International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V74I2P106

Volume 74 Issue 2, 88-97, February 2026

Original Article

Pipelined Hardware-Efficient Scalable Complex Multiplier for High-Performance DSP Applications

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: 02 June 2025 Revised: 07 January 2026 Accepted: 20 January 2026 Published: 14 February 2026

Abstract *- Complex numbers find significant use in Very Large-Scale Integration (VLSI) design, particularly in Digital Signal Processing (DSP) algorithms and hardware implementations. They are crucial for efficiently performing complex mathematical operations like the Fast Fourier Transform (FFT) and other signal processing techniques. In any system, multipliers take a longer time to process due to complexity, and therefore, efficient design of multipliers is crucial in the overall performance of the system. A 16x18 and a 64x64 Hardware-efficient complex multiplier are designed, which uses fewer multipliers than the direct approach. The design is suitable for high-speed applications because of its pipelined architecture, and its parameterized sizes of inputs offer scalability of designs with customized and target applications. This design also offers low power and minimum area to serve the very purpose of SoC, and even as an IP Core. The design is implemented in 90nm technology using the Cadence design suite. Thus, the design furnished reduces the number of operations at the architectural level itself and can save chip area, power, cost, and further increase speed.*

Keywords *- Complex multiplier, Hardware-efficient, Pipeline, SoC, VLSI.*

1. Introduction

The Very Large-Scale Integration (VLSI) design heavily relies on complex numbers, especially for hardware implementations and Digital Signal Processing (DSP) methods. Additionally, AC circuits and electromagnetic phenomena are represented and analysed using complex numbers. DSP and more scientific relevance methods rely heavily on complex number multipliers.

An essential basic function in mathematical operations is multiplication. Among the commonly used Computation-Intensive Arithmetic Functions (CIAF) presently employed in many DSP applications, including Fast Fourier Transform (FFT), convolution, filtering, ALUs of processors, logic units, Multiply and Accumulate (MAC), etc. High-speed multipliers are necessary since multiplication takes up the majority of DSP algorithms' execution time. The primary determinant of an instruction cycle time now is still multiplication time in the DSP block. The growing number of applications has led to an increase in the need for rapid processing units. In many real-time applications, arithmetic operations with higher throughput are necessary to provide the required performance for image processing applications.

Multiplication is a crucial mathematical operation in these kinds of applications, and the creation of fast multiplier circuits has drawn attention for many years. For many applications, lowering the power usage and time delay are crucial criteria. In this paper, many multiplier architectures are presented. One of the quick and low-power multipliers is the Vedic mathematics-based multiplier [1,2].

The research strategies to decrease the complexity of any design resulting from increasing process variability, and then shorten chip production turnaround times, are a clear problem facing the Integrated Circuit (IC) industry. Conventional approaches used are largely labour-intensive, time-consuming, and resource-intensive. On the other hand, the distinct learning mechanisms of Artificial Intelligence (AI) offer a variety of fascinating, automated methods for managing intricate and data-intensive activities in the design and testing of VLSI designs. The time, along with effort, is decreased when AI and ML techniques are used in VLSI manufacturing and design. Consequently, it lowers the manufacturing turnaround time and increases the IC yield. The AI/ML automated techniques previously introduced for VLSI design and manufacture are reviewed in detail. AI/ML applications could transform VLSI design in the future, with a focus on fast, highly intelligent, and effective implementations.

To build and construct Systems-on-Chip (SoCs) that use complex number arithmetic for certain applications, including signal processing or specific scientific computations, the term

"complex number as system on chip" is used. It refers to a specific aspect of chip architecture; however, it is not as common as a "system on a chip" [3,4]. Numerous scientific fields, such as signal processing, control theory, fluid dynamics, electromagnetic, quantum mechanics, mapping, and vibration analysis, use complex numbers.

Regular 2^n x2^n sizes of complex multiplier are unable to address the customized applications of varied sizes. In this paper, a customized design size is possible. This SOC can be used when very high throughput is required, and real-time processing is mandatory, and where large FFT/filter banks are used, like SDRs and AI accelerators.

2. Related works

The "Vedas" of ancient Indian mathematics served as an inspiration to the design of the multiplier, adder, and subtractor units. These formulas reduce the operations and supply partial products and the sum in a single step. [5] DSP methods like FFT frequently use the simple act of multiplying two complex values. Hundreds of these multipliers are necessary for large, highly parallel FFT designs. Therefore, enhancement is applied to all of the complex multipliers in an architecture; improving one of them results in large savings.

The Structure and Function of Complex Multipliers. Complex Multiplier IP can be used to accomplish multiplication. A multiplier (multiplier + adder) is used to accomplish the complex multiplication in the Gowin Complex Multiplier IP. Bit width of input data is changeable, and the resource usage of the complex multiplier varies depending on the bit width. To save resources when the demand is satisfied, the user should configure the IP as minimally as possible. The following are adjustable settings for the Gowin Complex Multiplier IP: The breadth of the input data, the data entered is either a signed or unsigned number, and the GUI allows the user to choose the input data bit width N. Currently, the signed or unsigned data N up to a maximum of 26 bits, and based on the bit width of the input data, a resource-efficient technique to do complex multiplication is selected. Bit width of output data is 19 bits when the GUI configuration input N<=9; 37 bits is the output data bit width when 9<N<=18; and 73 bits is the output data bit width when 18< N<=26. Selecting the data type and whether the data is signed or unsigned when configuring the IP core in the GUI interface is chosen.

Selecting the reset mode, can choose between synchronous and asynchronous reset modes when configuring the IP core in the GUI interface. The Structure and Function of a Complex Multiplier IPUG521-1.0E 8 Configures register parameters. The DSP's registers can be enabled to implement the delayed output. The minimum delay is zero, and the maximum delay is three cycles for the output delay period [6,7]. The microelectronics sector has grown significantly due to the constant scaling back of transistors across many technical generations, which has increased device density and performance [8]. A significant increase in demand for power-efficient designs with cutting-edge features has resulted from the growing appreciation of portable gadgets in recent years. The continually growing demand in the electronics sector is satisfied by extremely sophisticated and scalable VLSI designs. A factor propelling the development of IC technology using enhanced device performance is continuous device downscaling to sub-3-nm-gate and beyond.

Device engineers now face both new opportunities and several obstacles because of the aggressive downscaling of CMOS technology. Devices can suffer from several performance problems despite their tiny size, including increased leakage [9–11], lower gain, and greater sensitivity to changes in the manufacturing process [12]. The circuit operation is greatly impacted by the huge increase in process differences, which causes identical-sized transistors to perform differently [13]. One of the main reasons for parametric yield loss is growing process variability in the nanometre range. Aggressive scaling, however, also affects their performance metrics [14, 15].

To sustain performance, the VLSI design flow must incorporate sophisticated, reasonably priced design methodologies that enable finer optimization. The ability of EDA tools to overcome design restrictions determines a chip's turnaround time. Furthermore, the conventional methods are manual, which makes them time-sensitive and resource-intensive, causing delays in time to market. Furthermore, it takes a lot of effort and time to comprehend the original functionalities, that is, the underlying source of problems, to implement modifications, when necessary, once the data has been sent back [12]. These days, field-programmable gate arrays, or FPGAs, are frequently used in the creation of finished goods. Nevertheless, applications do not always fully utilize the FPGA's hardware capabilities. More effective designs result from improved FPGA utilization.

Logic blocks can be used to implement the same on FPGAs. Nonetheless, specialized modules for computing arithmetic operations are typically found on FPGAs. These modules, known as DSP slices, enable higher clock frequencies in Xilinx FPGAs. The DSP48E1 block is included in Xilinx's 7 Series FPGAs. A block called DSP48E2 [13] is found in Xilinx's UltraScale and UltraScale+FPGAs [15]. Two effective complex multiplier implementations that work with DSP48E1 and DSP48E2 slices are presented in this research. While the second version seeks a trade-off, the first implementation aims for maximum throughput, minimum area, and the greatest speed permitted by design. Development of SoC devices has been made feasible by significant developments in the speed, capability, and IC complexity in recent years, including ASICs, RAMs, and microprocessors. The suggested architecture attempts to provide a single-chip solution for the majority of popular high-performance applications. This minimizes the amount of external logic by

combining the entire logic design into a single FPGA. The paper states that the proposed design aims to provide a high-throughput complex multiplier implementation suitable for these FPGA architectures.

Numerous issues in a variety of sectors have found notable answers because of Artificial Intelligence (AI). AI is human intelligence, illuminated so that a machine can simply mimic and complete tasks of different complications. AI includes ML as a subset because learning, reasoning, predicting, and perceiving are the objectives of AI/ML. Customers make informed judgments by using AI/ML's rapid ability to spot patterns. High processing rates can be achieved by AI/ML algorithms when handling various kinds of data. As these algorithms continue to learn, their forecast accuracy and efficiency increase. Additionally, they make decision-making easier by streamlining pertinent procedures. AI/ML techniques have been widely used in VLSI design and technology within the past ten years.

From design entry to fully customized layouts, the competence of CAD tools determines the performance assessment of extremely complicated integrated circuits. With an enormous rise in devices per chip, developing CAD for VLSI is getting harder and more complicated. For rapid convergence, there are many chances in semiconductor and EDA technology to create or integrate AI/ML elucidations to computerize processes at different abstraction levels of VLSI. The goal of these clever learning algorithms is to provide effective, automated chip fabrication solutions with comparatively quick turnaround times.

In comparison with the existing designs, the work compares the proposed pipelined hardware-efficient scalable complex multiplier design with Vedic mathematics-based multipliers. These Vedic formulas are said to reduce the number of operations and provide partial products and sums in a single step. The work also discusses the Gowin Complex Multiplier IP, which can be used to perform complex multiplication with a feature that allows the bit width of the input data to be configurable, with the resource usage varying depending on the bit width. A high-performance full-custom 12x12 complex multiplier [16] was designed in terms of power and area. In an efficient pipelined complex multiplier [18], a pipeline scheme for efficient realization of a complex multiplier using distributed arithmetic was used. The pipelined multiplier consists of one conventional multiplier that is multiplexed and some small additional circuitry on the boundary.

Overall, the related works section provides a thorough review of existing complex multiplier architectures and highlights the key features and contributions of the proposed design, such as its scalability, hardware efficiency, and pipelined structure, which aim to address the performance requirements of DSP applications.

3. Methodology

The proposed Pipelined Hardware-Efficient Scalable Complex Multiplier is designed in 90nm technology using Cadence Innovus tools.

The overall methodology consists of six main stages.

1. Firstly, the algorithm formulation and architecture design are developed
2. Optimized algorithm formulation and architecture are coded in Verilog HDL
3. The RTL is validated through exhaustive simulations using NC Launch before synthesis
4. The functionally verified RTL is synthesized using Cadence Genus
5. The synthesized netlist is imported to Innovus for complete place and route, DRC, and CTS, Routing
6. Finally, the layout is verified

The design flow is as shown in figure 1. A hierarchical process is followed by defining and modeling the chip's design. System-level is planned, which traverses through design, verification, and physical realization.

Specifications include the functionality, various hardware components (processors, memories, etc.), external interfaces to other hardware (pins, buses, etc.), internal interfaces between hardware components, and other physical design details such as area and power. Once the SoC specifications are done, the architecture is designed.

3.1. Algorithm Formulation and Architecture Design

To reduce the power and area of the complex multiplication (a+ib) (c+id) is restructured using arithmetic optimization and sharing of common subexpressions. A scalable architecture is defined so that the design can be extended to higher bit widths with minimal architectural modification. A multistage pipelined architecture is introduced to achieve higher operating frequency, reduced critical path, and improved throughput. Pipeline registers are inserted after each arithmetic stage to balance delay and maximize clock frequency.

3.2. RTL Design and Functional Verification

The optimized architecture is coded in Verilog HDL, like a modular RTL design for add/sub blocks, partial product units, and pipeline registers. Simulation is done using NC launch to verify functional correctness and then with test bench generation with multiple corner cases like positive integers, negative integers, overflow, and complex quadrants, and hence the RTL is validated through exhaustive simulations before synthesis.

3.3. Logic Synthesis

Next, the functionally verified RTL is then synthesized using Cadence Genus with a 90nm standard cell library. During this stage, Synthesis Design Constraints (SDC) are

applied to the target clock frequency, area, and power timings are optimized, and multi-threshold (HVT/LVT) cells are used where appropriate. The pipelined stages are automatically preserved using do-not-touch attributes, and hence the output consists of an optimized gate-level netlist, timing reports, area and power estimation, and a mapped standard cell netlist for Innovus. The Logic Synthesis process involves three steps, which are translation, mapping, and optimization. These steps convert RTL code into a level netlist. The RTL elements are mapped into logic gates using standard cell libraries. Libraries consist of various basic logic gates like AND, OR, NOR, and complex cells like adders, multiplexers, memory, and flip-flops, etc. At this stage, often additional test logic is inserted to support Design For Testability (DFT) features and capabilities of the device. The DFT logic can be used after fabrication to test and debug any faults in the design.

3.4. Physical Design

This is done using Cadence Innovus. The synthesized netlist is imported to Innovus for complete place and route in 90nm technology, like floor planning, where the core area is defined based on the synthesized cell count, input output pin placement, macro placement, if any, aspect ratio, and utilization set to avoid congestion, placement blockages, and routing channels. Then comes Power planning: V_{DD}/V_{SS} power rings created, metal stripes added across the core region, power grid verified using connectivity checks, uniform IR drop, and robust power delivery.

3.4.1. Placement

Global and detailed placement of all standard cells, physical constraints applied to preserve pipeline structure, congestion estimation, and pre-CTS timing optimization performed.

3.4.2. Clock Tree Synthesis

Clock buffers and inverters inserted; skew and insertion delay minimized; multistage pipeline structure helps simplify CTS; Post-CTS timing closure done. Routing: Global routing followed by detailed routing, DRC clean routing ensured, cross-talk aware, a timing-driven routing used, and special routing applied to high fan-out and critical nets.

3.4.3. Post Route Optimization and Extraction

Setup and hold fixes applied, RC parasitics extracted, accurate power and timing analysis performed.

3.5. Verification and Sign Off

The final layout is verified using DRC design rule checking. Timing signoff is completed using extracted parasitics to ensure the design meets timing at the target operating frequency. Finally, the GDSII file is generated

3.6. Performance Evaluation

The implemented design is evaluated in terms of operating frequency, latency, throughput, total cell count, and area. Dynamic, internal, and leakage power, PPA comparison with non-pipelined and non-optimized architectures. The methodology ensures a highly efficient, scalable, low-area, and high-throughput complex multiplier suitable for DSP and communication applications.

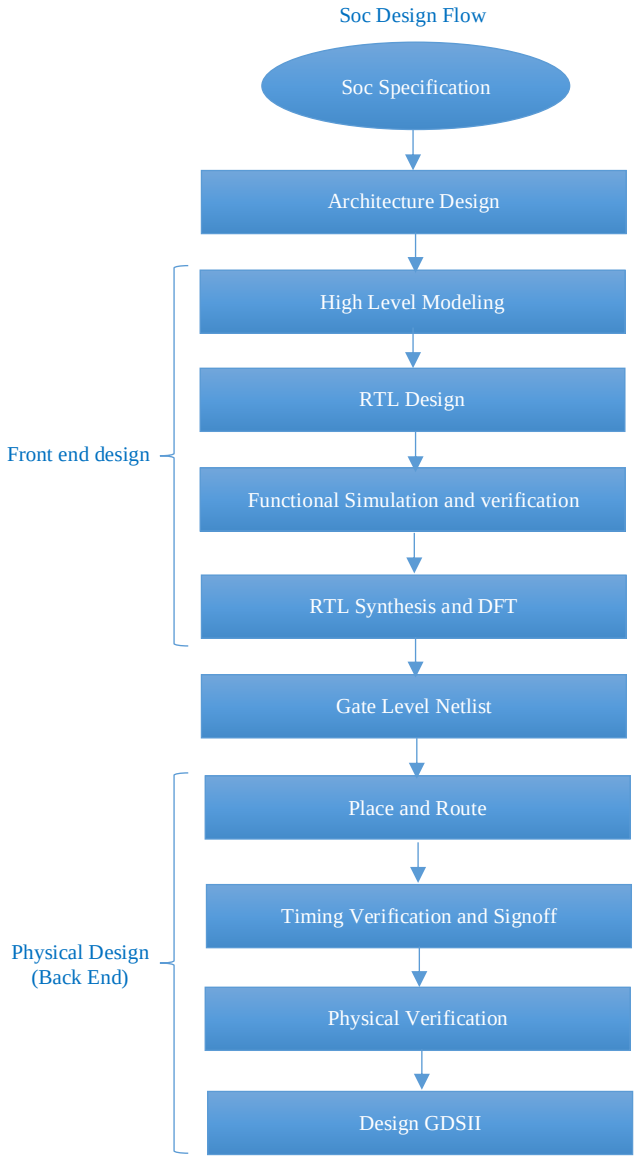

Fig. 1 Semicustom design flow [17]

4. Design of Complex Multiplier

The operations involved in the complex multiplier are four real multiplications and two real additions. However, by reordering the operations, real multiplications may be decreased to three, but the number of additions will increase to five. Xilinx IP cores have employed and shown several architectures based on this methodology. These architectural designs serve as a substitute for multiplier-less rotators. Adder compression is used in optimization to simplify the circuit. Lastly, a thorough analysis of the direct computation's

throughput, power, area, and latency in those structures is shown in comparison to the three-multiplier approach. The advantages of a System on a Chip (SoC) are space optimization, power efficiency, cheaper, reliability, and better performance. A complex multiplier is designed and implemented as an SoC.

4.1. Complex Multiplication

(a + ib) and (c + id) are two complex numbers and i is an imaginary value of -1:√-1.

The equation for this multiplication is given

$$(a+ib)(c+id) = ac+ibc+iad-bd=(ac-bd)+i(ad+bc) \quad (1)$$

Where real part, x = (ac-bd) and imaginary part, y = (ad+bc). The block diagram of a complex multiplier with four multipliers and two adders is presented in Figure 2.

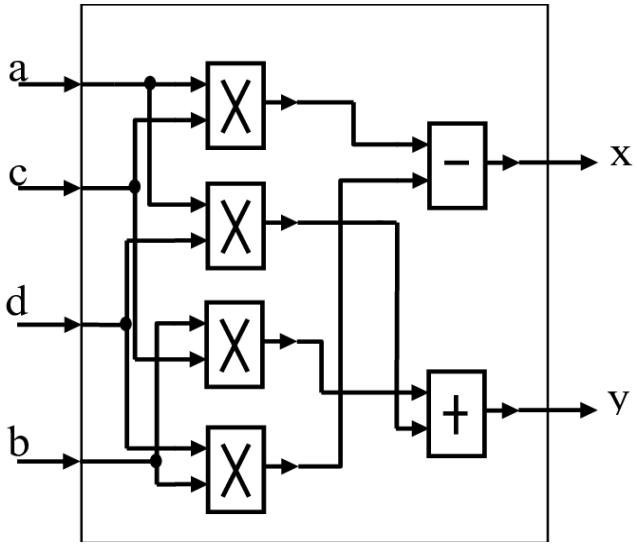

Fig. 2 Block diagram of complex multiplier

(Xr+jXi) and (Yr+jYi) are two complex numbers, and the real and imaginary product parts are,

$$Zr=XrYr-XiYi \quad (2)$$

$$Zi=XrYi+YrXi \quad (3)$$

$$Zr=Xr(Yr-Yi)+Yi(Xr-Xi) \quad (4)$$

$$Zi=Xi(Yr+Yi)+Yi(Xr-Xi) \quad (5)$$

One computation of Yi(Xr−Xi) is sufficient as it is in both calculations of Zr and Zi implementations. This will reduce the area as the multipliers needed are significantly reduced in overall implementation. The same structure can be redesigned using three real multipliers and five adders, as shown in Figure 3.

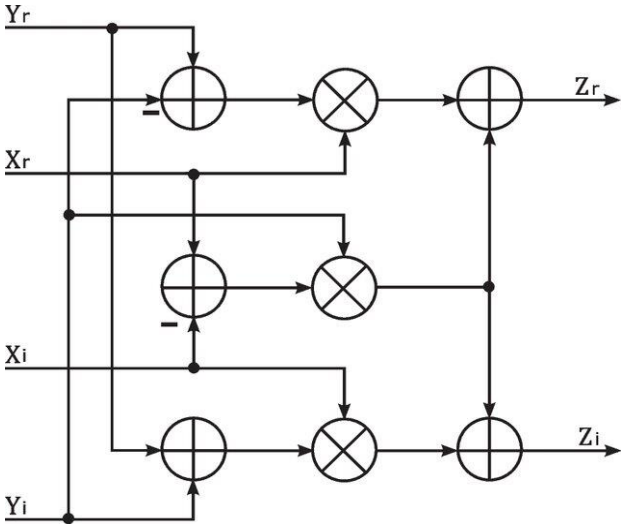

Fig. 3 Complex multiplier

However, the proposed design nullifies the trade-off among area, power, and speed parameters by optimizing all three characteristics at the same time, which is discussed below.

4.2. Hardware-Efficient Complex Multiplier

The low area leading to low cost and low power is attained by hardware-efficient design with a lower number of multipliers in the design, occupying minimum silicon area and consuming lower power. The proposed design uses only three multipliers instead of four through algebraic manipulation and pipelines the series of operations over a few clock cycles.

4.3. Pipelined Complex Multiplier

The pipeline concept is used to increase speed for any design by keeping subsequent data in the registers. In a single clock cycle, the data is assigned and ready for the next operation in the pipeline. The high-throughput complex multiplier is constructed for the highest frequency of operation required for DSP applications. Figure 4 shows the pipelined complex multiplier structure with varied sizes of inputs.

4.4. Scalable Complex Multiplier

The proposed complex multiplier works for different word length considerations, i.e., parameterized inputs, AWIDTH, and BWIDTH, which can be altered for the desired size of the design. The proposed implementation has two inputs with 16 bits and 18 bits. Input values will be in the range containing negative numbers. When these two numbers are added together, the output is 34 bits, to cover 16 bits of A and 18 bits of B, resulting in the product in AWIDTH + BWIDTH. It should be noted that 34 bits cannot include the sign, as shown for the input sizes mentioned in Figure 4. Consequently, the range of output values is also 34 bits on each register, P_{real} and $P_{imaginary}$. Similarly, a 64x64 bit complex multiplier has two inputs with 64-bit and 128-bit outputs [16, 17].

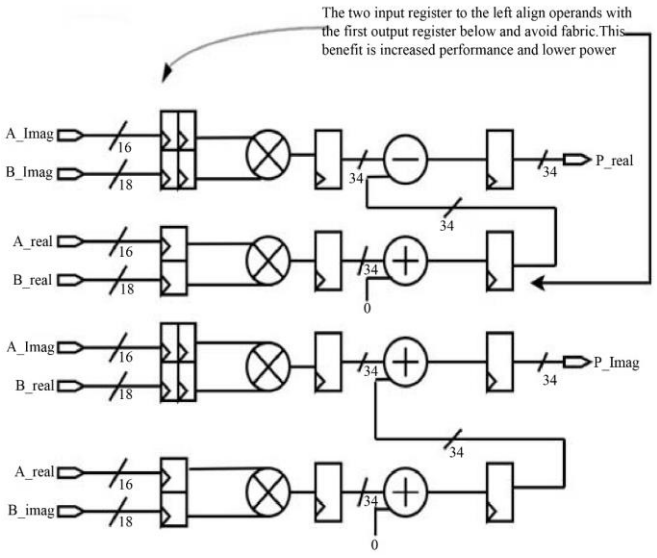

Fig. 4 16x18 Complex multiplier using the concept of Pipeline

5. Implementation of Complex Multiplier

Two Complex Multipliers with varied input sizes, for example, 16x18 and equal sizes 64x64, are designed and implemented using the Cadence design suite. Verilog HDL is written to design the complex multipliers and is simulated in NC Launch. The design is synthesized using Genus, and the layout is developed using Innovus. The designs are presented here before and after optimization for power and area. Though the design of the complex multipliers is meticulously done during coding, further optimization is done using the EDA tools. All the results during various stages of the design are presented below.

5.1. 16x18 complex multiplier

The scalability of input and output registers is explained in the previous section 4.4. The simulation outputs of the 16x18 multiplier are shown for various combinations of inputs in Figure 5.

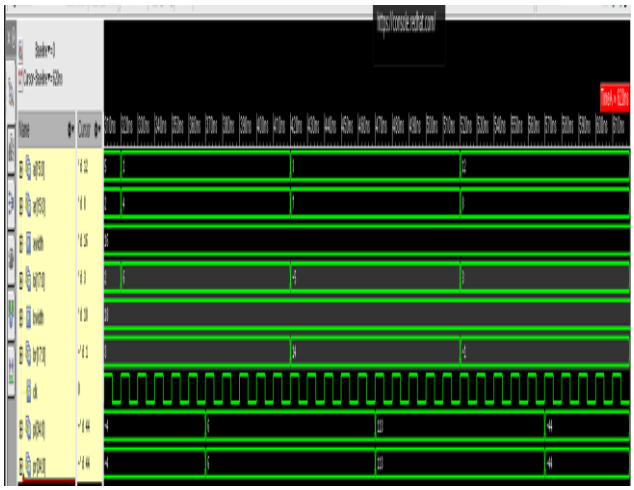

Fig. 5 Simulation outputs of 16x18 complex multiplier

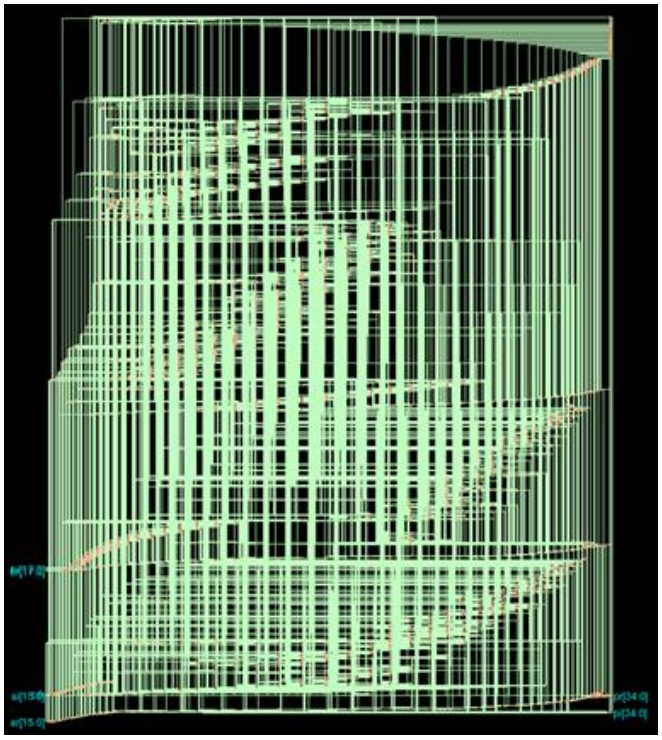

Fig. 6 RTL schematic of 16x18 complex multiplier before optimization

The RTL view of the 16x18 complex multiplier before and after optimization is shown in Figures 6 and 7. Figures 8 and 9 show the Layout of the 16x18 Complex Multiplier before and after optimization. Optimization is carried out for dynamic and leakage power, as well as speed.

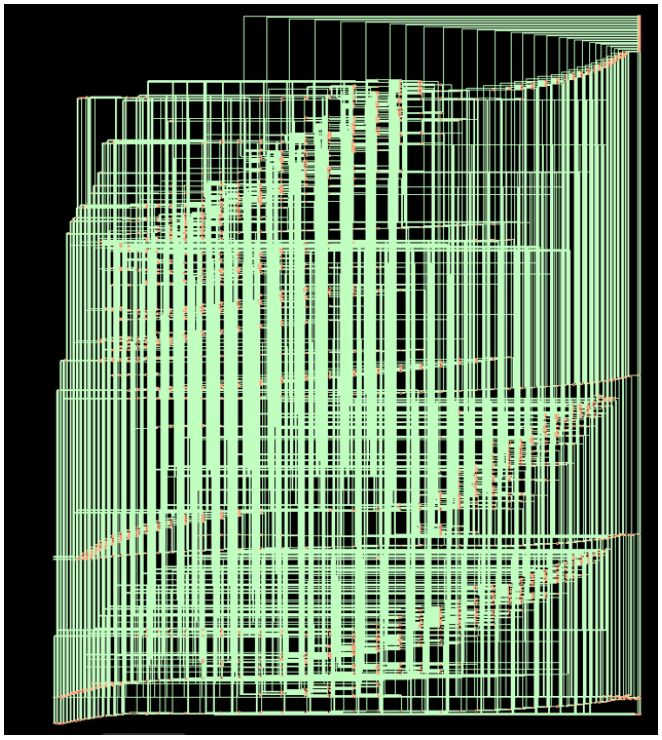

Fig. 7 RTL schematic of 16x18 complex multiplier after optimization

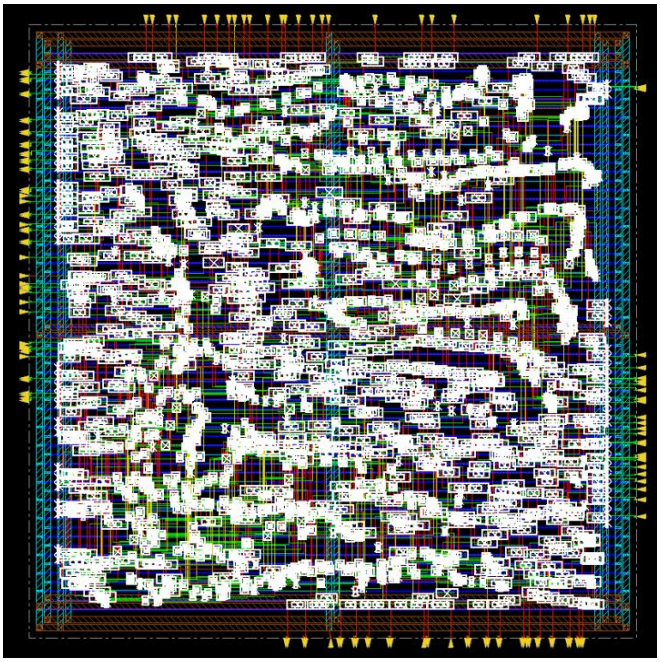

Fig. 8 Layout of 16x18 complex multiplier before optimization

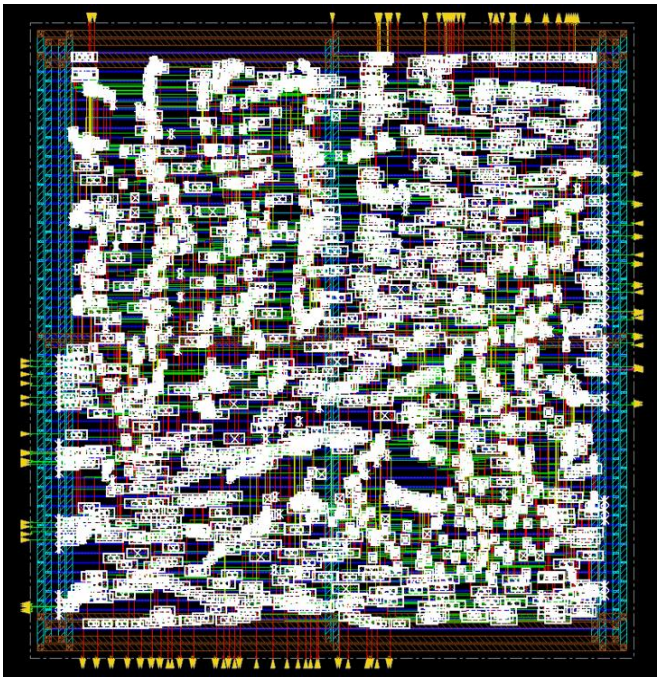

Fig. 9 Layout of 16x18 complex multiplier after optimization

5.2. 64x64 Complex Multiplier

The scalability of the input and output registers is explained in the previous section 4.4. The simulation outputs of the 64x64 multiplier for various input combinations are shown in Figure 10.

The Technology schematic view of the 64x64 complex multiplier before and after optimization is shown in Figures 11 and 12. Optimization is carried out on power and speed.

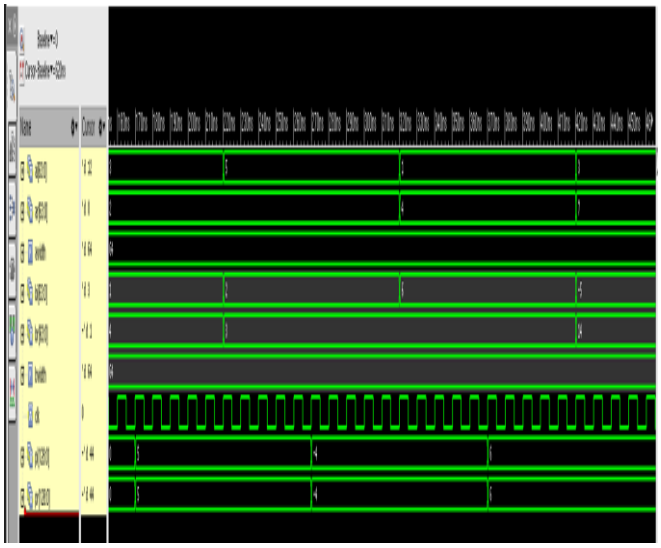

Fig. 10 Simulation outputs of 64x64 complex multiplier

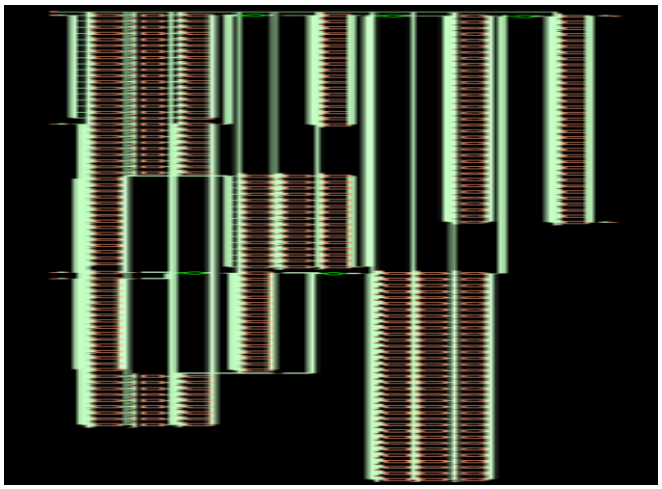

Fig. 11 Technology schematic of 64x64 complex multiplier before optimization

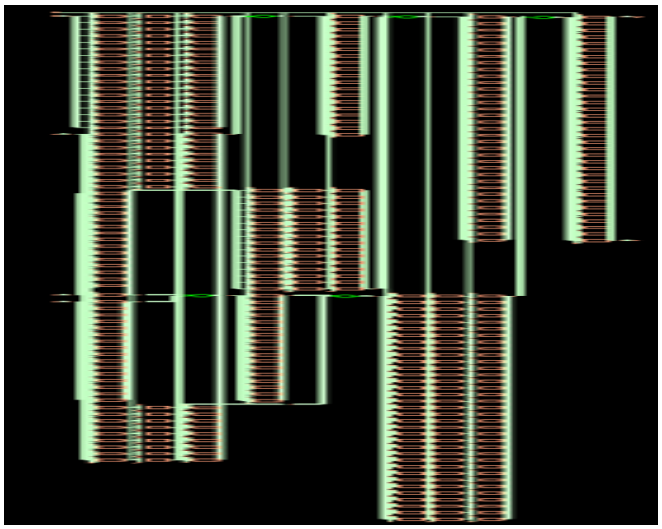

Fig. 12 Technology schematic of 64x64 complex multiplier after optimization

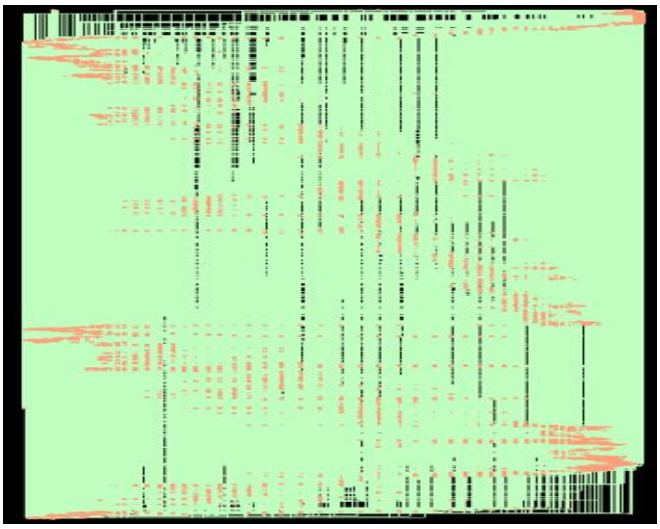

Fig. 13 RTL schematic of 64x64 complex multiplier before optimization

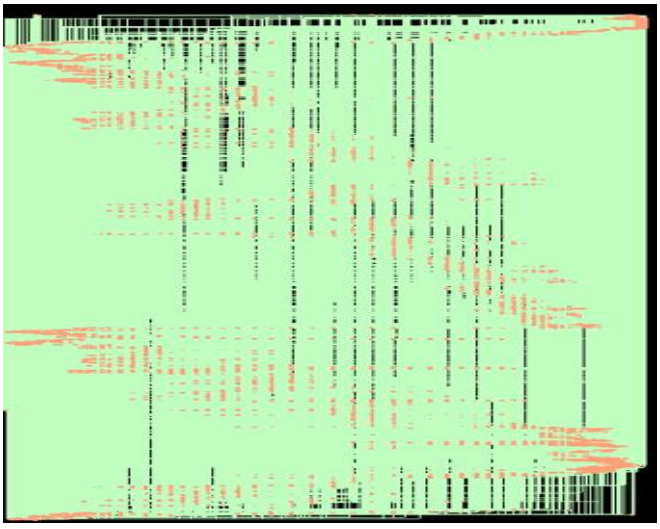

Fig. 14 RTL schematic of 64x64 complex multiplier after optimization

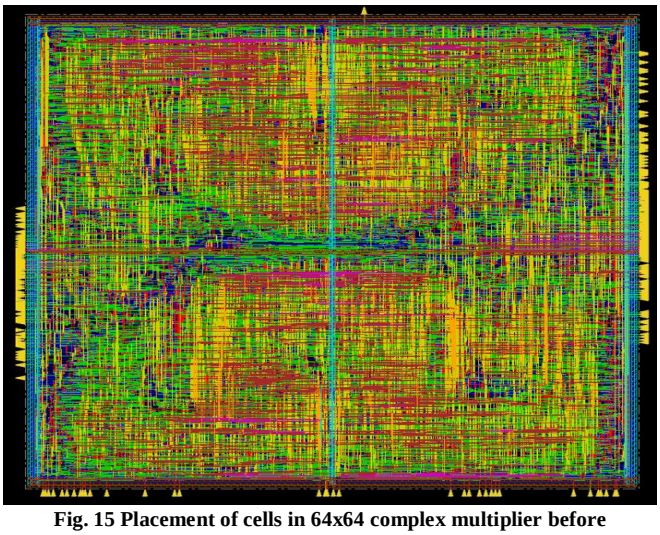

Fig. 15 Placement of cells in 64x64 complex multiplier before optimization

RTL Schematic of 64x64 Complex Multiplier before and after optimization is shown in Figures 13 and 14. Figures 15 and 16 show the Placement of cells in the 64x64 Complex Multiplier before and after optimization. Figures 17 and 18 show the Layout of the 64x64 Complex Multiplier before and after optimization.

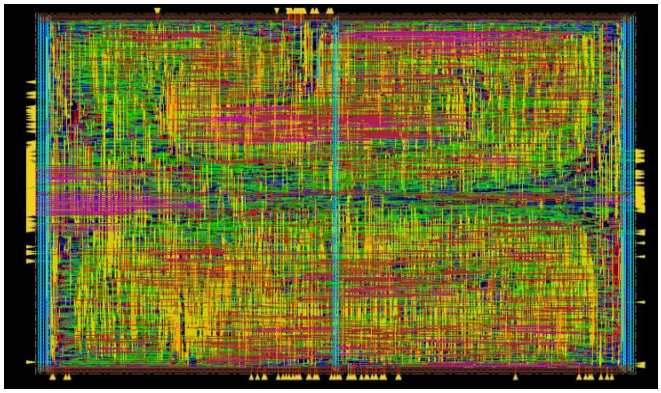

Fig. 16 Placement of cells in 64x64 complex multiplier after optimization

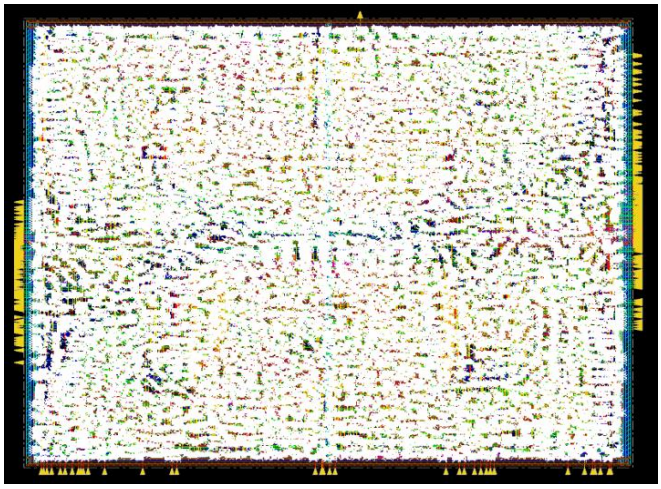

Fig. 17 Layout of 64x64 complex multiplier before optimization

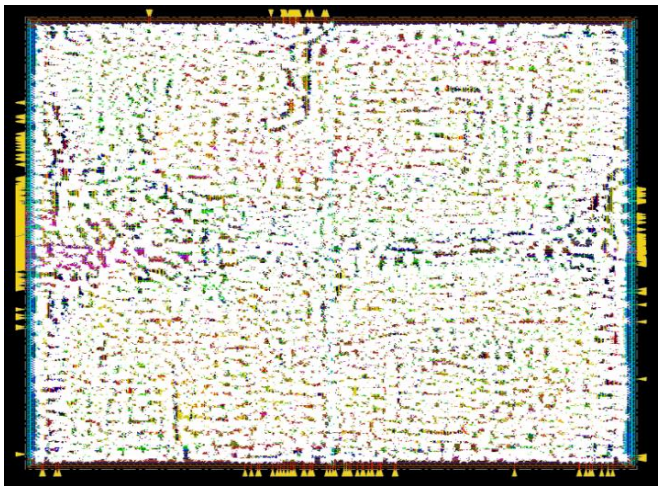

Fig. 18 Layout of 64x64 complex multiplier after optimization

5. Results and Discussion

The design of 16x18 and 64x64 complex multipliers, and the implementation is carried out as a single system on a chip or an IP core, in this work. The design includes efficient inclusion of hardware and high speed by pipeline method through effective HDL implementations. The scalable design allowed us to scale it from a 16x18 complex multiplier to a 64x64 complex multiplier.

The scalability is proved with the advantage of the design being that the design takes an equal required time to run the design in both cases, i.e., 16x18 and 64x64 complex multipliers. When power is concentrated. The standard cell count is increased by a very minute number to reduce the area occupied by the design in both cases of 16x18 and 64x64 complex multipliers, which in turn results in the cost of the chip.

Tables 1 and 2 show the Standard cells used in the design and the area occupied by these standard cells of 16x18 and 64x64 Complex Multipliers, respectively.

Table 1. Standard cells and area of 16x18 complex multipliers

Design Module	16x18 Complex Multiplier		
	Std. Cell Count	Cell Area (nm^2)	Required time (ns)
Before Optimization	1717	17832	9793
After Optimization	1712	17797	9762

Table 2. Standard cells and area of 64x64 complex multipliers

Design Module	64x64 Complex Multiplier		
	Std. Cell Count	Cell Area (nm^2)	Required time (ns)
Before Optimization	18577	169292	9823
After optimization	18564	169213	9792

The designs are optimized using Genus(Cadence) with a constraint of the dynamic power to leakage power ratio being 1:2. This optimization is important because, when the technology further advances, leakage power dominates the internal and switching power. Constrained dynamic to leakage power ratios can give chips customized for power.

Table 3. Power distribution components in 16x18 complex multipliers

Design Module	16x18 Complex Multiplier (Power in mW)			
	Leakage	Internal	Switching	Total
Before Optimization	0.083 (5.38%)	1.16 (75.71%)	0.2918 (18.91%)	1.54 100%
After Optimization	0.0765 5.30%	1.09 75.67%	0.274 19.03%	1.44

Table 4. Power distribution components in 64x64 complex multipliers

Design Module	64x64 Complex Multiplier (Power in mW)			
	Leakage	Internal	Switching	Total
Before Optimization	0.699 (6.1%)	7.97 (68.4%)	2.98 (25.5%)	11.6
After Optimization	0.638 (6.05%)	7.10 (68.2%)	2.68 (25.75%)	10.4

Tables 3 and 4 represent the power distribution components in 16x18 and 64x64 Complex Multipliers, respectively. The percentage contribution of each power distribution is mentioned in the table below. Total power reduction after optimization gained 10.3% of the total power. The proposed complex multipliers are compared with the 17x13 complex multiplier [18]. Table 5 shows the comparison of the area occupied by the 17x13 complex multiplier and the proposed 16x18 and 64x64 complex multiplier.

Table 5. Comparison of complex multipliers with respect to Area

Design Module	Area in mm^2
17x13 Complex multiplier[18]	0.28
16x18 Complex multiplier[Proposed]	0.0177
64x64 Complex multiplier[Proposed]	0.169

Finally, complex multipliers of variable sizes were designed with $2^n \pm P$ and $2^n \pm Q$. Power and area were optimized. Compared to existing work [16,18]. Area is significantly reduced when compared to the existing works.

6. Limitations

The HDL code is already written for the optimized design, which addresses the scalability to larger input sizes and adaptability to various technologies. The work evaluates key performance factors, such as leakage power, internal power, and switching power, for scalable complex multiplier designs. The work proved that the power and area were optimized. However, the work suggests that further research can be done to explore additional power reduction techniques. The area is automatically optimized by adopting advanced technologies along with the best architectures in the design. The work focuses on the design and implementation of complex multipliers, but the extension to other arithmetic functions can be done. A potential future research direction could extend the proposed techniques to the design of other arithmetic functions, such as accumulators, which are also crucial for DSP applications.

7. Conclusion

For the effective execution of intricate mathematical operations, such as the Fast Fourier Transform (FFT) and other signal processing methods, the efficient design of complex multipliers is critical. Hardware-efficient complex

multipliers with varied sizes of 16x18 and 64x64, resulting in 34-bit and 128-bit product terms, respectively, are designed that use fewer multipliers than the direct approach. The design is suitable for high-speed applications because of its pipelined architecture. The parameterized sizes of inputs offer scalability of designs with customized and target applications. This design also offers low power and minimum area to serve the very purpose of the SoC. The results are analysed before and after optimization constraints are applied to the power in the dynamic and leakage power ratios. HDL design to final layout is discussed in this work, presenting the advantages of the design being hardware efficient, pipeline, and scalable.

References

[1] B.S. Premananda et al., "Design of Area and Power Efficient Complex Number Multiplier," *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, Hefei, China, pp. 1-5, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[2] Anuja A. Bhat, and Mangesh N. Thakare, "Review on 32-Bit IEEE 754 Complex Number Multiplier Based on FFT Architecture using BOOTH Algorithm," *International Journal of Engineering and Computer Science*, vol. 6, no. 2, pp. 20308-20312, 2017. [Google Scholar] [Publisher Link]

[3] Pedro Paz, and Mario Garrido, "Efficient Implementation of Complex Multipliers on FPGAs using DSP Slices," *Journal of Signal Processing Systems*, vol. 95, no. 4, pp. 543-550, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] A.P. Pascual, J. Valls, and M.M. Peiro, "Efficient Complex Number Multipliers Mapped on FPGA," *ICECS'99. Proceedings of ICECS '99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.99EX357)*, Paphos, Cyprus, vol. 2, pp. 1123-1126, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[5] Razaidi Hussin et al., "An Efficient Modified Booth Multiplier Architecture," *2008 International Conference on Electronic Design*, Penang, Malaysia, pp. 1-4, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[6] D. Padma Shri, D. Subba Rao, and G. Vijay Goud, "VLSI Design of High Performance Complex Multiplier," *International Refereed Journal of Engineering and Science (IRJES)*, vol. 3, no. 4, pp. 77-84, 2014. [Google Scholar] [Publisher Link]

[7] Marc Belleville et al., "Designing Digital Circuits with Nano-Scale Devices: Challenges and Opportunities," *Solid-State Electronics*, vol. 84, pp. 38-45, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[8] Zia Abbas, and Mauro Olivieri, "Impact of Technology Scaling on Leakage Power in Nano-Scale Bulk CMOS Digital Standard Cells," *Microelectronics Journal*, vol. 45, no. 2, pp. 179-195, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] 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]

[10] T. Esther Rani, Dr. Rameshwar Rao, and 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]

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

[12] Juan-Antonio Carballo et al., "ITRS 2.0: Toward a Re-Framing of the Semiconductor Technology Roadmap," *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, Seoul, Korea (South), pp. 139-146, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[13] Gordon E. Moore, "Cramming More Components Onto Integrated Circuits, Reprinted from Electronics, Volume 38, Number 8, April 19, 1965, pp.114 ff.," *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33-35, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[14] H.-S.P. Wong et al., "Nanoscale CMOS," *Proceedings of the IEEE*, vol. 87, no. 4, pp. 537-570, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[15] What is a System on a Chip (SoC)?, Synopsys, 2025. [Online]. Available: https://www.synopsys.com/glossary/what-is-system-on-a-chip.html

[16] Ren Ping Wang, "Full-Custom Design and Implementation of High-Performance Multiplier," *Advanced Materials Research*, vol. 631-632, pp. 1445-1451, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[17] A. Lakshmi, P. Chandrasekhar Reddy, Esther Rani Thuraka, "Full Custom Design and Implementation of 12-Bit Complex Multiplier," *SSRG International Journal of Electronics and Communication Engineering*, vol. 12, no. 8, pp. 40-50, 2025. [CrossRef] [Publisher Link]

[18] SoC Development Overview, AnySilicon, 2020. [Online]. Available: https://anysilicon.com/soc-development-overview

[19] Weidong Li, Shengxian Zhuang, and Lars Wanhammar, "An Efficient Pipelined Complex Multiplier," 1997. [Google Scholar]

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