iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.69346

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Design and Implementation of 32-bit Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol using Cadence

P. Uma Maheshwari[1], J. Siddhu Nayak[2], Sreeja Mandava[3], M. Shoukath Ali[4]
[1, 2, 3]*4th Dept. of ECE, Geethanjali College of Engineering and Technology, India*
[4]*Associate. Professor, Dept. of ECE, Geethanjali College of Engineering and Technology, India*

Abstract: ***The Advanced Microcontroller Bus Architecture (AMBA), developed by ARM, is a widely adopted open standard that facilitates high-performance, low-power communication between functional blocks in System-on-Chip (SoC) designs. It enables modularity, scalability, and reusability in SoC development through a family of protocols including AHB (Advanced High-performance Bus), AXI (Advanced extensible Interface), and APB (Advanced Peripheral Bus). Among these, APB is specifically designed for connecting simple, low-bandwidth peripherals such as UARTs, timers, and GPIOs due to its low complexity and reduced power consumption. This work presents the Design and Implementation of 32-bit AMBA APB Protocol using Cadence, showcasing a complete digital VLSI design flow. The project encompasses RTL coding in Verilog HDL, functional verification using testbenches, and synthesis using Cadence Genus to evaluate area, power, and delay reports. Following synthesis, the design undergoes placement and routing in Innovus, resulting in a GDSII layout generation. Post-synthesis and post-layout analyses are carried out to ensure that the design meets performance and physical constraints. The final GDS file confirms successful implementation from RTL to layout, making the protocol ready for silicon realization. This project demonstrates the integration of bus protocol design within a professional EDA environment, reflecting industry practices in hardware development and physical design.***

Keywords: AMBA, APB, AXI, AHB, SoC, UART, GPIO, RTL, HDL, EDA, GDSII.

I. INTRODUCTION

In the era of rapidly evolving semiconductor technologies, System-on-Chip (SoC) designs have become the backbone of modern digital systems, integrating processors, memory units, and various peripherals onto a single silicon chip. As SoCs grow in complexity, the demand for efficient, low-power, and standardized communication protocols between internal components becomes critical. To address this need, ARM developed the Advanced Microcontroller Bus Architecture (AMBA), a widely adopted open standard that enables modular design, high-performance data exchange, and power-efficient communication across multiple functional blocks in a system. The AMBA protocol suite includes several key protocols—AXI (Advanced extensible Interface) for high-speed, high-throughput communication; AHB (Advanced High-performance Bus) for burst-based pipelined data transfer; and APB (Advanced Peripheral Bus), which is specifically tailored for low-bandwidth and low-power peripheral connectivity. APB is intentionally simplified to reduce area and power consumption, making it highly suitable for peripherals such as UARTs, timers, keypads, GPIOs, and interrupt controllers. Its non-pipelined, two-phase transfer mechanism ensures deterministic communication with minimal logic, aligning with the needs of energy-efficient embedded systems. Due to its practical utility, the APB protocol is not only prevalent in industry-grade SoC designs but is also a significant area of interest in academic and research domains. It provides an ideal foundation for understanding the principles of digital protocol design, signal interfacing, and subsystem integration. Mastery of such protocols equips engineers with the ability to develop and verify reliable interconnects in real-time embedded systems. To model, simulate, and validate such protocols, Verilog HDL (Hardware Description Language) remains the industry standard for register-transfer level (RTL) design. Its flexibility allows for precise definition of logic behavior, module hierarchies, and timing characteristics. Complementing this, EDA tools from Cadence, such as Genus for logic synthesis and Innovus for placement, routing, and layout generation, offer an end-to-end design environment that reflects real-world VLSI development flows.

©IJRASET: All Rights are Reserved

These tools enable designers to obtain critical metrics such as area, timing, and power, and ensure that the final layout complies with both logical functionality and physical design constraints. The increasing adoption of the AMBA APB protocol in both commercial and academic SoC applications underscores the importance of a comprehensive understanding of its architecture and implementation. By focusing on the design and implementation of a 32-bit AMBA APB protocol using Cadence tools, this work provides both a theoretical and practical exploration of the protocol's functionality. The implementation is carried out in Verilog HDL, with verification through simulation and synthesis, leading up to the generation of a GDSII layout. This enables a deeper insight into the entire digital VLSI design flow, from RTL to silicon-ready layout, and demonstrates how foundational bus protocols contribute to building efficient, low-power SoC subsystems.

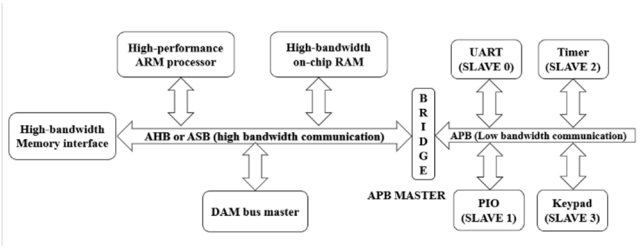


Fig. 1, AMBA Bus Architecture diagram

II. PROBLEM STATEMENT - PROPOSED SOLUTION

In modern SoC designs, there is a growing need for lightweight and efficient communication protocols to interface with low-bandwidth peripheral devices. The AMBA APB protocol is widely adopted for such use cases; however, designing a custom 32-bit APB interface that is both functionally accurate and physically realizable poses several design challenges. These include ensuring protocol correctness through thorough verification, achieving synthesis and timing closure, and managing physical design constraints such as area and congestion. Despite the availability of commercial IPs, educational and research contexts often lack access to fully developed design flows that span from RTL description to GDSII layout. This project addresses the need for a complete and practical implementation of a 32-bit AMBA APB protocol using the Cadence digital design flow, covering all stages from RTL design and functional verification to synthesis and physical layout. The problem lies in integrating these stages into a cohesive workflow that results in a silicon-ready design, while also ensuring compliance with protocol specifications and optimizing for design constraints.

To address the challenges of designing a verified and synthesizable 32-bit AMBA APB protocol, this work proposes a structured implementation methodology using the Cadence digital design toolchain. The solution begins with the RTL design of the APB master and slave modules in Verilog, adhering strictly to the AMBA APB protocol specifications. A comprehensive testbench is developed to simulate valid operational scenarios such as write and read operations, error handling, and finite state machine (FSM) transitions. Functional verification is performed using Cadence Nclaunch/NCSim, which provides cycle-accurate simulation and waveform analysis. Following successful simulation, the design is synthesized using Cadence Genus, with timing and design constraints specified via an SDC file to generate a gate-level netlist. This netlist is then passed through the Cadence Innovus Implementation System for full physical design, which includes floorplanning, power planning, placement, clock tree synthesis (CTS), global and detailed routing, and post-route optimization. The design is subjected to physical verification checks such as DRC and LVS to ensure fabrication readiness. This end-to-end solution not only ensures protocol compliance and functional correctness but also validates the design across synthesis and physical implementation stages. By using an industry-standard EDA flow, the project demonstrates a scalable and educationally valuable reference for digital IP development, especially in low-power peripheral communication systems.

III. MASTER-SLAVE COMMUNICATION IN APB

The Advanced Peripheral Bus (APB) is a part of the AMBA (Advanced Microcontroller Bus Architecture) protocol family developed by ARM, specifically designed for low-bandwidth and low-power peripheral communication in System-on-Chip (SoC) designs. APB is ideal for interfacing with simple peripherals such as UARTs, GPIOs, timers, and interrupt controllers, which do not require high-speed data transfers. The AMBA APB typically consists of an APB Bridge/Master and multiple APB Slaves, and it is designed to interface efficiently with several peripheral devices in a System-on-Chip (SoC). The APB Bridge, which acts as the master on the APB, receives high-speed transactions from an upstream bus like AHB or AXI and converts them into APB-compliant transactions. This bridge essentially serves as the control unit, managing all communication across the APB interface.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

As shown in the block diagram, the APB Bridge/Master initiates and controls read/write operations by generating the required control signals (PCLK, PADDR, PWRITE, PWDATA, PSEL, PENABLE) and ensuring synchronization throughout the transaction. Each APB Slave, such as UART, Timer, or GPIO, is connected via its dedicated select signal (PSELx) and is responsible for responding to the master's commands by returning data (PRDATA), indicating readiness (PREADY), or asserting errors (PSLVERR) when necessary.

The architecture is designed such that only one slave responds at a time, ensuring conflict-free data transfer. This hierarchical setup—with the bridge acting as both a slave on the high-performance system bus and a master on the APB—allows seamless integration of low-bandwidth peripherals into complex SoC environments. The simplicity and modularity of APB make it highly effective for low-power and area-constrained applications.

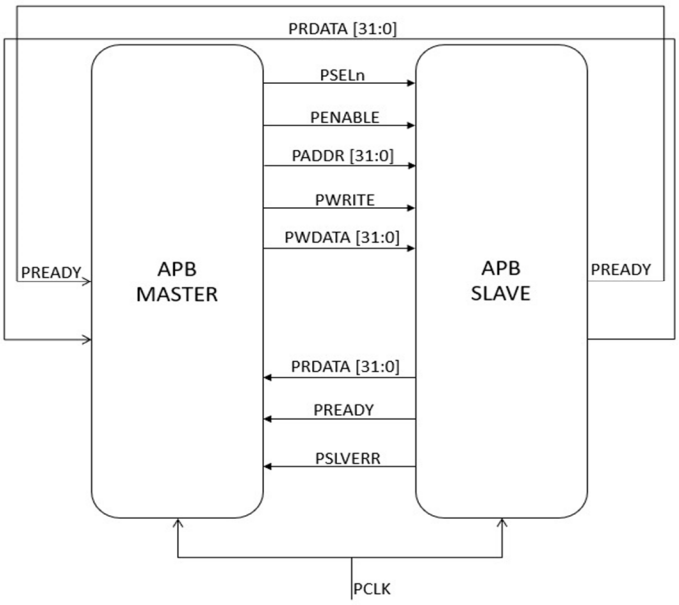


Fig. 2, Signal-Level Interface Between APB Master and Slave

A. APB Master Description

The APB Master is responsible for initiating all communication within the APB system. It controls the timing, direction, and flow of data to and from the slave devices. The master begins a transfer by asserting the PSELx signal to select the appropriate slave, placing the target register address on the PADDR bus, and setting the PWRITE signal to indicate the type of operation—write (high) or read (low). In the case of a write operation, the master also places valid data on the PWDATA bus. Once all the setup signals are valid, it asserts the PENABLE signal, indicating that the access phase has begun. The master then waits for the PREADY signal from the slave, which tells the master whether the slave is ready to proceed. If PREADY is low, the master will hold the transaction, effectively inserting wait states until the slave responds. During read operations, the master captures the data presented by the slave on the PRDATA bus once PREADY is high. If an error is encountered, the slave may assert the PSLVERR signal, which the master can use for error handling logic. The master manages all timing relative to the PCLK signal, ensuring synchronous data transfers.

B. APB Slave Description

APB Slave is the target of communication and responds to requests made by the master. It remains inactive until selected by the PSELx signal. Upon being selected, the slave decodes the address from the PADDR bus and either prepares to accept data (in a write operation) or to supply data (in a read operation). When PENABLE is asserted by the master, the slave performs the requested action. For a write, it captures the data on the PWDATA bus and stores it at the location specified by the PADDR. For a read, it places the requested data onto the PRDATA bus so the master can receive it. Once the slave completes its operation, it asserts the PREADY signal, informing the master that the transfer can finish. If the slave encounters any fault—such as an invalid address or unsupported operation—it may assert the PSLVERR signal to indicate an error condition. Like the master, the slave also operates in sync with the rising edge of the PCLK signal to maintain timing integrity.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

IV. RTL DESIGN USING VERILOG HDL

The Register Transfer Level (RTL) design of the AMBA APB protocol was developed using Verilog HDL, modelling a 32-bit communication interface between the master and slave. The design encapsulates all essential APB signals including PCLK, PRESETn, PADDR, PSEL, PENABLE, PWRITE, PWDATA, PREADY, PSLVERR, and PRDATA. The APB slave was implemented with a 32x32-bit memory block to store and retrieve data during read and write operations. A finite state machine (FSM) was used to control the protocol's operational flow through three defined states: IDLE, SETUP, and ACCESS.

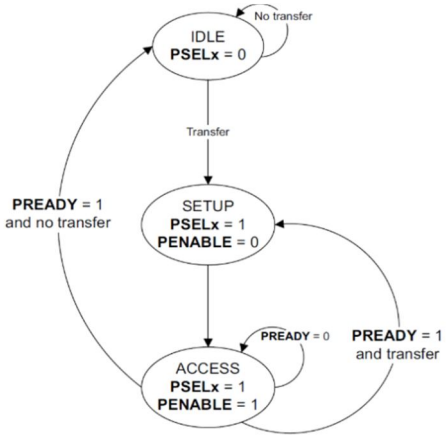


Fig. 3, FSM for APB Protocol

The control logic ensures that data transfers occur only when valid conditions are met—i.e., when the master asserts PSEL followed by PENABLE. On write cycles, the slave stores the incoming data (PWDATA) into memory at the specified address (PADDR), whereas in read cycles, the corresponding memory content is driven to the PRDATA output. Error detection was implemented via the PSLVERR signal to handle out-of-bound memory accesses. To ensure design correctness and timing closure during synthesis, the corresponding SDC file was created. The SDC file defines the clock (PCLK) period as 20 ns (50 MHz), sets appropriate input/output delays for all ports, accounts for clock uncertainty, and defines false paths (e.g., reset paths). The set_drive and set_load constraints help refine timing estimates for synthesis. Overall, the Verilog design was structured for compatibility with digital design tools and ease of integration in downstream physical design flows.

V. FUNCTIONAL VERIFICATION USING VERILOG TESTBENCH FOR APB

The functional verification of the 32-bit AMBA APB protocol was carried out using a comprehensive Verilog testbench within the Cadence digital design environment. The primary objective of the testbench was to validate the correctness of data communication and control handshaking between the APB master and slave modules. Key signals such as PSEL, PENABLE, PWRITE, PWDATA, PRDATA, and PREADY were closely monitored to ensure accurate execution of both write and read operations. The simulation confirmed that the design transitioned correctly through all three finite state machine (FSM) states—IDLE, SETUP, and ACCESS—based on the status of the control signals. During the write phase, when PWRITE is asserted, valid data from PWDATA is driven to the slave and correctly latched with the corresponding address. During read operations, PWRITE is deasserted, and the expected data is retrieved from the slave on PRDATA, verifying proper memory behaviour and data consistency. The waveform generated from the Cadence simulation (as shown in Figure 4) clearly illustrates these signal transitions over time, confirming protocol compliance and correct functionality. The PREADY signal was observed to be asserted during valid access phases, indicating that the slave was ready to complete the transaction. Additionally, while not explicitly shown in the waveform, the design includes logic to assert PSLVERR during invalid memory accesses, adding robustness to the protocol implementation. The synchronous nature of the protocol is maintained throughout, with address and data signals properly aligned to the clock edge, and the reset (PRESETn) held high to keep the system in an active operational state. The entire design, simulation, and implementation flow was executed using industry-standard Cadence EDA tools. Functional simulation and waveform analysis were performed using Cadence Nclaunch (NCSim), which provided cycle-accurate signal visualization for verification.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

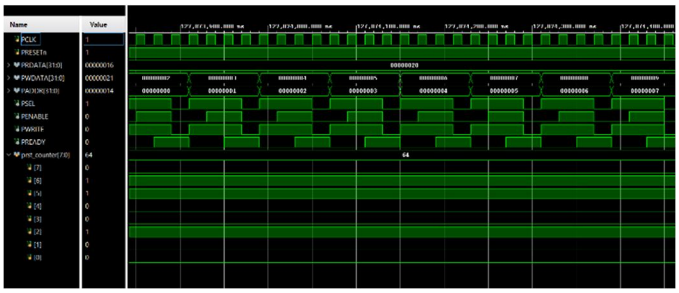


Fig. 4, APB protocol waveform result

VI. RTL SYNTHESIS

RTL (Register Transfer Level) synthesis is the process of transforming an RTL description of a design, typically written in Verilog or VHDL, into an optimized gate-level netlist consisting of logic gates, flip-flops, and other hardware components. This gate-level netlist serves as the foundation for subsequent stages of design, such as physical design, layout, and implementation, while ensuring that the design meets specific criteria such as timing, area, and power requirements. The synthesis process involves three key stages: translation, mapping, and optimization. During translation, the RTL code is converted into an intermediate representation, which is then mapped to specific target technology cells in the mapping stage. Finally, optimization techniques such as area minimization, timing refinement, and power reduction are applied to enhance the design's efficiency, ensuring that it meets the required constraints.

Cadence Genus is a comprehensive RTL synthesis tool used in the VLSI design flow to automate the conversion of RTL descriptions into optimized gate-level representations. By utilizing standard cell libraries for the target technology, Genus facilitates the synthesis process, ensuring efficient area, power, and timing performance. The tool supports a variety of optimization techniques, including area minimization, power reduction, and timing optimization, enabling designers to meet specified design constraints. Seamlessly integrating with other Cadence tools, Cadence Genus ensures a smooth transition from RTL to gate-level netlist, providing detailed reports that evaluate the synthesized design's performance and allowing for further analysis and refinement.

TABLE 1. POST-SYNTHESIS RESULT OF 32-BIT AMBA APB PROTOCOL

METRIC	PARAMETERS	VALUE
Area	Cell Count	1908
	Cell Area	20086.612
	Total Area	20086.612
Timming	Startpoint	PENABLE (R)
	Endpoint	PENABLE (R)
	Required Time	14500 ps
	Data Path Delay	2310 ps
	Arrival Time	7310 ps
	Slack	+7190 ps (MET)
Power	Leakage Power	128469.110 nW
	Dynamic Power	69023.249 nW
	Total Power	197492.359 nW

After synthesizing the 32-bit AMBA APB protocol using Cadence Genus, the design occupied a total cell area of approximately 20086.6 µm² with 1908 standard cells. The synthesis met all timing constraints, achieving a positive slack of +7190 ps between PENABLE and PSLVERR under the PCLK domain. Power analysis indicated a total power consumption of 197.49 µW, with leakage and dynamic power contributing 128.47 µW and 69.02 µW, respectively. These results validate the design's efficiency in terms of timing, area, and power under the specified constraints.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

VII. RESULT AND DISCUSSION

Physical Design is the process of converting a gate-level netlist into a layout ready for chip fabrication. Using Cadence Innovus, this flow includes key steps like floorplanning, placement, Clock Tree Synthesis (CTS), routing, and layout verification.

The images shown are the final layout views of our VLSI design project, generated using Cadence Innovus as part of the physical design flow from RTL to GDSII. These layouts represent the design both before and after optimization, showing the evolution of routing, placement, and overall cell organization.

The physical design process began with floor planning, where the core area was defined and power planning was configured. In this step, we inserted power rings and rails using horizontal Metal 9 and vertical Metal 8 layers. The I/O pads were placed around the periphery, and macro placement (if any) was adjusted to ensure routability and timing closure.

After floor planning, we performed power planning to distribute power (VDD) and ground (VSS) across the chip using metal layers, followed by placement of standard cells. Placement aimed to minimize wirelength and prepare the design for optimal routing. Post-placement, Clock Tree Synthesis (CTS) was implemented to ensure minimal skew and proper clock signal distribution across all sequential elements.

Routing was then performed in two phases: global and detailed. The layout views clearly show the dense interconnects between cells and modules across multiple metal layers. As shown in figure 5, (PRE-OPT) shows the layout before final optimization wires appear denser and less uniform, with some paths likely having longer delays or more congestion.

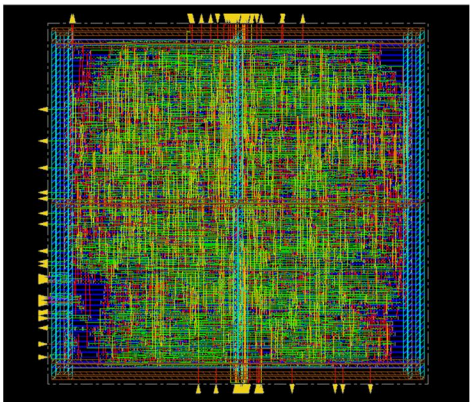


Fig. 5, Pre-Optimization Layout result

In contrast, figure 6, (POST-OPT) reflects improved routing after optimizations, such as better buffer insertion, delay fixing, and net rerouting for timing improvement and DRC clean-up

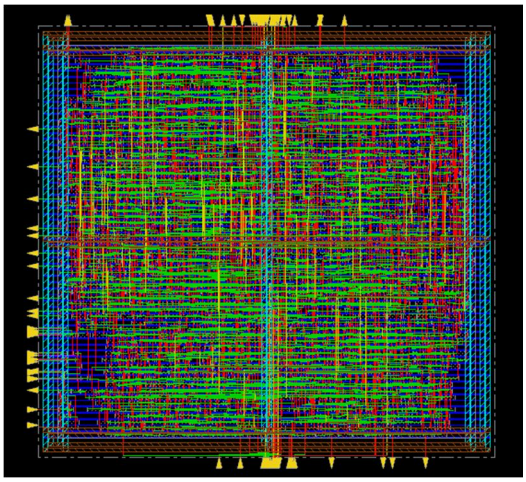


Fig. 6, Post-Optimization Layout result

Finally, we generated the GDSII file, which represents the design in a format ready for fabrication. Basic verification steps like DRC (Design Rule Check) and LVS (Layout Versus Schematic) were performed to confirm design correctness and consistency with the schematic.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

These layout results reflect the successful completion of our physical design project. The transition from pre- to post-optimization demonstrates our understanding of key physical design steps, including floor planning, power planning, placement, CTS, routing, and layout verification using Cadence Innovus.

VIII. CONCLUSION

In this project, the design and implementation of a 32-bit AMBA APB (Advanced Peripheral Bus) protocol were successfully carried out using Verilog HDL for RTL modeling and Cadence tools for synthesis and physical design. Verification was performed through an efficient Verilog Testbench, which tested various operational scenarios including Read, Write, and Error cycles, ensuring the functional correctness of the protocol. The testbench incorporated reusable tasks and self-checking mechanisms for validating different data paths and memory operations. The integration of design and testbench through interface logic allowed seamless simulation and verification. Random data and index-based test cases were used to verify the robustness of the design. Simulation results confirmed that the data retrieved during the read cycle accurately matched the data written, confirming the correctness of the protocol behaviour. The complete flow from RTL to GDSII was executed using Cadence Innovus, covering steps such as synthesis, floor planning, power planning, placement, clock tree synthesis (CTS), routing, and final GDSII generation. The final layout was verified for design rule correctness and functional equivalence.

This work demonstrates a full-cycle implementation of the AMBA APB protocol, from design and verification to physical realization. For future scope, low-power design techniques can be incorporated to further enhance the efficiency and performance of the APB protocol in SoC environments.

REFERENCES

[1] Santhi Priya Sarekokku , K.Rajasekhar,“ Design and Implementation of APB Bridge Based on AMBA AXI 4.0”, Volume 01, Issue 09 (November 2012)

[2] Shankar,Dipti Dipti Girdhar and Neeraj Kr.Shukla, “Design and Verification of AMBA APB Protocol”, International Journal of Computer Applications (IRJET), 95(21):29-35 DOI:10.5120 /16720-7047.

[3] J Mukunthan,Daniel Raj Androse, S Keerthivandana, R Kiruthika,T Shruthi and S Snegasri “Design and Implementation of AMBA APB Protocol”, IOP conference Series Materials Science and Engineering, 1084(1):012050. DOI: 10 .1088/1757-899X/1084/1/012050.

[4] T. Koundinya, “Design and Implementation of AMBA based AHB AHB2APB Bridge”, IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), Volume 7, Issue 3, pp. 42-45, 2017, Volume 11, Issue 06 (June 2022).

[5] G Prathibha . Ambika Sekhar, “APB Bridge Based on AMBA 4.0”, Volume 02, Issue 09 (September 2013).

[6] ARM, "AMBA Protocol Specification 4.0" www.arm.com, 2010.

[7] ARM, "AMBA APB Protocol Specification Version 2.0", www.arm.com

[8] M. kiran Kumar, Amrita Sajja, Dr, Fazal Noorbasha, Design and FPGA Implementation of AMBA APB bridge with clock skew minimization Technique (IOSR-JVSP),Vol.7,Issue.3,June 2017

[9] Fei Hong, Aviral Shrivastava, and Jongeun Lee, Return Data Interleaving for Multi-Channel Embedded CMPs Systems, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, No. 7, July 2012, pp 1351-1354.

[10] Chenghai Ma et.al., , University of Bologna, ForlÃ¬, Italy, Design and Implementation of APB Bridge based on AMBA 4.0, in Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC), 2010 5th.

[11] Shrivastav, A. , et. al., Dept. of Electron. & Comm., Priyatam Inst. of Technol. & Manage., Indore, India, Performance Comparison of AMBA Bus-Based System- On-Chip Communication Protocol, International Conference on Communication Systems and Network Technologies, 2011,pp 449-454.

[12] Muhlbach,S., et. al., Hamburg Univ. Of Technol., Hamburg, Secure And Authenticated Communication In Chip-Level Microcomputer Bus Systems With Tree Parity Machines, International Conference On Embedded Computer Systems: Architectures, Modeling And Simulation, 2007, Pp 201-208.

[13] Muhlbach,S., et. al., Hamburg Univ. Of Technol., Hamburg, Secure And Authenticated Communication In Chip-Level Microcomputer Bus Systems With Tree Parity Machines, International Conference On Embedded Computer Systems: Architectures, Modeling And Simulation, 2007, Pp 201-208.

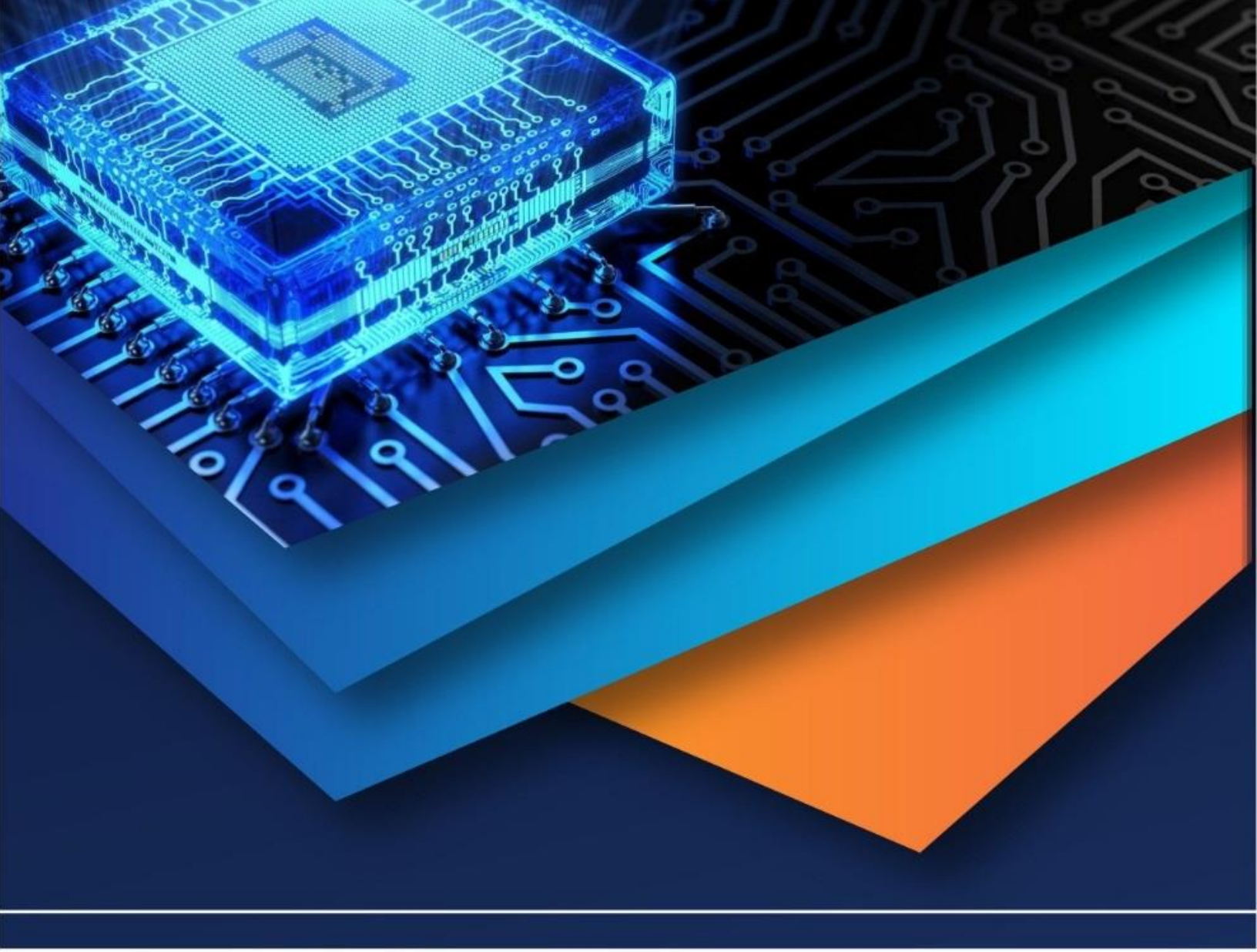
[14] Benini, Roberto Zafalon, Scalability Analysis of Evolving SoC Interconnect Protocols, Int. Symposium on System-on-Chip, 2004. Lukai Cai, Daniel Gajski, Transaction level modeling: an overview, in Proceedings of the 1st IEEE/ACM/IFIP international conference on[Hardware/software codesign and system synthesis, October 2003].

[15] S.Nithya, S.Shobaa,“ RTL to GDSII Implementation using Cadence Tool ”, IJARCCE, Volume 6, Issue 4, April 2017

[16] G. Dineshkumar, S. Muthuselvi,“ ASIC Implementation of 32-bit AMBA APB Protocol”, International Journal of Engineering Research and Technology (IJERT), Vol. 6, Issue 3, 2018.

[17] Dr.S.Uma Maheswari, M. Jothimani, “A Study on ASIC Physical Design Flow using EDA Tools”, Conference: IEEE International Conference on Circuits and Systems (ICCS), 2019.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)