SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I8P104

Volume 12 Issue 8, 40-50, August 2025
© 2025 Seventh Sense Research Group®

Original Article

Full Custom Design and Implementation of 12-Bit Complex Multiplier

A. Lakshmi[1], P. Chandrasekhar Reddy[2], Esther Rani Thuraka[3]

[1,2]Department of ECE, JNTUH University, Hyderabad, India.
[3]Department of ECE, CVR College of Engineering, Hyderabad, India.

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

Received: 02 June 2025 Revised: 04 July 2025 Accepted: 03 August 2025 Published: 30 August 2025

Abstract *- The complex multiplier is an important module used in co-processors, especially designed for signal processing in Graphical Processing Units (GPUs), Digital Signal Processors (DSPs), and certain Artificial Intelligence (AI) accelerators. These applications require a low area and low power. This work presents a novel strategy for complex number multiplication. The design is full custom and utilizes a circuit optimization technique. The complex multiplier is designed using the Bottom-up approach. It uses a radix-4 modified Booth encoder. These concepts are used for performance improvement. The process of multiplication is sped up as the radix-4 modified Booth encoder can decrease the rows of partial products to n/2, and carry-save adders are designed to add the partial products by using a smaller number of transistors to improve the speed of the addition process. Finally, an increase in speed, low power, and low area is achieved by the utilization of a smaller number of transistors overall. Hence, less silicon area is utilized. The design is implemented using Cadence tools for 12x12-bit signed and unsigned numbers and is simulated using ADE with Spectre simulator for both pre-layout and post-layout complex multiplier using 0.18µm technology. Novelty stems from its integrated approach of a new full-custom design strategy, meticulous circuit-level optimization, and the effective application of radix-4 Booth encoding to achieve a highly efficient 12-bit complex multiplier in terms of power, area, and speed.*

Keywords *- Booth encoder, Circuit optimization, Complex multiplier, Full custom design, Low power, Silicon area.*

1. Introduction

High-performance computing tasks like scientific computing, graphics, and signal processing utilize complex multipliers. Complex multiplication is used to compute rotations and scaling in 2D complex space. Complex multipliers are used in co-processors, especially in digital signal processors, GPUs, and certain AI accelerators, to efficiently handle operations involving complex numbers. They are commonly used in general digital signal processing applications and today's intelligent DSPs, neural networks, image processing with complex-valued applications of communication engineering, different types of data frequency domain analysis, and computer vision applications. Co-processors use complex multipliers, such as FFTs, to accelerate signal processing. The Fast Fourier Transform (FFT), which is computed using complex number arithmetic, is used in many of these applications. Every area of engineering and study uses the Fourier methods. In the past, Very Large-Scale Integration designers prioritized cost, performance, power, area, and reliability; power consideration was usually of secondary significance. This has begun to change in recent years, though, as power issues are being given the same consideration as area and speed problems.

The primary contributing factor has been the remarkable success and growth of the field of wireless communication systems (personal communicators and assistants) and personal computing devices (portable desktop computers, multimedia products based on audio and video) that require complex functionality and high-speed computation with low power consumption. In many situations, average power usage becomes an important design factor [1, 2].

If low-power design solutions are not employed, the current and future portable gadgets will either have a pack with a large battery or a battery with limited life. Inbuilt cooling systems become costly during packing and moving. Another significant motivation is the fact that excessive power usage is becoming a barrier more and more. From an environmental point of view, the less electricity is utilized, the less of an impact it has on the environment; the less stringent the environmental standards are for heat removal or power delivery. Improving complex multipliers leads to a better design of FFT architecture [14], as the large data FFT architecture of today's generation requires hundreds of multipliers.

The article describes the "full custom design and implementation of 12-bit complex multiplier". It makes use of circuit optimization techniques, radix-4 modified Booth encoding, and a bottom-up methodology. The main claims are that employing fewer transistors and decreasing partial product rows results in low power, speeding up multiplication, performance improvement, and low area.

2. Background

The importance of complex multipliers in high-performance computing and the changing objectives in VLSI design, especially regarding power consumption and space efficiency, provide the foundation for this work. One essential component for carrying out the multiplication of two complex numbers is a complex multiplier. For many algorithms, notably Fast Fourier Transforms (FFTs), this procedure is essential.

They are widely employed in high-performance computer applications like signal processing, graphics, and scientific computing. Complex multipliers are essential components of GPUs, 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.

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. Changing priorities for VLSI Design, power consumption was frequently a secondary consideration for VLSI designers in the past, who gave priority to cost, performance, area, and reliability. This has altered dramatically, though, with power considerations now being given the same weight as speed and area difficulties.

Complex functionality and high-speed computation with low power consumption are required due to the explosive rise of wireless communication systems and personal computing devices. Current and future portable devices might need larger batteries or have shorter battery lives if low-power design alternatives were not available.

Overuse of power generates a lot of heat, which makes it difficult to package and operate VLSI circuits and systems realistically. Less use of electricity has a smaller environmental impact, according to environmental experts. Two adders and four multipliers are normally needed for a traditional complicated multiplier. A generator of Partial Products (PP), an adder for the produced partial products, typically make up its architecture. Decreasing the partial product count and improving the method for combining them are usually ways to increase the Power, Performance, and Area (PPA) efficiency of multipliers.

Prior research has investigated several methods for implementing complicated multiplication with lower PPA complexity, including the use of vector-merging adders, Booth encoders, and CSA trees. Compressors for partial product decrease and Vedic mathematics are two more methods. Although some previous methods resulted in severe latency and large resource utilization, the research points out that complicated multipliers have been optimized in silicon for DSP applications. In recent years, multiplier design has also focused on modified Booth encoding.

This background lays the groundwork for the innovative design described in this work by highlighting the growing need for effective complex multipliers in a variety of applications, as well as the industry's move towards giving low power and small area in VLSI design priority.

3. Related Works

A traditional Complex multiplier requires 4 multipliers [1-4] and 2 adders. A general multiplier architecture consists of two parts: (i) a partial products generator, and (ii) generated partial products addition. Multiplier Power Performance Area (PPA) efficiency is generally enhanced by decreasing the partial product number and adding partial products. In [5], two Booth encoders, four carry-save adder trees, four vector-merging adders, and a pair of adders/subtractors are used to implement the complex multiplication. The power performance area complexity is less than [3, 6]. Using Vedic mathematics, the real multiplier architecture shown in Figure 1 was used to implement complex multiplication and is proposed in [8, 10].

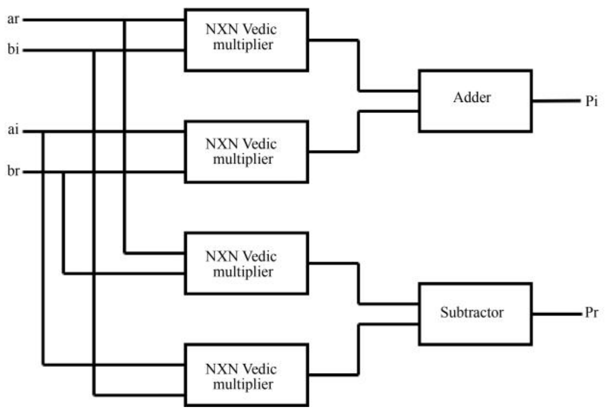

Fig. 1 Multiplier architecture using Vedic mathematics

Signed numbers of real and imaginary parts of complex operands cannot be utilized directly with this. In [9], it is suggested that compressors be used for low-power, high–performance partial products reduction. Complex multiplication is optimized in the hardware for DSP applications [7], and it takes more space than conventional [5]. Implementing a directly general complex multiplier, as is typical in [11-13], results in very high latency, 10416 look-up tables,187mW, and more space taken up by the hardware used for DSP applications. The suggested complex multipliers have lower performance. Output value is represented with 32 bits. The multiplier is the main block in the complex multiplier. Over recent years, the multiplier using modified Booth encoding by using different techniques [15-18] has been developed. They used different circuit-level realizations and implemented them on an FPGA or semi-custom designs.

The investigation was done carefully for the best-related works to design complex multipliers with high performance using circuit optimization techniques for the radix-4 Booth technique, multipliers, and adders. The proposed 12-bit complex multiplier is designed for 0.18 µm CMOS technology using Cadence tools in full custom design. Since systems with low power are in great need for the current situation, the results after post-simulation show that the proposed complex multiplier offers low area and power when compared to reported literature [5, 15- 18]. This is basically due to the circuit optimization technique in the architecture. The architecture is optimized to reduce the number of transistors at every stage. Thus, low power and low area were achieved when compared to the above proposed designs.

Designing digital circuits is challenging when dealing with the trade-off between power area and performance. Few designs are based only on high throughput, and to achieve the target applications' maximum performance is expected. Common Subexpression Elimination (CSE) techniques are used to optimize the coefficients of CSD encoded to minimize the adders/subtractors count [20]. A unique multiple-bit counter for effective binary multiplication is presented in this study. Three methods are used to propose, modify, and optimize a 7:3 counter: first, group-wise parallel addition; second, removing unnecessary carry-generators; and third, hardware optimization. Standard static-CMOS is used in the circuit's design and optimization [21, 22].

4. Concepts of Complex Multiplier

Complex multipliers are essential parts of the co-processors found in AI accelerators, GPUs, and DSPs. The suggested design aims to accelerate the multiplication process by using circuit optimization techniques and a radix-4 modified Booth encoder. This immediately results in these specialized hardware units being able to execute complex algorithms more quickly. A crucial technique in many scientific and technical domains, the Fast Fourier Transform (FFT) mainly depends on complex number arithmetic. The design can speed up FFT calculations by enhancing the complex multiplier, communication engineering, and other data frequency domain analyses. Power consumption used to be a minor consideration in VLSI design, but it is now a fundamental factor along with reliability, performance, and area. For present and future portable devices, the design places a high priority on low power and low area, eliminating the need for expensive cooling systems or huge batteries.

By using circuit optimization techniques that minimize the number of transistors at each level, the design achieves low power and minimal area. For large data FFT systems, which need hundreds of multipliers, this is very crucial. This is further aided by the utilization of radix-4 Booth encoding, which lowers partial product rows. The study proposes that scaling down to smaller technology nodes like 90nm, 45nm, and 28nm can further improve the design and possibly result in even larger space and power reductions. The architecture is built using 0.18µm CMOS technology. A complex multiplier plays an important role and is a fundamental building block used to perform multiplication, which is a basic operation. This operation is crucial for algorithms like fast Fourier transforms and other signal processing operations. The complex number multiplication concept is as follows.

Two complex numbers $(Ar + jAi)$ and $(Br + jBi)$ are multiplied as,

$$(Ar + jAi).(Br + jBi) = (Ar.Br - Ai.Bi) + j(Ar.Bi + Ai.Br) \quad (1)$$

The imaginary part is,

$$Pi = (Ar.Br - Ai.Bi) \quad (2)$$

and the real part is,

$$Pr = (Ar.Bi + Ai.Br) \quad (3)$$

Where Pr and Pi are products of real numbers and imaginary numbers. The direct computation requires 4real multipliers, 2 adders or subtractors. Figure 2 represents the complex multiplier.

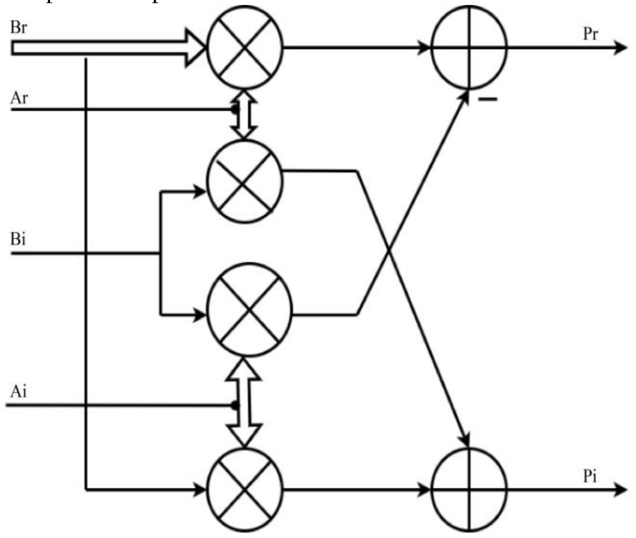

Fig. 2 Direct computation of complex multiplier

When a complex multiplier multiplies two complex numbers, these are often implemented, in fixed-point or floating-point, depending on the system's performance, area, and precision requirements, either in an FPGA, DSP, ASIC, or any other special processor like GPUs.

4.1. Fixed Point Complex Multipliers

The number of bits for integers and parts of a fraction is fixed, and the arithmetic used is fixed-point. Scaling and saturation are required to handle overflow and maintain precision. They are good for hardware implementation. They are fast and resource-efficient. Dynamic range and precision are limited. Scaling is needed to avoid overflow, which can reduce accuracy. They are used in DSPs and embedded systems. In real systems, such as embedded devices, fixed-point is used to save memory and CPU, often used in embedded systems, microcontrollers, DSP, etc., where memory or speed is limited. If they are stored as integers and scaled, it is a fixed-point complex multiplier. Example 4.5 + i5.5 for fixed point format, 45 and 55 by multiplying with 10, and once the result is obtained, it is divided by 10, and for floating, it can be the same given data. Also, the q1.15 format is used to convert decimals into integers for fixed-point complex multipliers.

4.2. Floating Point Complex Multipliers

Figure 2 consists of four multipliers, and multiplier blocks play an important role here. A general multiplier is represented in Figure 3 below.

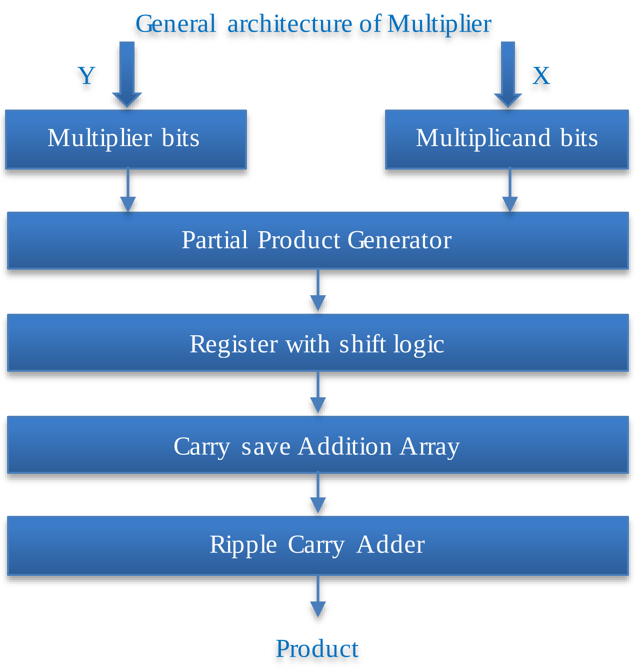
General architecture of Multiplier
Y
X
Multiplier bits
Multiplicand bits
Partial Product Generator
Register with shift logic
Carry save Addition Array
Ripple Carry Adder
Product

Fig. 3 General multiplier architecture

It consists of five sections. In the first stage, registers are required to store the bits, next is the generator of the partial product, and then the register is used for shifting the products when they are generated, then added with carry save adder to get carry and sum and finally added with ripple carry adder to get the result [19-22, 4]. In the second stage, the Radix-4 booth algorithm is utilized, in which partial product rows can be reduced to exactly half of the multiplier bits taken.

Time complexity also plays an important role in VLSI. For basic Booth encoding, the time complexity is O(n), where n represents several multiplier bits; the time complexity for modified Booth encoding is O(n/log2(r)), where r is radix-4 encoding. Here, a higher radix indicates a reduction in the number of operations because it can process a large number of bits per cycle. For radix-4, the complexity is O(n/2), and for radix-8, it is O(n/3). The time complexity of a complex multiplier depends on how the complex numbers are represented and the algorithm used for multiplication. In standard complex multiplication, if the real numbers have size n (for example, n-bit integers or floating point), each multiplication is O (n^2). Similarly, the complexity of time for adders also plays a crucial role.

5. Complex Multiplier Design Architecture

The Architecture of a complex multiplier circuit that applies Radix-4 modified Booth encoding, which is utilized in the study, is shown in Figure 4. All the blocks required for design are optimized using the circuit optimization technique, i.e. by reducing the number.

The transistors in the leaf cells are designed especially for a multiplexer designed with two transmission gates, adders designed with transmission gates,10T D-flip-flops, and an XOR gate.

Table 1. Radix-4 modified booth encoding

A2i+1	A2i	A2i-1	Generated Partial Products
0	0	0	0*Multiplicand
0	0	1	1*Multiplicand
0	1	0	1*Multiplicand
0	1	1	2*Multiplicand
1	0	0	-2*Multiplicand
1	0	1	-1*Multiplicand
1	1	0	-1*Multiplicand
1	1	1	0*Multiplicand

Architecture can be studied in two parts: the imaginary part and the real part. It uses two 12-bit registers as inputs, each of which is designed with a 10-transistor D flip flop and a 24-bit register for outputs that store the real and imaginary parts. A, B, C, and D are 12-bit numbers, based on the architecture. The imaginary part is (BC + AD) and the real part is (AC- BD). MBE, a generator that generates partial products, CSA trees, CPA, and ripple carry adder are other blocks used to get the final product. In the conventional method, four separate multipliers are required, that is, for AC, BD, AD, and BC to produce real part and imaginary part numbers, which will use four different modified Booth encoders.

For each multiplier with modified Booth encoding, the architecture is shown in Figure 5 below. The architecture clearly picturizes MBE separately for all four products, that is, AC, BD, BC, and AD, the chip area, power consumption increases, and speed decreases.

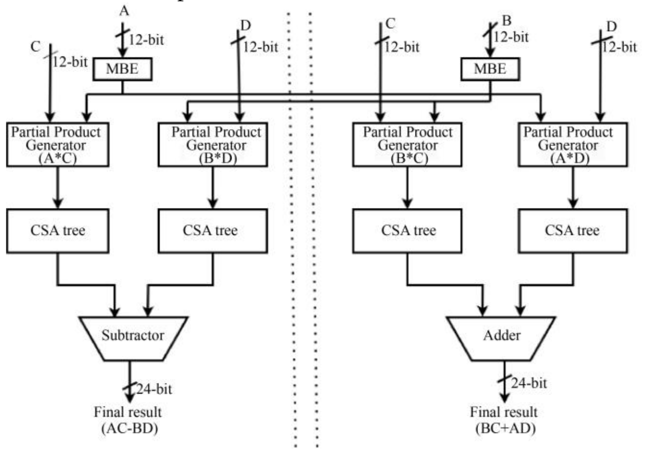

Fig. 4 Architecture of a complex multiplier

The architecture of complex multipliers is shown in Figure 4; only two modified Booth encoders are used to execute the process for multiplication of complex numbers. 12 bits of A and 12 bits of B are always set as multiplier bits, which are applied to modified Booth encoding and 12-bit C and 12-bit D are considered as multiplicand bits. With this approach, the number of transistors in the design is reduced.

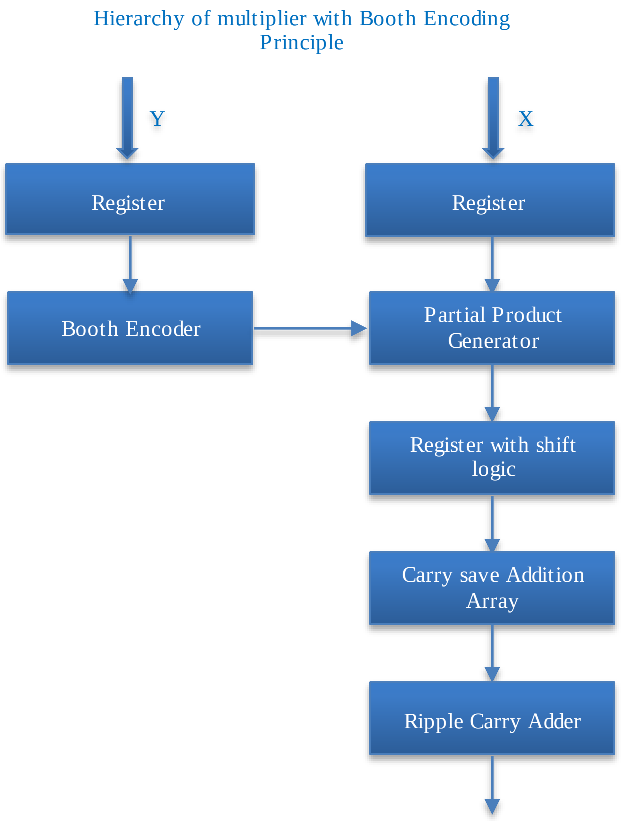
Hierarchy of multiplier with Booth Encoding Principle
Y
X
Register
Register
Booth Encoder
Partial Product Generator
Register with shift logic
Carry save Addition Array
Ripple Carry Adder

Fig. 5 Hierarchy of multiplier with booth encoding principle

5.1. Modified Booth Encoding Radix-4 Algorithm

In a conventional multiplier, the first shift and then add technique is used, but here the multiplier 12 bits are grouped into groups of three bits by appending a zero in the LSB. A stride of two and a window size of 3 bits is taken. So, the multiplier (12+1) bits of A and (12+1) bits of B are considered for grouping. Each group will produce the signal. Table 1 shows the signals that are generated for the group of three bits. The control signals are the inputs to a generator, which generates partial products.

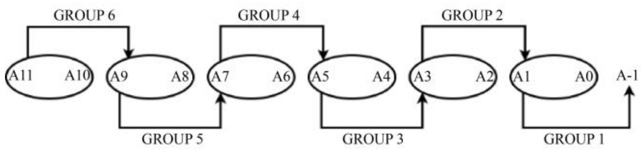

Fig. 6 Multiplier bits of A grouped according to booth encoding

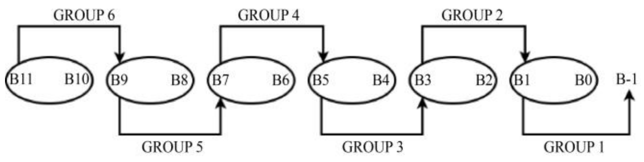

Fig. 7 Multiplier bits of B grouped according to booth encoding

By implementing this modified booth technique, for each product, partial product rows to be stored can be reduced from 12 to 6, i.e. for AC, BD, AD, and BC. When the circuit design runs, power consumption and propagation delay play an important role. In circuit optimization techniques and MBE, both power and area can be lowered.

5.2. Partial Product Generator

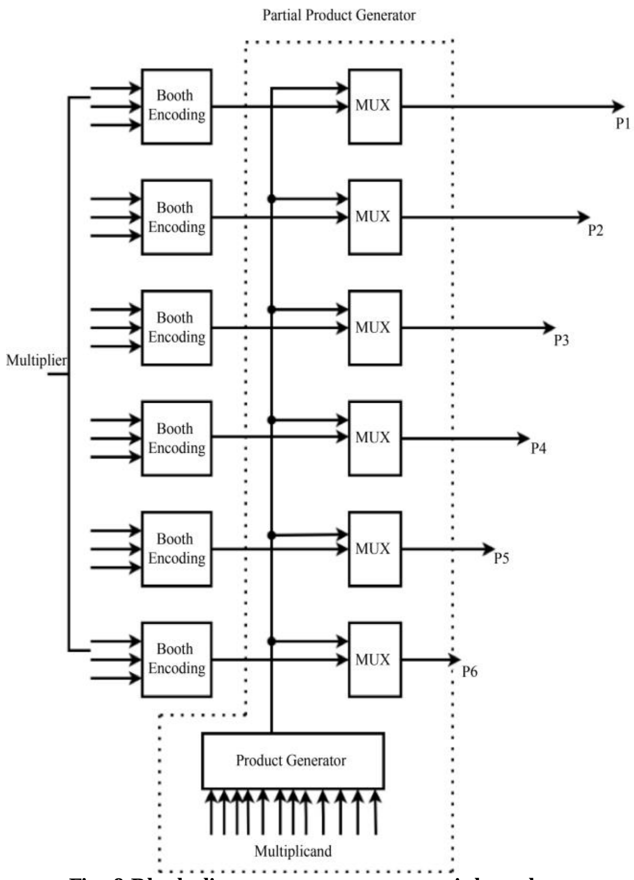

Fig. 8 Block diagram to generate partial products

Keeping in view the low power and low area of the design, many techniques to generate partial products were used in [19, 20]. Partial product rows go with 2's complement error correction and negation signal circuits. But in the design, this approach is not followed; therefore, multiplication uses fewer paths, which increases speed, utilizes less power, and uses less area. Using the Booth encoding technique, when signed numbers are multiplied, the operation requires two complements as the Booth encoder generates negative signals.

An integer's two's complement is typically created by adding one to the number after every bit is complemented. Six multiplexers are employed to handle the 12-bit complex multiplication process. Figure 8 above shows a partial product generator. Modified Radix-4, the main way that Booth encoding speeds up multiplication is by lowering the partial products that must be created and then added. Faster computing and increased efficiency in digital multiplier designs are directly correlated with this reduction.

Reducing partial product rows to half of the multiplier bits is the main advantage. In a traditional multiplier, a partial product row is produced by each multiplier bit. This would normally result in N rows of partial products for an N-bit multiplier. Partial products to N/2 are cut in half by using radix-4 encoding, which groups the multiplier bits so that each group processes two bits simultaneously.

5.3. Carry Save Adder

A simpler and quicker partial product addition stage results from fewer partial product rows. Radix-4 Booth encoding reduces the partial product rows for each product (AC, BD, AD, and BC) from 12 to 6. This is for a 12-bit complex multiplier. The whole multiplication procedure is sped up as a direct result of this decrease in the number of elements to be added together. It consists of an array of full adders. It adds three or more partial products without propagating carry. The design generates six rows of partial products. All six rows of partial products are added using a ripple carry adder. Since the ripple carries adder is too slow and latency is also important in design, carry-save adders are used for better speed. Four CSA trees are considered in the design, one for each product. Each CSA tree will compress a vector of partial products for its respective input products.

Carry-save adders, which are made especially to add partial products with fewer transistors, are also incorporated into the design to speed up the addition process. This is important because the efficiency of their summing has a major impact on the overall speed of multiplication, even though Booth encoding decreases the number of partial products. As each partial product of every product will be up to 24 bits, each CSA tree needs to handle partial products of up to 24 bits wide since the design is of a 12x12 complex multiplier. Each CSA tree takes six inputs and compresses them using 3:2 compressors that are full adders in layers. Sum and carry each up to 24 bits are the outputs. Figure 9 represents the blocks of the carry save adder tree. Therefore, each CSA tree outputs a sum of 24 bits and carry bits of 24, which are not the final product.

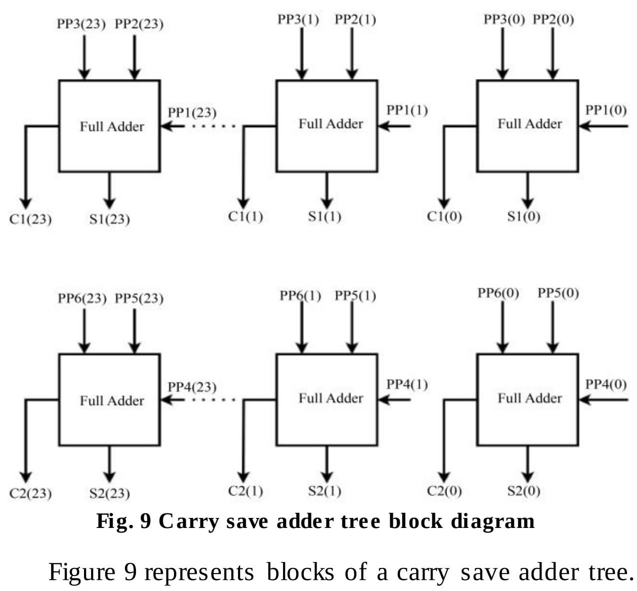

Fig. 9 Carry save adder tree block diagram

Figure 9 represents blocks of a carry save adder tree. Therefore, each CSA tree outputs a sum of 24 bits and carry bits of 24, which are not the final product.

5.4. Carry Propagate Adder (CPA)

Sum of 24 bits and carry of 24 bits of each CSA use CPA to get the final 24-bit products. For digital addition, Carry Propagate Adders (CPAs) provide a fair trade-off between speed and complexity, especially when contrasted with more straightforward ripple carry adders. Despite not being the fastest adder, CPAs are frequently utilized because of their reasonable hardware complexity and comparatively low latency when compared to other adder types.

5.5. Combination of Real Products and Imaginary Products

Here, an addition of (BC+AD) is performed to get the imaginary part and a subtraction of (AC-BD) to get the real part; therefore, the Ripple carry adder is used. It takes output from the CPA and finally generates the real part and imaginary part. Two's complement adders are used for the real part. Figure 10 shows how the outputs of the CPA are fed to the Ripple carry adder.

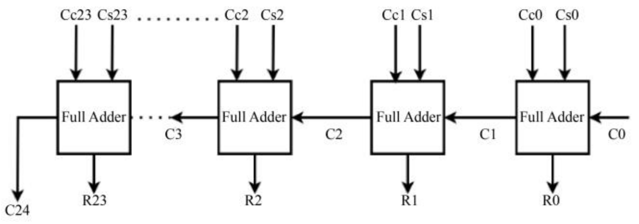

Fig. 10 Ripple carry adder block diagram

6. Implementation of Complex Multiplier

The sophisticated multiplier is constructed using a bottom-up methodology in this fully customized design. Unlike semi-custom or FPGA-based designs, this full custom methodology enables fine-grained circuit-level optimization.

The use of the "circuit optimization technique" throughout the design is a key component of the originality. To do this, the design must be optimized to use fewer transistors at each step, which will result in a smaller silicon area and reduced overall power consumption. These increases in power and area efficiency are a result of this careful transistor-level optimization.

Complex multipliers are implemented using Cadence tools in full custom design. The approach used is a bottom-up approach. The increase in speed, low power, and reduction of chip area are achieved by using radix-4 Booth encoding and reducing the number of transistors using the circuit optimization technique during the design of leaf cells [23, 24].

First, the leaf cells are designed. The leaf cells used in the implementation of the complex multiplier are an inverter, an input NOR gate, an XOR gate designed with only 4-T, a transmission gate designed with 2T, a multiplexer with two transmission gates, and a D-flip flop with 10-T, which is used in registers. Then using these leaf cells the next level blocks are designed like half adder, full adder, 12-bit registers, 24-bit registers, booth encoder, partial product generator (which generates the least significant bit, and P1 generating the remaining bits), carry-save adders, ripple carry adder and finally, multipliers and integrated all the blocks to obtain the final implementation of the complex multiplier.

The “radix-4 modified Booth encoder” is incorporated into the design. Because it efficiently reduces the number of partial product rows (to n/2), this method is essential for performance enhancement because it expedites the multiplication process. The study emphasizes that lowering the total number of transistors helps improve speed, low power, and low space.

The utilization of a “smaller number of transistors overall” is made possible by the combination of circuit optimization and radix-4 Booth encoding. This decrease in the number of transistors indicates “less silicon area” and helps the design achieve its “low power” and “increase in speed” goals. The suggested design provides “low power and less area as compared to reported literature” because of these optimization techniques; the article makes this clear.

6.1. Booth Encoder Radix-4 Block

Using the design rules, the leaf cells are designed. They are an inverter, three input gates, and a 2x1 multiplexer. When employing simple Booth encoding, a multiplier’s temporal complexity is usually O(n), where ‘n’ is the number of multiplier bits. On the other hand, the temporal complexity for modified Booth encoding with radix-r is $O(n/\log_2(r))$. The complexity is $O(n/\log_2(4)) = O(n/2)$ for radix-4 (r=4). A straight increase in speed is indicated by this mathematical reduction in complexity. A greater number of bits can be processed per cycle with higher radix encoding. This results in a faster execution time overall since more of the multiplication operation is finished for every clock cycle. When used in conjunction with circuit optimization approaches, radix-4 modified Booth encoding helps lower the overall number of transistors needed in the design. Faster operation and less propagation delay can result from fewer transistors.

6.1.1. Schematic

Using the design rules, the leaf cells are designed. They are an inverter, three input gates, and a 2x1 multiplexer. When employing simple Booth encoding, a multiplier’s temporal complexity is usually O(n), where ‘n’ is the number of multiplier bits. On the other hand, the temporal complexity for modified Booth encoding with radix-r is $O(n/\log_2(r))$. The complexity is $O(n/\log_2(4)) = O(n/2)$ for radix-4 (r=4). A straight increase in speed is indicated by this mathematical reduction in complexity.

A greater number of bits can be processed per cycle with higher radix encoding. This results in faster execution time overall since more of the multiplication operation is finished for every clock cycle. When used in conjunction with circuit optimization approaches, radix-4 modified Booth encoding helps lower the overall number of transistors needed in the design. Faster operation and less propagation delay can result from fewer transistors.

6.1.2. Symbol of the Booth Encoder

Grouping multiplier bits for A and B is shown in Figures 6 and 7, and these grouped signals are fed to the inputs. The cadence tool uses a symbol editor to create the symbol for the schematic drawing, which is shown below. The next step is the test bench. This step is called pre-layout simulation. This is done with a specter simulator. Figure 12 shows the symbol of the booth encoder.

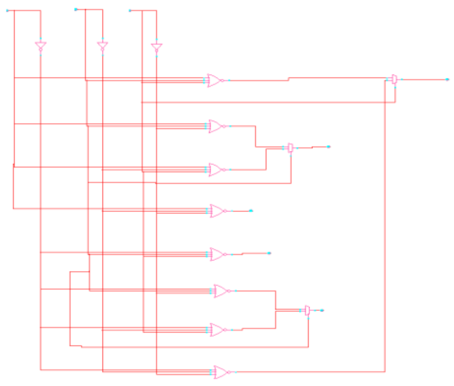
Fig. 11 Booth encoder schematic

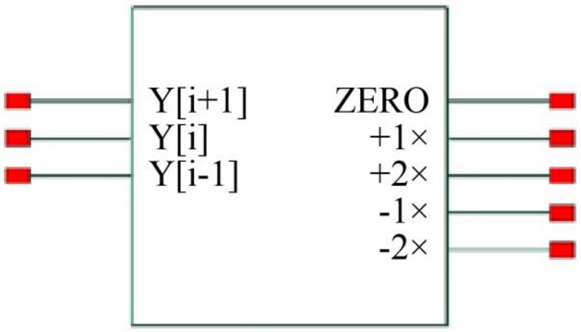

Fig. 12 Booth encoder symbol

6.1.3. Layout of the Booth Encoder

Working with the physical layout will be more challenging for the back-end designer. The design rules should be followed very carefully, depending on the technology; the layout should be drawn, and the contacts and vias should be properly placed. One should think properly and try to minimize the use of metal layers. The propagation delay is one of the important factors to be considered when layouts are drawn physically. By using the Virtuoso layout editor, a CMOS physical layout is drawn. Figure 13 shows the layout using 6 different metal layers for the schematic of Figure 11.

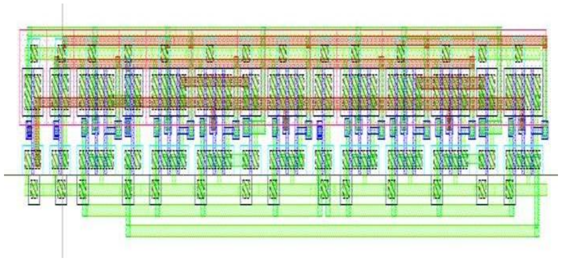
Fig. 13 Booth encoder layout

6.1.4. Layout Extraction

The Diva / Assura circuit extractor extracts the layout and clearly shows parasitic resistance and capacitance. The designer can check the propagation delay of the circuit at this point using τ=RC.

6.1.5. Layout vs Schematic

Using the hierarchy editor, both the layout symbol and schematic symbol are created, and a test bench is created for both.

6.1.6. Post Layout Simulation

Waveforms obtained can be compared with the layout and the schematic symbol for which the test bench was created.

The following individual blocks are used in the Modified Booth Encoder.

Multiplexer

The Multiplexer was designed with two transmission gates. It has two inputs and one select line. Output is generated based on the selection line.

The inverter is designed with PMOS and NMOS transistors. A three-input NOR gate is designed using a standard concept.

6.2. Registers

Leaf cells required for input 12-bit registers and output 24-bit registers are one-bit D flip-flops. 12-bit registers use 12-D flip-flops, and 24-bit registers use 24-D flip-flops. These are designed with only 10 transistors, a D flip-flop. Schematic symbols and layouts are shown in Figures 14,15, and 16.

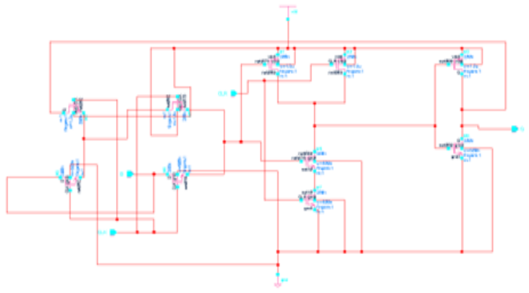

Fig. 14 Schematic of D-FF with 10 transistors

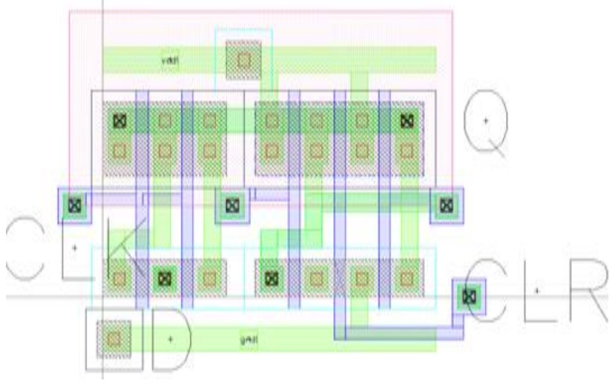

Fig. 15 D-FF layout with 10 transistors

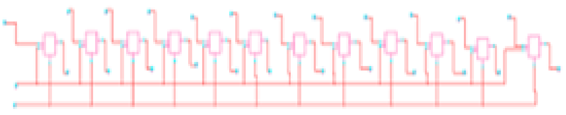

Fig. 16 12-bit register schematic

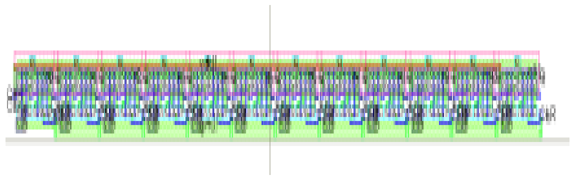

Fig. 17 12-bit register layout

6.3. Partial Product Generator

It generates six rows of 24-bit partial products. It consists of two 12-bit input registers, six Booth encoders, six 12-bit partial product generators, and six 24-bit registers with shift logic. The symbol was generated, and the layout was tested by Cadence tools after integrating all the required sub-systems, as shown below in Figure 18.

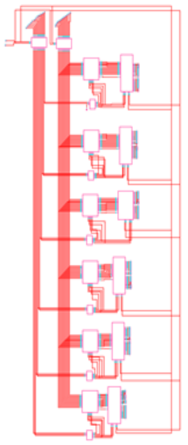

Fig. 18 Schematic of partial product register

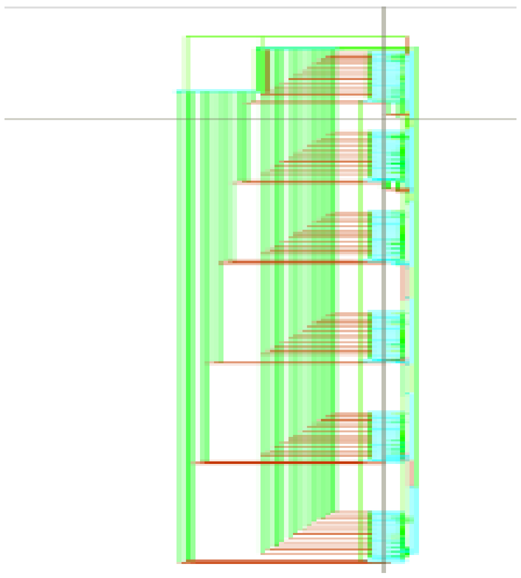

Fig. 19 Layout of partial product register

6.4. Complex Multiplier

The complex multiplier is designed with Radix – 4 algorithms for partial product rows reduction and 10-T D flip flop for registers also multiplexer is designed with two transmission gates which are used in the booth encoder and four transistor xor gate and transmission gate with two transistors which are used in full adders and finally, design utilized less power and low silicon area. Figure 20 shows the Schematic of a Complex multiplier.

The complex multiplier is verified functionally with the test bench, and the layout in Figures 21, 22 to 29 represents simulation results for the post-layout designs.

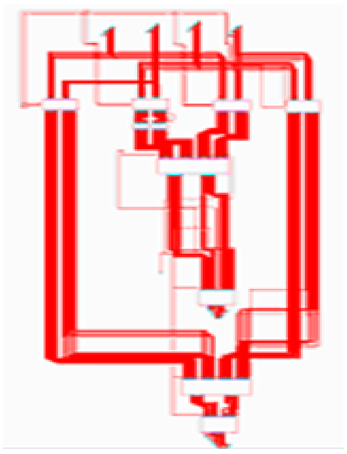
Fig. 20 Complex multiplier schematic

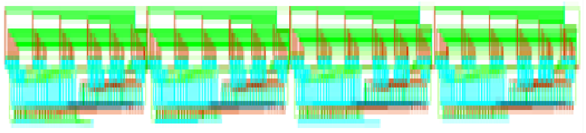
Fig. 21 Layout of complex multiplier

This work mentions several specific circuit optimization techniques used in the design to achieve less area and power consumption. Radix-4 Modified Booth Encoding (MBE) was used to reduce the number of partial product rows. Reduced the number of Modified Booth Encoders and used only two MBEs instead of four, which helped reduce the overall transistor count.

In the Optimized D flip-flop design, the registers use D flip-flops designed using only 10 transistors. Multiplexers with reduced transistor count are used in multiplexers designed with only two transistors. The 10-transistor D flip-flop was used as the leaf cell for the registers in the complex multiplier design. This suggests the 10-transistor design was chosen specifically for its efficiency and optimization. By using this optimized D flip-flop design, the design reduces the overall transistor count in the complex multiplier architecture, which likely contributed to the lower power consumption and area reported for the design.

Optimized XOR gate and transmission gate designs used in full adders use four-transistor XOR gates and two-transistor transmission gates. Carry-save adders are used to speed up the addition process of partial products. A bottom-up approach was used in implementing the complex multiplier, which likely contributed to optimizing the overall design. Though several techniques exist for low-power design, this work involves the circuit optimization method to reduce the power. Circuit optimization automatically leads to a reduction in area and less power consumption, which indirectly increases the performance of the design.

7. Results

The design is implemented using Cadence tools for 12x12-bit signed and unsigned numbers and is simulated using ADE with Spectre simulator for both pre-layout and post-layout complex multiplier using 0.18µm technology. The following unsigned numbers are given as inputs.

a=5	0000 0000 0101
b=3	0000 0000 0011
c=2	0000 0000 0010
d=4	0000 0000 0101

The observed outputs of the complex multiplier after post-simulation waveforms are shown below.

Real Part: -2 1111 1111 1111 1111 1111 1110
imaginary part: 26 0000 0000 0000 0000 0001 1010

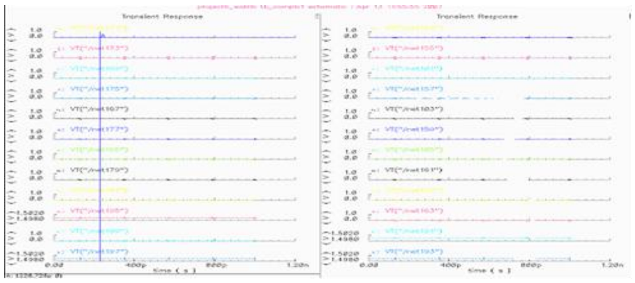
Fig. 22 (a), (b) Waveforms for input

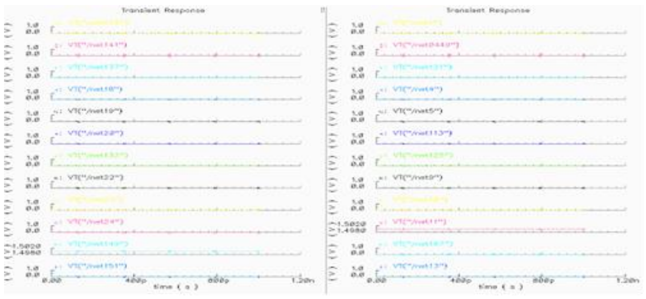
Fig. 23 (a), (b) Input waveforms

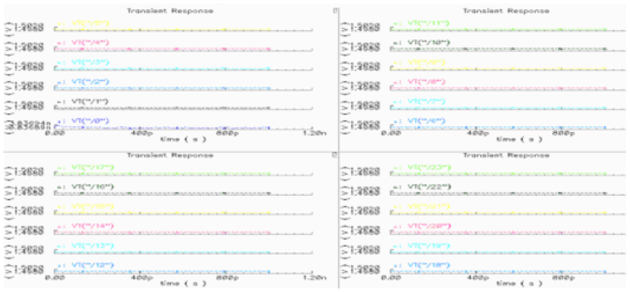
Fig. 24 Output waveforms for the real part with

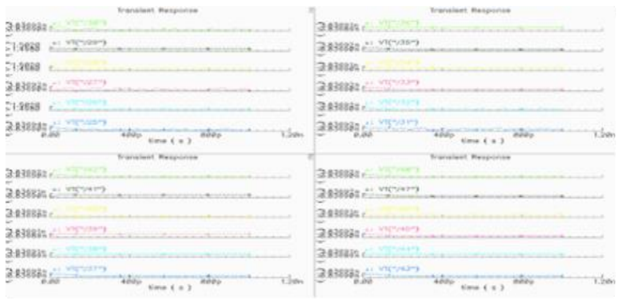
Fig. 25 Output waveforms for imaginary part

The following signed numbers are applied as inputs.
a= 5 0000 0000 0101
b= 3 0000 0000 0011
c= -2 1111 1111 1110
d= 4 0000 0000 0100

The observed outputs after post-layout simulation are as follows.
Real part -22 =1111 1111 1111 1111 1110 1010
imaginary part 14 =0000 0000 0000 0000 1110

Figure 26 shows the input waveform of a and b, and Figure 27 shows the input waveform for a and b.

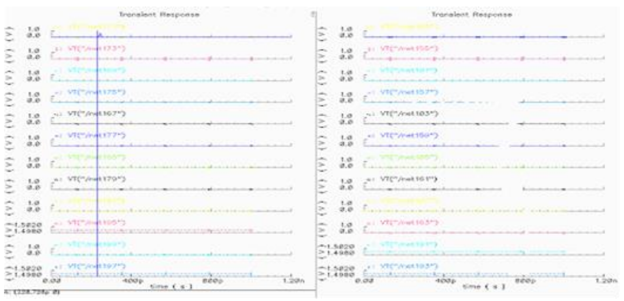
Fig. 26 (a), (b) Waveforms for input

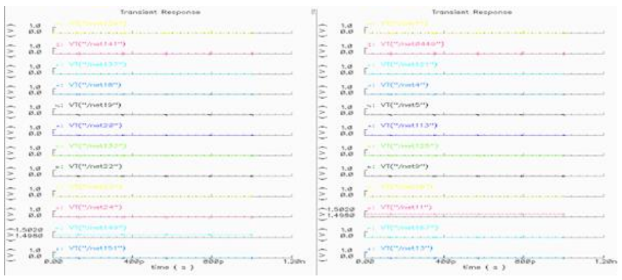
Fig. 27 (a), (b) Input waveforms

Output waveforms of the Real part are shown in Figure 28, and the output waveforms of the imaginary part are shown in Figure 28. Table 2 shows the analysis of dynamic, standby and total power.

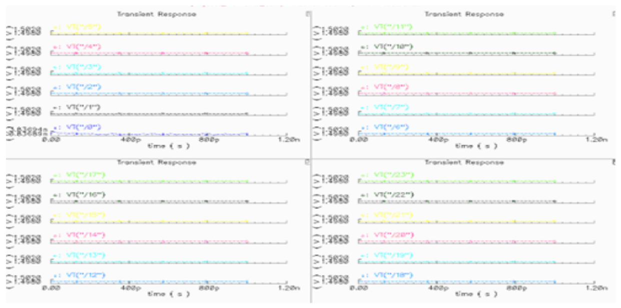
Fig. 28 Waveforms for real part imaginary part

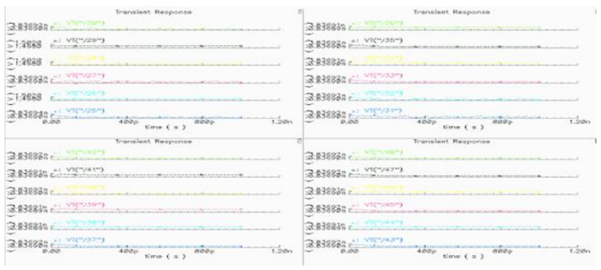
Fig. 29 Waveforms for imaginary part

Table 2. Analysis of power

Design Module	Dynamic Power	Standby Power	Total Power
Booth Encoder	108.7 µW	3.12 µW	111.82 µW
Multiplier	1828µW	697.4µW	2525.4 µW
Complex multiplier	6.5mW	3.25mW	9.75 mW

The key performance factors evaluated for the proposed 12-bit complex multiplier design and Quantitative findings are:

- Power consumption: The paper provides a quantitative breakdown of the dynamic power, standby power, and total power consumption for the Booth Encoder, Multiplier, and overall Complex Multiplier blocks.
- Area/Transistor count: The design approach, including techniques like using a reduced number of Booth Encoders, optimized leaf cell designs, and a bottom-up implementation, helped achieve lower overall transistor count and area.
- Speed/Latency: The use of Carry-Save Adders helped speed up the addition process compared to a Ripple Carry Adder.
- Functional verification: The design was implemented and functionally verified using Cadence tools, including pre-layout and post-layout simulations.

8. Conclusion

A thorough effort has been made to create a 12-bit complex multiplier with a focus on optimizing for low power, reduced area, and increased performance, which makes it distinctive.

Efficient Radix-4 Modified Booth Encoding, circuit optimization approaches, design strategy, full custom implementation, and performance gains through reduced transistor count are the main innovative components.

The proposed full custom design of a 12-bit complex multiplier is designed using Cadence tools. The need for this complex multiplier in digital processors is identified as Fast Fourier Transform (FFT), which is capable of handling new requirements in signal processing, which has mobilized the world of high-performance digital signal processing.

With the new emerging technologies, special co-processors are being designed and used for many applications where these processors need complex multipliers, depending on the applications.

The complex multiplier was designed with radix-4 by using a circuit optimization technique that reduces transistors during the design of leaf cells, to achieve less area and less power. Further design can be done using 90nm, 45nm, and 28nm to reduce the area. Optimization techniques like inserting sleep circuits in the architecture can be used to further reduce power and finally attain low-power area circuits.

References

[1] Andrew D. Booth, "A Signed Binary Multiplication Technique," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236-240, 1951. [CrossRef] [Google Scholar] [Publisher Link]

[2] C.S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, vol. 13, no. 1, pp. 14-17, 1964. [CrossRef] [Google Scholar] [Publisher Link]

[3] Taewhan Kim, William Jao, and Steve Weng Kiang Tjiang, "Arithmetic Optimization using Carry Save Adders," *Proceedings of the 35th Annual Design Automation Conference*, pp. 433-438, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jung-Yup Kang, and J.L. Gaudiot, "A Fast and Well-Structured Multiplier," *Euromicro Symposium on Digital System Design*, Rennes, France, pp. 508-515, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[5] Rizalafande Che Ismail, and Razaidi Hussin, "High Performance Complex Number Multiplier using Booth-Wallace Algorithm," *2006 IEEE International Conference on Semiconductor Electronics*, Kuala Lumpur, Malaysia, pp. 786-790, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[6] P.K. Saha, A. Banerjee, and A. Dandapat, "High Speed Low Power Complex Multiplier Design Using Parallel Adders and Subtractors," *International Journal on Electronic and Electrical Engineering*, vol. 7, no. 11, pp. 38-46, 2009. [Google Scholar]

[7] Monika Hemnani et al., "Hardware Optimization of Complex Multiplication Scheme for DSP Application," *2015 International Conference on Computer, Communication and Control*, Indore, India, pp. 1-4, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[8] K. Deergha Rao, Ch. Gangadhar, and Praveen K. Korrai, "FPGA Implementation of Complex Multiplier using Minimum Delay Vedic Real Multiplier Architecture," *2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON)*, Varanasi, India, pp. 580-584, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[9] Duy Manh Thi Nguyen et al., "Design and Implementation of Complex Multiplier with Low Power and High Speed," *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, Ho Chi Minh City, Vietnam, pp. 215-219, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ansha Noushad, and A.R. Abdul Rajak, "VLSI Implementation of Complex Multiplier using Vedic Mathematics and Study its Performance," *ARPN Journal of Engineering and Applied Sciences*, vol. 16, no. 10, pp. 1058-1061, 2021. [Publisher Link]

[11] Mario Garrido et al., *Hardware Architectures for the Fast Fourier Transform*, Handbook of Signal Processing Systems, pp. 613-647, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Fahad Qureshi, Jarmo Takala, and Shuvra Bhattacharyya, *Rotators in Fast Fourier Transforms*, Embedded, Cyber-Physical, and IoT Systems, pp. 245-262, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[13] Carl Ingemarsson et al., "Efficient FPGA Mapping of Pipeline SDF FFT Cores," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2486-2497, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jakub Žádník, and Jarmo Takala, "Low-Power Programmable Processor for Fast Fourier Transform based on Transport Triggered Architecture," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, pp. 1423-1427, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[15] Shiann-Rong Kuang, Jian-Ping Wang, and Cang-Yuan Guo, "Modified Booth Multipliers with a Regular Partial Product Array," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 5, pp. 404-408, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ravindra P. Rajput, and M.N. Shanmukha Swamy, "High Speed Modified Booth Encoder Multiplier for Signed and Unsigned Numbers," *2012 UKSim 14th International Conference on Computer Modelling and Simulation*, Cambridge, UK, pp. 649-654, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[17] Amir Fathi et al., "Low Latency, Glitch-Free Booth Encoder-Decoder for High-Speed Multipliers," *IEICE Electronics Express*, vol. 9, no. 16, pp. 1335-1341, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[18] Amir Fathi et al., "Ultra High Speed Modified Booth Encoding Architecture for High-Speed Parallel Accumulations," *IEICE Transactions on Electronics*, vol. 95, no. 4, pp. 706-709, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[19] Venkata Krishna Odugu, C. Venkata Narasimhulu, and K. Satya Prasad, "Design and Implementation of Low Complexity Circular Symmetric 2D FIR Filter Architectures," *Multidimensional Systems and Signal Processing*, vol. 31, pp. 1385-1410, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[20] Zahra Ebrahimi et al., "Rapid: Approximate Pipelined Soft Multipliers and Dividers for High Throughput and Energy Efficiency," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 3, pp. 712-725, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Aloke Saha et al., "Novel CMOS Multi-Bit Counter for Speed-Power Optimization in Multiplier Design," *AEU - International Journal of Electronics and Communications*, vol. 95, pp. 189-198, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[22] Lakshmanan, M. Othman, and M.A.M. Ali, "High Performance Parallel Multiplier Using Wallace-Booth Algorithm," *Proceedings of the 9th International Conference on Neural Information Processing. Computational Intelligence for the E-Age*, Penang, Malaysia, pp. 433-436, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[23] S.M. Sait, A.A. Farooqui, and G.F. Beckhoff, "A Novel Technique for Fast Multiplication," *Proceedings International Phoenix Conference on Computers and Communications*, USA, pp. 109-114, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[24] B.P.R. Che Ismail et al., "Performance Enhancement and Reduced Area Parallel Multiplier," *IEEE National Symposium on Microelectronics*, pp. 252-258, 2005. [Google Scholar]