International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I12P209
Volume 69 Issue 12, 66-73, December, 2021

Implementation of Multiplierless High-Speed Low Power Split Radix SDF FFT using NEDA Algorithm for Speech Enhancement

Mr.C.Ramesh Kumar[1], Dr.M.P.Chitra[2]

[1]*Research Scholar, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu*
[2]*Professor, Department of Electronics and Communication Engineering, Panimalar Institute of Technology, Chennai, Tamil Nadu, India*

[1]*rameshchand2006@gmail.com*

Abstract *- The Split Radix FFT processor is primarily used for low-power FFT processors. It necessitates fewer mathematical operations. FFT, in the traditional sense, demands greater power and area. However, the proposed SDF SRFFT approach uses less power, is faster, and has a shorter delay. In addition, the twiddle factor multiplication in this SRFFT is performed using the New Distributed Arithmetic (NEDA) technique, which lowers the chip area. The SRFFT implementation is designed with the XILINX ISE Tool and implemented on FPGA Spartan6. For various types of acoustic noise, the SNR is improved. Python3 is used to lower voice noise using the suggested SRFFT architecture.*

Keywords — *Fast Fourier Transform, Butter Fly Unit, Processing Element, Single Delay Feedback, New Distributed Arithmetic, Distributed Arithmetic.*

I. INTRODUCTION

In the field of digital signal processing, the FFT is critical; low power, high-speed processors, require OFDM systems, which require 256-to-8192-point DFT. The complexity of FFT was lowered by Cooley and turkey from DFT N^2 to FFT $N/2\log_2 N$. They create algorithms that are both fast and simple [2]. Where N=N1N2 and N1 and N2 are co-prime, a good mapping is utilized to partition the transfer function into different length FFTs [3]. To overcome the nesting of multiple multiplications, Winograd in 1974 created the Winograd Fourier Transform Algorithm [4]. This approach uses less multiplication but is more complex structurally and takes longer [5,6]. After dividing, the prime factor algorithm spits out two transforms: prime length and Multidimensional DFT's Higher Radix Algorithm, both of which are computed using FFTs. In [7,8,9], this approach minimizes the complexity of the Multiplication and Addition operations. The R4 method is less complex and, as a result, quite popular. Split Radix Fast Fourier Transform was devised by P. Dhumel and H. Hallman in 1984 [10]. (SRFFT). This algorithm calculates the even and odd components in Radix-2 and Radix-4, respectively. This demonstrates that the mixed-radix algorithm uses fewer sum and product calculations. This results in simple structural flow graphs, which are described in the sections that follow. The R2/R4/R8 computation is simpler than the SRFFT. The disadvantage of SRFFT is that it is unable to present with all powers of 2n. Each structure necessitates the use of two operations: sum and product. To achieve fast speed, low hardware cost, and efficient power usage, the Multiplication of Phase factor is applied. Booth Multiplier, CORDIC, DA, and CSD are four ways of implementing phase factors [11]. Booth and CORDIC take up more space and cost more money. As a result, when the canonical sign Digit Multiplier is applied in [12], the Area and Cost are likewise increased. To remedy this issue, in 1968, the Zohar [13] was published. It is the inventor of Distributed Arithmetic, which uses a Lookup Table to compute the multiplication (LUT). The majority of mathematical equations in DSP in SOP form can be easily obtained using DA. Abraham created an IIR digital filter based on the DA Algorithm in order to gain a better outcome while utilizing SRFFT with DA to save register word length. The key benefit of this technology is that it uses less space and power while also increasing the speed with which Radix 2 architecture can be performed [14,33,28]. By utilizing the data flow graph of the FFT [15], a novel algorithm introduces in ‘in place'.For various real-time DSP applications, such as DCT DWT, the MAC unit is implemented using Distributed Arithmetic. For storing product terms in a ROM, DA is implemented by MAC to a preset LUT. The inner product of two multidimensional vectors is computed by DA. To avoid rescaling of input data, Joshi implemented the DA Algorithm-based SRFFT in 2013 [14]. This increases the result's dimension while also increasing the memory requirement. To solve this drawback, pick both multiple terms as fixed coefficients. NEDA [16] is the name of this approach. For various real-time DSP applications, such as DCT, DWT, the MAC unit is implemented using Distributed Arithmetic. For storing product terms in a ROM, DA is implemented by MAC to a preset LUT. DA In NEDA, the inputs are replaced with coefficients, resulting in a memory-less DA architecture. The Shift adds algorithm is used in the NEDA computation. To come up with NEDA Mathematically, to show that the two's complement addition approach is superior to the internal product method, a minimal number of moves at the end [17]. DA.

The first section of this article contains a quick introduction to FFT. The Split Radix FFT is reviewed in

Section 2. The suggested NEDA-based SDFSRFFT is then shown in Section 2. The noise improvement utilizing FFT/IFFT with filtering is shown in Section 4. Section 5 contains the results and discussion, as well as the conclusion.

II. DESIGN OF SPLIT RADIX FFT

The signal flow graph for SRFFT is depicted in Figure 1. When N point DFT is decimated, it produces one N/2 point DFT without phase factor and two N/4-point DFTs with a phase factor. Figure 2 depicts the SRFFT Processing Element. The calculation is completed in one step.

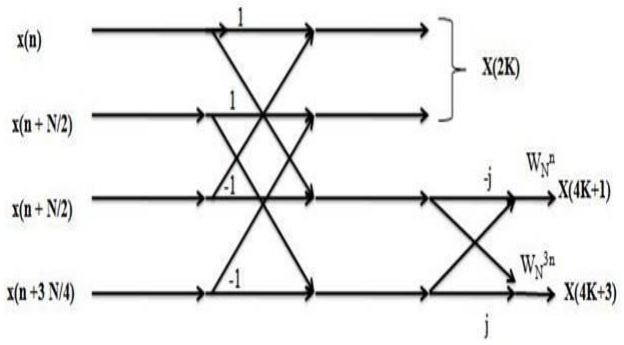


Fig. 1 Basic Block Diagram of Split Radix FFT

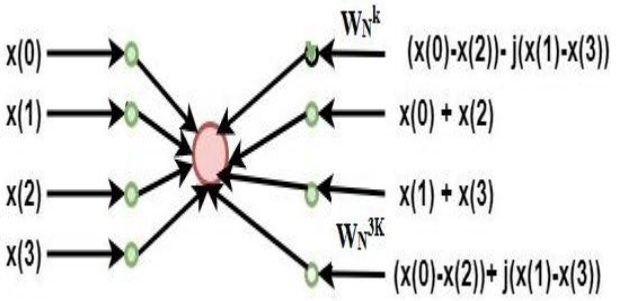


Fig. 2 Flow diagram for SRFFT with acceptable input

The number of complex multiplication and addition using Split Radix SDF FFT is given as

Number of complex Multiplication represented in

$$R_M = N \log_2 N - 3N + 4 \quad (1)$$

Amount of complex Addition represented in

$$R_A = 3N\log_2 N - 3N + 4 \quad (2)$$

The drawback of DFT is it requires more computation. If sample N increases, the amount of product and sum also increase. To overcome this drawback goes for FFT reduces the amount of product and sum is N $\log_2$N and N/2 $\log_2$N.

This section briefly presents the SR FFT and its signal flow diagrams. The Analysis function of DFT is represented as

$$X(K) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad (3)$$

Where K = 0 to N-1

Where W_N^{nk} It is called the twiddle factor or phase factor. The even and odd components of X(K) can be splatted into

$$X(2K) = \sum_{n=0}^{N/2-1} \left(x_n + x_{n+\left(\frac{N}{2}\right)}\right) W_{N/2}^{nk} \quad (4)$$

Where K = 0 to N/2-1

$$X(2K+1) = \sum_{n=0}^{N/2-1} \left(x_n - x_{n+\left(\frac{N}{2}\right)}\right) W_{N/2}^{nk} \quad (5)$$

Further decomposes odd component into X (4K + 1) and X(4K + 3), The even and odd component of split radix FFT expressed as

$$X(2K) = \sum_{n=0}^{N/2-1} \left(x_n + x_{n+\left(\frac{N}{2}\right)}\right) W_{N/2}^{nk} \quad (6)$$

$$X(4K+1) = \sum_{n=0}^{N/4-1} \left(x_n - jx_{n+\left(\frac{N}{4}\right)} - x_{n+\left(\frac{N}{2}\right)} + jx_{n+\left(\frac{3N}{4}\right)}\right) W_N^n W_{N/4}^{nk} \quad (7)$$

Where K =0 to N/4-1

$$X(4K+3) = \sum_{n=0}^{N/4-1} \left(x_n + jx_{n+\left(\frac{N}{4}\right)} - x_{n+\left(\frac{N}{2}\right)} + jx_{n+\left(\frac{3N}{4}\right)}\right) W_N^{3n} W_{N/4}^{nk} \quad (8)$$

Where K =0 to N/4-1

III. ANALYSIS OF NEDA

This approach is mostly used in MAC units for various DSP applications. It is primarily utilized in DCT, FFT, and other similar applications. Implementation of the National Economic Development Act (NEDA), which strengthens the qualities of Area, Power, and Swiftness. The NEDA Expression is calculated as follows:

The internal product can be expressed below

$$Z = \sum_{i=1}^{K} C_i X_i \quad (9)$$

Where C_i the constant in is fixed coefficient and X_i Is the input variable. Equation (9) expressed in the matrix form is given as

$$Z = \begin{bmatrix} C_1 & C_2 & \dots & C_K \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_K \end{bmatrix} \quad (10)$$

To represent both C_i and X_i in 2's complement form

$$C_i = -C_i^M 2^M + \sum_{K=N}^{M-1} C_i^K 2^K \quad (11)$$

Where C_i is either 0 or 1, K = N, N+1,…M. and C_i^M Is the sign bit. Apply eqn no (11) in (10)

$$Z = \begin{bmatrix} -2^0 & 2^{-1} & \dots & 2^{-12} \end{bmatrix} \begin{bmatrix} C_1^0 & \dots & C_K^0 \\ \dots & \dots & \vdots \\ C_1^{12} & \dots & C_K^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_K \end{bmatrix} \quad (12)$$

The matrix containing C_i^K is the sparse matrix either 0 or 1. eqn (9) expressed as

$$Z=[-2^0 2^{-1} \dots 2^{-12}] \begin{bmatrix} W_0 \\ \vdots \\ W_{12} \end{bmatrix} \quad (13)$$

$$\text{Where} \begin{bmatrix} W_0 \\ \vdots \\ W_{12} \end{bmatrix} = \begin{bmatrix} C_1^0 & \dots & C_K^0 \\ \dots & \dots & \vdots \\ C_1^{12} & \dots & C_K^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_K \end{bmatrix} \quad (14)$$

NEDA operation is discussed below

$$Y=\left[COS\ \frac{\pi}{8}\ \ COS\ \frac{\pi}{4}\right] \begin{bmatrix} W_0 \\ \vdots \\ W_{12} \end{bmatrix} \quad (14)$$

$$Y =[-2^0 2^{-1} \dots 2^{-12}] \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (15)$$

Equation (14) may be written as

$$Y=[-2^0 2^{-1} \dots 2^{-12}] \begin{bmatrix} 0 \\ X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ 0 \\ X_2 \\ X_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (16)$$

Apply precise shifting we rewrite the equation is

$$Y=[2^{-1}2^{-2}2^{-3}2^{-4}2^{-5}2^{-6}2^{-8}2^{-9}] \begin{bmatrix} X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ \\ X_2 \\ X_1 \\ \\ \end{bmatrix} \quad (17)$$

When eqn(16) is compared to eqn(15), the number of adders in eqn(16) is reduced (15). With the use of mathematics shift, a product can be comprehended. As a result, NEDA-based architectures are less complex than traditional MACs that lack a multiplier and ROM.

IV. PROPOSED ARCHITECTURE

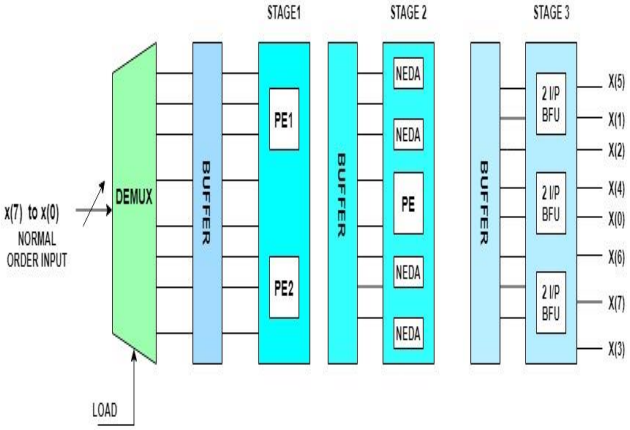
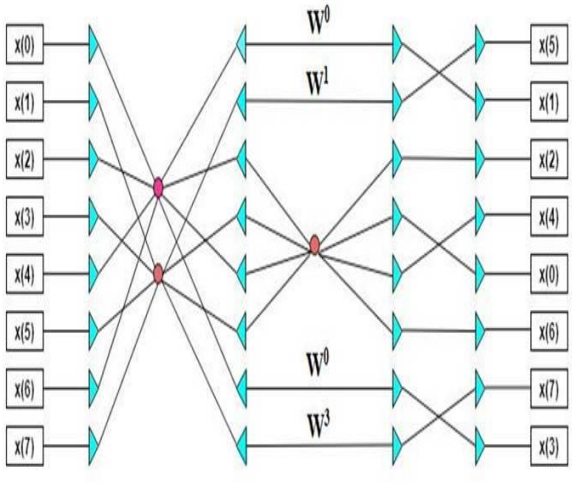


Fig.3. Functional diagram of proposed Split Radix FFT using NEDA

Figure 3 shows a functional diagram of a Split Radix FFT utilizing NEDA. Three phases are required in an 8-point FFT. (N= 2n) Here, the number of DFTs is N, and the number of stages is n = 3. The buffer can be connected between each step to store the outcome of that stage; for example, the yield of the first stage can be fed into the input of the second stage, and vice versa. NEDA was used to multiply the twiddle factors. FFT input is delivered through serial in this manner, allowing 'DEMUX' to be implemented on the input side. The suggested 64-point DFT with an 8 X 8 data width. Two input flow graphs can be implemented in the final stage of SRFFT. It does complex addition and multiplication with two inputs. As a result, a buffer can be used between the steps. It took three stages in total. Each stage is divided into PE and NEDA blocks. NEDA is made up of merely a shifter and an adder, with no ROM. Figure 4 depicts the data flow diagram for Pipeline SDF SRFFT. . In the first cycle, incoming sample data is kept in a FIFO until it reaches the inputs of the Radix-2 BFU, at which point it receives the feedback FIFO's input. Delay Feedback refers to the BFU's output being fed back to the FIFO. During each clock cycle, data is either read from or written to the memory. Each First In First Out has a 100% use percentage



.

Fig 4 signal flow graph of 8-point SRFFT

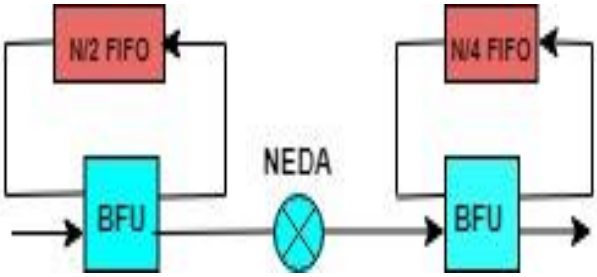


Fig. 5 Functional Diagram of Pipeline FFT Architecture (PFA)

V. EXPERIMENT IMPLEMENTATION OF R2^2 SRFFT

Simulations are run using XILINX ISE Tools, and the Sparton 3 FPGA board is configured using the system generator. MATLAB is used to simulate the procedure, which is supported by the XILINX ISE Platform.

A. SPEECH ENHANCEMENT USING SRFFT

The following procedure is used to degrade the noise from the original speech signal. Noise removal in speech signal using Python 3.

1. The noisy audio clip is subjected to an SRFFT.

2. Statistics are calculated using the noise's SRFFT (in frequency)

3. Based on the noise data, a threshold is calculated.

4. The signal is subjected to an SRFFT.

5. The signal FFT is compared to the threshold to create a mask.

6. A filter is used to smooth the mask over frequency and time.

7. The mask is inverted and applied to the signal's SRFFT.

VI. EXPERIMENTAL OUTPUTS AND DISCUSSIONS

The 16-point conventional FFT and R2^{2}SRFFT processor simulation results are obtained. In figure 6 shows the 16-point conventional FFT; it is observed to occupy more area and delay. To implement conventional FFT, more multipliers and adders are required. The hardware complexity is also more. Figure 7 shows the RTL schematic of 16-point conventional FFT output. Figure 8 shows the RTL schematic of 16 points conventional FFT. Figure 9 shows the timing report for 16 points conventional FFT. Figure 10 shows the timing report for 16-point SDF SRFFT. Figure 11 shows the simulation out Put of 16 point SDFSRFFT. Figure 12 shows the RTL schematic of 16-point SDFSRFFT In Figure 13 shows the Design utilization summary for conventional FFT. Figure 14 shows the Design utilization summary for SDF SRFFT. The audio input is shown in Figure 16. The audio input after the addition of White Gaussian noise is shown in Figure 17. The noise eliminated Enhanced Speech signal is shown in Figure 18. The output of the spectrogram is shown in figure 19. Performance analysis of before and after Filtering response shown in Figure.20.

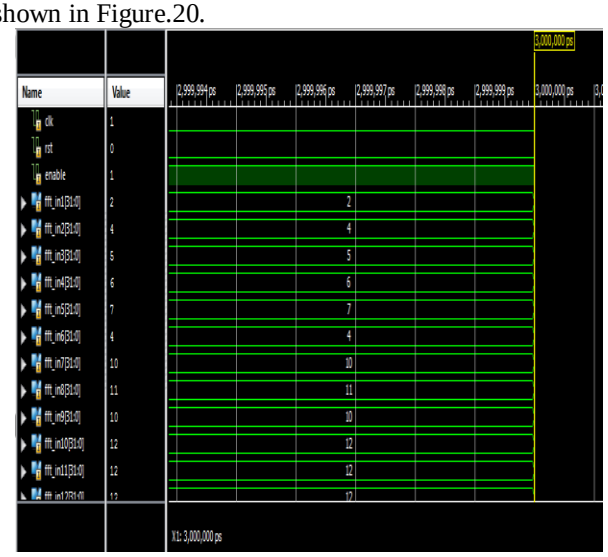


Fig .6 Simulation result of 16 points conventional FFT Input

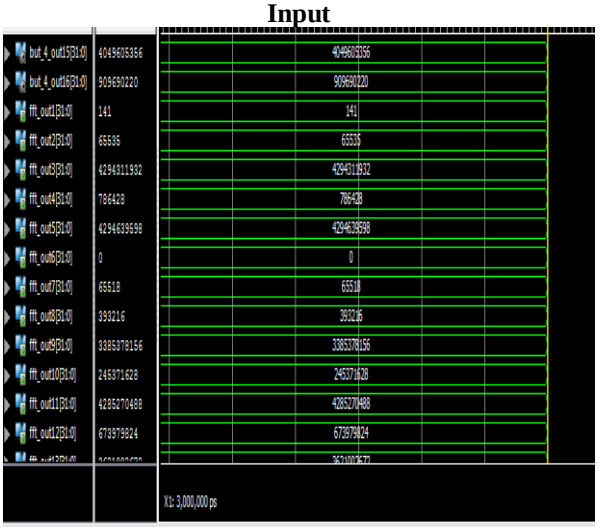


Fig 7 Simulation result of 16 points convolutional FFT Output

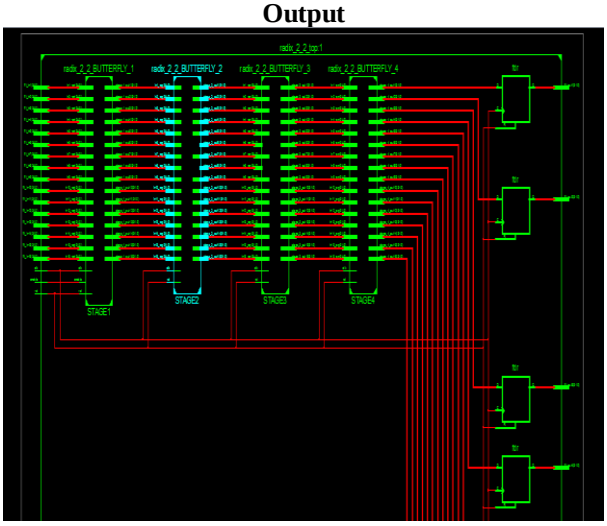


Fig 8 RTL schematic of 16 points conventional FFT

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 5.949ns (Maximum Frequency: 168.103MHz)
   Minimum input arrival time before clock: 4.363ns
   Maximum output required time after clock: 3.762ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 5.949ns (frequency: 168.103MHz)
  Total number of paths / destination ports: 730263 / 2604
-------------------------------------------------------------------------
Delay:               5.949ns (Levels of Logic = 14)
  Source:            STAGE2/c2/m1/B_in_9_14 (FF)
```

Fig 9 Timing report for conventional FFT

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	3101	11440	27%
Number of Slice LUTs	4911	5720	85%
Number of fully used LUT-FF pairs	2019	5993	33%
Number of bonded IOBs	1026	102	1005%
Number of BUFG/BUFGCTRLs	1	16	6%

Fig 10 Design utilization summary for conventional FFT

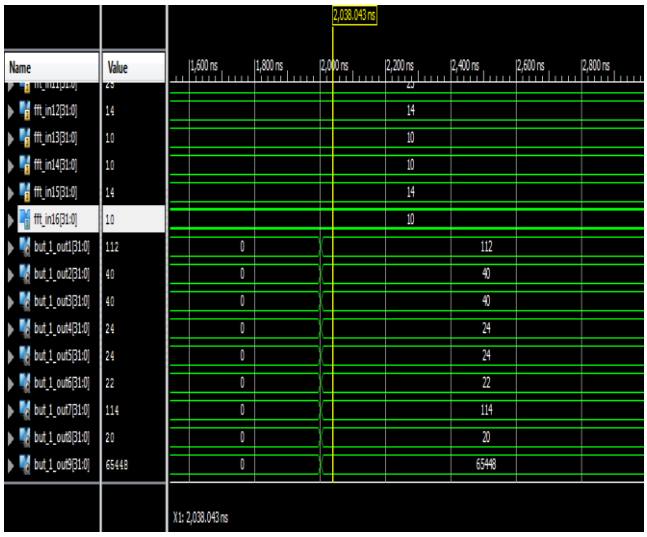


Fig 11 Simulation result of 16 point SRFFT Output

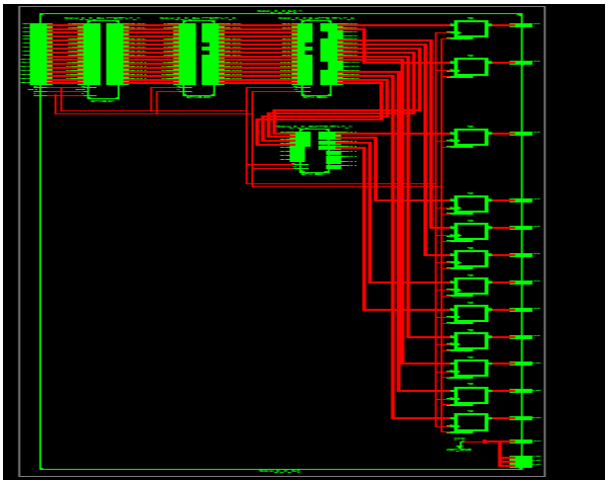


Fig 12 RTL schematic of 16 points SR FFT

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2377	11440	20%
Number of Slice LUTs	4366	5720	76%
Number of fully used LUT-FF pairs	1500	5263	28%
Number of bonded IOBs	1026	102	1005%
Number of BUFG/BUFGCTRLs	1	16	6%

Fig 13 Design utilization summary for SDF SRFFT.

```
=========================================================================
iming constraint: Default OFFSET IN BEFORE for Clock 'clk'
 Total number of paths / destination ports: 22569 / 2873
-------------------------------------------------------------------------
ffset:             4.189ns (Levels of Logic = 1)
 Source:           rst (PAD)
 Destination:      fft_out2_0 (FF)
 Destination Clock: clk rising

 Data Path: rst to fft_out2_0
                                Gate     Net
   Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
    IBUF:I->O          2377   1.222   2.537  rst_IBUF (rst_IBUF)
    FDR:R                     0.430          fft_out2_0
   ----------------------------------------
   Total                      4.189ns (1.652ns logic, 2.537ns route)
                                      (39.4% logic, 60.6% route)
```

Fig 14 Timing report for conventional FFT

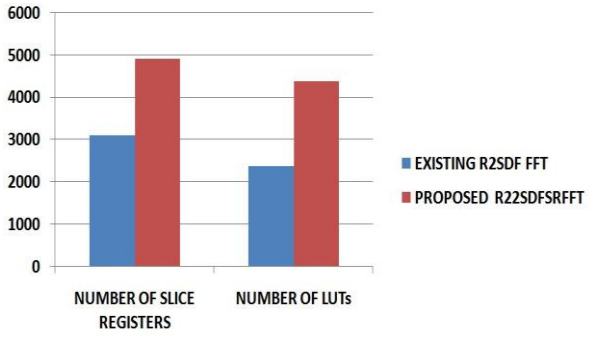


Fig15 Performance analysis of Existing SDFFFT and proposed R2^2 SDFFFT

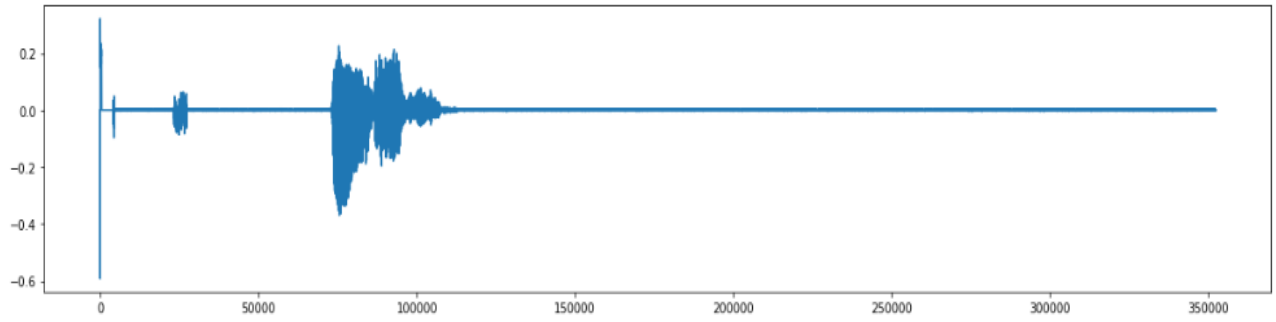


Fig 16. Audio input

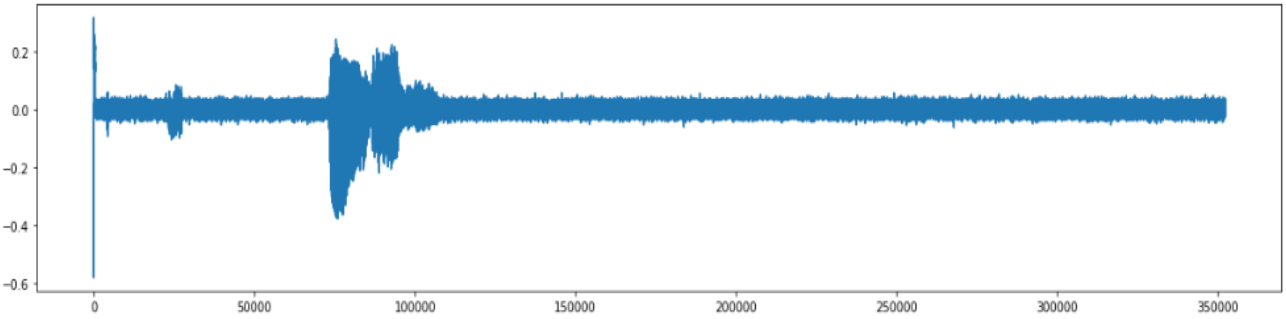


Fig 17. Plotting audio data after addition of White Gaussian Noise

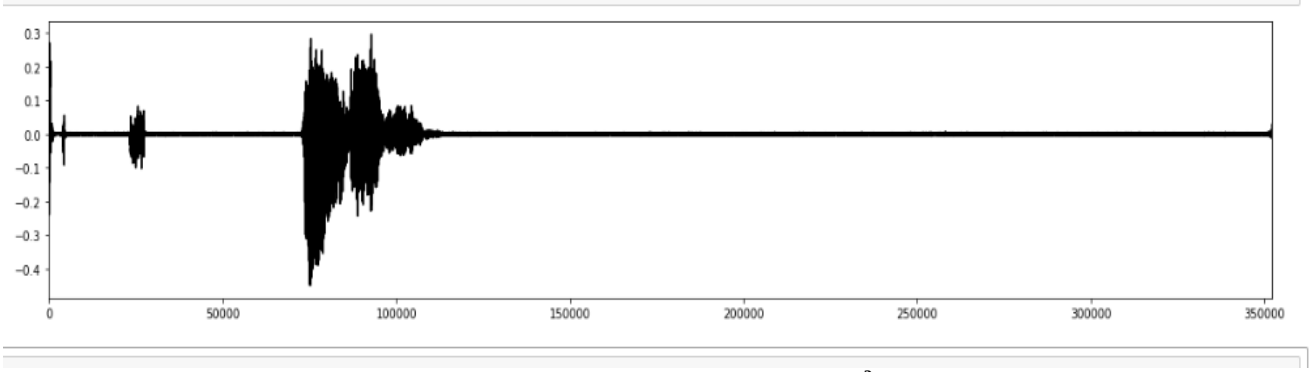


Fig 18. Noise added audio signal is removed by using R2^{2}SDFSRFFT output

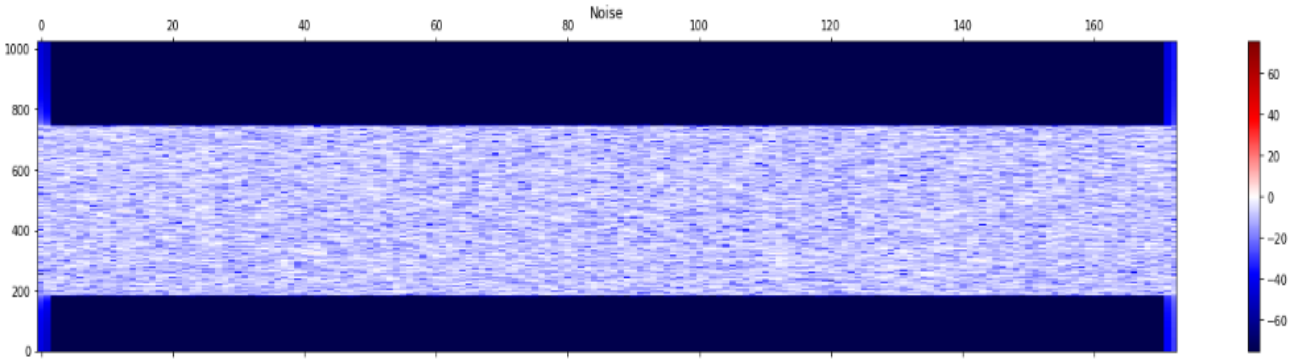


Fig 19. Output spectrogram

The Comparison result of conventional FFT and proposed R2^{2}SDFSRFFT is shown in Table 1. It is observed that, compared to Conventional FFT, the proposed FFT occupies less Area and time delay.

TABLE.1 RELATIVE RESULT OF CONVENTIONAL FFT VS. PROPOSED R2^2 SDF SRFFT

Types of Parameters	Existing R2SDF FFT	Proposed R2^{2}SDFSRFFT	Percentage Reduction %
Number of slice register	3101	2100	32.27
Number of Slice LUTs	4911	4386	10.69
Delay(ns)	5.949	4.189	29.58
Power(W)	2.197	1.519	44.63
Number of flip flops	2019	1500	20.18
Number of bonded IOBs	1026	926	10.25

The Development of SNR in different noises is shown in Table 2. It is observed that the 30 to 40% SNR ratio is improved after applying the proposed SRFFT Filter.

TABLE 2 DEVELOPMENT OF SNR RATIO FOR DIFFERENT TYPES OF NOISES

Various Noises	SNR Ratio input before Filtering (db)	SNR Ratio Output after Filtering (db)
White Gaussian Noise	11.4	22.3
Thunder	8.76	11.35
Engine	0.89	5.23
Boing	5.7	11.76

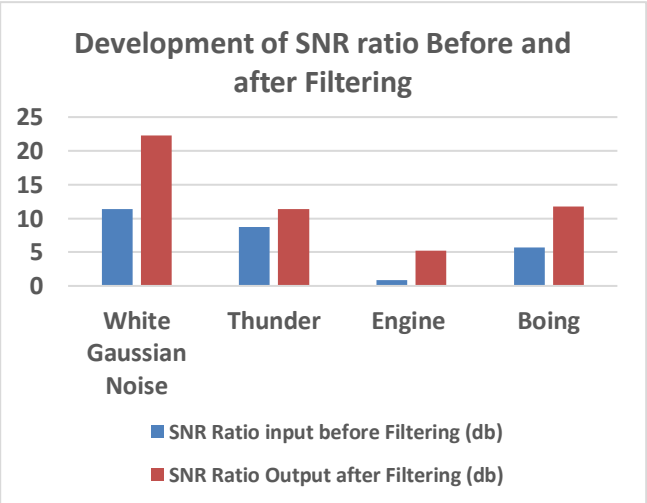


Fig 20. Performance Analysis of SNR Before and After Filtering

VII. CONCLUSION

This paper deals with the multiplier-less NEDA-based R2^{2}SDF SRFFT architecture. This architecture is implemented for 16-point complex inputs. Implementations are carried out using XILINX ISE Tools and configured using Spartan 6 FPGA. Comparing conventional FFT with Proposed SDFSRFFT shows the reduction in the following attributes: 10.69% reduction in LUTs, 32.27% reduction in Slices, 29.58% reduction in Delay, thus providing a reduced number of computations with better performance. Finally, speech enhancement is achieved by increasing the SNR ratios after SRFFT Filtering with various ambient noises. This method is suitable for DHA.

REFERENCES

[1] Cooley, J., & Tukey, J., An algorithm for the machine calculation of complex Fourier series. Mathematical Comparative,19 (1965) 297–301.
[2] N. Weste and D. J. Skellern, VLSI for OFDM, IEEE Common. Mag., 36 (1998) 127–131.
[3] Good, I., The interaction algorithm and practical Fourier analysis. Journal of the Royal Statistical Society, B-20 (1958) 361–372
[4] Winograd, S., On computing the discrete Fourier transform. Proceedings of the National Academy of science, (1976).
[5] Sciences of the United States of America, 73, 1005–1006.Morris, L., A comparative study of time-efficient FFT and WFTA programs for general-purpose computers. IEEE Transaction Acoustics Speech Signal Process, ASSP-26 (1978) 141–150
[6] Nawab, H., & McClellan, J., Bounds on the minimum number of data transfers in WFTA and FFT programs. IEEE Transaction Acoustics Speech Signal Process, ASSP-27 (1979) 394–398.
[7] Chen, T., Sunanda, T.G., Jin, J., COBRA: a 100-MOPS single-chip programmable and expandable FFT. IEEE Transactions Very Large Scale Integrative (VLSI) Systems, 7 (1999) 174–182.
[8] Maharatna, G., &Jagdhold, U., A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM. IEEE Journal Solid-State Circuits, 39 (2004) 484–493.
[9] Perez-Pascual, A., Sanaloni, T., Valls, J., FPGA-based radix-4 butterflies for HIPERLAN/2. In IEEE international symposium on circuits and systems , (2002) 277–280.
[10] Duhamel, P., & Hollmann, H., Implementation of split radix FFT algorithms for complex, real, and real-symmetric data.IEEE International Conference on Acoustics, Speech and Signal Processing, 10 (1985) 784–787
[11] Jiang, M., Yang, B., Huang, R., Zhang, T.Y., Wang, Y.Y.,Multiplierless fast Fourier transform architecture. Electronics Letters,43 (2007) 191–192.
[12] Maharatna, G., &Jagdhold, U., A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM. IEEE Journal Solid-State Circuits, 39 (2004) 484–493.
[13] Zohar, S., The counting recursive digital filter. IEEE Transaction on Computers, C-22 (1973) 338–347.
[14] Joshi, S.P., Paily, R. Distributed Arithmetic based Split-Radix FFT. J Sign Process Syst **75** (2014) 85–92.
[15] Duhamel, P., & Hollmann, H., Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data. IEEE International Conference on Acoustics, Speech and Signal Processing (1985).
[16] Ansuman Dipti Sankar Das, Abhishek Mankar, N Prasad, K. K. Mahapatra, Ayas Kanta Swain, Efficient VLSI Architectures of Split-Radix FFT using New Distributed Arithmetic, International Journal of Soft Computing and Engineering (IJSCE), (2013).
[17] Wendi Pan, Ahmed Shams, and Magdy A. Bayoumi, NEDA: A New Distributed Arithmetic Architecture and its Application to One Dimensional Discrete Cosine Transform, Proc. IEEE Workshop on Signal Processing Syst., (1999) 159 – 168.

[18] Y. Wang, Research Progress in Speech Enhancement Technology, 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), (2020) 222-226,

[19] M. Chen and K. -J. Chen, Research on Novel Normal Fuzzy Kalman Filter for Speech Enhancement in Noisy Environment, IEEE International Conference on Power Electronics, Computer Applications (ICPECA), (2021).

[20] G. M. Rao and U. S. Kumar, Speech enhancement using a combination of digital audio effects with Kalman filter, 2016 International Conference on Signal Processing, Communication, Power, and Embedded System (SCOPES), (2016).

[21] Hari Narayan Mishra, Agya Mishra Agya Mishra, Audio Enhancement using Remez Exchange Algorithm with DWT, SSRG International Journal of Electronics and Communication Engineering (SSRG-IJECE) – 2(2) (2015).

[22] P.Kabilamani, Dr.C.GomathyEfficient Modified Reduced FFT(MRFFT) Feedback-Commutator Architecture Design, SSRG International Journal of VLSI & Signal Processing (SSRG - IJVSP) - Volume 6(1) (2019).