electronics

MDPI

Article

A Pipelined FFT Processor Using an Optimal Hybrid Rotation Scheme for Complex Multiplication: Design, FPGA Implementation and Analysis

Hung Ngoc Nguyen [1], Sheraz Ali Khan [1], Cheol-Hong Kim [2] and Jong-Myon Kim [1,*]

[1] School of Computer Engineering and Information Technology, University of Ulsan, Ulsan 680-749, Korea; hungnguyenvldt@gmail.com (H.N.N.); sherazalik@gmail.com (S.A.K.)

[2] School of Electronics and Computer Engineering, Chonnam National University, Gwangju 501-757, Korea; chkim22@chonnam.ac.kr

* Correspondence: jmkim07@ulsan.ac.kr; Tel.: +82-52-259-2217

Received: 7 July 2018; Accepted: 1 August 2018; Published: 2 August 2018

Abstract: The fast Fourier transform (FFT) is the most prevalent algorithm for the spectral analysis of acoustic emission signals acquired at ultra-high sampling rates to monitor the condition of rotary machines. The complexity and cost of the associated hardware limit the use of FFT in real-time applications. In this paper, an efficient hardware architecture for FFT implementation is proposed based on the radix-2 decimation in frequency algorithm (R2DIF) and a feedback pipelined technique (FB) that allows effective sharing of storage between the input and output data at each stage of the FFT process via shift registers. The proposed design uses an optimal hybrid rotation scheme by combining the modified coordinate rotation digital computer (m-CORDIC) algorithm and a binary encoding technique based on canonical signed digit (CSD) for replacing the complex multipliers in FFT. The m-CORDIC algorithm, with an adaptive iterative monitoring process, improves the convergence of computation, whereas the CSD algorithm optimizes the multiplication of constants using a simple shift-add method. Therefore, the proposed design does not require the large memory typically entailed by existing designs to carry out twiddle factor multiplication in large-point FFT implementations, thereby reducing its area on the chip. Moreover, the proposed pipelined FFT processor uses only distributed logic resources and does not require expensive dedicated functional blocks. Experimental results show that the proposed design outperforms existing state-of-the-art approaches in speed by about 49% and in resource utilization by around 51%, while delivering the same accuracy and utilizing less chip area.

Keywords: radix-2 decimation in frequency; fast Fourier transform; feedback; pipelined; modified coordinate rotation digital computer; field programmable gate arrays

1. Introduction

State-of-the-art methods for the intelligent maintenance of rotary machines rely on the timely and accurate analysis of condition monitoring signals, such as acoustic emissions (AE) [1–4] and vibration acceleration signals [5,6]. AE signals are sampled at very high frequencies, typically 1 MHz, to capture ultrasonic sounds released during the initiation and propagation of cracks in machine components. Maintenance decisions are made by detecting those faults through spectral analysis of the envelope signal. The envelope signal is obtained by demodulating the raw AE signal using the Hilbert transform, and its spectral analysis is carried out using the fast Fourier transform (FFT) algorithm. Spectral analysis of the AE envelope signals reveals useful information about underlying bearing faults that is of significant importance in the field of machinery fault diagnosis.

The FFT is one of the most popular transform algorithms in digital applications, used to calculate the discrete Fourier transform (DFT), quickly and accurately. Ever since its introduction, the computational advantages of the FFT have made it an essential algorithm with widespread applications in science and engineering, such as communication, signal processing, image processing, bio-robotics, and intelligent maintenance [1,2,4,7–10]. The high-speed requirements of smart maintenance systems, such as fault diagnosis in rotary machines using the spectral analysis of AE signals, necessitate a high-performance FFT processor. Thus, design of an efficient FFT processor is of great significance in meeting the requirements of real-time applications in terms of speed, accuracy, low cost, and a smaller chip area. The FFT algorithm is mainly implemented on field-programmable gate arrays (FPGAs) [10–12], which offer advantages such as higher performance, less design time, and lower costs than digital signal processor–based systems (DSPs) [13–15]. Moreover, the increasing gate density of FPGAs in recent years has enabled designers to implement data-parallel signal processing algorithms by using massively parallel architectures that can meet high-speed processing requirements; this has resulted in implementations on FPGA that deliver outstanding performance in many applications.

With pervasive applications in signal processing, FFT is the most widely used block in DSP systems and often occupies most of the chip area in hardware implementations. Conventional FFT processors are usually implemented using the radix-2 decimation in frequency (R2DIF) algorithm, which reduces the complexity of computing DFT from $O(N^2)$ to $O(Nlog_2N)$ [16–18]. To increase a system's speed and throughput performance, and make it suitable for real-time signal processing, a pipelined architecture has commonly been applied to the hardware implementation of FFT processors [19–21]. However, the increasing complexity and cost of existing designs inhibits their use for large-point FFT architecture. Likewise, the butterfly operation uses complex adders and multipliers for twiddle factors (TF) in the FFT processor. These complex multipliers are the primary speed bottlenecks in the architecture. In addition, TF values are usually pre-computed and stored in memory. Therefore, a large-point FFT implementation requires a large amount of memory, making the design even more complex and hence, costly [18,22]. Coordinate rotation for digital computers (CORDIC) is an effective method for overcoming the memory problem of FFT computation [23]; it can significantly reduce the resources required to implement TF multiplication. In [24], a memoryless architecture for FFT computation was presented using CORDIC to minimize the memory requirements, but its hardware structure is quite complex. Another CORDIC-based FFT processor was proposed in [25], with a reduced memory footprint and relatively low power consumption. However, it was primarily a memory-based design and offered comparatively slower processing speed due to the high latency in its architecture.

In this paper, an efficient FFT processor is implemented using a feedback pipelined technique (FB), which yields a relatively simple architecture that uses fewer resources and occupies a comparatively smaller area on the chip than existing systems, while delivering high-speed performance. The most important feature of the FB architecture is a feedback loop that allows the outputs of the butterfly units to be fed back to the same memory that is used to store the inputs. This storage sharing at each stage of the FFT processor reduces the hardware complexity. This proposed R2DIF, FB pipelined-based FFT processor (R2FB) employs an optimal hybrid rotation scheme based on a modified CORDIC algorithm (m-CORDIC) and a binary encoding technique based on a canonical signed digit (CSD) to entirely replace the complex multipliers, reducing the memory required to store the TFs and improving system speed and resource utilization. The proposed m-CORDIC algorithm uses a controllable iterative mechanism to enhance the system response time, which is an efficient way to realize the butterfly operation without requiring any dedicated complex multipliers. As an alternative approach for storing the complex TF values, the proposed m-CORDIC and CSD–based hybrid rotation scheme stores only the real TF angles for the butterfly operations, making the hardware design very flexible and efficient by requiring only adders and shifters in FFT computation, thereby saving resources and chip area. Furthermore, the proposed R2FB pipelined FFT processor is implemented using only the distributed

logic resources in the architecture, thereby saving significant resources by eliminating the dedicated functional blocks that are widely used by existing implementations. The proposed R2FB pipelined FFT processor is synthesized in its entirety on the Xilinx Virtex-7 FPGA kit for testing and evaluation.

The main contributions of this paper are listed as follows:

- The design of an efficient FFT processor, in terms of speed, accuracy, and resource utilization that is suitable for real-time signal processing and requires less area on a chip, remains a challenging problem. The proposed R2FB design exploits the FB technique and reduces the memory required at each stage of the pipeline by sharing registers between the input and output at each stage, thereby improving both the utilization of hardware resources and execution time. The proposed pipelined architecture offers a more flexible and less complex hardware implementation of FFT than current designs.
- The m-CORDIC and CSD–based optimal hybrid rotation scheme is proposed to replace the complex TF multipliers in FFT, which reduces the memory requirements, optimizes the area on the chip, and improves the processing speed. The proposed m-CORDIC uses a conditional iterative mechanism via a predefined threshold to determine the number of effective iterations, which improves the convergence of the algorithm. The proposed optimal hybrid rotation scheme uses the shift-add method and does not require the dedicated functional blocks in the architecture, which are available in limited numbers on a target FPGA chip, while still guaranteeing high performance in terms of speed and precision. With these novel improvements, the proposed design uses only the distributed logic resources, yielding a highly efficient FFT processor, as demonstrated by the experimental results.
- Furthermore, the proposed pipelined design uses shift registers replacing the slower memory blocks to store both the input data and the outputs. This memory-sharing mechanism improves the speed of the FFT processor and saves more registers and memory in the architecture. The data paths for the proposed architecture are designed as signed fixed-points and use the number of variable fractional bits at each computational stage in a pipelined architecture to enhance the precision of the final results.

The remainder of this paper is organized into the following sections. An overview of the R2DIF algorithm for FFT computation is given in Section 2. Section 3 briefly describes the m-CORDIC and CSD–based optimal hybrid rotation scheme and its hardware implementation on FPGA. The hardware implementation of the proposed R2FB pipelined processor for the FFT computation is presented in Section 4, and Section 5 analyzes the experimental results and compares them with those of existing designs. Finally, this work is concluded in Section 6.

2. The Radix-2 Decimation in Frequency (R2DIF) Algorithm for Fast Fourier Transform

The N-point DFT transforms an input signal $x(n)$ into its equivalent representation in frequency domain $\mathrm{X}(k)$ using the following relation:

$$\mathrm{X}(k) = \sum_{n=0}^{\mathrm{N}-1} x(n)\mathrm{W}_{\mathrm{N}}^{kn}; \qquad 0 \leq k \leq \mathrm{N}-1 \tag{1}$$

where $\mathrm{W}_{\mathrm{N}}^{kn}$ is the twiddle factor (TF) for rotation, which is given as follows:

$$\mathrm{W}_{\mathrm{N}}^{kn} = \mathrm{e}^{(-\mathrm{j}2\pi kn/\mathrm{N})} = \cos(\frac{2\pi kn}{\mathrm{N}}) - \mathrm{jsin}(\frac{2\pi kn}{\mathrm{N}}) \tag{2}$$

The FFT is mostly used to compute the DFT quickly by reducing the number of operations from $\mathrm{O}(\mathrm{N}^2)$ to $O(\mathrm{N}\log_2 \mathrm{N})$. An efficient radix-2 decimation in frequency (R2DIF) algorithm for FFT is applied to decompose the N-point input $\mathrm{X}(k)$ into even samples $\mathrm{X}(2k)$, and odd samples $\mathrm{X}(2k+1)$, as given in Equations (3) and (4), respectively.

$$\mathrm{X}(2k) = \sum_{n=0}^{\frac{\mathrm{N}}{2}-1} \left(x(n) + x\left(n + \frac{\mathrm{N}}{2}\right) \right) \mathrm{W}_{\frac{\mathrm{N}}{2}}^{kn} \tag{3}$$

$$\mathrm{X}(2k+1) = \sum_{n=0}^{\frac{\mathrm{N}}{2}-1} \left(x(n) - x\left(n + \frac{\mathrm{N}}{2}\right) \right) \mathrm{W}_{\frac{\mathrm{N}}{2}}^{kn} \mathrm{W}_{\mathrm{N}}^{n} \tag{4}$$

Here, $0 \leq k \leq \mathrm{N}/2 - 1$. A radix-2 butterfly unit for FFT computation is shown in Figure 1; it involves an adder and a subtractor followed by a TF multiplier. Because the R2DIF yields the smallest butterfly unit in the architecture, it makes the design space more flexible relative to other algorithms. Calculating N-point FFT requires $\log_2 \mathrm{N}$ stages, and each stage includes $\mathrm{N}/2$ butterfly units, i.e., a sum of $(\mathrm{N}/2)\log_2 \mathrm{N}$ butterfly units for N-point FFT. The signal flow graph of a 16-point FFT using R2DIF algorithm, wshich consists of 32 butterfly operation units in the entire structure, is shown in Figure 2.

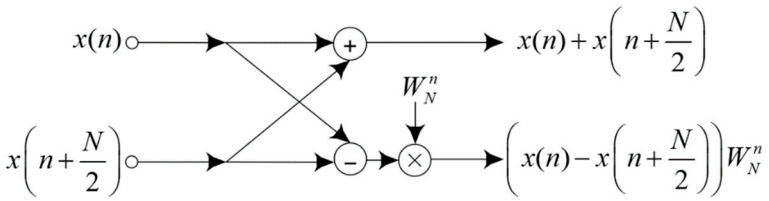

Figure 1. A radix-2 butterfly unit of the radix-2 decimation in frequency (R2DIF) algorithm.

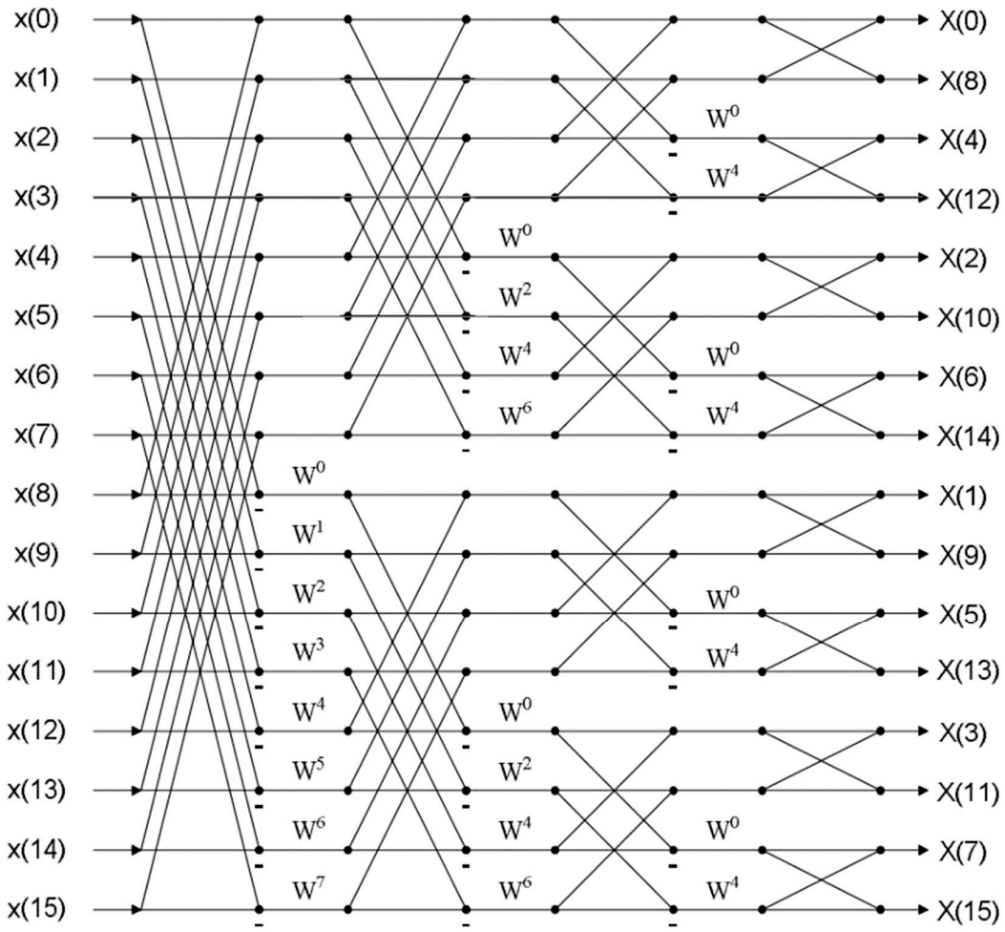

Figure 2. The signal flow graph of a 16-point fast Fourier transform (FFT) on the radix-2 decimation in frequency (R2DIF) algorithm.

3. The Proposed Modified Coordinate Rotation Digital Computer (m-CORDIC) and Canonical Signed Digit (CSD)-Based Rotation Scheme

The CORDIC is an effective alternative approach that can be utilized to compute complex arithmetic functions, including logarithmic, hyperbolic, and trigonometric, only using basic operations such as shifting and adding [23]. It can be applied to multiply the complex twiddle coefficients for butterfly operation units in an FFT processor, without requiring any dedicated multiplier blocks, thereby reducing the area required on the chip and hence its cost and power consumption. The CORDIC also saves memory resources that would otherwise be required to store the complex TF values. These savings in time and memory resources make the proposed design faster and more resource efficient than existing systems. Furthermore, the CSD technique is used to optimize resources of the constant coefficient multipliers on hardware [26]. The proposed m-CORDIC and CSD–based optimal hybrid rotation methodology implemented in this study is explained below.

3.1. CORDIC-Based Complex Multipliers

In this paper, the multiplication of the input data $x(n)$ by the complex TF values $\mathrm{w}_{\mathrm{N}}^{n}$, i.e., $x(n).\mathrm{w}_{\mathrm{N}}^{n}$ is equivalent to the rotation of an input vector by the angle $\theta = {}^{2\pi n}/_{\mathrm{N}}$, such that $\mathrm{w}_{\mathrm{N}}^{n} = \mathrm{e}^{-\mathrm{j}\theta}$, as shown in Figure 3a. Assuming rotation of a vector at the coordinates (x_i, y_i) is performed by an angle θ in the x-y plane to the new coordinates (x_{i+1}, y_{i+1}), its equation is given in matrix form as follows:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{5}$$

Equation (5) can be modified to obtain the following form:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{6}$$

The presence of trigonometric functions makes the implementation of Equation (5) and Equation (6) complex. Therefore, a practical approach is proposed to break the arbitrary angle θ into a series of smaller angles α_i such that $\theta = \sum_{i=0}^{n-1} k_i \alpha_i$, with $k_i = [-1; +1]$ representing the direction of rotation for each iterative step $(i = 0, 1, \ldots, n-1)$, as presented in Figure 3b. By using the arithmetic property of the tangent function, i.e., $\tan\alpha_{\mathrm{i}} = 2^{-i}$, the complexity of the tangent function is reduced into a series of simple shift operations corresponding to each step i. Thus, the desired angular rotation is performed using an iterative series of rotation steps via the smaller angles α_i. The equation for the rotation can be rewritten as follows:

$$\begin{bmatrix} \mathrm{x}_{i+1} \\ \mathrm{y}_{i+1} \end{bmatrix} = \mathrm{m}_i \begin{bmatrix} 1 & -k_i 2^{-i} \\ k_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{7}$$

where m_i is the scale coefficient for each iteration step:

$$\mathrm{m}_i = \cos\theta = \cos(\arctan(2^{-i})) = \frac{1}{\sqrt{1+\tan^2\theta}} = \frac{1}{\sqrt{1+2^{-2i}}} \tag{8}$$

Equation (8) is derived from the trigonometric identities. After n iterative steps, the scaling coefficient M is obtained as follows:

$$\mathrm{M} = \prod_{i=0}^{n-1} \mathrm{m}_i = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1+2^{-2i}}} \tag{9}$$

To compute the R2DIF algorithm, M $\approx$ 0.6072523 when the number of iterations is sufficiently large. The rotating direction k_i is chosen appropriately depending on the accumulated angle z_i. The value of z_i shows the angle difference between the expected rotation and the iterative rotation:

$$k_i = \text{sign}(z_i) = \begin{cases} -1, & z_i < 0 \\ +1, & z_i \geq 0 \end{cases} \tag{10}$$

z_i is determined as follows:

$$z_{i+1} = z_i - k_i \arctan\left(2^{-i}\right) \tag{11}$$

In a more general form, the rotation equations of the CORDIC for computing complex numbers are given in Equation (12), after dropping the scale coefficient M for hardware simplicity. Because this scale factor contains only magnitude information and is independent of the rotation angle, the final scaling coefficient M reaches a certain constant value after n iterative steps. The magnitude of the final output is thus scaled by M at the end, instead of scaling it for each rotation step.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -k_i 2^{-i} \\ k_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{12}$$

In addition, $z_{i+1} = z_i - k_i \alpha_i$, with $\alpha_i = \arctan\left(2^{-i}\right)$.

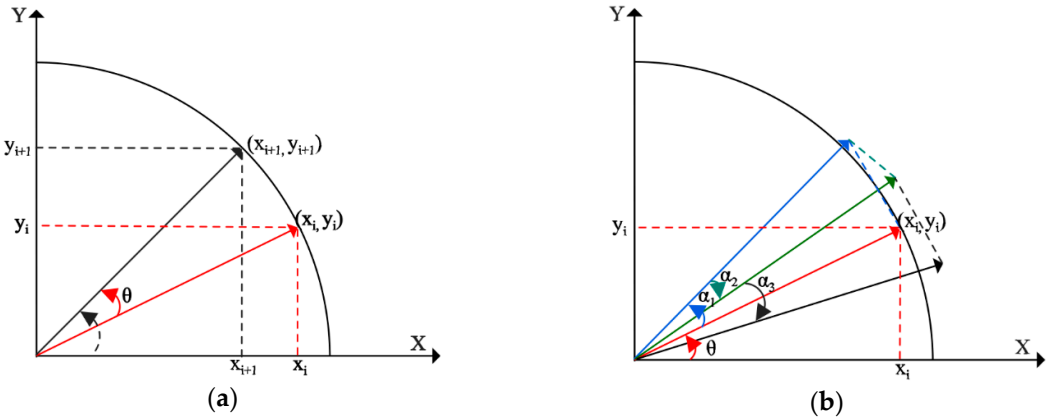

Figure 3. Rotation of a vector at the coordinates (x_i, y_i) in the x-y plane.

Instead of performing the rotation directly through an angle θ, CORDIC performs it using a number of rotations via the smaller angles α_i Assuming that we would like to perform $n = 20$ iterations with number of angle constants α_i, as presented in Table 1, the CORDIC algorithm exploits all these angle values and their direction k_i to compute the desired result, as described in Equation (7). It means that the rotating angle $\theta = \frac{\pi}{6}$ is computed in 20 iterations: i.e., $\theta = \frac{\pi}{6} = \alpha_0 - \alpha_1 + \alpha_2 - \alpha_3 + \alpha_4 + \alpha_5 - \alpha_6 + \alpha_7 - \alpha_8 - \alpha_9 + \alpha_{10} + \alpha_{11} - \alpha_{14} - \alpha_{15} - \alpha_{16} + \alpha_{17} + \alpha_{18} - \alpha_{19}$. The rotations around the desired angle corresponding to clockwise or anticlockwise via the sequential values of the α_i angle are efficiently computed using only shift and add operations.

Table 1. The pre-computed α_i angle values set for rotation.

i	$\tan\alpha_i = 2^{-i}$	$\alpha_i = \arctan(2^{-i})$ (Degree)	α_i (Radian)
0	1.000000000	45.000000000	0.785398163
1	0.500000000	26.565051177	0.463647609
2	0.250000000	14.036243468	0.244978663
3	0.125000000	7.125016349	0.124354995
4	0.062500000	3.576334375	0.062418810
5	0.031250000	1.789910608	0.031239833
6	0.015625000	0.895173710	0.015623729
7	0.007812500	0.447614171	0.007812341
8	0.003906250	0.223810500	0.003906230
9	0.001953125	0.111905677	0.001953123
10	0.000976563	0.055952892	0.000976562
11	0.000488281	0.027976453	0.000488281
12	0.000244141	0.013988227	0.000244141
13	0.000122070	0.006994114	0.000122070
14	0.000061035	0.003497057	0.000061035
15	0.000030518	0.001748528	0.000030518
16	0.000015259	0.000874264	0.000015259
17	0.000007629	0.000437132	0.000007629
18	0.000003815	0.000218566	0.000003815
19	0.000001907	0.000109283	0.000001907

3.2. Implementation of the Pipelined-Based Modified CORDIC Algorithm

There is no general consensus on the number of iterations that the CORDIC algorithm requires for convergence. Thus, the conventional CORDIC usually has a slow response time because of the indeterminate nature of the number of iterations that it requires to converge. This paper proposes a modified CORDIC algorithm (m-CORDIC) in which the iterations are investigated automatically to improve its convergence and save computational time. The proposed m-CORDIC algorithm overcomes the disadvantage of the conventional algorithm using an integrated comparator block to compare the angle accumulator z_i to a hard threshold value T after each iteration i such that z_i converges to zero. In this paper, the threshold value is set appropriately at a very small variance of the rotation angle constants around the desired angle, which ensures high accuracy of the computation. The iteration stops when z_i approaches a value smaller than the predefined threshold. For the rotation angle values in Table 1, the threshold value is chosen to achieve the desired convergence, as illustrated in Figure 4. The proposed algorithm is given below. When the input angle is $\pi/6$, as mentioned above, the conventional CORDIC requires almost 20 iterations to achieve the desired angle. The proposed method takes only nine iterations to converge to a very small residual value of 7.16×10^{-4}.

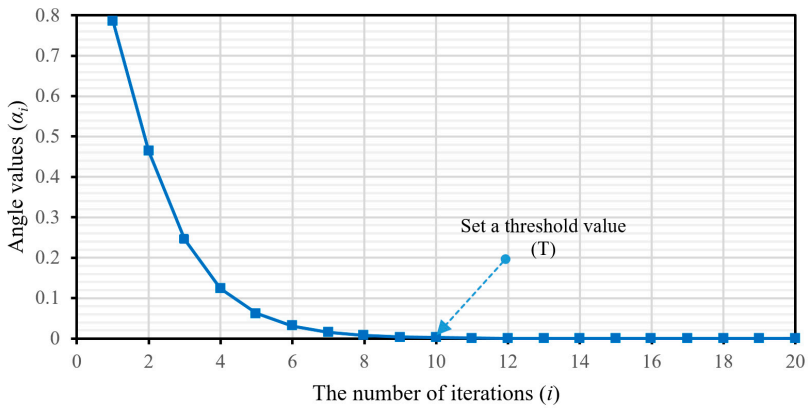

Figure 4. Evaluating the change of rotation angle by the number of iterations.

Algorithm 1 The m-CORDIC using the controllable iterative mechanism

Start with the input values: x_{in}, y_{in}, z_{in}
With *i from 0 to* $n-1$, calculation: $\alpha_i = arctan(2^{-i})$
Definition of the initial z_0 angle accumulator
Select the threshold value: $T = \alpha_{10}$
The initial iteration: $i = 0$ *and* $i \leq n-1$
Consideration of the conditions:
while $z_i > T$ *and select* $= 1$ **do**
update the data values
if $z_i \geq \alpha_i$
$x_{i+1} = x_i - y_i \cdot 2^{-i}$
$y_{i+1} = y_i + x_i \cdot 2^{-i}$
$z_{i+1} = z_i - \alpha_i$
else
$x_{i+1} = x_i + y_i \cdot 2^{-i}$
$y_{i+1} = y_i - x_i \cdot 2^{-i}$
$z_{i+1} = z_i + \alpha_i$
end if *condition*
$i = i + 1$
end while *condition*

Hardware implementation of the m-CORDIC is performed using the pipelined technique, as presented in Figure 5, where each pipelined stage mainly includes shifters, adders, and comparators for conditional multiplication; whereas, the registers for storing data are located between stages. The iterative operations in Equation (12) are performed in parallel in an unrolled manner such that each processing element in the architecture always executes in the same iteration. This iterative structure is easily pipelined by inserting registers between different stages to store the intermediate data for further computation. Therefore, the m-CORDIC computes directly using the input data, which allows for a substantial reduction in buffer memory blocks and the complete removal of the complex multipliers in FFT rotation. The controller correctly determines the amount of shift and the kind of calculation to execute at each clock cycle. The rotated directions of the original vector for the next iteration are determined from the signed bit of the previous accumulated angle z_i. Through these rotated directions, the types of arithmetic operations $\left(\frac{+}{-}\right)$ are defined at each pipelined stage in the architecture. The initial angle $z_0 = z_{in}$ equals the desired angle value; after n iterations are completed using the aforementioned approach, the achieved angle z_{n-1} is equal to or smaller than the specified threshold, which is also the convergence condition of the algorithm. The computational precision of the algorithm is accumulated via each stage in the architecture, such that a pipelined stage corresponds to an iterative step of the algorithm.

For a sufficient number of iterations n, the accumulated scaling coefficient is M $\approx$ 0.6072523. The final results need to be multiplied by this scaling coefficient. Because M is a constant value, an improvement of its multiplication can be performed by the optimal hybrid rotation scheme for the m-CORDIC architecture using the CSD technique. This scheme further reduces chip area requirements while saving processing time.

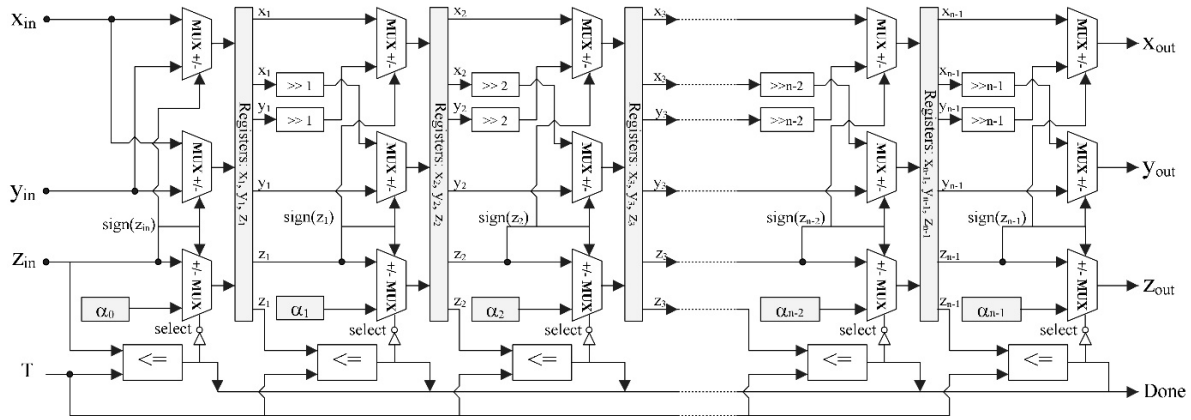

Figure 5. The hardware architecture of the pipelined modified coordinate rotation digital computer (m-CORDIC) algorithm with n-stage.

3.3. CSD Representation in Optimizing the Constant Multipliers

In general, multipliers often take a lot of area on the chip. However, the constant coefficient multipliers can be realized using adders and shifters only. The shifters and adders depend on the number of non-zero bits in the binary constant. In this study, CSD is used as an efficient solution for the binary representation of a constant because it has the least number of non-zero bits that are non-consecutive; therefore, it requires the fewest shifters and adders to determine the product. The CSD is presented in Algorithm 2.

Algorithm 2 The CSD for a constant representation

1: d = count of the number of consecutive "1" bits in a binary series
2: if $d \geq 2$, replacing them by a new sequence $10 \ldots 0\underline{1}$, where 1: +1 and $\underline{1}$: − 1
e.g., $11 \rightarrow 10\underline{1}$; $111 \rightarrow 100\underline{1}$; $1101111 \rightarrow 10\underline{1}10000\underline{1}$
3: continue checking and replacing the value pairs: $1\underline{1} \rightarrow 01$; $\underline{1}1 \rightarrow 0\underline{1}$; $\underline{11} \rightarrow \underline{1}01$
e.g., $10\underline{1}10001 \rightarrow 100\underline{1}0001$

The CSD can significantly reduce the number of shifters and adders required for the multiplication of a constant, thereby reducing its area and power consumption and enhancing its performance. This study uses CSD to optimize the multiplication of the final results by the constant scaling factor M after determining the number of iterations n of the m-CORDIC. The 16-bit conventional binary and CSD-based representations for M are given in Table 2, along with a comparison of resource utilization for the multiplier in each case, as shown in Figure 6. The results show that implementation of constant multipliers using the proposed approach needs about 37.5% fewer hardware resources.

Table 2. Binary representation of the constant multiplier in conventional way and canonical signed digit (CSD).

	Conventional Binary	**CSD**
Constant scaling factor	M = 0.607253	M = 0.607253
Binary expression	0100110110111010	0101001001001010
# of shifters	8	5
# of adders	8	5

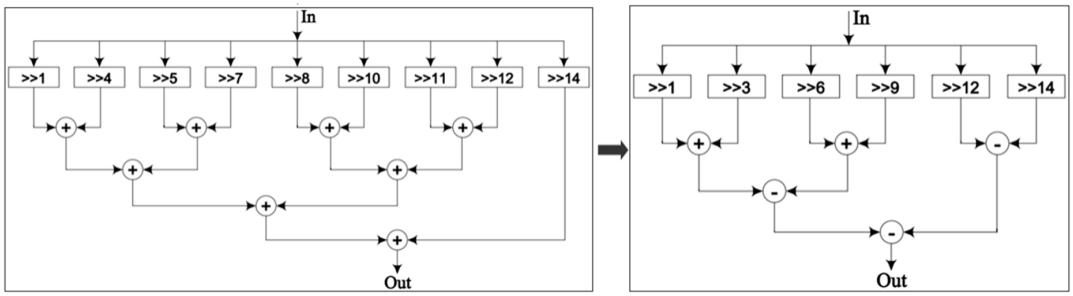

Figure 6. Hardware implementation of the constant multiplier in conventional way and canonical signed digit (CSD).

4. Hardware Implementations of Pipelined Fast Fourier Transform (FFT) Processor

In this paper, an effective FB pipelined design is proposed for the implementation of an FFT processor on hardware with minimal requirement for multipliers and memory. The control logic of the proposed design is also rather simple, making the proposed R2FB pipelined design highly flexible and with low complexity for system implementation on FPGAs. Using the proposed m-CORDIC and CSD–based optimal hybrid rotation method, the performance of the FFT processor is improved in terms of execution time and resource consumption, thus making this design more suitable for applications requiring real-time signal processing.

4.1. Implementation of R2DIF Feedback (FB) Pipelined FFT Architecture

To realize a high performance FFT processor on hardware, an R2DIF butterfly unit is designed based on the FB pipelined technique. Instead of using a large number of memory blocks, leading to lower efficiency of the resultant hardware implementation, the FB pipelined architecture uses only shift registers (data buffers) to store the immediate data at each pipelined stage. The shift registers are designed in a feedback manner using FB to share the storage between input and output data at each stage. A block diagram of the proposed R2FB butterfly unit for FFT computation, shown in Figure 7, implements the complex multipliers using the m-CORDIC and CSD–based optimal hybrid rotation approach instead of the conventional complex multipliers used in typical FFT implementations.

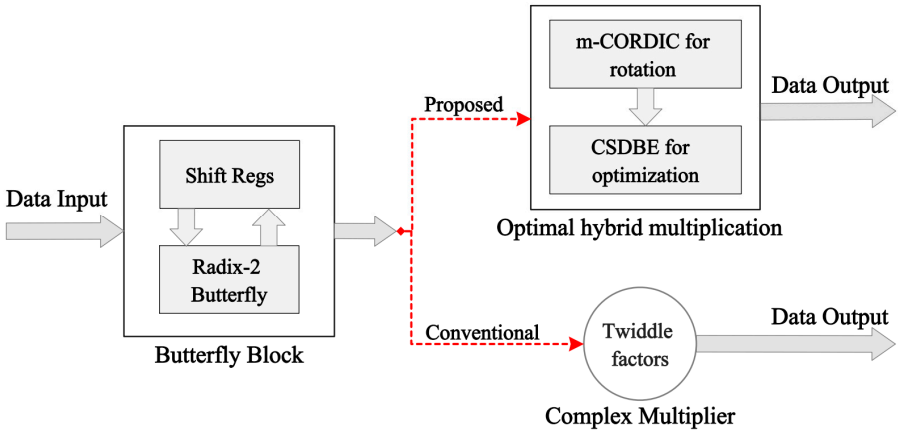

Figure 7. Block diagram of the R2DIF, FB FFT processor (R2FB) pipelined stage with two different implementation approaches.

In a conventional R2FB butterfly unit based on the FB pipelined architecture, as presented in Figure 8, the multiplication of TF values involves calculating four real products, two real sums, and one imaginary sum. On an FPGA, the complex multiplication is done either by using a specialized DSP

block or by using real multipliers and adders. The proposed R2FB design works on a single data stream that is sequentially forwarded and directly computed through the butterfly unit at each stage. The dimensions of the FB shift register banks for storage decreases by half in the architecture when the number of stages increases by one. If N is the number of required FFT points, $\frac{N}{2^s}$ is the size of the data buffer for storage at the s-th pipelined stage of the FFT processor $(s = 1, 2, \ldots, S)$. Each pipelined stage is an effective R2FB module for quickly calculating the DFT using various memory capacities. The pipelined architecture for N points has a similar R2FB module, which is repeated for $S = \log_2 N$ stages. This design is flexible and scalable to any number of FFT points.

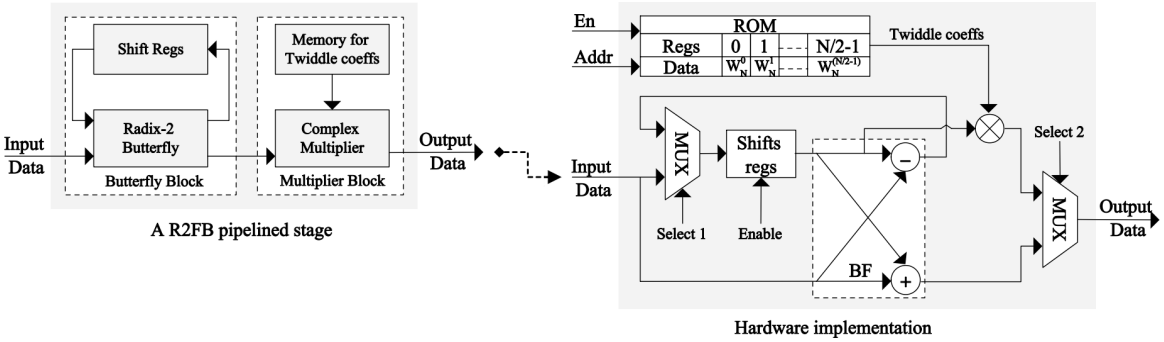

Figure 8. The hardware architecture of a conventional R2FB pipelined stage.

The function of each stage in the R2FB pipeline is illustrated in Figure 8, and we refer to it as the processing element. The processing element contains butterfly units for the addition and subtraction of the input data stream at each stage, a block of the data buffer to store intermediate data, and a complex multiplier for the rotation computation in FFT, as shown in Figure 8. The controller block plays a vital role in generating and scheduling the control signal in accordance with the system. A read-only memory (ROM) is used to store and provide the TF values for rotation, which can be designed using distributed logic resources such as slice registers, configurable logic blocks (CLBs), lookup tables (LUTs), or the available memory blocks. In conventional approaches, implementation of a large-point FFT processor thus requires a large amount of memory, which occupies many hardware resources. In addition, an addressing controller is needed to prevent memory conflicts in data access for the butterfly unit. Thus, designs that require huge memories can be inefficient for the pipelined implementation of FFT on FPGAs.

4.2. The Proposed R2FB Pipelined FFT Processor Using m-CORDIC and CSD-Based Hybrid Rotation Scheme

The FFT processors designed in a conventional way offer poor performance in execution time and resource utilization. To correct this issue, the m-CORDIC algorithm is the most appropriate candidate for implementation of rotation in the FFT computation on an FPGA to achieve efficiency in terms of execution time, accuracy, and resource utilization on the hardware. For the computation of $N = 1024$ points FFT, the proposed R2FB FFT processor would be too large to clearly illustrate its implementation on a FPGA; therefore, the proposed R2FB design for 16-point FFT is given in Figure 9 to highlight its salient features and advantages. It has four stages, each of which consists of a butterfly unit, a CORDIC multiplier, a shift register bank for holding data, a ROM for storing angle values, and a switching logic to control the data streams for computation. The switching module contains mostly multiplexers that select and rearrange the processed data streams on the previous stage in the correct order before transferring them to the next processing stage.

The proposed m-CORDIC, with improvements in controlling the number of effective iterations required for convergence, optimizes the use of hardware resources and the area needed on a chip. Instead of holding the complex TF values, it stores only the real rotation angle values, which reduces the memory requirements and enhances the processing speed. The proposed algorithm is further

improved by using CSD to optimize the constant coefficient multipliers, as presented in Section 3.3. The m-CORDIC and CSD-based optimal hybrid rotation method effectively and completely replaces the conventional multipliers at each processing stage of the $\log_2 N$ pipelined stages in the proposed R2FB design, as shown in Figure 10. The implementation of m-CORDIC in pipelined architecture is discussed particularly in Section 3.2. At each R2FB stage, the multiplier's input (Input_Mul) and the adder's input (Input_Add) to the butterfly unit of the next stage are outputs from the previous stage. The switching arbitrates the data streams between pipelined stages in the architecture. Sharing the feedback shift registers for storage saves significant memory for the system. Moreover, as given in Section 3.3, optimizing the constant multipliers using CSD can reduce the design complexity by 37.5%. Additionally, the data samples consist of only real values at the first stage; thus the m-CORDIC is also optimized effectively for those real values. Furthermore, the rotation of the coefficient $W_N^{N/2} = -j$ in FFT can be computed by merely interchanging the value of the real part and the imaginary part in a product using only the multiplexers and inverters, which saves more resources and further improves the processing speed.

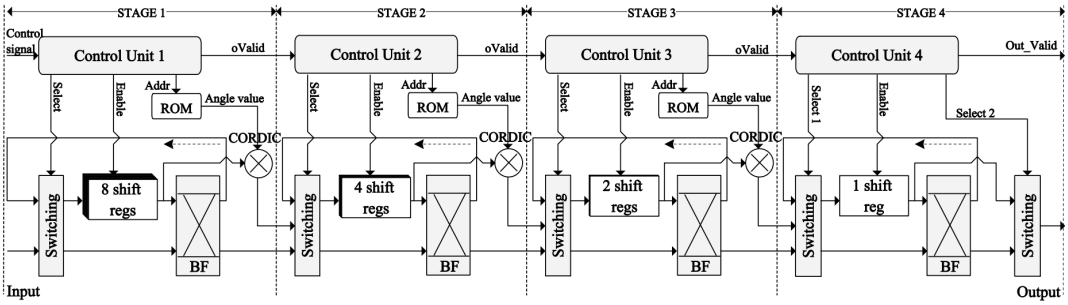

Figure 9. The proposed R2FB pipelined FFT processor for 16 points on field-programmable gate array (FPGA) hardware.

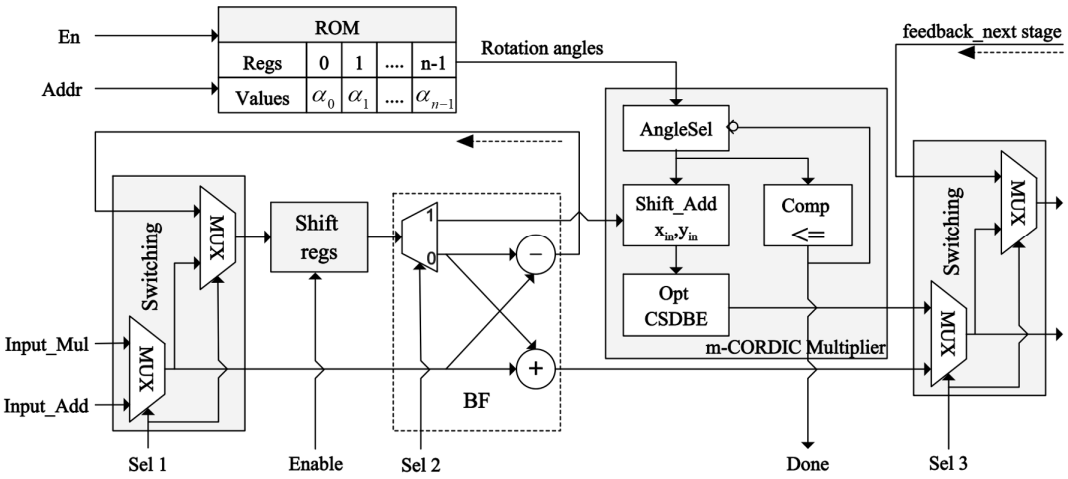

Figure 10. The hardware architecture of a proposed R2FB pipelined stage using m-CORDIC and CSD-based hybrid rotation scheme.

Efficiency of the proposed R2FB design is validated by evaluating its performance and comparing it with that of the conventional architecture, which mainly uses the memory blocks to hold the complex TF values pre-computed for FFT, and a modified design of the conventional architecture that uses distributed logic resources instead of expensive dedicated functional blocks. The efficiency of the aforementioned FFT processors is evaluated while computing FFT for different points or signals

of different lengths in the experimental results. Furthermore, the proposed R2FB processor is also compared with state-of-the-art FFT implementation.

5. Experimental Results

The proposed R2FB pipelined FFT processor and the other aforementioned designs for comparison were implemented on a Virtex 7 XC7VX485T FPGA using the Verilog hardware description language. The functional verification, timing simulation, and synthesis were performed using the specialized Xilinx Vivado Design Suite tool. The data paths are designed in a signed fixed-point format with 16-bit word length and 10-bit precision for the fractional part. The output is formatted dynamically such that the number of fractional bits is variable at each computational stage to improve the system precision.

The conventional R2FB design (A) was implemented using mainly the memory blocks for storing the complex TF coefficients that were pre-computed for FFT. The register slices and LUTs were utilized to compute and store the generated intermediate data during the operation process, whereas the complex multiplications were realized using DSP blocks embedded in the FPGAs. The modified R2FB design (B) was implemented using the distributed logic of CLBs and LUTs for computation instead of a large number of embedded dedicated blocks on the FPGAs. In this case, the slice registers and LUTs effectively replaced the memory blocks, storing both the complex TF coefficients and the intermediate data. In hardware implementations, multiplications by the complex TF values in FFT are commonly processed by the DSP blocks, which usually occupy a significant area space on FPGAs. Thus, a hardware implementation of the complex multipliers must be carefully considered to optimize chip resources and enhance processing speed. For this system (B), using the radix-2 algorithm and the number of FFT points defined for a system to determine the TF values as constants using Equation (2), and those values are then stored as a lookup table. Hence, the FFT rotation becomes a series of complex constant multiplications that can be calculated using the shift-add method instead of embedded DSP blocks. This way, this approach saves 100% of the dedicated functional blocks on FPGAs, but it consumes many of the distributed logic resources. In contrast to those conventional approaches, the proposed R2FB pipelined FFT processor (C) also uses only the slices of distributed logic on a chip without requiring any complex multiplication blocks or memory blocks for storage in architecture. The m-CORDIC and CSD–based optimal hybrid rotation method for this design improves resource utilization, reduces hardware complexity, and lowers costs, resulting in higher efficiency both in terms of resources and speed compared to the other designs. An evaluation of the hardware complexity and performance of the three approaches for a 1024-point FFT processor is provided in Table 3.

Table 3. The achieved hardware results of an FFT implementation with the different approaches.

Methods \ Usage	# of Slice Registers	# of Slice LUTs	# of IOBs	# of BRAMs	# of DSP48Es	# of Clocking Buffers	Total # of Clock Cycles	Execution Time (μS)
(A)	2076	3159	92	4	30	1	2066	10.332
(B)	1611	16,336	92	0	0	1	2024	10.118
(C)	1393	11,865	92	0	0	1	1034	5.168

The proposed R2FB pipelined FFT processor uses the input data directly for computation, enabling it to process faster and at less power than the conventional designs leaning more on memory based architecture, which use the slower and more power intensive load instructions to fetch the TFs. As shown in Table 3, the proposed architecture (C) for computing 1024-point FFT can save the distributed logic resources about 13.53% in slice registers and 27.37% in slice LUTs and improve the processing speed by about 49% compared to the modified approach (B).

In this paper, all three designs used the distributed logic resources of slice registers and LUTs to implement their logic functions, which is a general measure of the area on FPGAs. However, the conventional memory-based designs also use dedicated block random-access memory (BRAM) and 48-bit DSP element (DSP48E) functional blocks. The area of design is usually measured through

the number of slice registers, LUTs, BRAMs, DSP48Es in the architecture. It is essential to estimate the area of all designs precisely using an effective common metric. In this paper, we measured the area required by all designs using the number of slices, the primary element in all FPGAs, as the main metric. In the Virtex-7 family, each DSP48E block consists of an adder, a 25 × 18 multiplier, and an accumulator, whereas each BRAM block is fundamentally 36 Kb in size and can be used for storing data. With those capacity values, we obtained the equivalent area of the DSP and BRAM blocks in terms of slices [27], as given in Table 4. The experimental results for computing 1024-point FFT by the aforementioned designs for FFT processors and a comparison of their efficiencies in area and execution time are shown in Table 4. The proposed R2FB pipelined FFT processor uses about 51% fewer slices than the conventional design and does not require any dedicated functional blocks in the architecture. The m-CORDIC and CSD–based optimal hybrid rotation scheme reduces the necessary chip area and improves the performance of the proposed design.

Table 4. An evaluation the slice area of a 1024-point FFT implementation with the various architectures.

Resources / Methods	# of Slices	DSP48Es		BRAMs		Total # of Slices	Frequency (MHz)	Execution Time (μS)
		# of Blocks	Equivalent Slices	# of Blocks	Equivalent Slices			
(A)	5235	30	15,000	4	6800	27,035	200	10.332
(B)	17,947	0	0	0	0	17,947	200	10.118
(C)	13,258	0	0	0	0	13,258	200	5.168

The precision of the proposed R2FB pipelined FFT processor was evaluated by measuring the average relative percentage error value across 1024 points, as shown in Equation (13). The results obtained via the standard functions of Matlab were used as a baseline for comparison with the results on hardware. The average relative percentage error value of the conventional memory-based design was 0.52%, whereas the proposed design yielded an error of 0.72%. The advantages in performance and savings in chip area and hardware resources of the proposed design outweigh its rather insignificant disadvantage in accuracy. The achieved results show the high precision of the proposed FFT processor: less than a 1% error rate with better design than the systems in [16] (more than 1%) and [17] (3.22%).

$$\frac{\sum_{i=1}^{N} \frac{|A(i) - B(i)|}{B(i)}}{N_{points}} \times 100(\%) \tag{13}$$

A comparison of the results achieved on hardware between the proposed design and existing state-of-the-art designs is given Table 5. In any hardware implementation of an FFT processor, the balance struck between the amount of resources consumed and the processing speed depends on the complexity of the architecture. The larger the number of FFT points, the higher the complexity. The extended results for the FFT processor with a various number of points are shown in Table 6. An evaluation of the area on the chip, in terms of the number of slices, required by each of the aforementioned approaches for FFTs of various lengths on the same FPGAs is presented in Figure 11. The proposed R2FB pipelined design requires the fewest number of slices for all cases of FFT computation in this study, and the resource utilization of the proposed design is also more optimized as the number of FFT points increases greatly. In the case of 4096-point calculation, the proposed R2FB pipelined FFT processor consumes about 59.56% less slices than the conventional design, which uses mainly the dedicated functional blocks in the architecture. The experimental results obviously show that the proposed FFT processor requires the fewest clock cycles, has faster execution time, and requires fewer hardware resources than the other systems for FFT computation. Compared to the other approaches, the proposed design reaches the goal of high speed, accuracy, and efficiency in resource utilization and area, and eliminates the need for a significant number of memory and DSP blocks on the chip.

Table 5. Comparison of the hardware resources needed by the various architectures for an N-point FFT implementation.

Hardware Architecture		Shift Registers	Adders		Multipliers		Overall Storage	# of Clock Cycles
Designs	**Schemes**		**Complex**	**Constant**	**Complex**	**Constant**		
[20]	Feedforward and feedback	$\frac{3}{2}N+\frac{3}{2}(\log_2 N-1)$	$\log_2 N+1$	N/A	$\frac{1}{2}\log_2 N-\frac{1}{2}$	N/A	$2N+\frac{3}{2}(\log_2 N-1)$	$2N+\log_2 N-1$
[28]	Feedforward for radix-2^2	$N-2$	$2\log_2 N$	N/A	$\log_2 N-2$	N/A	$3N-2$	$\frac{3N}{2}-1$
[29]	Pipelined and parallel	$\frac{3N}{2}-2$	$2\log_2 N$	N/A	$\log_2 N-2$	N/A	$\frac{17N}{8}-5$	$\frac{11N}{8}-4$
[30]	Dual-path delay	$\frac{3N}{2}$	$\log_2 N+1$	N/A	$\log_2 N-1$	N/A	$2N$	$2N-1$
Proposed	Feedback and m-CORDIC	$N-1$	$2\log_2 N$	$3\log_2 N$	0	$2\log_2 N-2$	$N+\log_2 N$	$N+\log_2 N-2$

Table 6. Evaluation of the performance of the R2FB pipelined FFT processor with various points.

FFT Points	Hardware Resource Utilization					Execution Time (μS)
	Slice Registers	*Slice LUTs*	*BRAMs*	*DSP48Es*	*Total # of Slices*	
A conventional architecture						
512	1826	2314	3	26	22,240	5.202
1024	2076	3159	4	30	27,035	10.332
2048	2306	4525	6	34	34,031	20.582
4096	2639	7163	14	38	52,602	41.072
A modified architecture only using distributed logic of slice registers and lookup tables (LUTs)						
512	1386	11,920	0	0	13,306	5.091
1024	1611	16,336	0	0	17,947	10.118
2048	1840	19,806	0	0	21,646	20.163
4096	2282	27,229	0	0	29,511	40.243
The proposed pipelined architecture						
512	1226	9352	0	0	10,578	2.602
1024	1393	11,865	0	0	13,258	5.168
2048	1594	14,576	0	0	16,170	10.292
4096	1959	19,312	0	0	21,271	20.537

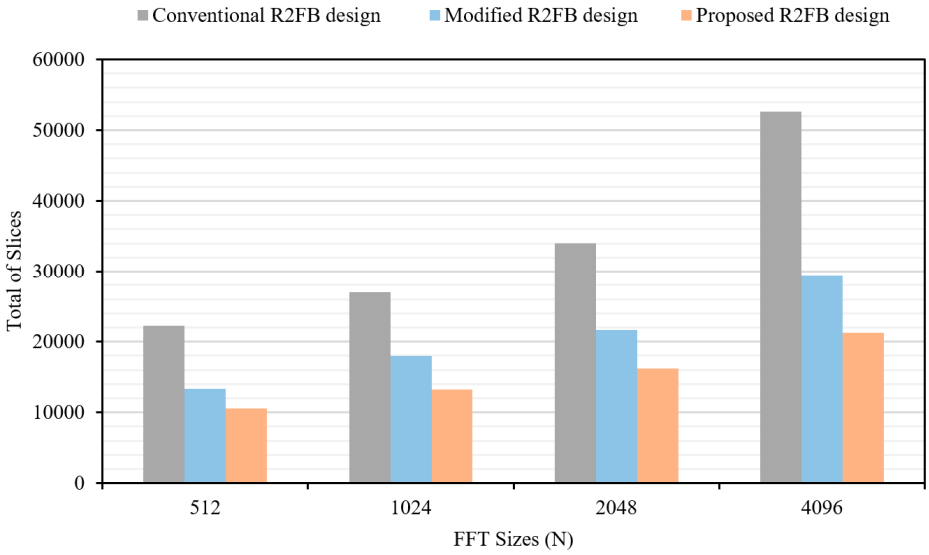

Figure 11. Slice area consumption of different approaches by the number of various FFT points.

6. Conclusions

In this paper, a R2FB pipelined architecture for the computation of FFT was proposed and its implementation on the Xilinx Virtex-7 FPGA kit was presented. The proposed design was verified and compared to existing approaches in terms of speed, the requisite hardware resources, area on the chip occupied by each design, and the hardware complexity. The feedback pipelined technique was successfully exploited for FFT implementation. The sharing of shift registers for storage between the inputs and outputs in a feedback architecture reduces the memory footprint of the proposed architecture and saves essential hardware resources. The m-CORDIC and CSD–based optimal hybrid rotation scheme was proposed to replace complex TF multipliers in the butterfly unit, resulting in faster convergence and fewer resource requirements with low hardware complexity. The improvements of the proposed design resulted in an FFT processor that requires only distributed logic resources on the FPGA instead of expensive dedicated functional blocks on a chip, making the proposed design flexible, fast, simple, low cost, and with reduced area on the chip. The achieved experimental results proved that the proposed R2FB pipelined FFT processor is better than existing FPGA-based designs in terms of speed by around 49% and in terms of resource utilization by about 51%, while delivering the same accuracy and using less area on the chip. The proposed design for the computation of FFT delivers significantly better processing speed and requires fewer hardware resources. However, since this design is based on the feedback pipelined technique and works on a single data-path stream that only allows sequential data processing at a rate of one sample per clock cycle, its computation throughput needs further improvement. To enhance the throughput of this design, multiple data-path, parallel processing approaches will be investigated in our future work.

Author Contributions: All the authors contributed equally to the conception of the idea, the implementation of the processor, setting up simulations, generating experimental results, and the writing and revision of this manuscript.

Funding: This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry and Energy (MOTIE) of the Republic of Korea (No. 20181510102160, No. 20162220100050, No. 20161120100350, and No. 20172510102130). It was also funded in part by the Leading Human Resource Training Program of Regional Neo Industry through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning (NRF-2016H1D5A1910564), in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1D1A3B03931927), and in part by the "Leaders in INdustry-university Cooperation +" Project, supported by the Ministry of Education and National Research Foundation of Korea.

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

References

1. Kang, M.; Kim, J.; Wills, L.M.; Kim, J.-M. Time-Varying and Multiresolution Envelope Analysis and Discriminative Feature Analysis for Bearing Fault Diagnosis. *IEEE Trans. Ind. Electron.* **2015**, *62*, 7749–7761. [CrossRef]
2. Islam, R.; Khan, S.A.; Kim, J.-M. Discriminant Feature Distribution Analysis-Based Hybrid Feature Selection for Online Bearing Fault Diagnosis in Induction Motors. *J. Sens.* **2016**, *2016*, 1–16. [CrossRef]
3. Tra, V.; Kim, J.; Khan, S.A.; Kim, J.-M. Incipient fault diagnosis in bearings under variable speed conditions using multiresolution analysis and a weighted committee machine. *J. Acoust. Soc. Am.* **2017**, *142*, EL35–EL41. [CrossRef] [PubMed]
4. Nguyen, N.H.; Kim, J.; Kim, J.-M. Optimal Sub-Band Analysis Based on the Envelope Power Spectrum for Effective Fault Detection in Bearing under Variable, Low Speeds. *Sensors* **2018**, *18*, 1389. [CrossRef] [PubMed]
5. Khan, S.A.; Kim, J.-M. Rotational speed invariant fault diagnosis in bearings using vibration signal imaging and local binary patterns. *J. Acoust. Soc. Am.* **2016**, *139*, EL100–EL104. [CrossRef] [PubMed]
6. Khan, S.A.; Kim, J.-M. Automated Bearing Fault Diagnosis Using 2D Analysis of Vibration Acceleration Signals under Variable Speed Conditions. *Shock Vib.* **2016**, *2016*, 8729572. [CrossRef]
7. Zhang, W.; Su, T. Reference Beam Pattern Design for Frequency Invariant Beamforming Based on Fast Fourier Transform. *Sensors* **2016**, *16*, 1554. [CrossRef] [PubMed]
8. Ganjikunta, G.K.; Sahoo, S.K. An area-efficient and low-power 64-point pipeline Fast Fourier Transform for OFDM applications. *Integr. VLSI J.* **2017**, *57*, 125–131. [CrossRef]
9. Sundararajan, M.; Govindaswamy, U. Multicarrier Spread Spectrum Modulation Schemes and Efficient FFT Algorithms for Cognitive Radio Systems. *Electronics* **2014**, *3*, 419–443. [CrossRef]
10. Sanchez, M.A.; Garrido, M.; Lopez-Vallejo, M.; Grajal, J. Implementing FFT-based digital channelized receivers on FPGA platforms. *IEEE Trans. Aerosp. Electron. Syst.* **2008**, *44*, 1567–1585. [CrossRef]
11. Iglesias, V.; Grajal, J.; Sanchez, M.A.; López-Vallejo, M. Implementation of a real-time spectrum analyzer on FPGA platforms. *IEEE Trans. Instrum. Meas.* **2015**, *64*, 338–355. [CrossRef]
12. Nguyen, N.H.; Khan, S.A.; Kim, C.-H.; Kim, J.-M. A high-performance, resource-efficient, reconfigurable parallel-pipelined FFT processor for FPGA platforms. *Microprocess. Microsyst.* **2018**, *60*, 96–106. [CrossRef]
13. Wang, Y.; Tang, Y.; Jiang, Y.; Chung, J.-G.; Song, S.-S.; Lim, M.-S. Novel memory reference reduction methods for FFT implementations on DSP processors. *IEEE Trans. Signal Process.* **2007**, *55*, 2338–2349. [CrossRef]
14. Sun, T.-Y.; Yu, Y.-H. Memory usage reduction method for FFT implementations on DSP based embedded system. In Proceedings of the 2009 IEEE 13th International Symposium on Consumer Electronics, Kyoto, Japan, 25–28 May 2009; pp. 812–815.
15. Pitkänen, T.O.; Takala, J. Low-power application-specific processor for FFT computations. *J. Signal Process. Syst.* **2011**, *63*, 165–176. [CrossRef]
16. Derafshi, Z.H.; Frounchi, J.; Taghipour, H. A high speed FPGA implementation of a 1024-point complex FFT processor. In Proceedings of the Computer and Network Technology, International Conference on(ICCNT), Bangkok, Thailand, 23–25 April 2010; pp. 312–315.
17. Kumar, M.; Selvakumar, A.; Sobha, P. Area and frequency optimized 1024 point Radix-2 FFT processor on FPGA. In Proceedings of the 2015 International Conference on VLSI Systems, Architecture, Technology and Applications, Bangalore, India, 8–10 January 2015; pp. 1–6.
18. Ma, Z.-G.; Yin, X.-B.; Yu, F. A Novel Memory-Based FFT Architecture for Real-Valued Signals Based on a Radix-2 Decimation-In-Frequency Algorithm. *IEEE Trans. Circuits Syst. Express Briefs* **2015**, *62*, 876–880. [CrossRef]
19. Garrido, M.; Parhi, K.K.; Grajal, J. A pipelined FFT architecture for real-valued signals. *IEEE Trans. Circuits Syst. Regul. Pap.* **2009**, *56*, 2634–2643. [CrossRef]
20. Wang, Z.; Liu, X.; He, B.; Yu, F. A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2015**, *23*, 973–977. [CrossRef]
21. Liu, J.; Xing, Q.; Yin, X.; Mao, X.; Yu, F. Pipelined Architecture for a Radix-2 Fast Walsh–Hadamard–Fourier Transform Algorithm. *IEEE Trans. Circuits Syst. Express Briefs* **2015**, *62*, 1083–1087. [CrossRef]
22. Luo, H.-F.; Liu, Y.-J.; Shieh, M.-D. Efficient memory-addressing algorithms for FFT processor design. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2015**, *23*, 2162–2172. [CrossRef]

23. Meher, P.K.; Valls, J.; Juang, T.-B.; Sridharan, K.; Maharatna, K. 50 years of CORDIC: Algorithms, architectures, and applications. *IEEE Trans. Circuits Syst. Regul. Pap.* **2009**, *56*, 1893–1907. [CrossRef]
24. Garrido, M.; Grajal, J. Efficient Memoryless Cordic for FFT Computation. In Proceedings of the 2007 International Conference on Acoustics, Speech and Signal Processing, Honolulu, HI, USA, 15–20 April 2007; pp. 113–116.
25. Oruklu, E.; Xiao, X.; Saniie, J. Reduced memory and low power architectures for CORDIC-based FFT processors. *J. Signal Process. Syst.* **2012**, *66*, 129–134. [CrossRef]
26. Pan, S.-T. A canonic-signed-digit coded genetic algorithm for designing finite impulse response digital filter. *Digital Signal Process.* **2010**, *20*, 314–327. [CrossRef]
27. Nguyen, N.-H.; Khan, S.A.; Kim, C.-H.; Kim, J.-M. An FPGA-Based Implementation of a Pipelined FFT Processor for High-Speed Signal Processing Applications. In Proceedings of the 13th International Symposium on Applied Reconfigurable Computing, Delft, The Netherlands, 3–7 April 2017; pp. 81–89.
28. Garrido, M.; Grajal, J.; Sánchez, M.; Gustafsson, O. Pipelined radix-2^k feedforward FFT architectures. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2013**, *21*, 23–32. [CrossRef]
29. Ayinala, M.; Brown, M.; Parhi, K.K. Pipelined parallel FFT architectures via folding transformation. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2012**, *20*, 1068–1081. [CrossRef]
30. Chang, Y.-N. An efficient VLSI architecture for normal I/O order pipeline FFT design. *IEEE Trans. Circuits Syst. Express Briefs* **2008**, *55*, 1234–1238. [CrossRef]