Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676, p-ISSN: 2320-3331, Volume 20, Issue 3 Ser. 1 (May. – June. 2025), PP 15-22
www.iosrjournals.org

FFT Structure Implementation With Varying Radices In Verilog

Dr. Dasari Yugandhar
Korada Vishalakshi
Konchada Rakshita
Battula Hrushikesh
Andavarapu Yaswanth
Dept. Of Ece
Aditya Institute Of Technology And Management (Affiliated To Jntugv) Srikakulam, India

Abstract :
The Fast Fourier Transform (FFT)is a basic algorithm in digital signal processing (DSP) and communication systems, and it is a computationally very efficient algorithm for calculating the Discrete Fourier Transform (DFT). A DFT has a time complexity of ***O (N^2)*** *while an FFT has time complexity* ***O (N log N)****. DFT is an integral part of FFT. The Very Large Scale Integration (VLSI) implementation of the FFT is extremely computationally efficient by optimizing parameters like power, speed and area. Different FFT architectures employ different radix methods, such as Radix-2, Radix-4, and Split-Radix, to enhance performance. So in this paper it is mainly focused to implement a Decimation in Frequency Fast Fourier Transform (DIF-FFT) algorithm in Verilog HDL on AMD Vivado. The Verilog implementation of FFT which includes complex numbers addition and multiplication due to twiddle factor. FFT Verilog model outputs further validated with outputs obtained using Matlab. The findings are helpful in the design of optimized FFT structures that are suitable for high-speed and low-power DSP applications.*

Keywords: *DFT, FFT, Radix 2, Radix 4, Split Radix, VLSI, Verilog.*

Date of Submission: 25-04-2025 Date of Acceptance: 05-05-2025

I. Introduction

The Fast Fourier Transform (FFT) is a technique that is classically used to calculate the Discrete Fourier Transform (DFT). Despite being useful in calculating Fourier transforms, the DFT has certain problems during computing. As a solution to DFT's problems, Cooley-Tukey [8] proposed a new algorithm called Fast Fourier Transform (FFT), which is faster than DFT. If the samples in the signal is a power of two, the FFT algorithm can be adopted. The computation of FFT takes (N/2) × log2 N multiplications and N × log2 N additions. When comparing DFT, FFT takes fewer numbers of computations.

The technique of computing DFT using FFT algorithm is an implementation of a divide and conquer strategy. There are two main methods that the FFT algorithm can be implemented; the first is Decimation in Time (DIT) and the second one is Decimation in Frequency (DIF). Both DIT and DIF use the butterfly structure to compute FFT. FFT can accelerate calculations of DFT and decreases the time the DFT takes to be computed. A FFT is an algorithm to compute the discrete Fourier transform (DFT) and it's inverse. A Fourier transform converts time domain to frequency and inverse FFT (IFFT) converts frequency domain to time domain. Different radix structures, including Radix-2 [1], Radix-4 [3], and Split Radix [2], offer unique benefits in VLSI implementations of the DIF-FFT structure. While Radix-2 is simple and widely used, Radix-4 provides improved speed by reducing the number of computational stages. Split-Radix FFT combines the advantages of both Radix-2 and Radix-4, achieving a balance between computational efficiency and hardware complexity.

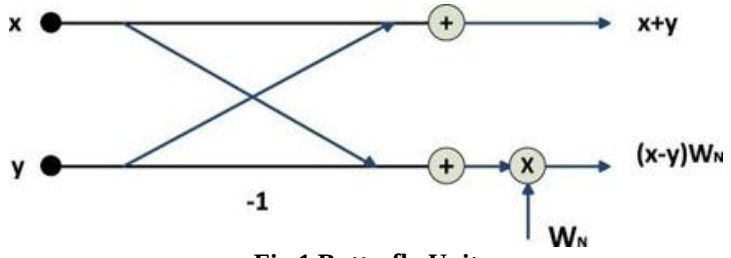

Fig.1 Butterfly Unit

FFT divides the data into smaller sets. The decimation in time takes inputs in bit reversed order and produces output in natural order while decimation is frequency takes normal input order and generates bit reversed output .This paper uses the decimation in frequency method. Each stages, the results of previous stage are combined. The Fig. 1 shows the butterfly diagram of FFT. The name butterfly implies that the shape of the flow of data. In every stage first inputs are add or subtract to other input and then multiplied by twiddle factor. In FFT calculation in DIF there is concept of taking the inputs in normal order to produce the outputs bits in the bit reversal order. The main contribution of this paper is to design FFT algorithm using radix-2 using AMD Vivado 2024.1tool with the help of Verilog code.

II. Literature Survey

The FFT algorithms using radix-2, radix-4, and radix-8, break the DFT into smaller sub problems in different ways. These algorithms can be implemented efficiently using VLSI technology, which allows for real-time signal processing [3-5]. When implemented using VLSI (Very Large Scale Integration), FFT benefits from low power consumption, making it ideal for mobile and battery-powered systems. Additionally, VLSI enables compact and efficient hardware designs, well-suited for embedded systems. Radix based FFT algorithms naturally exhibit parallelism, enhancing throughput and overall performance. Furthermore, VLSI-based FFT implementations offer scalability, allowing them to handle large datasets efficiently, making them valuable for big data processing applications [6].

FFT plays a crucial role in various applications across multiple domains. In telecommunications, it is used for modulation, demodulation, and error correction, particularly in systems like OFDM (Orthogonal Frequency Division Multiplexing). In radar and sonar, FFT aids in object detection and measuring distance and speed. It is also essential in audio and speech processing, enabling real-time speech recognition and audio compression formats like MP3 and AAC. In image processing, FFT is employed for transformations, filtering, and compression, as seen in JPEG. Medical imaging technologies such as MRI and CT scans rely on FFT for image reconstruction. Additionally, FFT is widely used in scientific simulations, including wave propagation, fluid dynamics, and quantum mechanics, making it a fundamental tool in computational science [7].

The leveraged a processor capable of performing one billion radix-2 butterfly operations per second, utilizing current VLSI technologies and components to optimize performance. The architecture was designed to demonstrate feasibility and efficiency for complex DSP tasks. A structured design approach combining state-of-the-art integrated circuits and advanced signal processing techniques. The design was guided by practical experience and requirements for high-performance DSP systems highlights the potential of modern integrated circuits in realizing high-performance DSP systems. It emphasizes the need for future VLSI research to be informed by insights from such designs to meet growing computational demands. They incorporated a modified Booth's algorithm to optimize Add/Subtract operations and eliminated carry propagation chains for parallel multiplication. The design uses a regular array structure for efficient VLSI implementation. They utilized signed digital number systems and optimized multiplication by breaking carry chains and leveraging a modified Booth's algorithm. A regular array structure was employed to enhance computational efficiency and simplify VLSI implementation. The proposed multiplier achieved nearly double the speed of Wallace tree multipliers, with computing time proportional to log(N).The number of computing cells was proportional to $N \cdot \log2(N/2)$, fewer than conventional multipliers.[9],[10].

They employed a low-level parallel filter structure with changes to device configurations and operations to reduce complexity while maintaining performance. Hybrid programming enabled effective simulation and analysis of the proposed algorithm. The new FIR filter [11] design reduced computational complexity and device usage while achieving efficient implementation. The design requirements of fixed algorithms in communication systems and demonstrated methods to tailor VLSI implementations to specific applications. The approach was illustrated using examples that optimize at the algorithm, word, and bit levels. The three level design methodology effectively reduces the complexity of VLSI implementation by tailoring it to specific DSP algorithms. This approach bridges the algorithm-architecture gap, enabling efficient and optimized solutions. The proposed three-level design approach is well suited for the VLSI implementation of DSP tasks in communication systems, ensuring efficient and application-specific solutions. It provides a structured framework to overcome implementation challenges in VLSI design [12]. The proposed FFT scheme achieves fewer computation operations compared to state-of-the-art methods. The multi-stage image encryption algorithm demonstrated strong encryption performance and robustness through experimental results and security analysis.

III. Methodology And Methods

In DIF-FFT, the butterfly operations are performed by taking input sequence in natural order, making it different from DIT, where input ordering happens before computations. A structured design approach combining state-of-the-art integrated circuits and advanced signal processing techniques. The design was guided by practical experience and requirements for high-performance DSP systems highlights the potential of modern integrated

circuits in realizing high-performance DSP systems. It emphasizes the need for future VLSI research to be informed by insights from such designs to meet growing computational demands. Defining and solving the problem by implementing the Radix-2, Radix-4 and Split-Radix FFT algorithms in Verilog as shown in Fig.2. The simulation results are analyzed to verify correctness and efficiency of different radices.

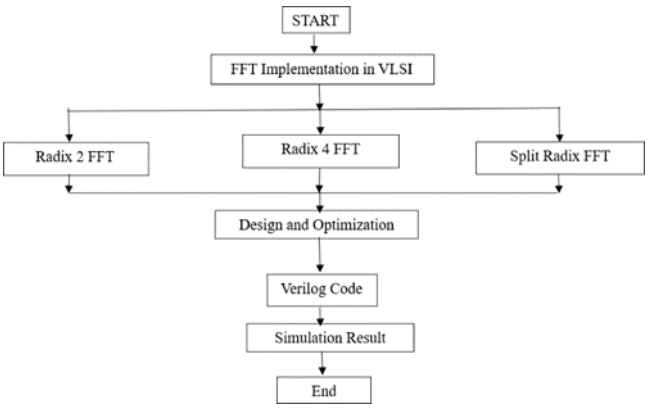

Fig.2 Methodology- Flow Chart

The Radix-2 approach divides the given input sequence recursively into smaller Discrete Fourier Transforms (DFTs) in order to efficiently compute the DFT for data lengths that are powers of two. This algorithm's core function, known as the "butterfly", uses certain complex exponential multipliers known as "twiddle factors" to merge the output of two smaller DFTs. The data flow via these butterfly operations' stages is graphically depicted in the butterfly diagram. The Butterfly Structure has $N=2^n$, where, n is the number of Stages. Radix-2 butterfly structure shown in Fig. 3 to Fig. 5 for an input sequence of length 4, 8 and 16 respectively.

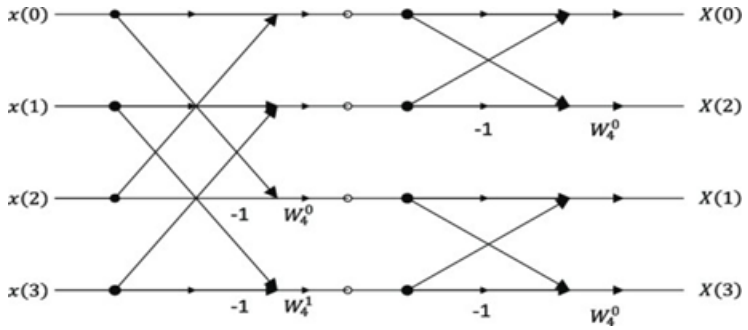

Fig.3 A 4 Point input sequence Radix-2 Butterfly Structure

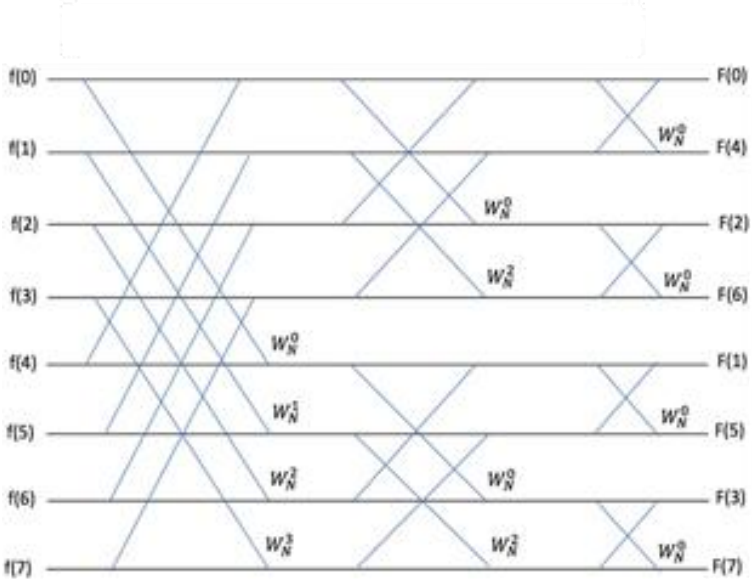

Fig.4 A 8-Point input sequence radix-2 Butterfly Structure

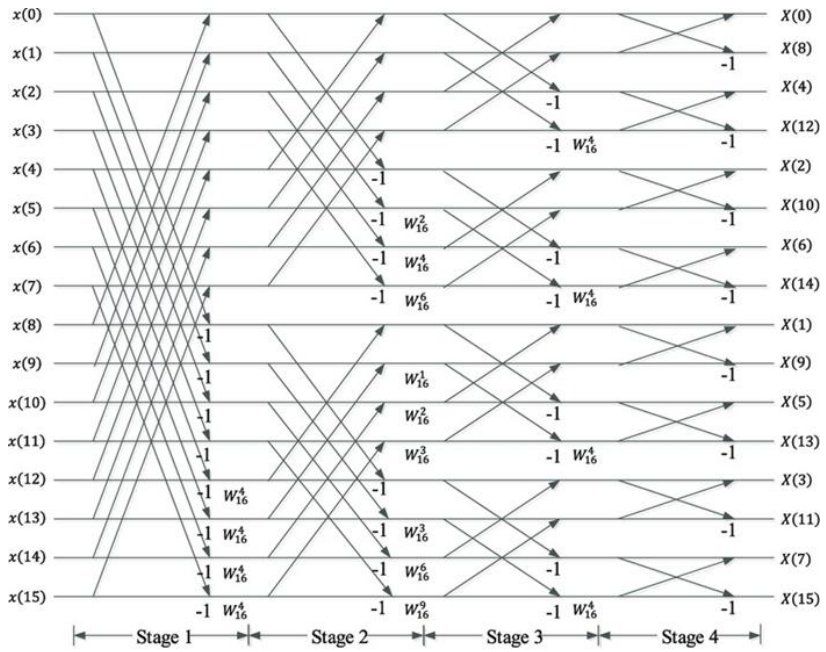

Fig.5 A 16 Point input sequence Radix-2 Butterfly Structure

The Radix-4 FFT algorithm is an extension of the Radix-2 FFT that processes four data points at a time instead of two shown in Fig.6 and Fig. 7 for an input sequence of 4 and 16 respectively. It used to further reduce the computational complexity of the DFT when the input sequence length N is a power of 4. By grouping the input data into blocks of four, the Radix-4 algorithm reduces the number of required arithmetic operations compared to Radix-2, making it more efficient for large inputs. This algorithm is particularly advantageous for high-speed FFT implementations in hardware and software when the input size is highly composite and divisible by 4.

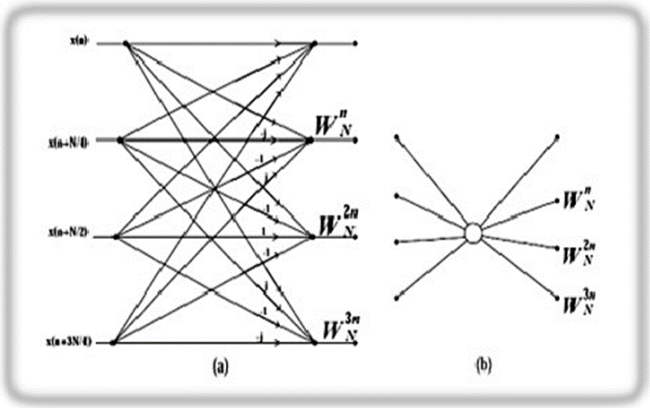

Fig.6 A 4-Point input sequence Radix-4 Butterfly Structure

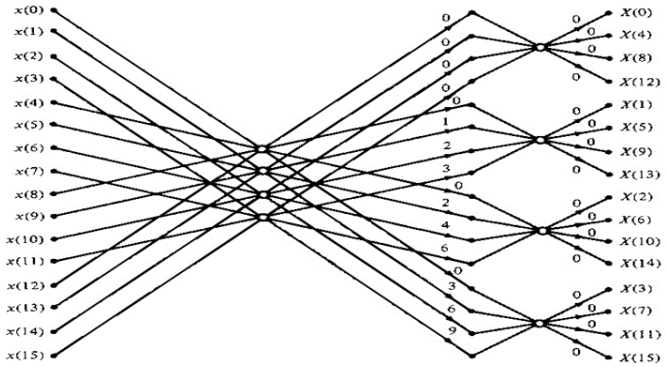

Fig.7 A 16 Point input sequence Radix-4 Butterfly Structure

The Split-Radix [13] FFT algorithm is an advanced and highly efficient method for computing the DFT of a sequence, especially when the sequence length N is a power of 2. It cleverly combines the strengths of both the Radix-2 and Radix-4 algorithms by splitting the DFT into a mix of smaller DFTs—specifically, one DFT of size N/2 and two DFTs of size N/4. This unique structure allows it to use fewer arithmetic operations than standard Radix-2 and even Radix-4 FFT algorithms, making it one of the most efficient algorithms in terms of number of multiplications and additions. Despite the increased complexity in programming due to its irregular structure, the Split-Radix FFT is widely appreciated for its lower computational cost and is commonly used in high-performance DSP applications. Split radix algorithm for an input sequence 4 and 8 is shown in Fig.8 and Fig. 9 respectively.

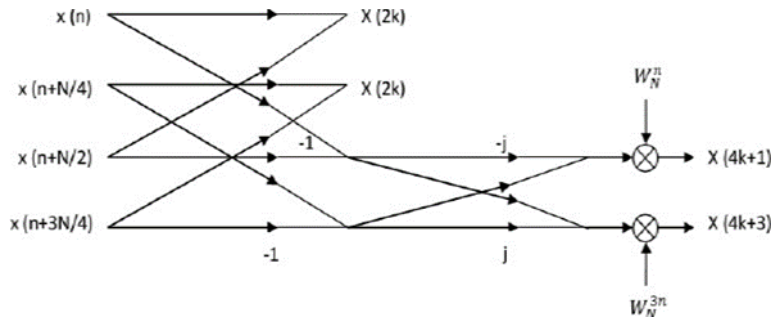

Fig.8 A 4-Point sequence Split Radix Butterfly Structure

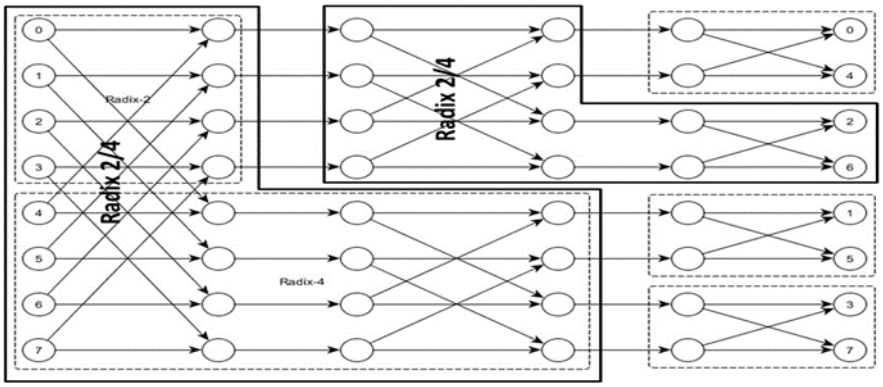

Fig.9 A 8-Point sequence Split Radix Butterfly Structure

Table 1 Comparison of different Radices

Radix Type	Computational Stages	Complexity
Radix-2	$\log_2(N)$	O (N log N)
Radix-4	$\log_4(N)$	O (N log N)
Split Radix	$\log_2(N)$ and $\log_4(N)$	O (N log N)

IV. Results And Discussions

FFT algorithm successfully implemented in Verilog HDL on AMD Vivado and validated by comparing its output with MATLAB results, ensuring accuracy. The design efficiently handled complex number operations, including twiddle factor multiplications and additions, leading to improved computational efficiency. Performance metrics such as power consumption, speed, and area were analyzed, demonstrating reduced hardware complexity while maintaining precision.

The implementation of the varying radices in Verilog results the corresponding simulation results. The below Fig.10 shows the simulation input for the 4-point sequence for the Split Radix and the output is shown in Fig.11. The implementation of the varying radices in Verilog results the simulation results. The below Fig.12 shows the simulation input for the 8-point sequence for the Radix-2 and the output is shown in Fig.13. The implementation of the varying radices in Verilog results the simulation results. The below Fig.14 shows the simulation input for the 16-point sequence for the Radix-4 and the output is shown in Fig.15.

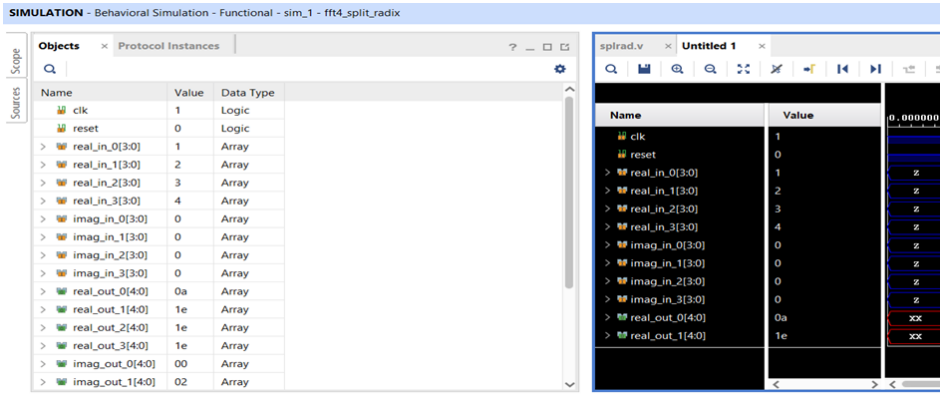

Fig.10 Simulation input for the 4-Point split radix for the input sequence

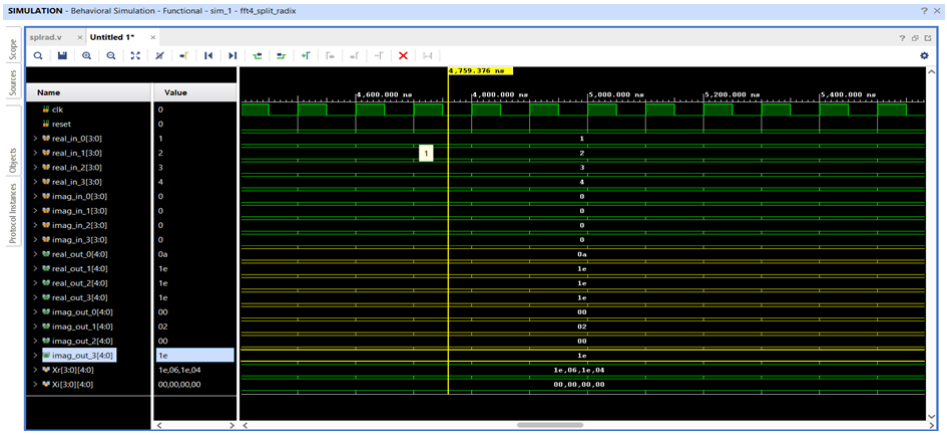

Fig.11 FFT Output for 4-point split radix

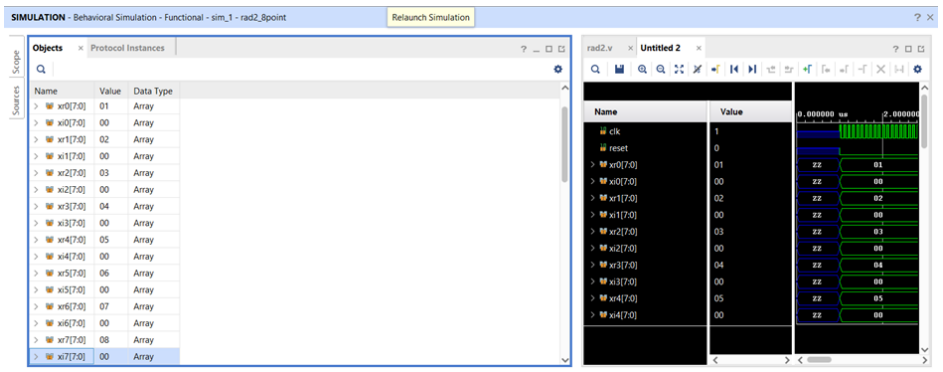

Fig.12 Simulation input for the 8-point Radix-2 for the input sequence

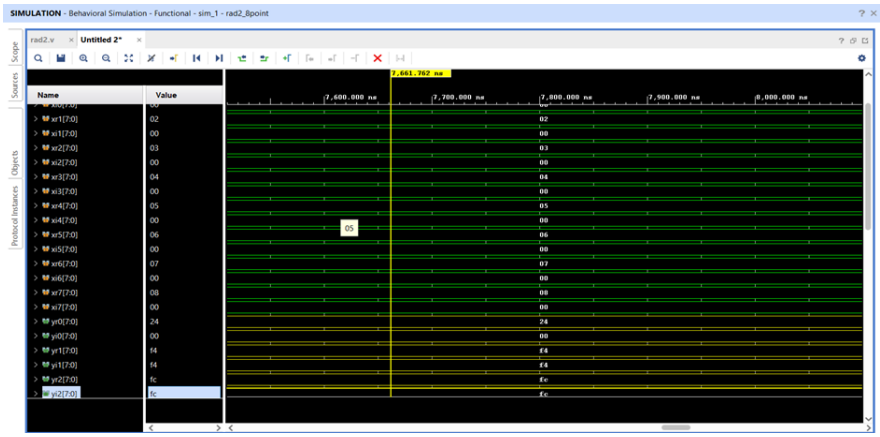

Fig.13 FFT Output for 8-point radix-2

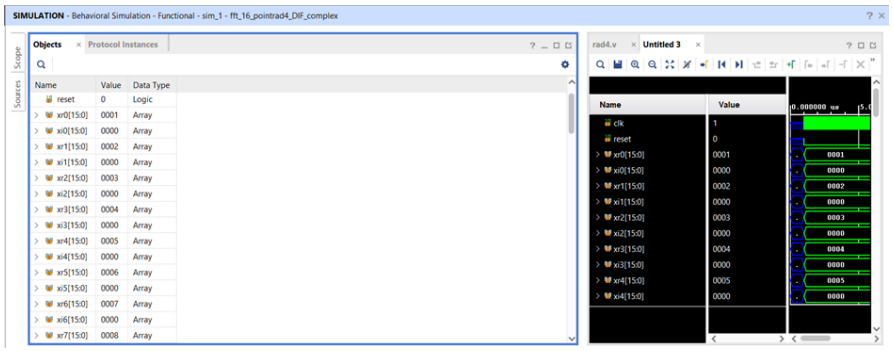

Fig.14 Simulation input for the 16-point Radix-4 for the input sequence

Fig.15 FFT Output for 16-point radix-4

As we know that the output for the different radices of the same number of bits are same even if the structure is DIT-FFT or DIF-FFT. So here, the we can say that the output for the radix-2,radix-4 and split radix for 4-point,8-point and16-point respectively are same . So here only one input and corresponding output values simulation result is attached.

Table 2 Comparison of different radices for a 4-point input sequence

Radix	LUTs (63400)	Registers (126800)	IOBs (210)
Radix-2	91	39	74
Radix-4	86	36	74
SplitRadix	31	32	74

Table 3: Comparison of different radices for a 8-point input sequence

Radix	LUTs (63400)	Registers (126800)	IOBs (210)
Radix-2	468	525	258
SplitRadix	296	525	258

Table 4 Comparison of different radices for a 16 point input sequence

Radix	LUTs (63400)	Registers (126800)	IOBs (210)
Radix-2	1709	2446	1026
Radix-4	1725	1599	994

V. Conclusions

This paper presented a comparative analysis of FFT implementations using different radix structures such as Radix 2, Radix 4, Split Radix in Verilog. While Radix-2 provides simplicity and low power consumption, Radix-4 improves speed by reducing computational stages. Split-Radix FFT offers a hybrid approach that optimizes computational efficiency and hardware complexity. The study highlights the importance of selecting the appropriate radix for FFT implementations based on application requirements. And here, the hardware utilization for different radices for different points are observed.

References

[1] L. Santhosh And A. Thomas, "Implementation Of Radix 2 And Radix 22 Fft Algorithms On Spartan6 Fpga," In 2013 Fourth International Conference On Computing, Communications And Networking Technologies (Icccnt), Tiruchengode: Ieee, Jul. 2013, Pp. 1–4. Doi: 10.1109/Icccnt.2013.6726840.

[2] P. Duhamel, "Implementation Of 'Split-Radix' Fft Algorithms For Complex, Real, And Real-Symmetric Data," Ieee Trans. A Coust. Speech Signal Process., Vol. 34, No. 2, Pp. 285–295, Apr. 1986, Doi: 10.1109/Tassp.1986.1164811.
[3] R. Barma Venkata And N. Fazal, "Fpga Implementation Of Optimized Radix 4 And Radix 8 Booth Algorithm," Int. J. Perform. Eng., Vol. 17, No. 6, P. 552, 2021, Doi: 10.23940/Ijpe.21.06.P8.552558.
[4] M. Butorac And M. Vucic, "Fpga Implementation Of Simple Digital Signal Processor," In 2012 19th Ieee International Conference On Electronics, Circuits, And Systems (Icecs 2012), Seville, Seville, Spain: Ieee, Dec. 2012, Pp. 137–140. Doi: 10.1109/Icecs.2012.6463781.
[5] F. D. Nunes And J. M. N. Leitao, "Signal Processing Aspects Of Fusion Plasma Broadband Reflectometry," Ieee Trans. Signal Process., Vol. 47, No. 2, Pp. 378–388, Feb. 1999, Doi: 10.1109/78.740123.
[6] D. Goyal, C. Mongia, And S. Sehgal, "Applications Of Digital Signal Processing In Monitoring Machining Processes And Rotary Components: A Review," Ieee Sens. J., Vol. 21, No. 7, Pp. 8780–8804, Apr. 2021, Doi: 10.1109/Jsen.2021.3050718.
[7] A. Rai And S. H. Upadhyay, "A Review On Signal Processing Techniques Utilized In The Fault Diagnosis Of Rolling Element Bearings," Tribol. Int., Vol. 96, Pp. 289–306, Apr. 2016, Doi: 10.1016/J.Triboint.2015.12.037.
[8] Cooley, J. W., & Tukey, J. W. (1965). An Algorithm For The Machine Calculation Of Complex Fourier Series. Mathematics Of Computation, 19(90), 297-301.
[9] Singleton, R. C. (1969). On Computing The Fast Fourier Transform. Communications Of The Acm, 12(10), 527-531.
[10] Duhamel, P., & Hollmann, H. (1984). Split-Radix Fft Algorithm. Electronics Letters, 20(14), 586-588. 51
[11] Sorensen, H. V., Jones, D. L., Heideman, M. T., & Burrus, C. S. (1990). Real-Valued Fast Fourier Transform Algorithms. Ieee Transactions On Acoustics, Speech, And Signal Processing, 38(10), 1681-1691.
[12] Chen, Y., Li, W., & Li, J. (2013). A New Split-Radix Fft Algorithm For Length-N Dft. Ieee Transactions On Signal Processing, 61(10), 2643-2653.
[13] Li, J., Chen, Y., & Li, W. (2020). A High-Performance Split-Radix Fft Algorithm On Gpus. Ieee Transactions On Parallel And Distributed Systems, 31(10), 2331-2344.