ezzaidi, H. FPGA implementation of a feature extraction technique based on Fourier transform. In Proceedings of the 2012 24th International Conference on Microelectronics (ICM), Algiers, Algeria, 16–20 December 2012; pp. 1–4. [CrossRef]
19. Khatua, P.; Ray, K.C. VLSI Architecture of DCT-Based Harmonic Wavelet Transform for Time–Frequency Analysis. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 6502108. [CrossRef]
20. Krishna, B.M.; Krishna, B.T.; Babulu, K. Linear and quadratic time frequency transforms on FPGA using folding technique. *Microprocess. Microsyst.* **2021**, *80*, 103635. [CrossRef]
21. Dai, Y.-J.; Bi, Z. CORDIC algorithm based on FPGA. *J. Shanghai Univ. (Engl. Ed.)* **2011**, *15*, 304–309. [CrossRef]
22. Abubeker, K.M.; Backer, S.; Varghese, A.M. Serial and Parallel Implementation of CORDIC Architecture: A Comparative Approach. In Proceedings of the 2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communication and Renewable Energy, (ICMiCR-2013), Kanjirapally, India, 4–6 June 2013.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.