ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN:2454-9940 www.ijsem.org

Vol 20, Issuse.2 April 2026

Synthesis for AMBA APB BUS

[1] K.Anuradha,[2] Burra Madhavi,[3] Samala Chandana,[4] Pothu Vivek,[5] Yamjala Dheeraj Kumar,

[1]Assistant Professor, Department of ECE, Narsimha Reddy Engineering Collage, Maisammaguda(V), Kompally, Telangana.

[2,3,4,5] Student, Department of ECE, Narsimha Reddy Engineering Collage, Maisammaguda(V), Kompally, Telangana.

Abstract—

A modular and scalable architecture that minimizes redesign efforts and integration time is provided by the AMBA (Advanced Microcontroller Bus Architecture) standard, which is extensively used in System-on-Chip (SoC) designs and allows effective administration of IP cores. Versions of AMBA include new communication interfaces such APB, ASB, AHB, AXI, ACE, and CHI, which stand for Advanced eXtensible Interface, Advanced System Bus, Advanced High-Performance Bus, Coherent Hub Interface, and Advanced eXtensible Interface, respectively. Using Verilog HDL as a foundation and governed by VLSI principles, comprehensive RTL (Register Transfer Level) schematics facilitate design optimization. The suggested architecture was synthesised using a brand new codebase that was designed from the ground up. With a total power optimization of 17% and switching power optimization of 34% and cell area optimization of 7%, the optimal performance is seen for effort high condition. Signals, Real-Time Language, APB Bus Protocol, AMBA Bus Architecture, Master, Slave

I. INTRODUCTION

The very large scale integration (VLSI) design flow is a methodical procedure for making ICs. Specifications for the architecture, system, and sub-blocks must be defined at an early stage. The RTL Design team creates these building blocks with the use of AMBA buses like APB, AHB, and AXI, which improve communication, and hardware description languages like Verilog and System Verilog. To distribute electricity efficiently throughout the chip, essential phases are floor layout and power planning. Testing the final layout, also known as the GDSII stream, ensures the chip satisfies all requirements before it is released. To control on-chip interconnects and ease SoC designs, ARM established the open-standard protocol AMBA in 1996. This makes it possible to simplify designs with several processors by integrating and coordinating their functional blocks on a single chip. Notable changes to AMBA throughout the years include the introduction of the speedier AHB bus in version 2, and the addition of sophisticated features like ACE and the CHI in versions 4 and 5, respectively. A major player in contemporary chip design, AMBA has been greatly improved upon.

II. LITERATURE SURVEY

Today, the semiconductor industry uses ARM's AMBA protocol as its standard bus protocol. The APB protocol is a part of the AMBA architecture. ARM's AMVA protocol, which is the current standard for bus protocols in the semiconductor industry, includes APB as one of its protocols. In [1], they built an APB using a master-and-slave system. According to ARM's APB whitepaper, the design is at an operational condition. According to their blueprint, the master initiates activities, while the slave acts as a go-between. In light of the obtained performance reports, the results are evaluated using the Cadence tool. They used Xilinx Vivado to examine the design reports, and as stated in [1], they employed a master and slave design to develop an APB design in [2]. In [3], the protocols were developed and the APB controller and interface, in addition to the AHB master and slave interface, were constructed. The designs were linked using a bridge top that was placed between the APB and the AHB. An Artix-7 FPGA kit is used to implement the findings once they have been verified using Xilinx ISE. They have included the design verification method with the design modeling in [4-5]. Function coverage driven verification and assertion-based verification were used by the authors of [4] to validate the design using System Verilog. On the other hand, functional verification is done in [5], with

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN:2454-9940 www.ijsem.org
Vol 20, Issuse.2 April 2026

the master design acting as the test bed and the slave design as the design being tested. In addition, the AHB-APB protocols are the main emphasis of the AMBA-defined protocol interface implementation. According to their findings in [6], Roopa M. and colleagues developed a novel approach to AMBA-APB slave construction. The APB-UART is a smart UART controller that makes use of "status registers" to allow for extensive controller customization. This unique controller is capable of more than simply transferring serial data; it can also take on the role of either a leader or a follower when communicating with other devices over UART. For flawless UART data exchanges, the APB interface truly shines because to its dual-purpose functionality. The research examined many components, such as the APB slave, UART, register bank, reception engine, and transmitter engine. In order to comprehend the interplay of all parts, they used computer simulations. On top of everything else, the design seamlessly incorporates critical components such as start bits, data bits, parity bits, and stop bits, guaranteeing accurate data transmission and reception. Both AMBA 4.0 and AXI 4.0 were released by ARM in 2010. To address this, Ma and colleagues [7] developed a crucial APB bridge IP core that translates AXI 4.0-lite transactions into APB 4.0 ones, thereby bridging the gap between low-power APBs and the high-performance AXI bus. For high bandwidth, this system uses APB, while for low bandwidth, it uses AHB-to-APB and ASB-to-APB connections to link peripherals. Important parts of the 32-bit AMBA-ASB-APB bridge [8] include an arbitrator, a decoder, an APB bridge, a reset controller, a remap, and a suspend controller, all of which were constructed in Verilog and verified on SPARTAN 3E. Power usage is also addressed in a related study [9], which is comparable to [5]. The physical implementation of a block was also addressed by Gupta and colleagues [10]. They used a 3-bit ripple counter to control clock skew and stressed the need of a well-designed clock utilizing tools like as the RTL compiler, Xilinx ISE, and Encounter Digital Implements.

III. DESIGN DESCRIPTION

A. Proposed Architecture

This document details the proposed hardware architecture, which includes three units: two slaves and a master that are all parallel-operating and synced to the rising edge of the APB BUS Protocol's clock. Signal priority, including PSEL(P-Select), PRESET, PWRITE, PENABLE, and PREADY, are carefully handled by the design. The low-to-high shift of PENABLE initiates data transmission, whereas the high-to-low shift of PREADY indicates its end. Section B. Schematic Figure 1 shows that the suggested configuration supports two APB slave circuits, each of which may store up to 256 bits. The APB slave circuit is their common link; it decodes the APB protocol and makes it easy to read and write. Utilizing an associated APB Wrapper may simplify verification setup, which might lead to an improvement in efficiency.

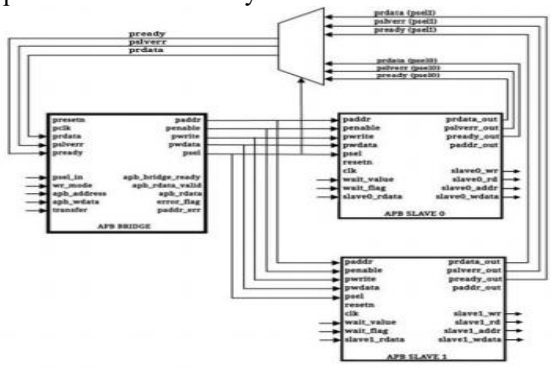

Fig. 1: Block Diagram of APB Bus Protocol Module

C. Flow Chart

The project's APB BUS protocol flowchart is shown in Figure 2. A block reset enters the process, which then enters the IDLE state. Then, one of the two slaves, Psel1 or Psel2, is designated by using PSelx. When figuring out when the system changes states, PSelx and PEnable are crucial. You can see that it's in the IDLE, setup, and access phases. In the IDLE state, with PSelx=0, everything is left alone. When PSelx=1 and PEnable=0, the system enters the SETUP state. When PSelx=1 and PEnable=1, the system enters the ACCESS state.

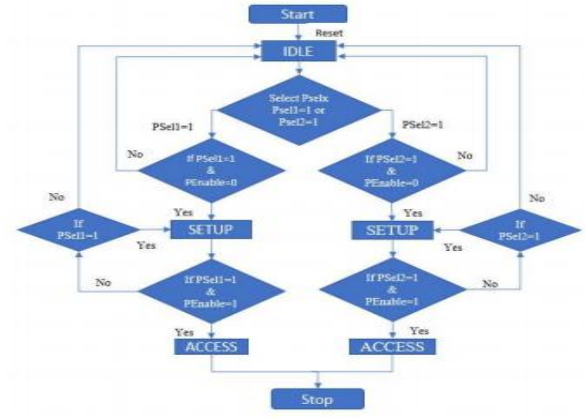

Fig. 2: Flow Chart of APB Bus Protocol Module

D. Finite State Machine

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN:2454-9940 www.ijsem.org

Vol 20, Issuse.2 April 2026

Figure 3 shows the FSM flow diagram, which describes the SETUP and ACCESS states that are taken into account for the WRITE and READ operations. The FSM enters the WRITE_SETUP and WRITE_ACCESS states in response to a write transfer request, and it is in the IDLE state by default. The READ_ACCESS state is claimed upon PEnable assertion while the READ_SETUP state is asserted upon read transfer requests.

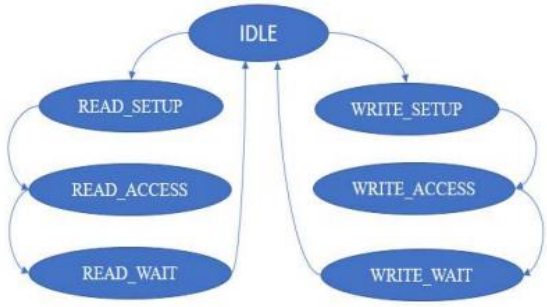

Fig. 3: FSM of APB Bus Protocol

IV. METHODOLOGY & IMPLEMENTATION

A. Methodology

In a bottom-up design approach, the first step is to identify the parts that are easily accessible for bigger cells. These parts are then used for blocks at higher levels. One Master and two Slaves are selected for this project, with IO Blocks provided for each. The Memory Block's RAM 1 and RAM 2 are linked to the APB BUS Protocol Block. Together, they form the APB BUS Protocol Module.

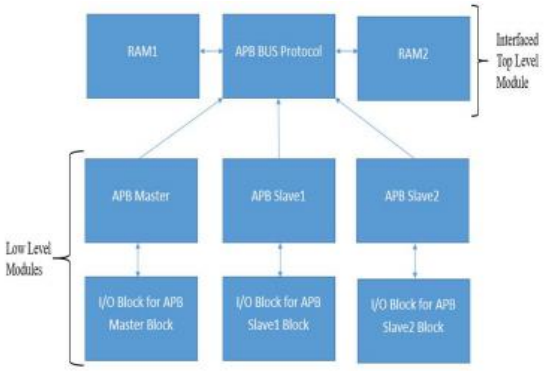

Fig. 4. Methodology of APB Bus Protocol Module

B. Implementation

Six example programs demonstrating APB-BUS's RTL capabilities and metrics like time, area, and power were used to test the protocol. The following six examples have been discussed: apb_edgeriq, apb_fastdecode, apb_fifo, apb_pulse, apb_regs1, and apb_wrtsetclr. The apb_edgeriq module handles level interrupts, for example. In this module, interrupt pulses are detected, stored, and managed. With its four incoming interrupt lines, control register, and status register, it makes managing and monitoring interrupts a breeze. Figure 5a shows this. The APB bus is partitioned into several address zones by the fastdecode module, which is an address decoder. This decoder is smaller and quicker than others since its design stresses simplicity over flexibility. This is accomplished by the decoder by dividing the psel signal into distinct select signals for each area. To guarantee effective signal processing throughout the separated address areas, the result signals — prdata, pready, and pslverr — are multiplexed appropriately. Figure 5b shows this. Learn how to use the APB bus to access a synchronous FIFO with the help of the apb_fifo module. It shows how to access FIFO status signals, read data from the FIFO, and write data to the FIFO. Furthermore, it guarantees correct synchronization and control by using a "pulse" method to produce a FIFO clear pulse. Figure 5c shows this. Apb Pulse is a prime example of a program that can create a high-frequency pulse that lasts for one apb-clock cycle. Once the APB access is finished, all of the pulses that were created here will be seen in the cycle. Figure 5d shows this. A simple example of reading status signals and generating control signals from the APB bus is provided by the apb regs1 module. The set consists of static read bits, read-only status bits, and read/write bits. This example also shows a variant that combines read/write and read-only bits at the same address, which is a simple but effective way to manage an APB register. This is seen in Figure 5e. To avoid expensive semaphores, the apb_wrtsetclr module shows how to use write-to-set and write-to-clear registers, which allow several threads or jobs to share a control register. A high value may be written by users to set certain bits high, which is an efficient approach for applications with many threads. This is seen in Figure 5f.

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN:2454-9940 www.ijsem.org

Vol 20, Issuse.2 April 2026

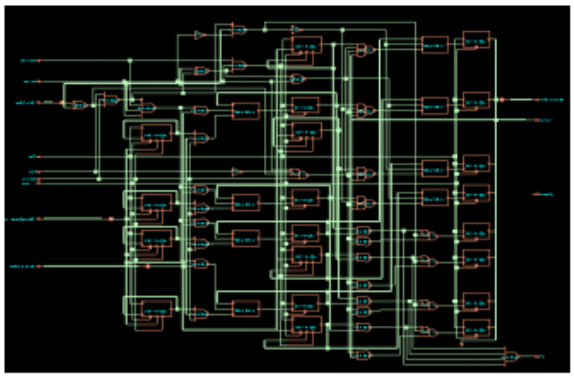

Fig. 5a: apb_edgeirq rtl

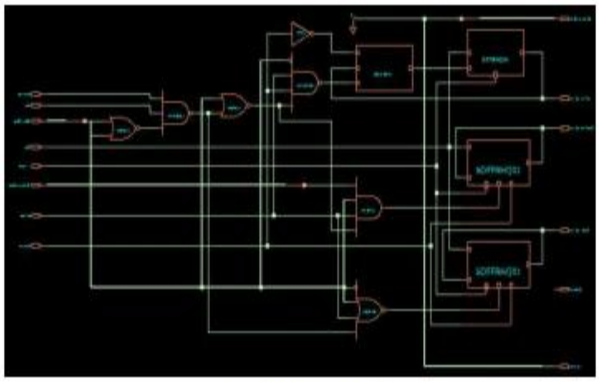

Fig. 5d apb_pulse rtl

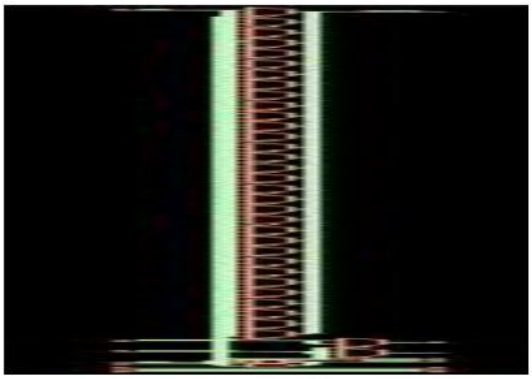

Fig. 5b: apb_fastdecode rtl

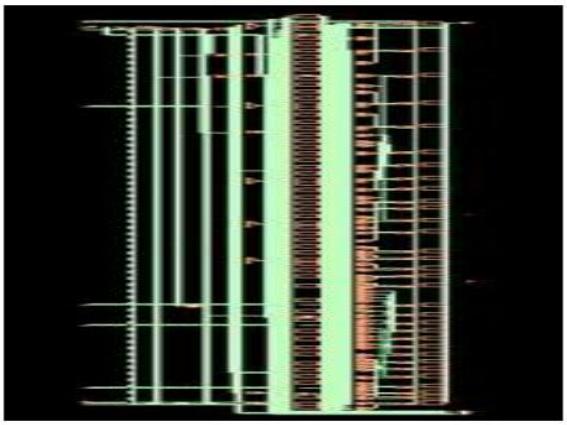

Fig. 5e: apb_rgs1 rtl

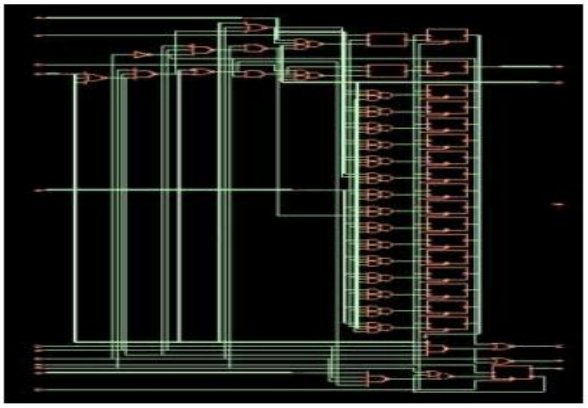

Fig. 5c: apb_fifo rtl

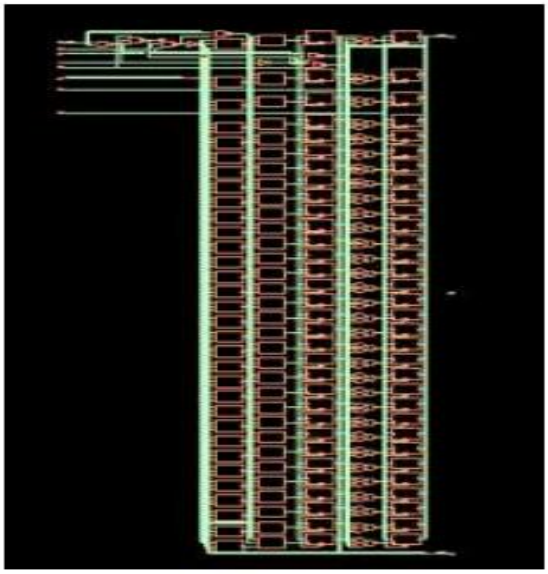

Fig. 5f: apb_wrtsetclr

To create a connection, one must first determine the master's address and then use the PSel line to initiate

a transaction with a slave. This signal is repeated for the vector master interface, and the acknowledged PSel is used to transmit the slave answers. At least two clock cycles are needed for an APB transaction: the SETUP phase starts the data exchange and the ACCESS phase finishes it. The slave is prompted to react with PRDATA when PWRITE is set low during a read operation. To enable data flow during a write operation, set PWRITE to high. Delays in operations could occur if the access phase takes more than one clock cycle. PREADY guarantees continuous transmission by signaling when data is available; low PREADY adds wait states, which are helpful for slower memory. To make things easier, slaves that can do read/write operations in a single cycle maintain PREADY high. While APB read and write transactions are almost indistinguishable, the kind of operation is controlled by PWRITE.The APB Bus Protocol has RTLs and reports for low, medium, and high effort levels, allowing for restriction-free synthesis. From Figure 6a to Figure 6c, you can see the RTL schematics. The data is shown in Table I.

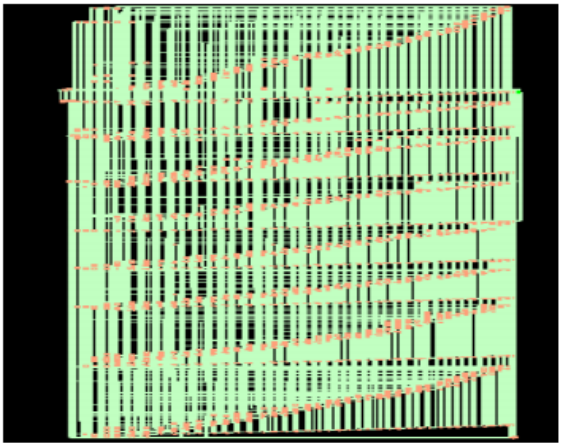

Fig. 6a: Effort Low RTL synthesis of APB Bus

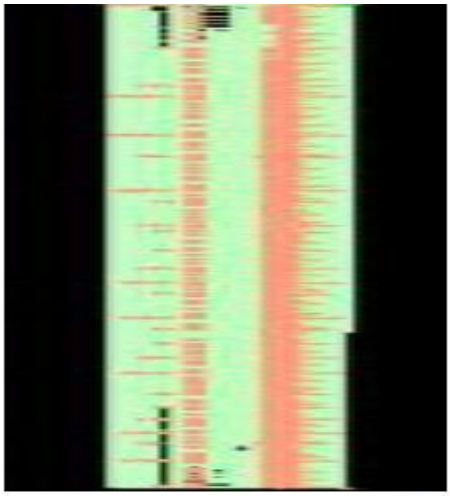

Fig. 6b: Effort Medium RTL synthesis of APB Bus

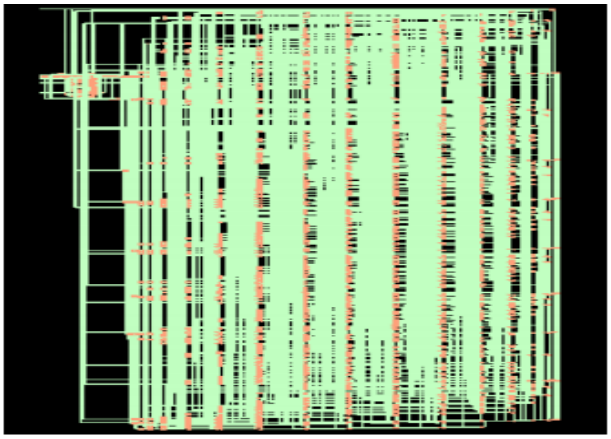

Fig. 6c: Effort High RTL synthesis of APB Bus

TABLE I: RTL REPORT

Condition/ Output Data	Cell Count	Cell Area/Power Percentage	Leakage Power Percentage	Internal Power Percentage	Switching Power Percentage	Setup Delay Time (ps)	Setup Arrival Time (ps)
apb_edgeriq	65	205.542	0.30%	87.55%	12.15%	130	1001
apb_fastcode	36	84.132	0.16%	75.97%	23.87%	0	128
apb_fifo	50	179.208	0.27%	84.32%	15.42%	131	968
apb_pulse	11	35.910	0.30%	81.49%	18.20%	208	764
apb_regs1	207	866.286	0.30%	86.61%	13.09%	120	1074
apb_wrtselclr	168	603.972	0.26%	87.05%	12.70%	74	1092

Condition/ Output Data	Cell Count	Cell Area/ Total Area	Leakage Power Percentage	Internal Power Percentage	Switching Power Percentage	Setup Delay Time (ps)	Setup Arrival Time (ps)
APB Bus without any Constraints	2349	8288.370	0.20%	83.45%	16.35%	117	1832 (Rising Edge)
APB Bus with Effort Low	2223	7856.766	0.23%	83.91%	15.87%	141	6090 (Falling Edge)
APB Bus with Effort Medium	2349	8288.370	0.20%	83.45%	16.35%	117	1832 (Rising Edge)
APB Bus with Effort High	2027	7672.086	0.20%	82.66%	17.14%	118	2684 (Rising Edge)

According to the statistics, APB Bus with Effort results in the best performance of the APB Bus Protocol under the conditions mentioned above.

For optimal switching power, high synthesis uses the least amount of space.

V. CONCLUSION

Area compromise is the primary topic of this study, which centers on the AMBA bus design and the APB bus protocol in particular. Checking the simultaneous slave communication using a multiplexer is done using Cadence tools with master-slave components. There is integration of priority signals like PRESET, PSel, PEnable, PREADY, and PWRITE. This project gives you everything you need to do time, area, and power analyses; it includes reports and six code examples for the APB module. The APB protocol was able to successfully validate read and write operations by integrating many activities into the testbench and examining waveforms. There is a 17.2644% optimization of effort, a 34.7480% optimization of switching power, and a 7.4355% optimization of cell area. A huge step forward, this effort might pave the way for much greater things to come.

REFERENCES

[1] E Ravikumar, K B Ganesha, C Y Chethana, and T M Parikshith Roy, “Design and Verification of Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) Protocol”, 2024 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), May 2024, DOI: 10.1109/ICSSES62373.2024.10561369

[2] Sirimalla Karun Kalyan Yadav, P. Satyanarayana, Bavineni Vaibhav Krishna, Kolli Ranjith Kesav Sai, Arun M, and V. Gokula Krishnan, “Design and Implementation of AMBA-APB Protocol for System-on Chip (SoC) Designs”, 2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), April 2024, DOI: 10.1109/ICDCECE60827.2024.10549635.

[3] Shanthi, G., Sravani, K.G., Vali, S.S. et al. Design and FPGA implementation of AHB-to-APB bridge. Microsyst Technol (2024). https://doi.org/10.1007/s00542-024-05703-1

[4] Prashant Dwivedi, Neha Mishra, and Amit Singh-Rajput, “Assertion & Functional Coverage Driven Verification of AMBA Advance Peripheral Bus Protocol Using System Verilog”, 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), February 2021, DOI: 10.1109/ICAECT49130.2021.9392518

[5] Padmaprabha Jain, and Satheesh Rao, “Design and Verification of Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol”, 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), February 2021, DOI: 10.1109/ICICV50876.2021.9388549

[6] Roopa, M., Vani, R. M., & Hunagund, P. V. (2013). UART controller as AMBA APB slave. National Conference on Challenges in Research & Technology in the Coming Decades (CRT 2013). doi:10.1049/cp.2013.2507. [7] C. Ma, Z. Liu and X. Ma, "Design and implementation of APB bridge based on AMBA 4.0," 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet), 2011, pp. 193-196, DOI: 10.1109/CECNET.2011.5768692.

[8] B. N. Manu and P. Prabhavathi, "Design and implementation of AMBA ASB APB bridge," 2013 International Conference on Fuzzy Theory and Its pplications (iFUZZY), 2013, pp. 234-238, DOI: 10.1109/iFuzzy.2013.6825442.

[9] K. Rawat, K. Sahni and S. Pandey, "RTL implementation for AMBA ASB APB protocol at system on chip level," 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 2015, pp. 927-930, DOI: 10.1109/SPIN.2015.7095347.

[10] A. Gupta, K. Rawat, S. Pandey, P. Kumar, S. Kumar and H. P. Singh, "Physical design implementation of 32-bit AMBA ASB APB module with improved performance," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 3121-3124, DOI: 10.1109/ICEEOT.2016.7755276.

[11] K. Akhila, N. Karuna and Y. J. M. Shirur, "Design and Implementation of Power Efficient Logic BIST With High Fault Coverage Using Verilog," 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, 2018, pp. 1-6, doi: 10.1109/ICNEWS.2018.8903923.

[12] Y. J. M. Shirur, B. C. Bhimashankar and V. S. Chakravarthi, "Performance analysis of low power microcode based asynchronous P-MBIST," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 2015, pp. 555-560, doi: 10.1109/ICACCI.2015.7275667