International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Low Power 16-Point FFT Processor Using Radix 2&4 Butterfly Units

Deekshitha K N[1], Divya Lakshmi M[2], Keerthana B K[3], Jalaja S[4]

[1]Student of Bangalore Institute of Technology, Department of Electronics Engineering (VDT), Karnataka, India

[2]Student of Bangalore Institute of Technology, Department of Electronics Engineering (VDT), Karnataka, India
Email: *msld.divya22[at]gmail.com*

[3]Student of Bangalore Institute of Technology, Department of Electronics Engineering (VDT), Karnataka, India

[4]Faculty of Bangalore Institute of Technology, Department of Electronics Engineering (VDT), Karnataka, India
Email: *jalajas[at]bit-bangalore.edu.in*
https://orcid.org/0000-0001-6676-7827

Abstract: ***This paper presents the implementation of a low power 16-point FFT processor utilizing a mixed radix approach. The design includes sub modules like twiddle multiplication, stage wise computation, butterfly computations. The main objective relies on the 16-point FFT, achieved by integrating the radix 2/4/2 algorithm (mixed radix) techniques along with low power strategies such as clock gating, FSM grey encoding and additional methods such as RAM memory. Likewise, we focus on obtaining the low power consumption. This approach results in a reasonable balance between power consumption and computational efficiency. The design process is conducted using Vivado and Cadence for simulation and synthesis to enhance power analysis reports. Overall, the dynamic power obtained using mixed approach is quiet high while, the best low power consumption was obtained after implementing FSM grey encoding. After optimization the dynamic power contribution is reduced to 14.60% of the total power consumption. This marks our approach to be effective for reducing the power consumption to 30.27%, significantly.***

Keywords: low power FFT processor, mixed radix FFT, clock gating techniques, FSM grey encoding, power optimization in VLSI.

1. Introduction

1.1 Discrete Fourier transform (DFT)

Traditionally, O(N2) complexity is needed for the direct implementation of the N-point discrete Fourier transform (DFT). Turkey-Cooley the FFT algorithm, which was first published in 1965, is an effective algorithm that greatly reduces the computational requirements of DFT to only O(N log N) computations by taking advantage of its symmetry and periodicity properties [1]. Other technologies, particularly embedded systems and system-on-a-chip (SoC), thrive as semiconductor technology advances. The hardware implementation of FFT is becoming feasible due to this evolution. DSP is a specialized microcontroller that has optimized architecture for fast operation. By sampling fourier transform X(e^jw) , we obtain a frequency domain discrete sequence X(K) known as DFT which is powerful computation tool for frequency of DFT signal.

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn} \quad (1)$$

$$W_N^{kn} = e^{-j\left(\frac{2\pi kn}{N}\right)} = \cos\left(\frac{2\pi kn}{N}\right) - j\sin\left(\frac{2\pi kn}{N}\right) \quad (2)$$

1.2 Fast Fourier transform (FFT)

Table 1: Comparison of Computational Complexity for Direct Computation and FFT Algorithms

No. of Points (N)	Direct computation complex additions (N^2-N)	Direct computation complex multiplications (N^2)	FFT complex multiplications (1/2 N $\log_2$ N)	FFT complex additions (N $\log_2$ N)
4 pt	12	16	4	8
8 pt	56	64	12	24
16 pt	240	256	32	64

The table illustrates how the FFT algorithm significantly reduces workload, whereas direct DFT computation necessitates a very large number of complex additions and multiplications as N increases. The Fast Fourier Transform (FFT) is a crucial method for calculating the Discrete Fourier Transform (DFT) in smaller segments, converting time-based signals into their frequency domain representations. As the name implies, it represents one of the quickest methods of the DFT algorithm, employing a divide and conquer strategy. Various algorithms such as Radix 2, 4,8, 2^2, and 2^k are part of this divide and conquer methodology [2].

In the case of advantages in FFT, implementing hardware with a higher bit radix proves to be more efficient due to its advantage of high throughput, and reduction in power, and reduced latency [4]. The design employs the DIF domain, where the inputs are in their natural form, where twiddle factor multiplication occurs post-butterfly operation. This method offers greater advantage over DIT, where twiddle factor multiplication is executed after the butterfly operation.

The FFT applications immense in communication systems, including orthogonal frequency division multiplexing

(OFDM) and single carrier frequency division multiple access (SC-FDMA). FFT is essential for 5G communication technologies. The ideology of OFDM was studied [3]. OFDM systems encompass ultra- wideband, asymmetric digital subscriber line, digital audio broadcasting, and digital video broadcasting.

1.3 Radix 2 DIF FFT

This algorithm efficiently evaluates the DFT using a divide and conquer strategy. The N-point DFT is decomposed into progressively smaller DFTs; if N is factored as N=r1, r2, r3, ……. rn, then $n=r^v$ implies $N=2^v$. In this context, r represents the radix of FFT algorithm. For the Fast Fourier Transforms, a number of algorithms have been proposed, including Radix-2, Radix-4, Radix-8, and several other higher order radix. Each of these algorithms has a unique advantage in terms of simplicity. Although Radix-2 is the best, it is not as fast as Radix-4 FFT implementation because it requires four more clock cycles [5]. When compared to the radix-2 algorithm, the radix-4 and radix-8 algorithms perform better arithmetic operations and data transfer [6].

1.4 Computation

In order to reduce complexity, the computation is divided into several stages where the original N-point DFT is gradually broken down into smaller parts (G(k), H(k), A(k), C(k), and B(k)) using twiddle factors. The FFT algorithm is based on this staged decomposition, which reduces the overall number of multiplications and additions when compared to direct DFT computation [9].

{First Stage}

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}$$

$$X(k) = \sum_{n=0}^{N-2} x(n)W_N^{kn} + \sum_{n=1}^{N-1} x(n)W_N^{kn}$$

$$X(k) = G(k) + W_N^k H(k) \quad (3)$$

{Second Stage}

$$G(k) = A(k) + W_{\frac{N}{2}}^k B(k) \quad (4)$$

$$H(k) = C(k) + W_{\frac{N}{2}}^k D(k) \quad (5)$$

$$\beta = \frac{N^2}{4} + 2N \quad (6)$$

{Third Stage }

$$A(k) = \sum_{n=0}^{\frac{N}{4}-1} x(n)W_{\frac{N}{4}}^{nk} = \sum_{n=0}^{N-1} x(n)W_2^{nk} \quad (7)$$

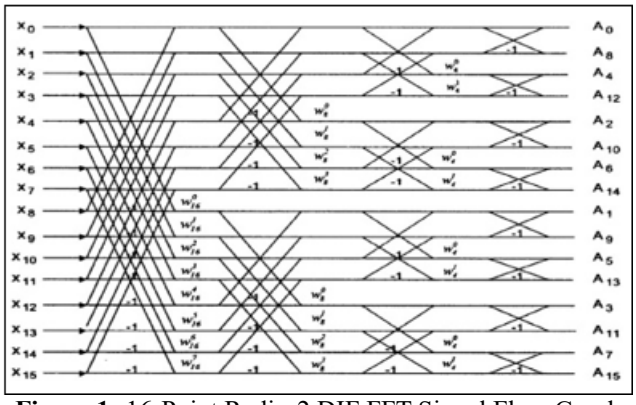


Figure 1: 16-Point Radix-2 DIF FFT Signal Flow Graph

The 16-point input sequence is gradually divided into smaller sub-sequences over four stages in the Decimation-in-Frequency (DIF) FFT shown in the figure[8,13]. Each stage multiplies using the proper twiddle factors after performing butterfly operations (add/subtract).

1.5 Radix 4 DIF FFT

The Radix-4 butterfly operation is more efficient in computing the DFT than the Radix-2 method. It processes four inputs and generates four outputs. This algorithm is applicable to points that are powers of 4, such as 4, 8, 16, and so forth. The Radix-4 algorithm is utilized for DFT computation where speed is crucial, it has a more complex structure than lower radix algorithms but delivers results in fewer stages with fewer multipliers [6].

$$X(k) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) + (-1)^2 x\left(n + \frac{N}{4}\right) + (-1)^2 x\left(n + \frac{N}{4}\right) + (-j)^2 x\left(n + \frac{3N}{4}\right)\right] W_N^{kn} \quad (8)$$

The radix-4 algorithm breaks down the N-point DFT into N/4. This method can be divided into four distinct subsets [7].

$$X(4k) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right)\right] W_N^0 W_{\frac{N}{4}}^{kn} \quad (9)$$

$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) - jx\left(n + \frac{N}{4}\right) - x\left(n + \frac{N}{2}\right) + jx\left(n + \frac{3N}{4}\right)\right] W_N^n W_{\frac{N}{4}}^{kn} \quad (10)$$

$$X(4k+2) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) - x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) - x\left(n + \frac{3N}{4}\right)\right] W_N^{2n} W_{\frac{N}{4}}^{kn} \quad (11)$$

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) + jx\left(n + \frac{N}{4}\right) - x\left(n + \frac{N}{2}\right) - jx\left(n + \frac{3N}{4}\right)\right] W_N^{3n} W_{\frac{N}{4}}^{kn} \quad (12)$$

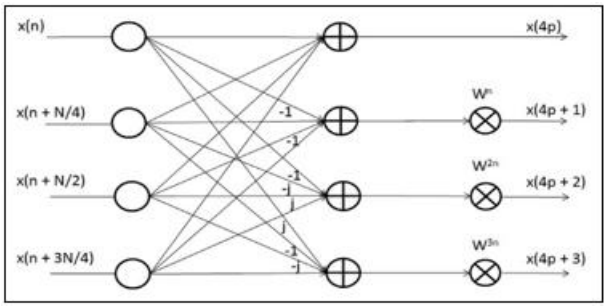


Figure 2: Radix-4 DIF FFT Butterfly Structure

This diagram illustrates the sequential operation of a 16-point FFT. Twiddle factors are used to multiply the input values after they have been mixed and combined using "butterfly" connections. The final frequency outputs are shown on the right after each stage [8].

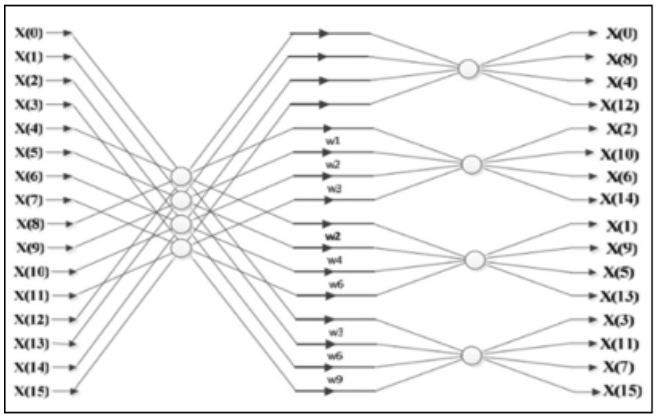


Figure 3: 16-point Radix-4 DIF FFT signal flow graph

The 16-point Radix-4 DIF FFT is depicted in the figure, where inputs are processed through Radix-4 butterfly units after being grouped in sets of four. Each butterfly generates the intermediate values that result in the final FFT outputs by performing addition, subtraction, and twiddle-factor multiplications [8].

Table 2: Comparison table of Radix -2 and Radix -4

No. of Points (N)	FFT Radix	No. of Multiplications	No. of Additions
4	Radix-2	4	8
16	Radix-2	32	64
4	Radix-4	4	12
16	Radix-4	24	48

The above table illustrates the number of multiplications and additions needed for each method when comparing Radix-2 and Radix-4 FFTs for 4-point and 16-point sizes. It makes it abundantly evident that Radix-4 is more effective for larger FFT sizes like 16-point because it requires fewer operations than Radix-2.

2. Proposed Architecture

2.1 Overview of block diagram

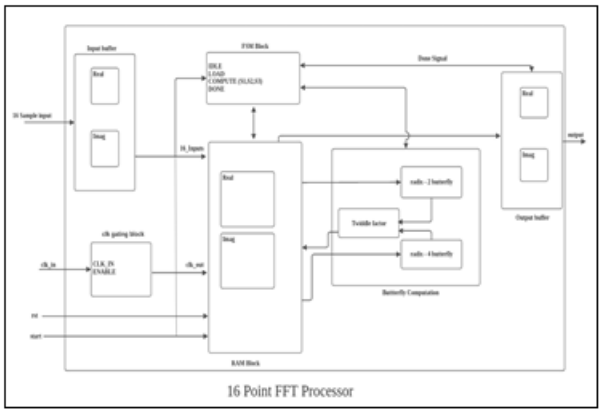


Figure 4: Overview of block diagram

The diagram shows the layout of a 16-point FFT processor [11]. It begins with an input buffer that holds the real and imaginary parts of the 16 samples. The FSM (Finite State Machine) manages the entire operation through states like IDLE, LOAD, COMPUTE, and DONE, making sure data moves properly [12]. A clock-gating block controls the clock to save power by sending a regulated clock signal to the RAM block [10]. The RAM holds intermediate FFT values and sends them to the butterfly computation block. This block includes Radix-2 and Radix-4 butterfly units, along with the twiddle-factor generator. These butterfly units carry out the necessary additions, subtractions, and multiplications to compute the FFT in stages. Finally, the processed frequency-domain results go to the output buffer, where the real and imaginary outputs are generated.

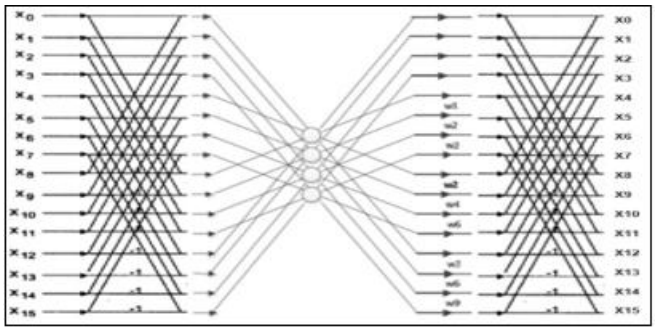


Figure 5: Mixed radix 2/4/2 signal flow graph

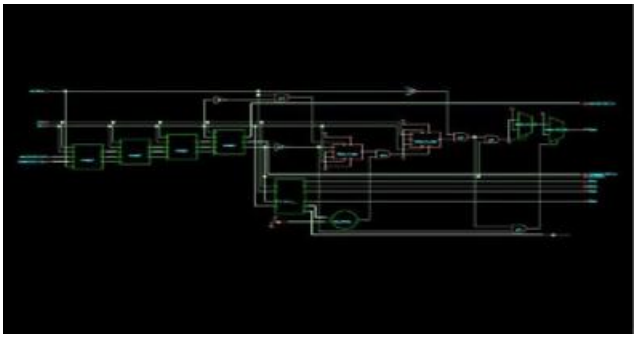
Input samples are loaded into the RAM block under the FSM controller, in stage-1 Radix-2 butterfly operations are performed which is read from the RAM processed and written back, similarly in stage-2 Radix-4 butterfly is computed with the intermediate results which under goes twiddle multiplication and are written back to RAM, in stage-3 Radix-2 butterfly operation is performed. These stages are operated by gray encoded FSM sequence while minimizing the switching power. Finally 16-point FFT outputs are sequentially read from the RAM. In this context, the mixed radix consists of a radix format of 2/4/2, with each component calculated at every stage. Each stage involves the computation of the different radix formats, specifically 2, 4, and 2.

Table 3: Stage-wise Computational Complexity of Mixed-Radix FFT

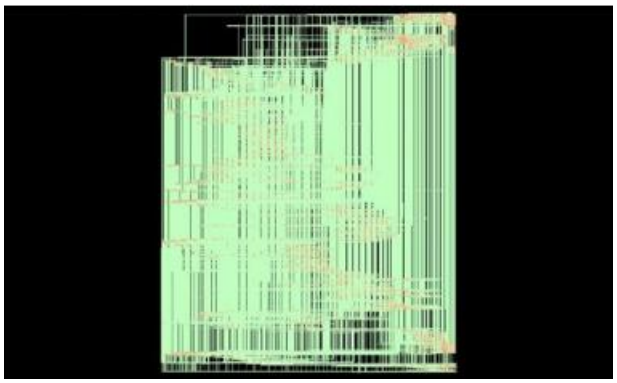
Stage	Radix FFT	No. of Twiddle Factors	No. of Multiplications	No. of Additions
Stage-1	Radix-2	16	8	8
Stage-1	Radix-2	24	12	8
Stage-1	Radix-2	16	8	8
Total		56	28	24

3. Results and Observations

Figure 6 shows RTL view of Mixed radix FFT before and after optimization. The Verilog code for the 16-point low-power FFT processor is written and arranged using mixed radix-2 and radix-4 algorithms as part of the RTL design implementation. The RTL explains the data flow through the control logic, twiddle factor multipliers, and butterfly units. The proper FFT operation is then confirmed by simulating this RTL code in Vivado. After simulation, Vivado synthesizes and implements the RTL design onto the FPGA, generating the final hardware structure for the 16-point FFT processor.



(a)



(b)

Figure 6: (a) RTL view of Mixed radix FFT before & **(b)** after optimization

Functionality Analysis

In fig 3.2 there is an image depicting the simulation waveforms for a 16-Point Fast Fourier Transform (FFT) using a mixed radix, stage-wise approach, which has been validated using Xilinx Vivado. The above-mentioned waveforms demonstrate how data samples progress through stages in an FFT process, emphasizing radix transform functions in butterflies. The highest signals represent the clock signals as well as the control signals, which manage the FFT computation timing. Following groups of signals represent the real as well as imaginary parts of the input data at various stages of pipelining. Each pipelining stage performs partial FFT, in which butterfly processing elements combine input samples using twiddle factors.

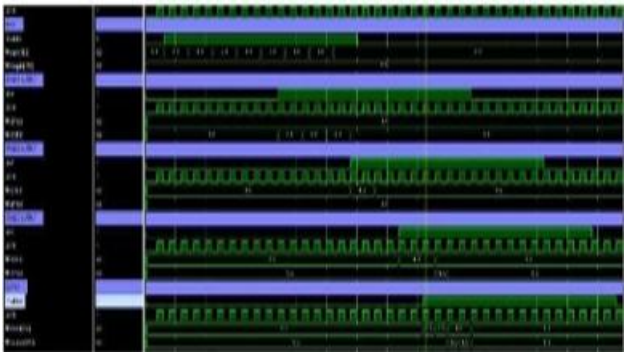


Figure 7: 16-Point FFT mixed radix stage wise using vivado

Using mixed radix helps in the reduction of the complexity of operations by doing operations of multiple radices in stages, thus making it faster than the radix-2 method. The output from the stages, shown in the waveform, verifies the process of the data pass and execution of the butterfly stages. These stable transitions and the expected output patterns in the final stage confirm the functional validity of the 16-point FFT architecture. However, the purpose of the second simulation example in the experimentation section was to showcase the functionality of the FFT module design in performing the fast Fourier transform operation.

Performance Analysis

The design demonstrates a significant decrease in switching activity by combining radix-2/4/2 mixed-radix computation with low-power strategies like clock gating, FSM grey encoding, and RAM-based storage.

Table 4: Comparison of power

S. No	FFT-Architecture	Points	Algorithm	Power(mw)
1	Proposed work: mixed radix	16	Radix 2/4/2	3.18184
2	[11] Pipelined Architecture	16	Radix-2 Radix-4	29.49 26.27
3	[10] Pipelined FET Processor	16	Radix-4 Radix-8	3.66 19.80

The mixed-radix architecture lowers dynamic power to a quiet high percent, while FSM grey encoding decreases power efficiency to 30.27%, according to cadence-based simulation and power analysis. All things considered, the design achieves both substantial power reduction and functional correctness.

4. Conclusion

This paper successfully demonstrates the implementation as well as analysis of a low power 16-point FFT processor with Radix-2/4 butterfly cells. Removal of redundant components in the design and improvement in data flow are the two most important factors towards which the optimization is aimed. Based on the analysis, the proposed design not only preserves a high degree of computation accuracy but also shows a noticeable power savings of 30.27% over the traditional Radix-2 processor.

Although the Radix-2 architecture always supports flexible data processing, the Radix-4 butterfly helps to result in smaller dynamic power consumption, as well as a reduced number of multiplications. From the above implementation outcomes, it can be concluded that the Radix-2/Radix-4 FFT processor supports a portable DSP implementation.

References

[1] L. P. Thakare and A. Y. Deshmukh, "Area Efficient FFT/IFFT Processor Design for MIMO OFDM System in Wireless Communication," Int. Conf. Emerg. Trends Eng. Technol. ICETET, vol. 2016-March, pp. 10–13, 2016.
[2] Cooley, J. W., & Tukey, J. W. (1965). "An algorithm for the machine calculation of complex Fourier series." *Mathematics of Computation*, 19(90), 297–301.
[3] Dr. R. Poovendran and R. Moulika "Low Power and Efficiency Pipelined FFT Processor for OFDM Communication System"- 2020
[4] M. Garrido, K. K. Parhi and J. Grajal, A Pipelined FFT Architecture for Real-Valued Signals, IEEE Transactions on Circuits and Systems –I: Regular Papers, Vol.56, No.12, December 2009
[5] K. Sreekanth, V. Charishma and Neelima koppala, Design and simulation of 64 point FFT using Radix 4 algorithm for FPGA Implementation, International Journal of Engineering Trends and Technology-Volume 4 Issue 2- 2013.
[6] N. Sulaiman, "Design of a reconfigurable FFT processor using Multi-objective Genetic Algorithm," 2010 Int. Conf. Intell. Adv. Syst., pp. 1–5, Jun. 2010
[7] Anirban Ganguly, and Ayan Banerjee "VLSI Design of Analog DFT Processor for Demodulation of QAM-OFDM Signal"- 2019
[8] P. Augusta Sophy, R. Srinivasan "Analysis and Design of Low Power Radix-4 FFT Processor using Pipelined Architecture" -2021
[9] Ngoc Le Ba, Tony Tae-Hyoung Kim "An Area Efficient 1024-Point Low Power Radix-22 FFT Processor With Feed-Forward Multiple Delay Commutators", -2018
[10] Siti Lailatul Mohd Hassan "Pipelined Fast Fourier Transform (FFT) Processor, Power Optimization" -2019
[11] Shafiqul Hai and Tella Rajashekhar Reddy "FPGA implementation of an Image Classifier using Pipelined FFT Architecture" -2025
[12] V. Sarada and T. Vigneswaran "Low Power 64 Point FFT Processor"- 2016
[13] Kevin H. Viglianco, Daniel R. Garcia, James J.W. Kunst "Implementation of a 4-Parallel 128 Point Radix-8 FFT Processor via Folding Transformation" -2023