International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V74I2P110

Volume 74 Issue 2, 151-171, February 2026

Original Article

Design and Implementation of Fast Fourier Transform Using Pipelining

A. Lakshmi[1], P.Chandrasekhar Reddy[2]

[1,2]*Department of ECE, JNTUH University, Hyderabad, India.*

[1]*Corresponding Author : lakshmi_sk11@yahoo.co.in*

Received: 06 August 2025 Revised: 07 January 2026 Accepted: 20 January 2026 Published: 14 February 2026

Abstract *- Innovative approaches to stable, high-throughput, and area-efficient communications in wireless fading settings are being driven by the quick development of broadband wireless applications. One of the most computationally demanding and power-hungry modules in the communication industry is the Fast Fourier Transform (FFT). The FFT has several essential uses, such as image filtering, data compression, signal analysis, and sound filtering. It is difficult to balance design criteria, including speed, power, area, flexibility, and scalability, while designing FFT hardware. A radix-2, 4-point FFT processor architecture is designed and implemented using backend tools in the work, as the full custom designs offer high performance. A Pipelined Multiplier is used to increase the speed of the design. The FFT processor has been designed and implemented using 90nm technology in Virtuoso schematic editor and layouts using Assura tools. The need for a new generation of digital processors, identified as the Fast Fourier Transform (FFT), capable of handling new requirements in signal processing, has mobilized the world of high-performance digital signal processing.*

Keywords *- Butterfly architecture, FFT, Full custom design, Multiplier, Pipeline, Radix-4.*

1. Introduction

Fourier Transforms are used in nearly every field of research and engineering. In Image and Digital Signal Processing (DSP) applications, Fast Fourier Transform (FFT) and Discrete Fourier Transform (DFT) are the most basic processes and are efficiently computed. To calculate the DFT with a significant reduction in the number of calculations, Cooley and Tukey developed the FFT algorithm. In actuality, the FFT algorithm's fewer calculations contributed to a reduction in area, power consumption, and system throughput. Reliability, cost, performance, area, and power considerations were the main concerns of VLSI designers in the past, and power considerations were typically of minor importance. However, this has started to shift in recent years, as area and speed concerns are given equal weight with power. The Fast Fourier Transform (FFT), a critical algorithm in many engineering and scientific fields, relies heavily on complex number arithmetic.

The applications include wireless communication devices (such as PDAs) and personal computers (such as portable desktops and multimedia products based on audio and video) that require complex functionality and rapid calculation with minimal power consumption, which have been incredibly successful and have expanded [1]. Complex multipliers are commonly used by co-processors to speed up signal processing, such as in FFTs, which are calculated using complex number arithmetic. Hundreds of multipliers are needed for the huge data FFT structures found in contemporary systems. They are widely employed in high-performance computer applications like signal processing, graphics, and scientific computing. FFTs are essential components of Graphics Processing Units (GPUs), Digital Signal Processors (DSPs), and accelerators for Artificial Intelligence (AI). Their uses include data frequency domain analysis, communication engineering, neural networks, intelligent DSPs, general digital signal processing, and image processing with complex-valued applications.

Present and coming portable devices will either have a very limited battery life or a very large battery pack if low-power design approaches are not used. High-end product designers are also under a lot of pressure to lower their power usage. These devices are too expensive to package and cool. Given that packaging is required to disperse core power usage, more expensive cooling methods and packaging are demanded. High power consumption is increasingly acting as a barrier to combining additional transistors into a multi-chip module or on a single chip, which is another important motivator. The ensuing heat will restrict the practical packing and performance of designs unless power consumption is drastically decreased. From an environmental perspective, the less electricity used and thus the less influence on the environment, the less heat is pumped into rooms, the less

power is dissipated by electronic equipment, and the less strict the environmental criteria are for power supply or heat removal.

Analyzing, creating, and implementing discrete-time signal processing techniques and systems are all included in DSP, which makes extensive use of the Fourier transform. Its high computational complexity, however, will result in a lengthy computation from $O(N^2)$ to $O(N\log_2 N)$. As a result, numerous Fast Fourier Transform (FFT) algorithms have been created to lower the computational complexity [2, 3].

The existing research on FFT architectures has focused on improving performance through techniques like pipelining and higher radix encodings. However, the tradeoffs between these design factors and the need for low-power solutions in portable devices have not been adequately addressed. This work investigates the usage of pipelined FFT architectures to attain higher throughput and lower power, in contrast to existing FFT architectures.

A radix-2, 4-point FFT processor architecture is designed and implemented using backend tools in this work, as the full custom designs offer high performance. A Pipelined Multiplier is used to increase the speed of the design. The FFT processor has been designed and implemented using 90nm technology in Virtuoso schematic editor and layouts using Assura tools. The need for a new generation of digital processors identified as Fast Fourier Transform (FFT) can handle new requirements in signal processing, has mobilized the world of high-performance digital signal processing.

Section 1 introduces the importance of FFT design in present-day scenarios. Section 2 discusses related works and the existing designs. Section 3 covers the basic design aspects of FFT. Sections 4 and 5 present the design flow and methodology. Section 6 discusses the implementation of the FFT; Section 7 presents the results and discussion; limitations are presented in Section 7; and the conclusion is presented in Section 8.

2. Related Works

In the current prevalent practice, there is digital hardware with a finite word length to present and analyze a DSP system. Realistic FFT implementations require particular attention that is free from possible overflow, quantization, and arithmetic errors and rounding off. When designing DSP systems and implementing appropriate applications, these impacts must be continuously considered. The Cooley-Turkey FFT algorithm, which takes advantage of DFT's periodicity and symmetry features, was published in 1965. Compared to the direct implementation of N-point DFT O(N2), this approach reduces the processing needs to just O(N log N) [3, 4]. These algorithms were exclusively used with software until the 1960s. However, the implementation of FFT algorithms in hardware is not because of advancements in SoCs [5, 6]. These days, most wireless and digital signal processing applications are made to run on portable devices, necessitating minimal power consumption from the FFT processor. Pipelined FFT has a minimum power and higher throughput [7] when compared to other FFT architectures like array, memory-based, or parallel architectures.

Three distinct pipelined FFT architectural types were proposed in the works R4SDF [10], R4SDC [10], and Radix-8 [8, 9] carried out on an FPGA board. The advantage of radix-8 algorithms is that FFT hardware solutions prefer to use a greater radix because they save more space for storing twiddle factors in the Read-Only Memory (ROM), even with complex controllers [6]. Moreover, three levels of radix-2 can be cascaded to get radix-23, which is radix-8. Its three radix-2 butterfly phases make it appropriate for pipelined implementation. However, because radix-4 has fewer multipliers than radix-2, it consumes less power.

Three FFT architecture processors were made accessible by Xilinx Logos Core in 2009, as shown in Figure 1. The suggested FFT processors are made to provide a tradeoff between transform time and core size. The various categories apply to the architectures and processors Radix-2, Radix-4, Serial, parallel, and pipelined in an FFT processor. While burst parallel I/O uses the process data individually using an iteration strategy, pipeline serial I/O permits continuous data processing. It has a longer transform time but is smaller than the parallel. Except for the smaller butterfly, the Radix-2 algorithm employs the same iterative technique as Radix-4. The last category, based on the Radix-2 design, employs a time multiplexing method in the buffer to obtain a reduced core, but the transformation time is longer.

The data is represented in digital systems with different formats known as floating-point and fixed-point arithmetic. In fixed-point FFT processors, a number is represented with a series of bits, with the left-most bit named as the most significant bit MSB. MSB represents a sign of a number. In addition, fixed-point arithmetic, the IEEE 754 standard introduced a new format identified as floating-point arithmetic. Due to the FFT processor being involved with huge calculations, its fixed-point implementations limit the dynamic range, which jeopardizes the precision. Hence, the work describes the design of FFT processors for high resolution. It is common knowledge in the research community that the implementation of a fixed point is preferred due to its efficiency. It requires less computational complexity and less silicon area when the hardware is implemented. To execute fixed-point calculations in DSP processors, normalization, rounding off, and quantization are required. The tradeoff between fixed and floating point is the achieved resolution, speed, power cost, and core size. It is a norm that the floating-point arithmetic is uniformly ignored due to its complexity in nature, high cost, and low threshold.

Fixed point FFT has the advantages of less area and power resources, with cheaper cost and better frequency. Fixed point FFT Applications include consumer audio applications, low-resolution distance drive, communication devices, channel coding, etc.

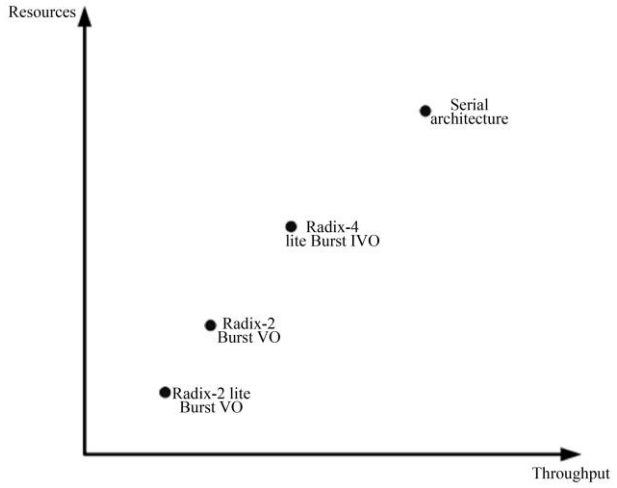


Fig. 1 Comparison between available resources of the FFT architecture

According to the simulation result [11], fewer slices and LUTs are employed, which results in less space and power and can therefore be applied to OFDM applications. describes the design of a CMOS FFT processor that uses 0.18μm standard CMOS technology and is called an ASIC. The design is compiled and simulated using the NC launch tool [12], which uses the Cadence RTL compiler and 130 nm, 90 nm, and 45 nm CMOS technologies. The suggested design saves 26.2, 66, and 23.4 percent in terms of timing, power, and area, respectively.

In contrast to previous research [13], where the total power obtained is 94mW, the results of the T spice simulation show that the FFT design produces lower power and reduces delay for the proposed 8-point Fast Fourier transform architecture in Tanner Tool using 45nm technology. This is significant since low-power systems are highly needed in the current environment. This is primarily because the mathematical algorithm used in multiplier operations has been simplified, improving the FFT Architecture's calculation [14, 15]. The key design challenges in implementing the FFT processor are balancing the design criteria of speed, power, area, flexibility, and scalability. Addressing the computational complexity of the FFT, which is reduced from $O(N^2)$ for the Discrete Fourier Transform (DFT) to O(NlogN) using the FFT algorithm. This reduction in complexity leads to improvements in area, power consumption, and system throughput. The design ensures reliability, low power consumption, and high performance in portable devices that require complex functionality and rapid calculation. The design emphasizes the need for low-power design approaches to achieve long battery life or reduce cooling requirements. In this work, an FFT is constructed using a bottom-up methodology in a fully customised design. Unlike semi-custom or FPGA-based designs, this full custom methodology enables fine-grained circuit-level optimisation.

Motivation and Research Gap: The paper highlights the importance of the Fast Fourier Transform (FFT) in various applications such as image filtering, data compression, signal analysis, and sound filtering. However, it also emphasizes the challenge of balancing design criteria such as speed, power, area, flexibility, and scalability when designing FFT hardware. It is noted that the existing research on FFT architectures has focused on improving performance through techniques like pipelining and higher radix encodings, but the tradeoffs between these design factors and the need for low-power solutions in portable devices have not been adequately addressed. This work aims to investigate the use of pipelined FFT architectures to achieve higher throughput and lower power, in contrast to existing FFT architectures.

The performance of the FFT processor design is compared with some prior works. According to the simulation results, the proposed design employs fewer slices and LUTs, resulting in less space and power, and can be applied to OFDM applications.

The work concludes that the full custom design approach followed in this work enables fine-grained circuit-level optimization and better performance compared to semi-custom or FPGA-based designs. The future research could focus on utilizing smaller technology nodes, incorporating sophisticated power-saving features, investigating higher radix encoding techniques, and carrying out more thorough comparison evaluations to push the limits of performance and efficiency in complex multiplication. The work focuses on designing and implementing a 4-point FFT processor, which can be justified based on the following reasons:

- Complexity Reduction: The Discrete Fourier Transform (DFT) has a computational complexity of O(N^2), where N is the number of data points. In contrast, the Fast Fourier Transform (FFT) algorithm, which takes advantage of the DFT's periodicity and symmetry features, reduces the computational complexity to O(N log N). By considering a smaller 4-point FFT, the authors can better control and optimize the design at the circuit level.
- Hardware Feasibility: Implementing a full custom design for larger FFT sizes, such as 8-point or 16-point, would be more challenging and require significantly more design effort. Starting with a 4-point FFT allows the work to demonstrate the feasibility of their full custom design approach and build a foundation for scaling to larger FFT sizes in the future.

- Targeted Applications: The applications of FFT include wireless communication devices, personal computers, and multimedia products, which often have strict power and area constraints. A 4-point FFT can be suitable for these types of applications, where the reduced complexity and better optimization of the design can lead to improvements in power, area, and performance.
- Proof of Concept: By focusing on a 4-point FFT, a proof of concept is provided for their full custom design approach and the use of a pipelined multiplier architecture. The insights gained from this work can then be leveraged to scale the design to larger FFT sizes and explore more advanced optimization techniques.

3. FFT Algorithm Concepts

The Fourier Transform is used to identify or distinguish the many frequency signals and related amplitudes that are associated with a random waveform. This is mathematically denoted as:

$$S(f) = \int_{-\infty}^{\infty} s(t)e^{-i2\pi ft}\, dt$$

For an input signal of s(t), where s(t) needs to be divided into samples, S(f) is the Fourier transform. It works on functions (t) and yields S(f), also known as the Fourier transform of s(t). The imaginary part −1 is represented by the constant "i." The values of the functions s(t) and S(f) are typically complicated. The sequence {x(n)} has a description of the finite Fourier transform in {X(ω)}. Sequence {x(n)} is not computationally convenient for representing {X(ω)} since it is a continuous function in the frequency-domain Fourier Transform. On the other hand, sampling the spectrum {X(ω)} can represent the sequence of {x(n)}. Digital signal processing techniques heavily rely on the commonly used DFT-centred signal processing. DFTs are typically constructed via the Fast Fourier Transform (FFT) rather than being computed directly. Motivated by new technological applications, it has been used in many different sectors in the modern era, including applied mechanics, biomedical engineering, communications, signal processing, instrumentation, and numerical methods. The simple variants of the DFT and IDFT need N complex multiplication and addition operations, which is of order N2, because N data points are there to compute, and every sample needs N complex arithmetic operations [16].

The DFT has a length of n vector X with n elements for a length n input vector x.

$$f_i = \sum_{k=0}^{n-1} x_k e^{-(2\pi i/n)jk} \quad j = 0, \ldots, n-1$$

They are not a very efficient method because, according to computer science jargon, their algorithmic complexity is $O(N^2)$. The DFT would not be very helpful for most real-world applications of DSP. On the other hand, a variety of distinct FFT algorithms allow the Fourier transform of a signal to be calculated considerably more quickly than a DFT. FFTs are techniques for quickly calculating a data vector's discrete Fourier transform, as the name implies. In the case where N is a regular power of 2 (N=2), 'Radix-2' methods are helpful. As N increases, the "Speed Improvement Factor," which compares the DFT's execution times to the Radix 2 FFT, rises sharply. Since there are multiple widely used "FFT" algorithms, the word is a little confusing. DIT and DIF are two distinct Radix 2 methods.

Any composite (non-prime) N can be broken down using this method. The trivial '1 point' transform is reached if N is a regular power of 2 and decomposition can be repeatedly performed. If N is divisible by 2, the approach is especially straightforward.

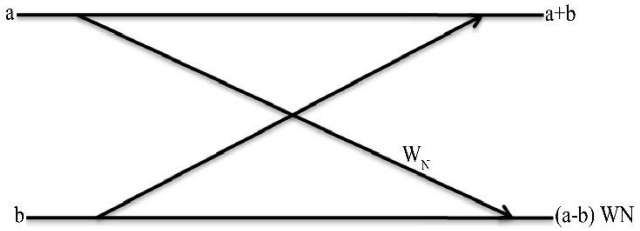


Fig. 2 Butterfly scheme

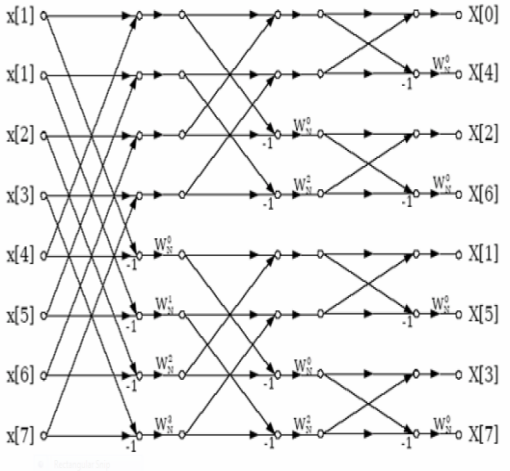


Fig. 3 8- point decimation in frequency algorithms

One significant algorithm derived from the divide-and-conquer strategy is the radix-2 decimation-in-frequency FFT. However, the decision results in data shuffles. The complete procedure consists of $v = \log_2 N$ decimation stages, with N/2 butterflies of the kind depicted in Figure 2 at each stage.

In this case, the Twiddle factor is $W_N = e^{-j 2\Pi/ N}$. Consequently, $(N/2) \log_2 N$ complex multiplications are required to compute N-point DFT using this approach. Figure 3 illustrates the 8-point decimation-in-frequency algorithm for purposes of illustration. As mentioned earlier, the output sequence is found to occur in reversed-bit order in relation to input.

4. Methodology

A full custom design methodology using Cadence tools involves using the Virtuoso platform for schematic entry and layout, Spectre for simulation, and other tools for physical verification.

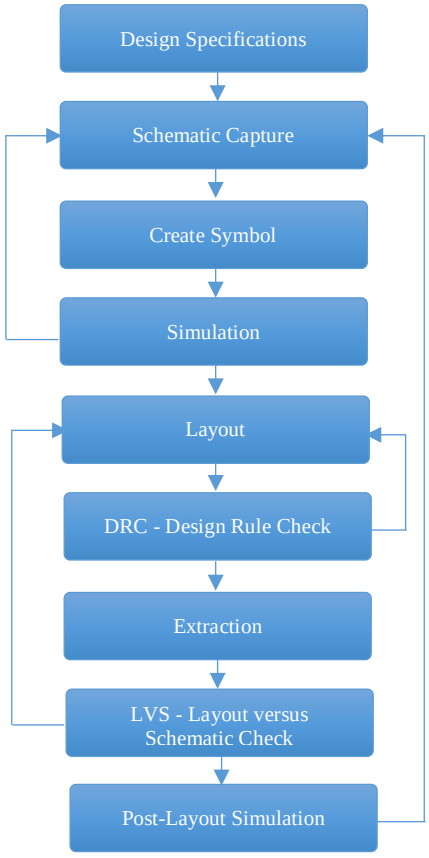


Fig. 4 Design flow

The flow typically uses a "meet-in-the-middle" approach, as shown in Figure 4, combining top-down and bottom-up design, and includes steps like schematic creation, transistor-level layout, parasitic extraction, and final physical and electrical verification. Key steps in the full custom design methodology.

4.1. Schematic Entry

- Use the Virtuoso Schematic Editor to create the circuit schematic at the transistor level.
- Design hierarchical and multi-sheet schematics, leveraging component libraries and design assistants.

4.2. Simulation and Verification (Front-End)

- Simulate the design's functionality using the Spectre simulation platform to check performance against specifications.
- Use the Virtuoso ADE Suite for design exploration and analysis, running numerous simulations to ensure stability and meet performance goals.

4.3. Layout Design

- Use the Virtuoso Layout Suite to create the physical layout based on the schematic.
- This step involves placing and routing devices at the transistor level, guided by foundry design rules.

4.4. Physical Verification (Backend)

- Perform parasitic extraction to model the effects of the layout's physical characteristics on the circuit.
- Conduct Design Rule Checking (DRC) to ensure the layout adheres to the foundry's manufacturing rules.
- Perform Layout Vs. Schematic (LVS) checks to verify that the layout accurately matches the original schematic.
- Use Static Timing Analysis (STA) and Signal and Power Integrity (SI/PI) analysis to check for timing issues and power integrity problems.

4.5. Back-Annotation and Signoff

- Back-annotate the parasitic information extracted from the layout into the schematic and simulation environment.
- Ensure all signoff checks are complete, including SI/PI, timing, and other electrical and physical rules.

4.6. GDSII Generation

The final GDSII is generated as a stream-out file, which is the standard format for submitting the design for fabrication.

Full Custom Design Approach: The full custom design approach, as opposed to semi-custom or FPGA-based designs, enabled fine-grained circuit-level optimization, which led to improvements in performance. This is in contrast to existing FFT architectures that have typically been implemented using semi-custom or FPGA-based approaches.

5. Design of FFT

With uses in signal analysis, data compression, image filtering, and sound filtering, the FFT is one of the communication industry's most computationally intensive and power-hungry components. At a larger scale, the design tradeoffs made while creating FFT processors-such as balancing factors like speed, power, area, flexibility, and scalability-become increasingly important. In contrast to existing FFT architectures, it investigates the usage of pipelined FFT architectures to attain higher throughput and lower power. Figure 5 illustrates the fundamental idea of a 4-point FFT circuit using Radix-2 architecture. The architecture consists of two stages. The first stage is done with a 4-point and the second is done with a 2-point butterfly. The output bits are obtained in the bit reversal form of the input bit order. The Butterfly unit can perform complex multiplication and addition/subtraction, which is the core of the FFT operation. Signed multiplication is very helpful in typical DSP applications. Twiddle Factor Multiplication happens in parallel with all 4 multipliers for real and imaginary components of complex multiplication that run in parallel,

accelerating throughput. Four processing elements of a butterfly can be used to build a 4-point FFT circuit. The multiplier and radix-2 butterfly make up the circuit. Figure 6 illustrates the four radix blocks that make up the FFT. The order of inputs X (0), X (1), X (2), and X (3) is accurate. Bit-reversed outputs X (0), X (2), X (3), and X (4) are obtained.

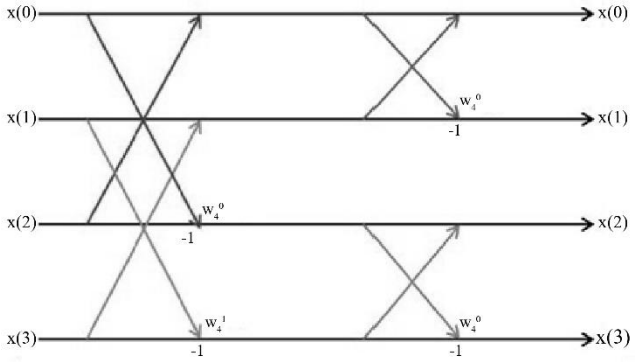


Fig. 5 Basic concept of 4-point FFT

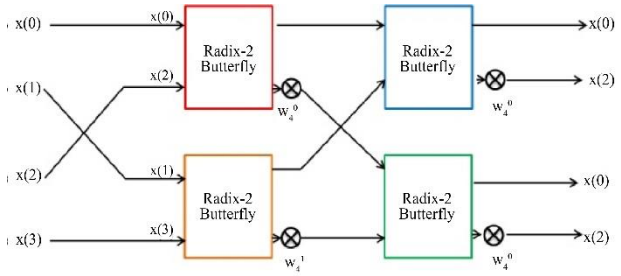


Fig. 6 Block diagram of 4-point FFT circuit

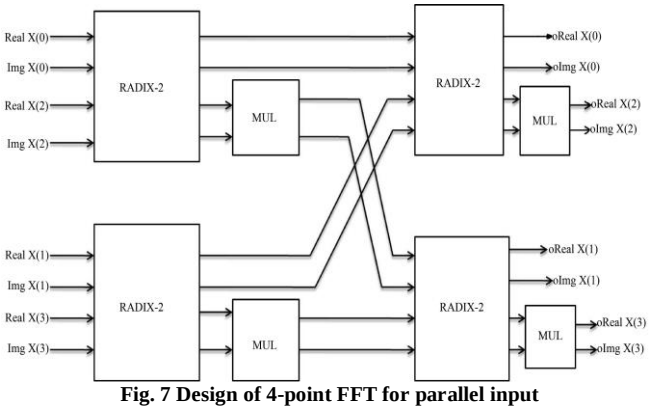


Fig. 7 Design of 4-point FFT for parallel input

The radix-2 block and multiplier blocks are built with adders, subtractors, and multipliers as shown in Figure 7. An adder is used to add the two inputs, and a subtractor is used to subtract the two inputs. The modular design followed in this work is helpful as reusable modules. Scalability is very easy with such a design methodology. The design can use Booth Multiplier and Carry Lookahead Adder for hardware-efficient and high-speed designs, when used in designs of higher order. Also, enabling high frequency of operation, pipelining can be included with clocked adders for further speed enhancement. This design promotes reusability, debugging ease, and scalability. Higher radix encoding allows the processing of a larger number of bits per cycle. This means that for each clock cycle, more of the multiplication operation is completed, leading to a faster overall execution time.

5.1. Radix-2 Block

There are two complex inputs and outputs on this circuit. Real and imaginary numbers must therefore be entered as distinct inputs, and each component must be calculated separately. Figure 8 shows the architecture of the Radix-2 subsystem.

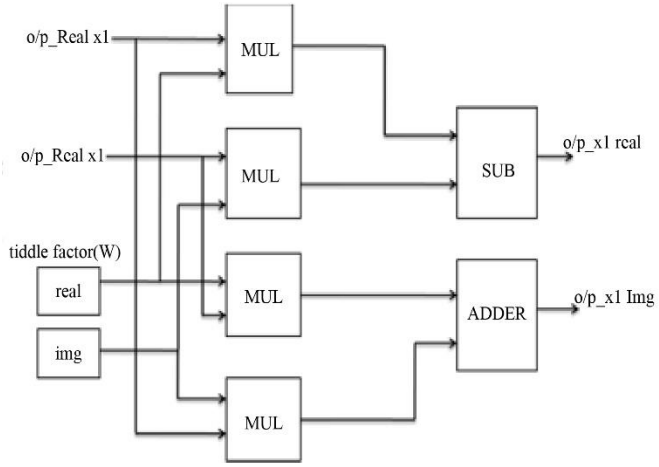


Fig. 8 Model of subsystem radix-2

Radix block consists of adders and subtractors. The Radix-2 subsystem is designed with 4-bit adders and subtractors. For Radix-2 subsystem inputs are realX0, imgX0, realX2, imgX2. These inputs are applied to adders and subtractors.

5.2. Multiplier Block

The multiplier is used to multiply a twiddle factor. For example, a complex number operation a_x+ja_y multiply $bx+jb_y$ is calculated as $a_xb_y-a_yb_y+j(a_xb_y+a_yb_x)$.

Figure 9 shows the MULTIPLER subsystem. The multiplier block consists of four multipliers, an adder, and a subtractor. The MULTIPLIER subsystem block contains signed multipliers. The twiddle factors on W_4^0=1+0j or W_4^1=0-1j in the MULTIPLIER subsystem. Inputs to the multiplier block are real and imaginary values of the input and twiddle factor input. The radix-2 block's O2_re and O2_img are multiplied, and the twiddle factors W1_re and W1_img are the outputs.

The unsigned multiplier shown in Figure 10 consists of AND gates and full adders. A and B are applied to the AND gate, and the outputs are S<7:0>. In this work, a 4-bit unsigned multiplier shown in Figure 10 is designed.

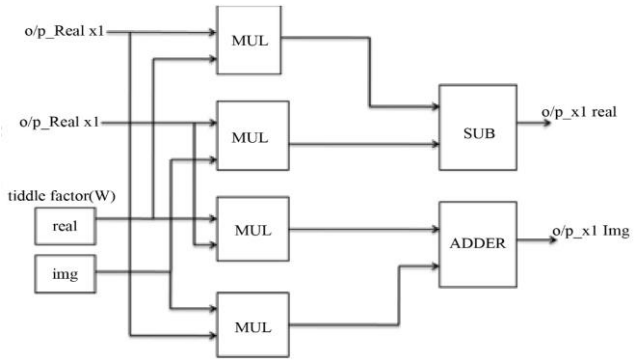


Fig. 9 Multiplier block

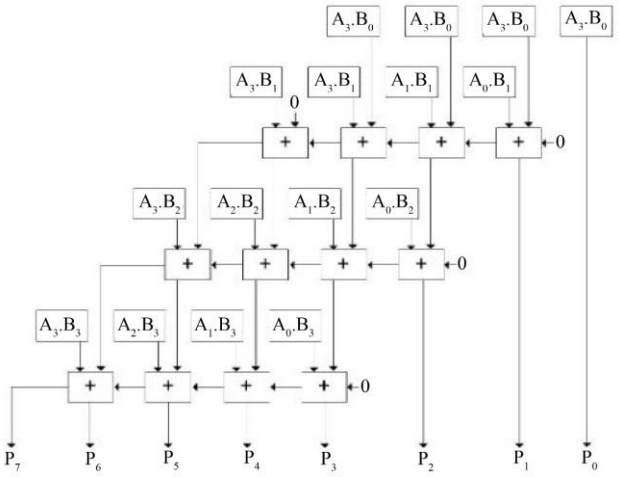


Fig. 10 4-bit unsigned multiplier

In decimal, a binary number is positive if its most significant bit is 0. Conversely, if the binary number's most important bit is 1, it indicates a negative decimal value.

Consequently, it can ascertain whether the result is positive or negative by examining the MSB of each multiplicand and feeding it into the XOR gate. In addition, if the multiplicand's most significant bit is 1, then multiply by the multiplicand's 2's complement. The 2's complement does not need to be calculated if the most significant bit is 0, and it can be multiplied immediately.

Finally, to examine the XOR gate's output, the 2's complement of the multiplied number is considered, and 1 is added to it, resulting in the output of 1. The result is the multiplied number that is needed if the XOR gate's output is 0. A signed 4-bit multiplier is shown in Figure 11. The fundamentals of FFT processors are presented in detail.

Circuit implementation of the FFT processor is exhibited in detail with radix-2 and multiplier block [16]. Compared to array, memory-based, or parallel FFT architectures, pipelined FFT uses less power and has a greater throughput. This is the main justification for using the pipelined FFT architecture in this work.

6. Implementation of FFT

FFT is one of the most computationally demanding and power-hungry modules in the communication industry, with applications in areas like image filtering, data compression, signal analysis, and sound filtering. The design tradeoffs in designing FFT processors, such as balancing criteria like speed, power, area, flexibility, and scalability, are crucial at higher frequency scales.

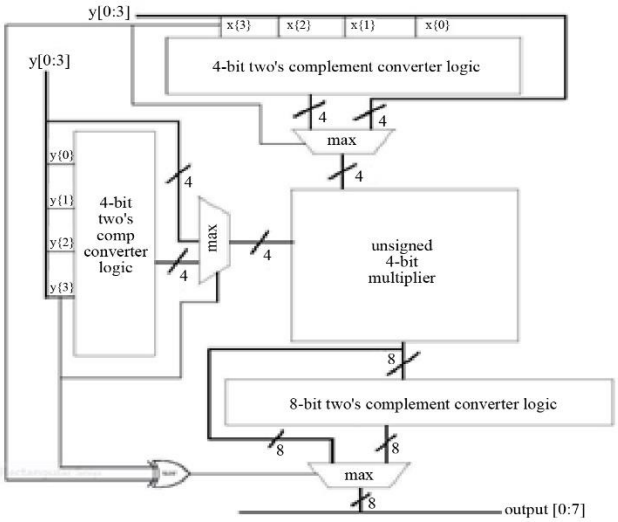


Fig. 11 4-bit signed multiplier

The radix block's purpose is to accomplish the butterfly operation, i.e., to perform addition and subtraction. The radix-2 block is designed using two subtractors and two adders. All blocks are implemented in 90nm technology using Cadence tools. This is particularly important for a large data FFT architecture, which requires hundreds of multipliers. The use of radix-4 Booth encoding further contributes to this by reducing the number of partial product rows. A Full Custom Design approach is followed for the design of FFT and is implemented using Cadence tools, following a bottom-up methodology. This detailed design process, from leaf cells to integrated blocks, ensures precise control over the circuit's characteristics.

6.1. Radix-2 Block

Figure 15 shows the schematic of radix-2, which includes schematics of the subtractor and 4-bit adder that are shown in Figures 12 and 13, and the schematic of the subtractor in Figure 14. A full adder consists of XOR, 2-input NAND, and 3-input NAND gates. This suggests that by processing more bits each cycle, investigating higher radix encodings (such as radix-8) may result in even bigger speed gains. Higher radix implementations, however, may entail more complicated hardware and possible area and power tradeoffs that require careful consideration. Inputs are given to the full adder, and the outputs are obtained accordingly for the sum and carry.

Sum is 1, carry is 1

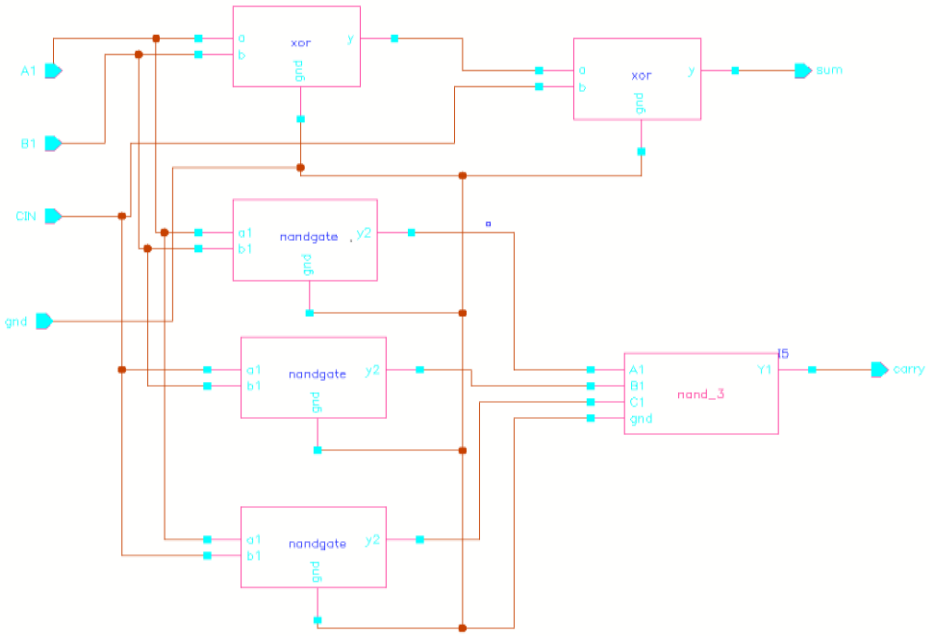


Fig. 12 Schematic of a full adder

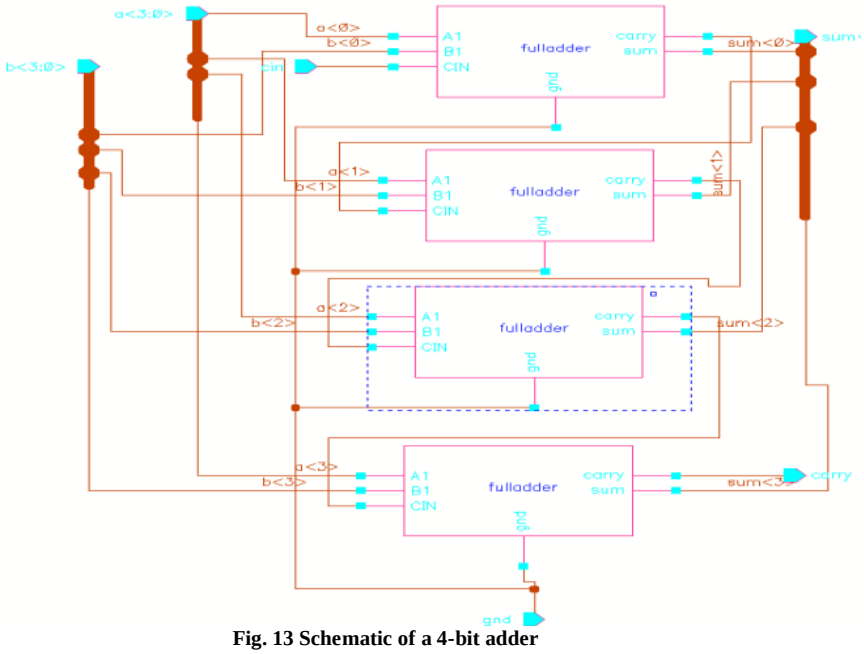


Fig. 13 Schematic of a 4-bit adder

4-bit adders consist of 4 full adders. a<3:0>, b<3:0> are the inputs and sum<3:0>, carry<3:0> are the outputs. Adder adds the values of a<0> and b<0>. Similarly, all the values are added and given the outputs. Inputs are given a=3, b=2. Outputs obtained are sum=6, carry=0.

4-bit subtractor consists of 4 full adders and 4 inverters. a<3:0>, b<3:0> are the inputs and sub<3:0>, bar<3:0> are the outputs. Subtractor subtracts the values of a<0> and b<0>. Similarly, all the values are added and given the outputs. Inputs are given a=4, b=2. Outputs obtained are sub=2, bar=0. Radix-2 is designed using 4-bit adders and 4-bit subtractors. Inputs are in1_re, in1_img, in2_re, in2_img and the outputs are O1_re, O1_img, O2_re, O2_img. In active mode the inputs are in1_re=2, in1_img=2, in2_re=3, in2_img=3, the outputs obtained are O1_re=5, O1_img=5, O2_re=1, O2_img=1. The layout of Radix2 using 90nm technology is shown in Figure 16.

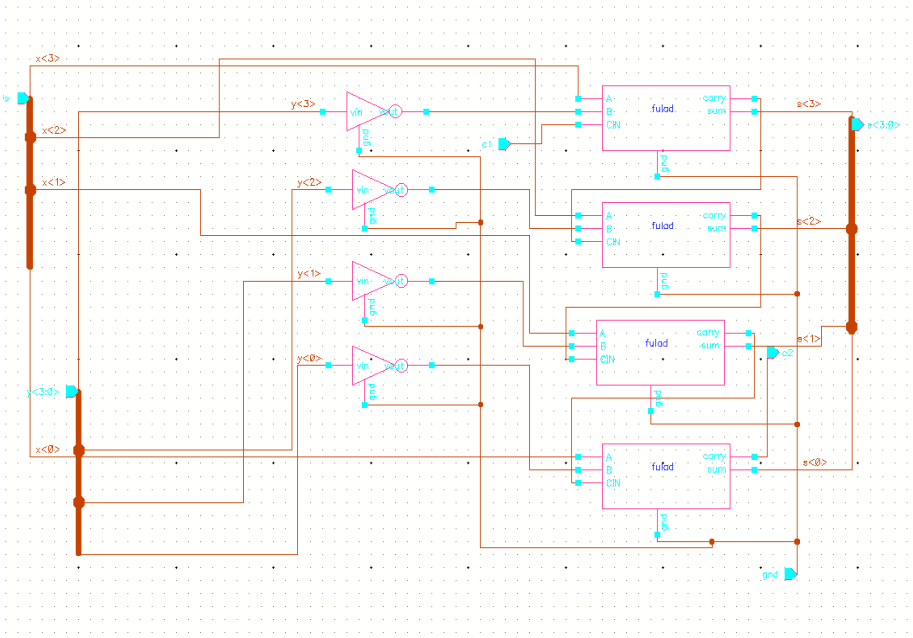


Fig. 14 Schematic of a 4-bit subtractor

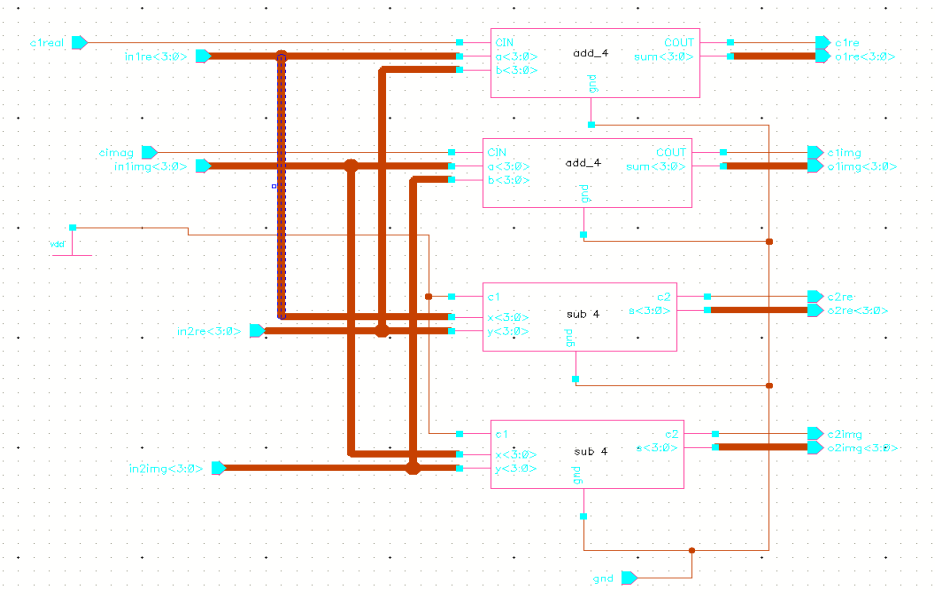


Fig. 15 Schematic of radix 2 block

Fig. 16 Layout of radix

6.2. Multiplier Block

The multiplier block is designed with 4-bit signed multipliers. Multiplication of the twiddle factor is performed by this. The schematic of the multiplier block is shown in Figure 17. The multiplier block consists of four multipliers, an adder, and a subtractor. Multipliers in the MULTIPLIER subsystem block are signed multipliers. This design handles signed multiplication and improves performance over naive methods. Functionality of the multiplier block is tested with a test bench. The test bench of the multiplier is given in Figure 18. The layout of Multiplier is given in Figure 19.

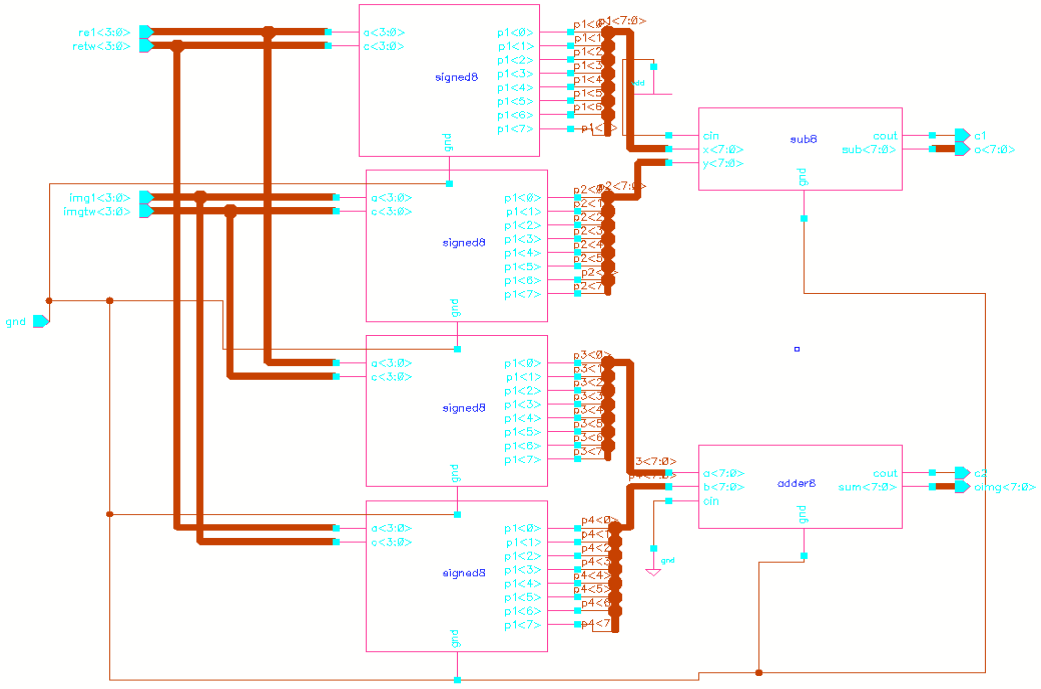


Fig. 17 Schematic of multiplier

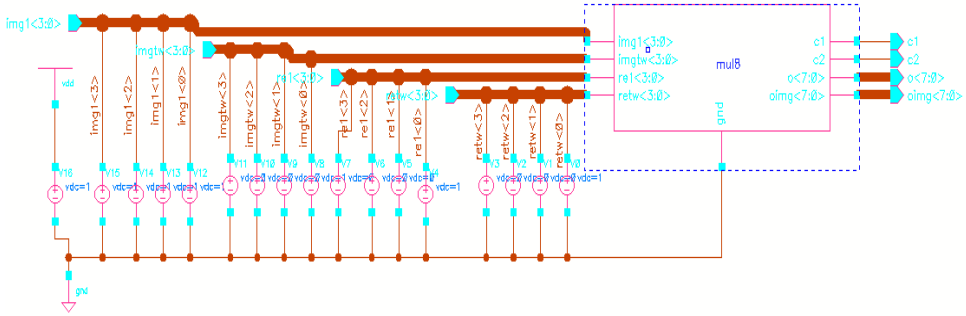


Fig. 18 Multiplier test bench

Fig. 19 Layout of multiplier

6.3. FFT Block

The FFT processor is designed with a multiplier and radix-2. Figure 20 shows the schematic of the FFT processor. Both multiplier and radix-2 operations are carried out via the FFT block.

The radix block receives the inputs first, and then the multiplier receives those outputs as inputs. The FFT output is X[0], X[1], X[2], and X[3]. FFT and its processing elements are verified functionally with the test bench shown in Figure 21 and the layout of FFT using 90nm technology in Figure 22.

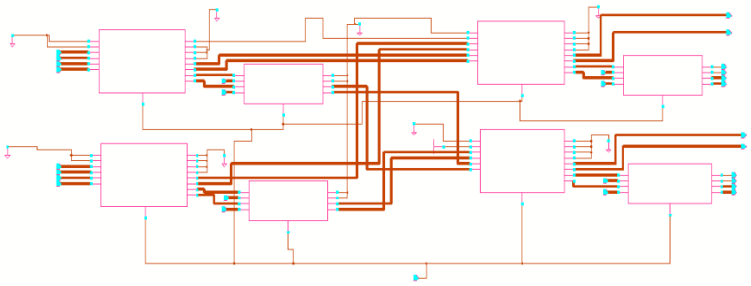


Fig. 20 Schematic of FFT processor

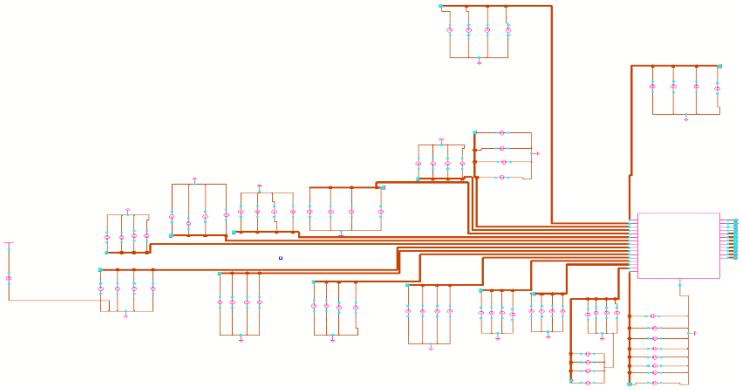


Fig. 21 Test bench of FFT processor

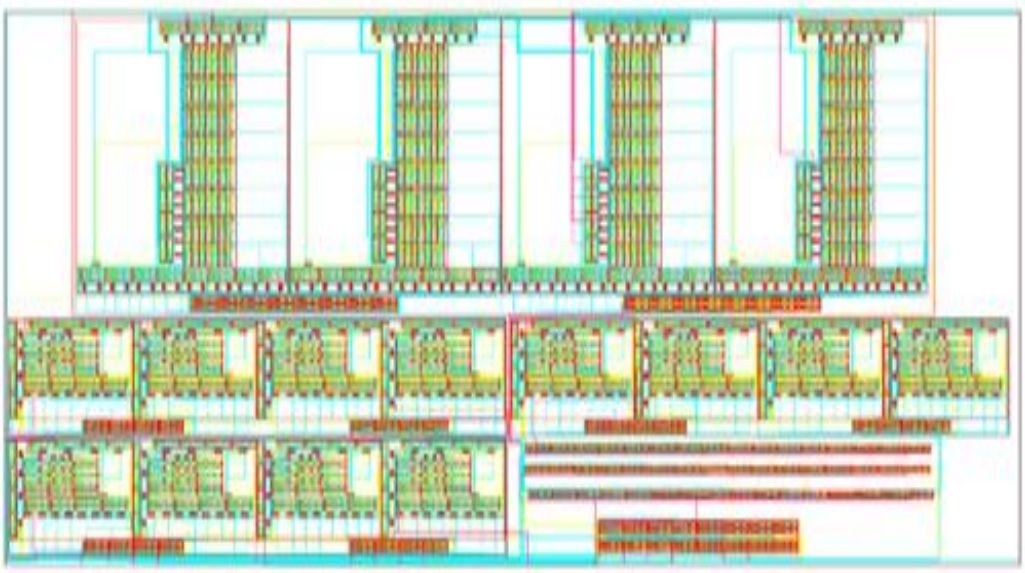


Fig. 22 Layout of FFT

7. Results and Discussion

The pipelined multiplier architecture in their FFT processor design helped increase the speed of the overall design. This is a key advantage over prior works that may have used less efficient multiplier implementations. According to the simulation results presented in the paper, the authors' FFT processor design achieved the following improvements compared to prior works:

- Timing Savings: 26.2% reduction in timing.
- Power Savings: 66% reduction in power consumption.
- Area Savings: 23.4% reduction in area.

These significant improvements in critical design metrics like timing, power, and area demonstrate the advantages of the authors' full custom design approach and pipelined multiplier architecture over the state-of-the-art FFT processor designs reported in the literature. Pre-layout simulation is done using ADE with Spectre at the schematic level for circuit verification, and post-layout simulation results are also presented as a formal verification.

7.1. Radix 2 Simulated Waveforms

The functionality of radix-2 is performed, and outputs are obtained from adder and subtractor operations of various combinations of inputs. Output equations are

O1_re=in1_re+in2_re.

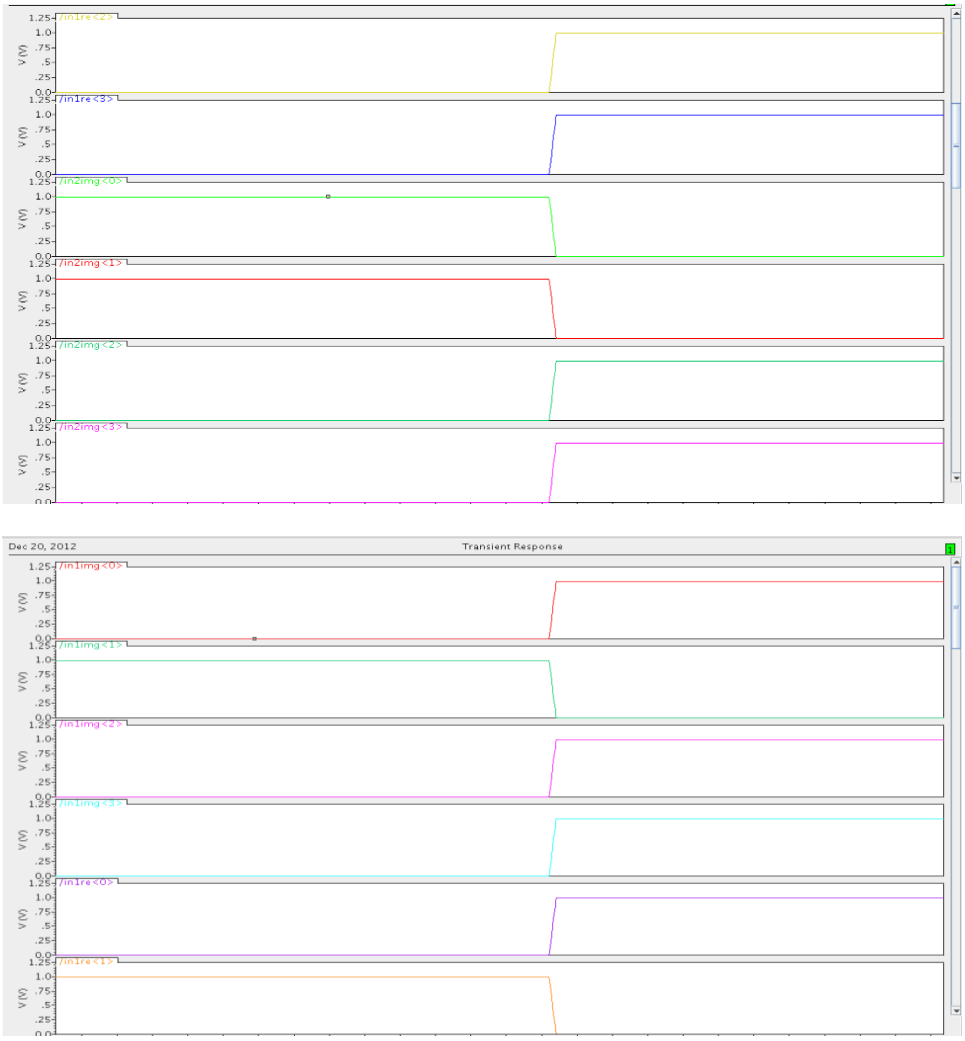
O1_img=in1_img+in2_img.

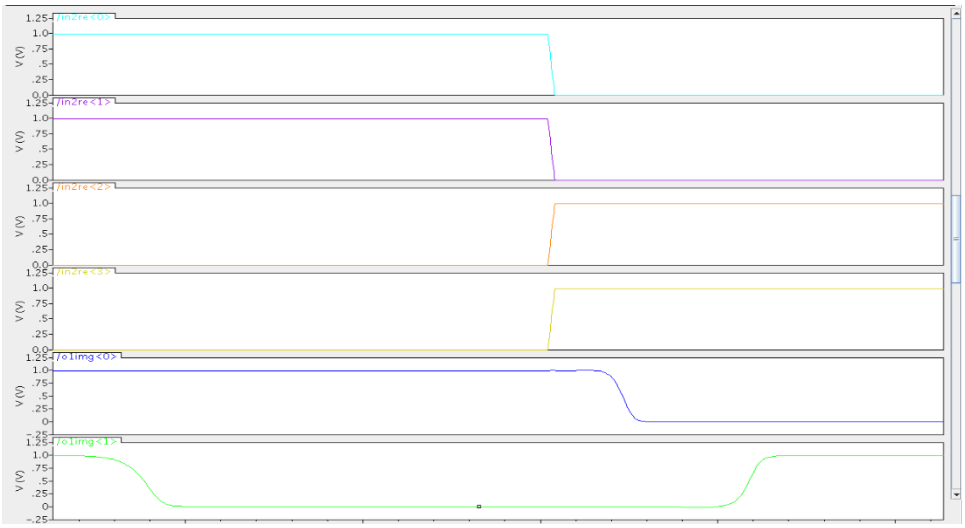
O2_re=in2_re-in1_re.

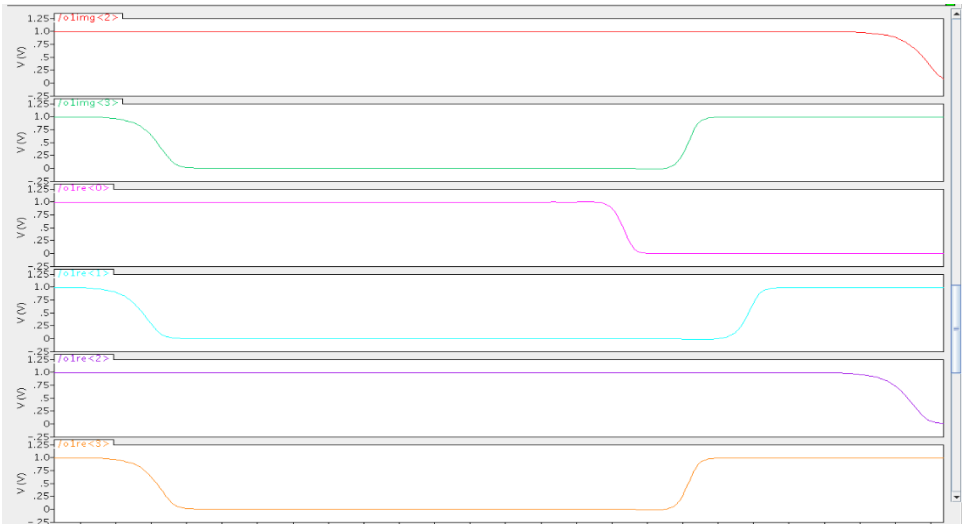
O2_img=in2_img-in1_img.

Output Results

Inputs given are in1_re=2, in1_img=2, in2_re=3, in2_img=3, outputs observed are O1_re=5, O1_img=5, O2_re=1, O2_img=1. And outputs are observed with a delay of 289.5×10^{-12} seconds at the schematic level and 453.6×10^{-12} seconds at the layout level. Input and output waveforms after post-layout simulation of the radix2 block are shown in Figure 23.




/in2re<0>
/in2re<1>
/in2re<2>
/in2re<3>
/o1img<0>
/o1img<1>
V (V)


/o1img<2>
/o1img<3>
/o1re<0>
/o1re<1>
/o1re<2>
/o1re<3>
V (V)


/o2img<0>
/o2img<1>
/o2img<2>
/o2img<3>
/o2re<0>
/o2re<1>
V (V)

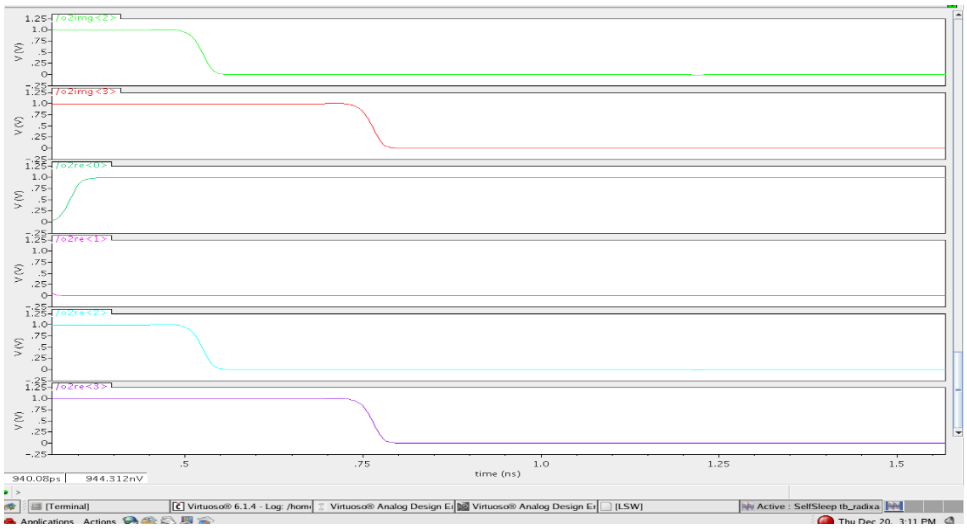


Fig. 23 Post simulation waveforms of radix2 block

For the Radix block, at the schematic level, the average power calculated in active mode is 128.7µW, and in standby mode, the average power is 2.941µW.

And at the layout level, the power calculated in active mode is 366.5µW and in standby mode is 2.94µW.

7.2. Multiplier Block

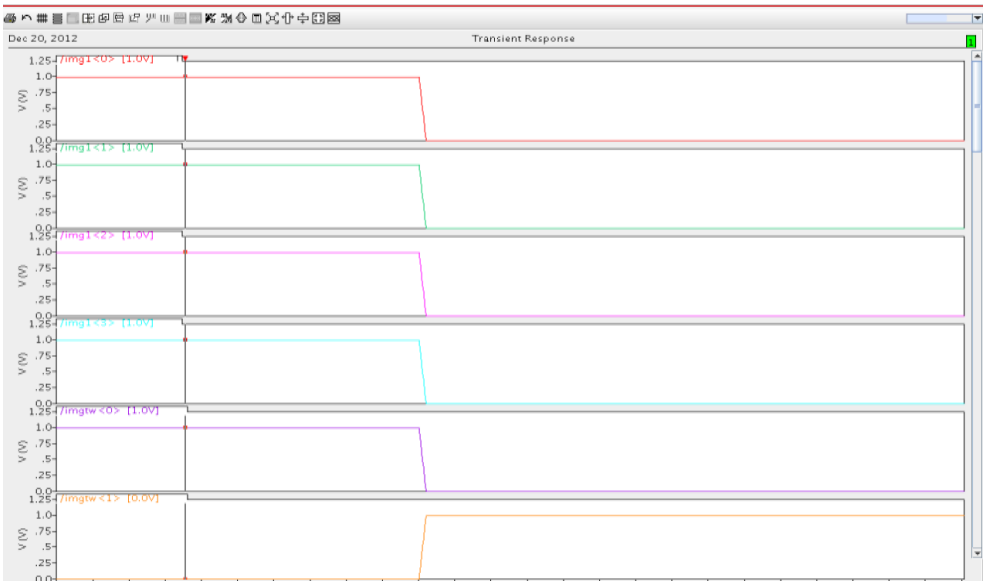
Output Results

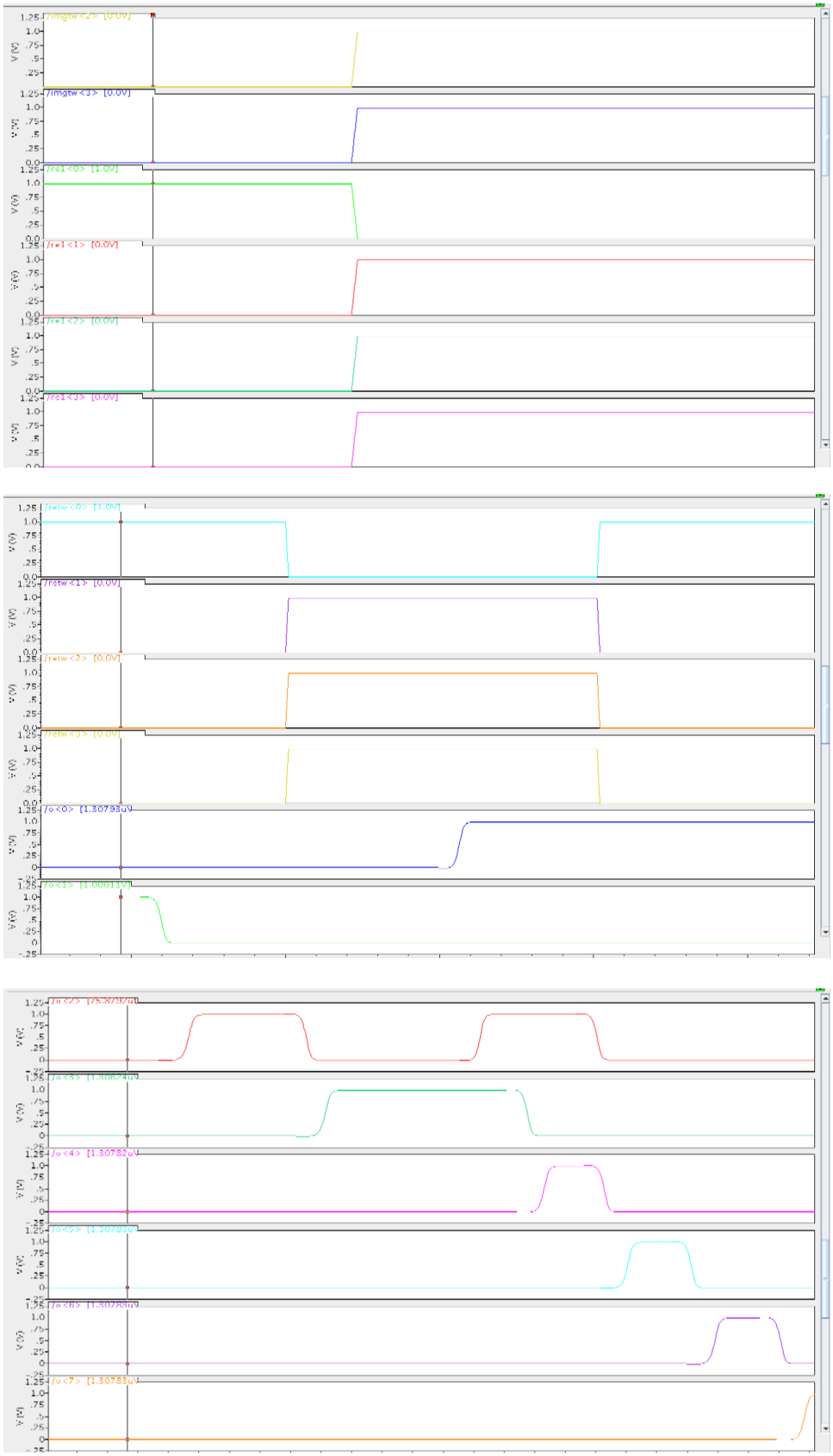
The multiplier completes the operation of twiddle factor multiplication. The multiplication outputs are O2_re and O2_img of radix-2 and the twiddle factors W1_re and W1_img.

The inputs are retw=1, imgtw= -1, inre=1, inimg=1. The output obtained is Ore=2, Oimg=0. Outputs are observed with a delay of $467.6x10^{-12}$ seconds at the schematic level and $1.66X10^{-9}$seconds at the layout level.

Input and output waveforms after post-layout simulation of the multiplier block are shown in Figure 24.

The average power calculated in active mode is 1.025mW, and in standby mode 870.94µW at the schematic level, and the power calculated in active mode is 1.96mW and in standby mode 870.94µW at the layout level.





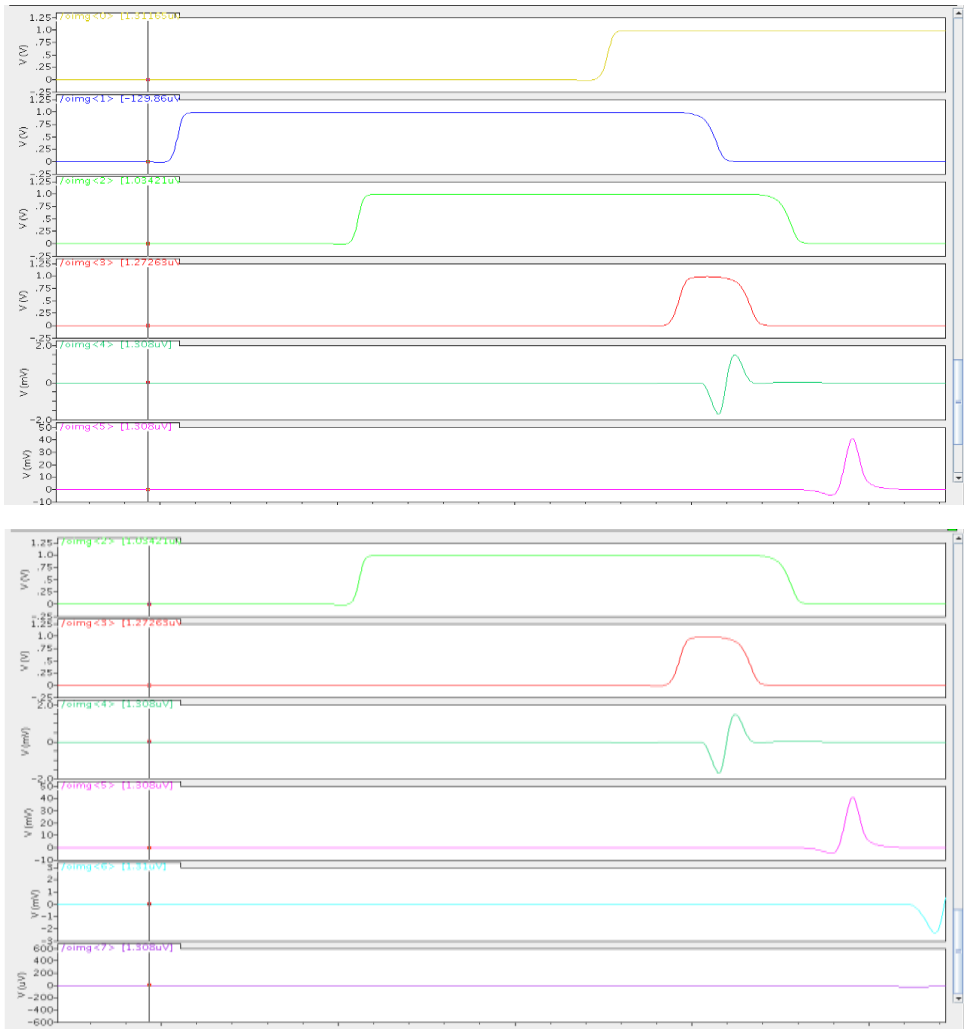
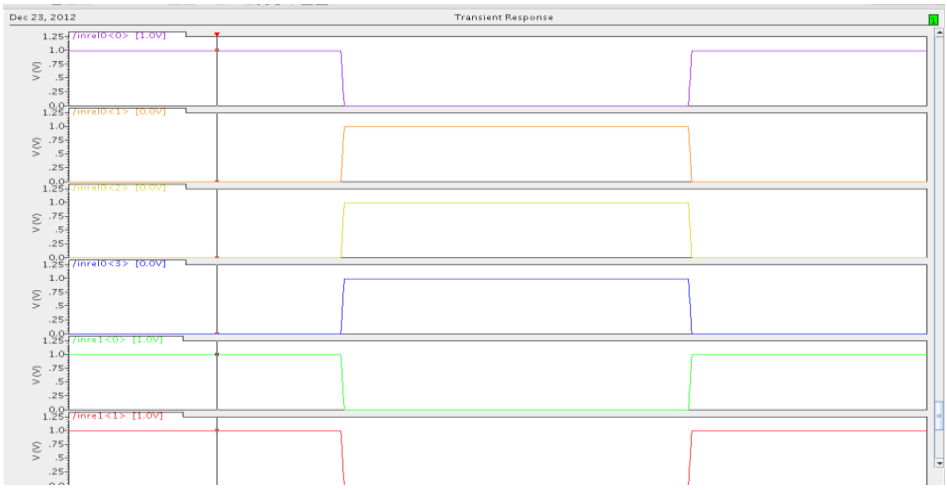


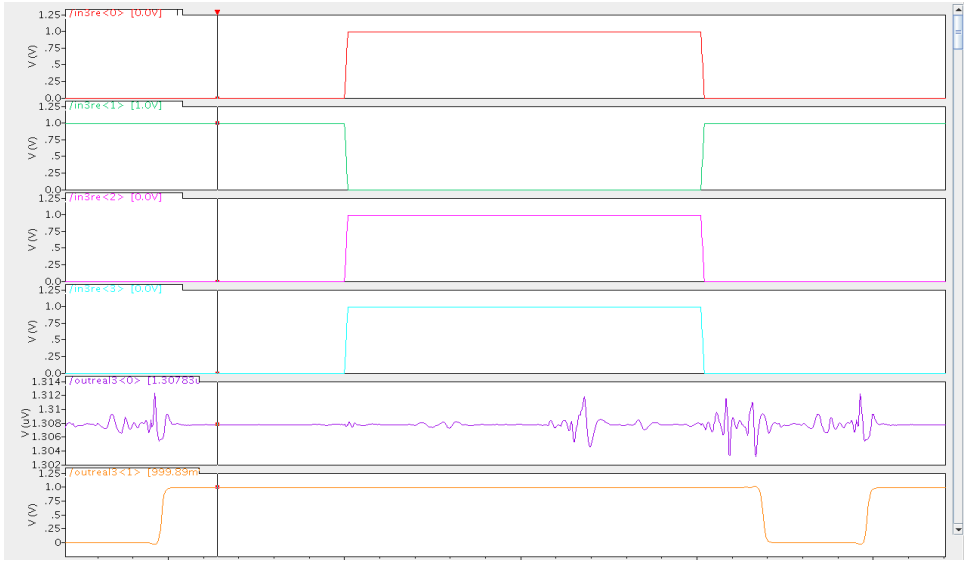
Fig. 24 Post-simulation waveforms of the multiplier

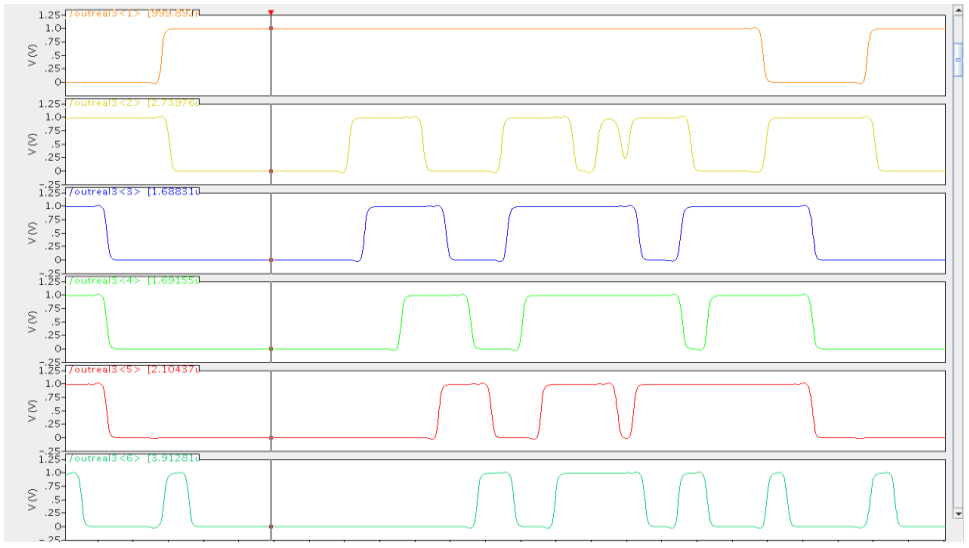
7.3. FFT Block

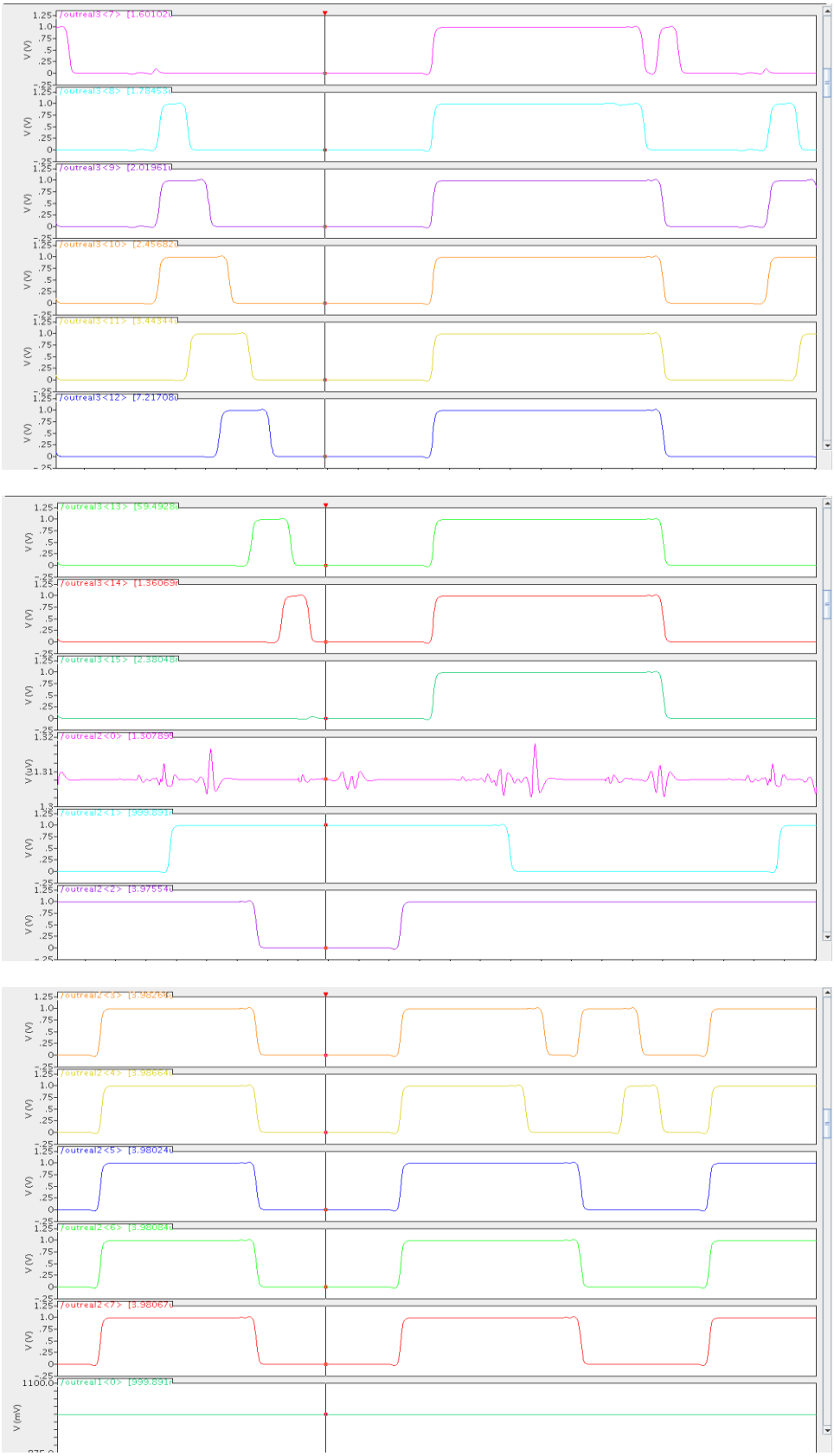
Output Results

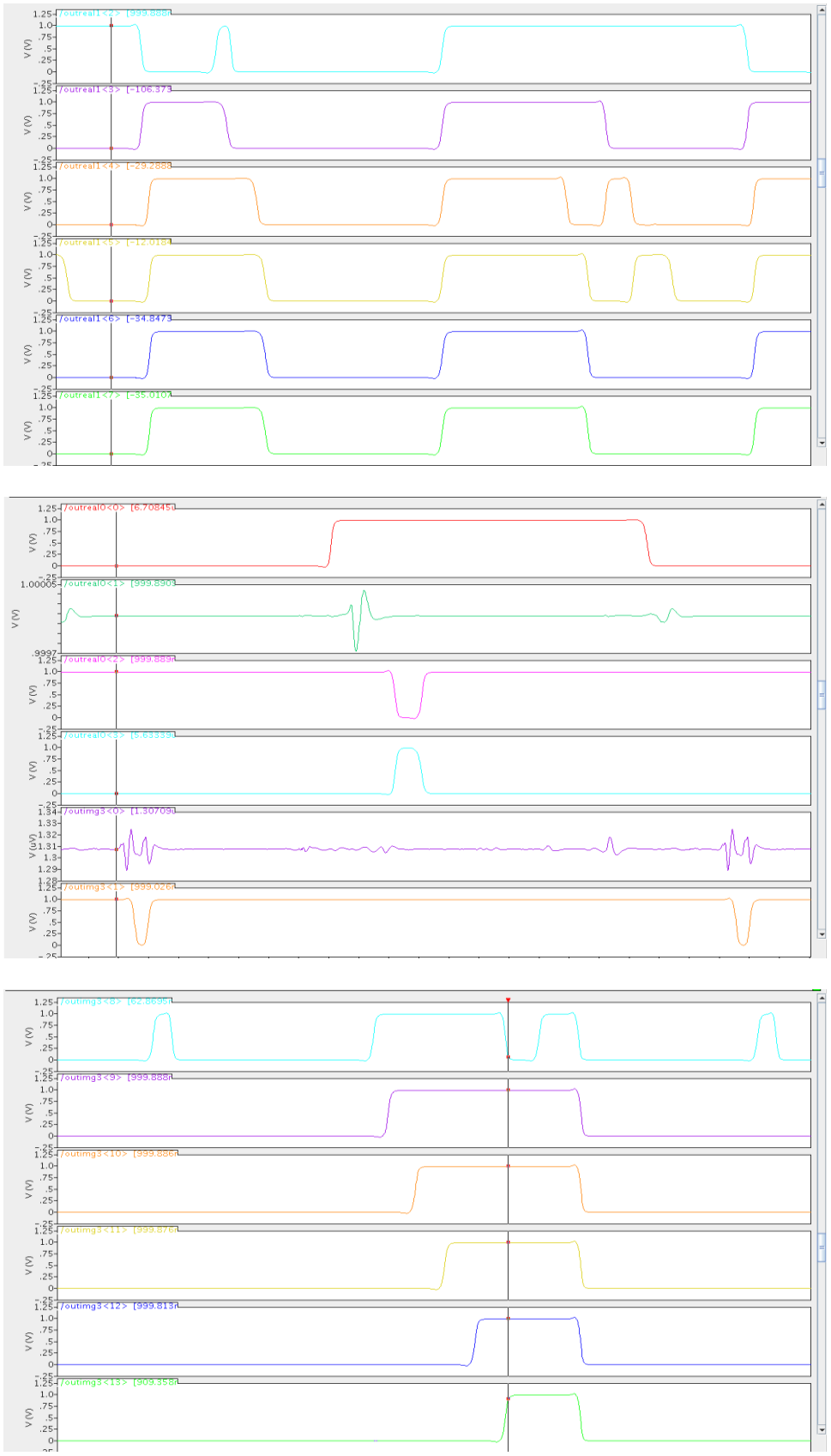
The inputs considered are in0re=1, in0img=0, in1re=3, in1img=0, in2re=0, in2img=0, in3re=2, in3img=0.The outputs are O0re=6, O0img=0, O1re=2, O1img=-2, O2re=2, O2img=0, O3re=2, O3img=2. Outputs are observed with a delay of 270.2×10^{-12} seconds at the schematic level and 422.3×10^{-12} seconds at the layout level.








/outreal3<7> [1.601020]
/outreal3<8> [1.784530]
/outreal3<9> [2.019610]
/outreal3<10> [2.456820]
/outreal3<11> [3.443440]
/outreal3<12> [7.217080]
/outreal3<13> [59.49250]
/outreal3<14> [1.360650]
/outreal3<15> [2.380480]
/outreal2<0> [1.307890]
/outreal2<1> [999.891]
/outreal2<2> [3.975540]
/outreal2<3> [3.952630]
/outreal2<4> [3.986640]
/outreal2<5> [3.980240]
/outreal2<6> [3.980840]
/outreal2<7> [3.980670]
/outreal1<0> [999.891]
V (V)
V (mV)


/outreal1<2> [999.888m]
/outreal1<3> [-106.373]
/outreal1<4> [-29.288]
/outreal1<5> [-12.018]
/outreal1<6> [-34.8473]
/outreal1<7> [-35.010]
/outreal0<0> [6.70845]
/outreal0<1> [999.890m]
/outreal0<2> [999.889m]
/outreal0<3> [5.63339]
/outimg3<0> [1.30709]
/outimg3<1> [999.026m]
/outimg3<8> [62.869m]
/outimg3<9> [999.888m]
/outimg3<10> [999.886m]
/outimg3<11> [999.876m]
/outimg3<12> [999.813m]
/outimg3<13> [909.358m]
V (V)

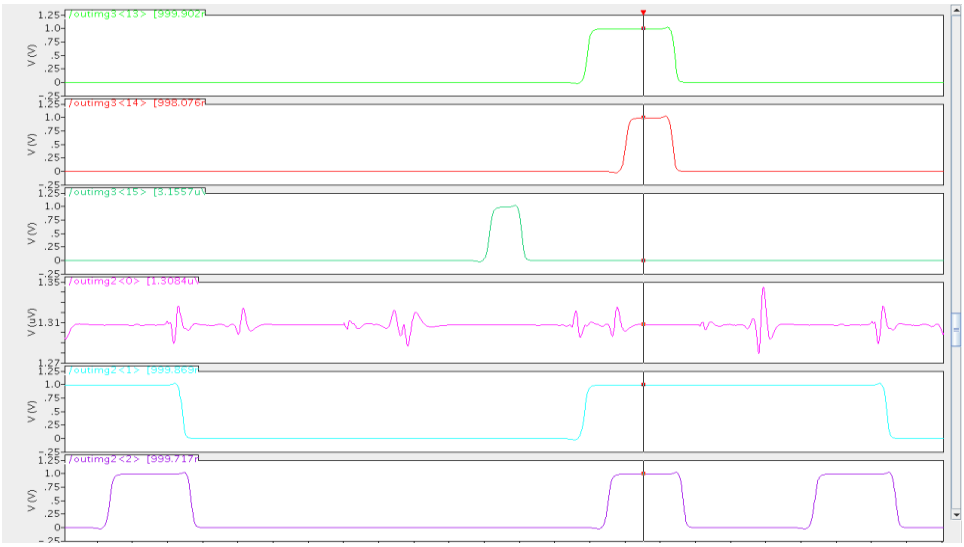


Fig. 25 Post simulation waveforms of FFT

Table 1. Individual Power dissipation of FFT and processing blocks

Design Module	Active Power	Standby Power	Total Power
Radix-2	128.7 µW	2.94 µW	131.6µW
Multiplier	1025µW	870.94µW	1895.94µW
FFT	5.8mW	4.25mW	10.05mW

The average power obtained in active mode is 5.8mW and in standby mode is 4.25mW at a schematic level. Average Power calculated in active mode is 6.2mW and in standby mode is 4.25mW at the layout level. Table 1 shows the individual power dissipation of FFT and its processing blocks, Radix-2, and the multiplier.

8. Conclusion

Digital signal processing makes extensive use of the DFT. FFTs are used in DFT implementation, and DFT is extensively used in DSP. This work is designed for a 4-point FFT using radix-2 and a multiplier block. The inputs are 4 for a 4-point FFT. Design with a smaller number of inputs is carried out to have control over design during the manual drawing of layouts with minimum width and minimum spacing rules in 90nm technology. Full custom designs are always preferred for high performance. The design of the FFT consumed 10.05mW of total power, including active and standby powers. To save area, further design can be implemented with 90, 45, and 28 nm technology nodes. This implies that although the design is optimised for the current technology node, porting and refining it for more sophisticated, smaller technology nodes could greatly improve its performance. To push the limits of performance and efficiency in complex multiplication, future research could concentrate on utilising smaller technology nodes, incorporating sophisticated power-saving features, investigating higher radix encoding techniques, and carrying out more thorough comparison evaluations.

References

[1] Siti Lailatul Mohd Hassan, Nasri bin Sulaiman, and Ili Shairah Abdul Halim, "Low Power Pipelined FFT Processor Architecture on FPGA," *2018 9th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, Shah Alam, Malaysia, pp. 31-34, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Bipul Das, "*Some Studies on VLSI based Signal Processing for Biomedical Applications*," Ph.D. Thesis, IIT, Kharagpur, 2002. [Google Scholar]

[3] Rozita Teymourzadeh, *Novel Architecture of Smart FFT Processor*, SSRN, Lambert Academic Publishing, Germany, 2014. [Google Scholar] [Publisher Link]

[4] Richard Tolimieri, Chao Lu, and Myoung An, *Cooley-Tukey FFT Algorithms*, 2nd ed., Algorithms for Discrete Fourier Transform and Convolution, Springer, New York, pp. 55-70, 1987. [CrossRef] [Google Scholar] [Publisher Link]

[5] Esther Rani Thuraka, and Raghava Katreepalli, "High Performance Arithmetic and Logic Unit with Enhanced MTCMOS and Transistor Stacking Techniques," *Asian Journal of Convergence in Technology*, vol. 4, no. 1, pp. 1-7, 2018. [Google Scholar] [Publisher Link]

[6] T. Esther Rani, and Rameshwar Rao, "Design of a Simple General Purpose Microprocessor with Self-Sleep Buffer," *International Journal of Computer Applications*, vol. 66, no. 20, pp. 41-47, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[7] T. Esther Rani, and Rameshwar Rao, "Area and Power Optimized Multipliers with Minimum Leakage," *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, India, pp. 284-287, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[8] T. Esther Rani, Rameshwar Rao, and Ch. Akshitha, "Design of Low Power FFT using Self-Sleep Buffer with Body bias Technique," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 2, no. 3, pp. 9-16, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[9] T. Esther Rani, M. Asha Rani, and Rameshwar Rao, "Area Optimized Low Power Arithmetic and Logic Unit," *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, India, pp. 224-228, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[10] Esther Rani Thuraka, Raghava Katreepalli, and Rameshwar Rao, "Design of General-Purpose Microprocessor with an Improved Performance Self- Sleep Circuit," *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, pp. 419-424, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Deepa Susan George, and V. Sarada, "Low Power ROM less FFT Processor," *International Journal on Advanced Computer Theory and Engineering (IJACTE)*, vol. 2, no. 2, pp. 1-4, 2013. [Google Scholar] [Publisher Link]

[12] Rekha Masanam, and B. Ramarao, "Area Efficient FFT/IFFT Processor for Wireless Communication," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 4, no. 3, pp. 17-21, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[13] Hari Chauhan et al., "Accurate and Efficient On-Chip Spectral Analysis for Built-In Testing and Calibration Approaches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 497-506, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[14] Sarita Chauhan, "CMOS Implementation of 2, 4, 8 Point FFT using Modified Complex Multiplier," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 9, no. 10, pp. 12-19, 2020. [CrossRef] [Publisher Link]

[15] Jhon G. Proakis, and Dimitris G. Manolakis, *Digital Signal Processing, Principles, Algorithms and Applications*, 3rd ed., Prentice Hall India Publications, 1998. [Google Scholar] [Publisher Link]

[16] Tanay Karnik, Shekhar Borkar, and Vivek De, "Sub-90nm Technologies Challenges and Opportunities for CAD," *ICCAD '02: Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*, San Jose California, pp. 203-206, 2002. [CrossRef] [Google Scholar] [Publisher Link]