Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 45 No. 9
September 2024

Implementation and Verification of AHB 5.0 to APB Bridge

Dr. Kiran V, Bhavar Santosh Appasaheb
Associate Professor,
Dept. of Electronics and Communication Engg.
RV college of engineering Bengaluru, India- 560059 kiranv@rvce.edu.in'
Student-M.Tech
Dept. of Electronics and Communication Engg.
RV college of engineering Bengaluru, India- 560059 bsappasaheb.lvs22@ rvce.edu.in

***Abstract*—**The integration of distinct bus protocols, such as the Advanced High-performance Bus (AHB) and the Advanced Peripheral Bus (APB), is essential in modern System-on-Chip (SoC) designs to enable efficient communication between high-performance computational units and slower peripheral devices. This research presents the design and verification of an AHB-to-APB bridge, a critical component that facilitates this integration by ensuring seamless data transfer and protocol translation between the two bus systems. The focus of this work is the development of a verification testbench environment tailored for a single master and single slave AHB-to-APB bridge. Utilizing SystemVerilog, the testbench is designed to achieve robust functional coverage, thoroughly validating the bridge's operation across a wide range of scenarios. In addition to the functional verification, this study also delves into the

Register-Transfer Level (RTL) to GDSII implementation flow of the AHB-to-APB bridge. Emphasis is placed on optimizing the design for both performance and area efficiency, addressing the critical trade-offs inherent in digital design. The proposed verification environment not only ensures compliance with design specifications but also contributes to the overall reliability and efficiency of the bridge within the SoC architecture. This work represents a significant step towards the development of highly efficient and reliable AHB-to-APB bridges, ultimately contributing to the advancement of SoC design methodologies.

***Index Terms*—**RTL , GDS , AMBA , AHB , APB ,Physical design, AHB to APB Bridge

introduction

High-performance integrated microcontroller development is facilitated by the Advanced Microcontroller Bus Architecture, which creates on-chip connectivity. The Advanced High-Performance Bus (AHB), the Improved System Bus (ASB), and the Improved External Bus (APB) are the three bus types covered by the ARM AHB standard. An AMBA-based microcontroller has integrated mem- ory, a central processor unit (CPU) core, direct memory access (DMA) devices, and a highly ef- ficient systems bus (AMBA AHB or ASB). It describes the interactions between various parts, including slaves, masters, and interconnects.

The AHB to APB bridge serves as a vital component in such designs, facilitating communication between high- speed AHB masters and slower APB peripherals.Conversion from AHB to APB ensures that high-performance components can communi- cate with low-speed peripherals seamlessly, en- abling comprehensive system integration[14]. The bridge manages bus access arbitration between mul- tiple AHB masters and APB peripherals, ensuring fair and efficient utilization of the APB bus band- width. The bridge decodes addresses from AHB transactions to determine the target peripheral on the APB bus, enabling precise routing[2] of data and control signals.

The verification of the AHB to APB bridge protocol using System Verilog is crucial to ensure its correct operation within the SoC environment. System Verilog, a hardware description and verifi- cation language, provides powerful constructs for creating comprehensive verification environments and validating digital designs.

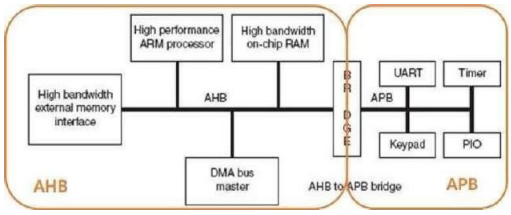

Fig. 1. AHB & APB in typical AMBA system

The CPU, on-chip memory, and other Direct Memory Access (DMA)[15] devices are often housed on a high-performance system backbone bus (AMBA AHB or AMBA ASB) that can support the external memory bandwidth, as shown in Fig. 1. A high-bandwidth link is offered by this bus between the components that are involved in the majority of transfers. A bridge connecting the high-performance bus to the lower-bandwidth APB, which houses the majority of the system's peripheral devices, is also present.

The Advanced Microcontroller Bus Architec- ture (AMBA) standard is a communications ar- chitecture designed to facilitate the creation of high-performance embedded microcontrollers. It of- fers three types of buses: the Advanced System Bus (ASB), the Advanced High-performance Bus (AHB), and the Advanced Peripheral Bus (APB). To build and evaluate a Verilog programmable AHB to APB bridge architecture, both AHB Master and APB Slave test benches are utilized for single-read and single-write tests. The primary function of this bridge is to convert transfers from system buses to APB transfers [4]. Additionally, the bridge unit is responsible for several key functions. It holds the address for the duration of the transfer and decodes this address to generate a PSEL (peripheral select) signal, ensuring that only one chosen signal is active during the transfer. The bridge also establishes aPENABLE timing strobe to coordinate the transfer, drives data onto the APB for write operations, and reads data from the APB[5] back to the system bus for read operations.

I. DESIGN SPECIFICATION

TABLE I PORT DEFINITIONS FOR AHB-TO-APB BRIDGE

Signal Name	Direction	Width
Hclk	Input	1-bit
Hresetn	Input	1-bit
Hwrite	Input	1-bit
Hreadyin	Input	1-bit
Hwdata	Input	32-bit
Haddr	Input	32-bit
Prdata	Input	32-bit
Htrans	Input	2-bit
Penable	Output	1-bit
Pwrite	Output	1-bit
Hreadyout	Output	1-bit
Hresp	Output	2-bit
Pselx	Output	3-bit
Paddr	Output	32-bit
Pwdata	Output	32-bit
Hrdata	Output	32-bit

The ports of the AHB-to-APB bridge serve spe- cific roles in facilitating communication between the Advanced High-performance Bus (AHB) and the Advanced Peripheral Bus (APB). The input signals include Hclk, the clock signal for the AHB, and Hresetn, an active-low reset signal that initial- izes the bridge. Control signals such as Hwrite, Hreadyin, and Htrans manage the direction of data flow and the state of transactions on the AHB side. The data and address buses, Hwdata and Haddr, carry the data to be written and the address of the target peripheral, respectively. The bridge also receives data from the APB through the Prdata input.

On the output side, the bridge provides several key signals to the APB. Penable and Pwrite control the timing and direction of operations on the APB, while Pselx selects the specific peripheral that the bridge is communicating with. The output buses Paddr and Pwdata correspond to the address and data lines on the APB, transmitting the necessary information for peripheral operations. The bridge also manages the handshaking and response signals

back to the AHB through Hreadyout and Hresp, ensuring synchronization between the two buses. Finally, Hrdata provides the AHB with the data read from the APB, completing the data transfer process.

A. APB Signals

TABLE II APB PORTS

Signal Name	Direction	Width
Hclk	Input	1-bit
Hresetn	Input	1-bit
valid	Input	1-bit
Hwrite	Input	1-bit
Hwritereg	Input	1-bit
Hwdata	Input	32-bit
Haddr	Input	32-bit
Haddr1	Input	32-bit
Haddr2	Input	32-bit
Hwdata1	Input	32-bit
Hwdata2	Input	32-bit
Prdata	Input	32-bit
tempselx	Input	3-bit
Pwrite	Output reg	1-bit
Penable	Output reg	1-bit
Hreadyout	Output reg	1-bit
Pselx	Output reg	3-bit
Paddr	Output reg	32-bit
Pwdata	Output reg	32-bit

TABLE III AHB PORTS

The ports for the design represent various signals critical to the operation of the AHB-to-APB bridge, facilitating communication between the two bus protocols. The input signals include Hclk, which is the clock signal that synchronizes operations, and Hresetn, an active-low reset signal that initial- izes the system. The valid, Hwrite, and Hwritereg signals are control signals that determine whether data is being written to the APB, while Hwdata and Haddr are 32-bit buses carrying the data to be written and the target address on the APB, re- spectively. The additional address (Haddr1, Haddr2) and data (Hwdata1, Hwdata2) inputs offer flexibility for handling multiple transactions or configurations. The Prdata signal is the 32-bit data bus that carries data read from the APB back to the AHB.

On the output side, the Pwrite and Penable sig- nals are used to control write operations and enable transactions on the APB, with both being declared as reg, indicating they hold their values across clock cycles. The Hreadyout signal is used to indicate the bridge's readiness to accept new transactions, en- suring synchronization between the AHB and APB. The Pselx signal selects the appropriate peripheral on the APB, while the Paddr and Pwdata outputs carry the address and data, respectively, for write operations. All these output signals are registered (reg), meaning they can maintain state, which is crucial for ensuring stable and correct operation within the bridge.

B. AHB Signals

Signal Name	Direction	Width
Hclk	Input	1-bit
Hresetn	Input	1-bit
Hwrite	Input	1-bit
Hreadyin	Input	1-bit
Htrans	Input	2-bit
Haddr	Input	32-bit
Hwdata	Input	32-bit
Prdata	Input	32-bit
valid	Output reg	1-bit
Haddr1	Output reg	32-bit
Haddr2	Output reg	32-bit
Hwdata1	Output reg	32-bit
Hwdata2	Output reg	32-bit
Hrdata	Output	32-bit
Hwritereg	Output reg	1-bit
tempselx	Output reg	3-bit
Hresp	Output	2-bit

The ports for this design manage various aspects of data communication and control between dif- ferent components. The input ports include Hclk and Hresetn, which provide the clock signal and an active-low reset signal, respectively. Control signals such as Hwrite and Hreadyin handle write opera- tions and readiness, while Htrans indicates the type of transfer being performed. The data and address buses, Haddr and Hwdata, carry the address and data values for transactions, and Prdata carries data read from the peripheral.

On the output side, the ports include valid, Hwritereg, and tempselx, which are registered out- puts used to indicate the validity of operations, control writing operations, and select temporary set- tings, respectively. The Haddr1, Haddr2, Hwdata1,

and Hwdata2 outputs provide additional address and data lines for extended transaction capabilities. The Hrdata output delivers data read from the AHB, and Hresp provides response signals indicating the status of transactions. This configuration ensures effective data management and synchronization be- tween the AHB and peripheral devices.

Implementation Of AHB To APB Bridge

A. APB bridge

An APB (Advanced Peripheral Bus) bridge is a hardware component used to connect two different systems or buses, typically with different protocols or interfaces, and facilitate communication between them. In the context of microcontroller or system- on-chip (SoC) designs, an APB bridge is often used to connect an APB bus to another bus, like an Advanced eXtensible Interface (AXI) bus or an Advanced High-performance Bus (AHB). On the AMBA APB, there is only one bus master: the APB bridge. Furthermore, the higher-level system bus also has the APB bridge as a slave.

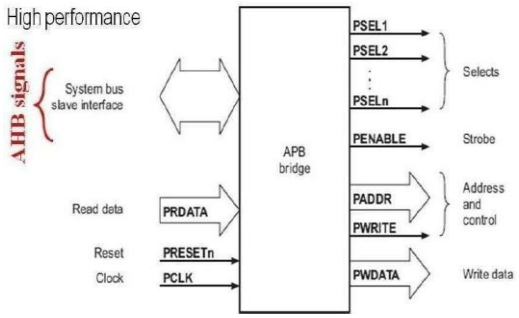

Fig. 2. APB bridge interface transfer

Fig. 2. shows APB bridge interface which trans- lates transactions from the APB protocol to the protocol of the target bus and vice versa. It handles address mapping between the APB address space and the target bus address space. This ensures that transactions are correctly routed to the intended pe- ripherals or memory regions. The bridge facilitates data transfer between the APB bus and the target bus, ensuring that data is correctly buffered and synchronized. In cases where the target bus supports multiple masters, the APB bridge may perform bus arbitration to prioritize and schedule transactions from different masters.

The bridge unit is responsible for converting system bus transfers into APB transfers, and it per- forms several critical functions to ensure accurate data communication. It first latches the address and maintains its validity throughout the duration of the transfer. This address is then decoded to generate a specific peripheral select signal, PSELx, ensuring that only one select signal is active during any given transfer. For write operations, the bridge unit drives the appropriate data onto the APB. Conversely, for read operations, it facilitates the transfer of data from the APB back onto the system bus. Addi- tionally, the bridge unit generates the timing strobe signal PENABLE, which is crucial for coordinating the timing of the transfer. These functions collec- tively enable the efficient and precise conversion of system bus signals to APB signals.

B. Interfacing APB to AHB

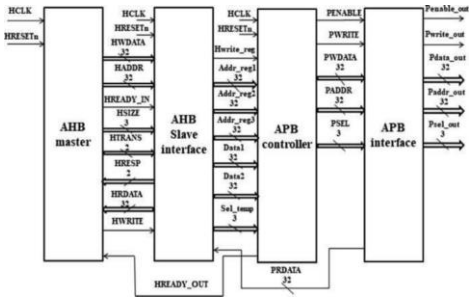

Fig. 3. AHB2APB bridge Architecture

Interfacing an APB (Advanced Peripheral Bus) to an AHB (Advanced High- performance Bus) involves connecting peripherals or IP blocks with APB interfaces to the AHB bus, enabling them to communicate with the rest of the system. This inter- facing can be accomplished using an APB-to-AHB bridge, which translates transactions between the APB and AHB protocols and manages communica- tion between devices on both buses. Fig. 3. depicts AHB2APB bridge Architecture interfacing APB to AHB through master and slave configuration for the communication[1] between the devices.

1) APB-to-AHB Read Transfer: APB-to-AHB Read Transfer mode enables peripherals connected to the slower APB bus to read data from modules connected to the higher-performance AHB bus, facilitating efficient communication and data transfer within an AMBA-based system. Below steps indicates the process of Read transfer in APB-to-AHB. The APB-to-AHB read transfer mode facilitates communication between peripherals on the slower APB bus and modules on the higher-performance AHB bus, allowing for efficient data exchange within an AMBA-based system. The process begins with the APB peripheral initiating a read request by asserting the necessary signals on the APB bus, in- cluding the read address. The APB-to-AHB bridge [6] then receives this request and translates it into an AHB-compatible transaction, ensuring that the read address is accurately represented for the AHB bus. The bridge subsequently decodes the AHB address to identify the target AHB slave device for the read operation.

Once the address is decoded, the bridge initiates a read transaction on the AHB bus, targeting the specified AHB slave device and providing the necessary read address. The AHB slave device responds to this transaction by placing the requested data onto the AHB bus. The bridge then receives this data and translates it into the appropriate APB format. Finally, the translated data is sent back to the APB peripheral that originally initiated the read request, completing the transfer process and ensuring that data from the AHB bus is effectively communicated to the APB peripheral.

2) APB-to-AHB Write Transfer: The APB-to- AHB write transfer mode allows peripherals on the APB bus to write data to modules on the higher-performance AHB bus, ensuring efficient communication and data transfer within an AMBA-based system. The process begins with the APB peripheral initiating a write request by asserting the appropriate signals on the APB bus, including the write address and data. The APB-to-AHB bridge receives this request and translates it into an AHB-compatible transaction, accurately conveying the write address and data for the AHB bus. Subsequently, the bridge decodes the AHB address to determine the target AHB slave device for the write operation. It then initiates a write transaction on the AHB bus, targeting the identified AHB slave device and delivering the write address and data.

During the data transfer phase, the AHB slave captures the data from the AHB bus and updates its internal registers or memory accordingly. Upon successful completion of the write transaction, the AHB slave [12] may send an acknowledgment signal, indicating the successful write operation. The bridge can translate this acknowledgment into a suitable response for the APB peripheral. Finally, the translated response is sent back to the APB peripheral that initiated the write request, confirm- ing the successful completion of the transaction and ensuring that the peripheral is informed of the outcome.

II. VERIFICATION USING SYSTEMVERILOG ENVIRONMENT

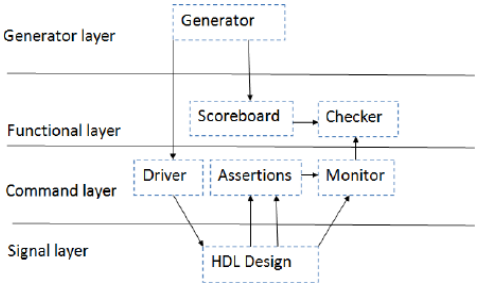

Fig. 4. System Verilog Environment[4]

Different components of verification environ- ment, shown in Fig. 4. & Design of the testbench architecture using SystemVerilog includes creating agents for the AHB and APB interfaces, scoreboard, driver, monitor, and sequencer components[9]. Sequences and drivers are generated for various AHB and APB transactions according to the test plan. This involves generating read and write transactions with different data, addresses, burst lengths, etc. The performance aspects of the bridge, such as latency, throughput, and timing constraints[11] have been

verified. This may involve running stress tests and analyzing timing diagrams.

A typical testbench environment in SystemVer- ilog consists of several key components. The gen- erator is responsible for creating randomized or directed stimulus to drive the design under test (DUT). The driver receives this stimulus and con- verts it into signals that interact with the DUT. The monitor observes the DUT outputs, extracting data for analysis. The scoreboard compares the DUT's actual outputs against expected outputs to check for correctness. The sequencer controls the flow of stimulus to the driver, allowing for complex test scenarios. The environment encapsulates these components, providing a structured and organized setup for verification. Additionally, coverage col- lectors measure how thoroughly the DUT has been tested, and assertions embedded within the DUT or testbench check for specific conditions and be- haviors. This modular and hierarchical approach in SystemVerilog testbenches enhances reusability and maintainability, making it easier to manage complex verification tasks.

III. RTL TO GDS FLOW

The RTL to GDS-II flow shown in Fig. 5. in this VLSI design starts with creating a Verilog file containing the RTL description of the digital circuit. During synthesis, this code is transformed into a gate-level netlist. Placement then determines the exact positions of cells on the silicon die [10] , optimizing for area, performance, and power while ensuring timing constraints. Static Timing Analysis (STA) verifies timing performance, followed by routing, which establishes physical connections be- tween cells. Post-Route STA ensures the routed de- sign meets timing constraints. Migration adapts the design to different technologies if needed. Design Rule Check (DRC) and Layout Versus Schematic (LVS) ensure manufacturability and correctness. The final output is a GDS-II file, describing the lay- out for fabrication. This flow ensures the accurate transformation of the digital design from high-level description to manufacturable layout, optimizing and verifying at each stage.

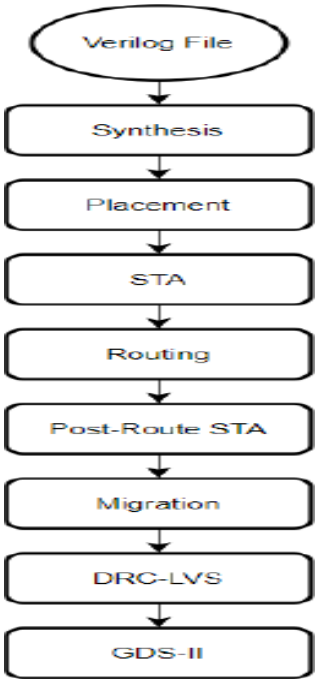

Fig. 5. RTL to GDS-II Flow[10]

IV. **Results**

A. Write/Read Transfer

In Fig. 6. Write/Read Transfer output illustrates the operation of an AHB to APB bridge, highlight- ing both write and read transactions. Initially, the hresetn signal is low, indicating a reset state. Once it goes high, the system becomes operational. The AHB side sets up the haddr and hwdata signals to specify the address and data for the transaction, respectively. For instance, haddr might be set to 80000000 while hwdata holds the value 00000024. In a write operation, indicated by the hwrite signal being high, data from the AHB bus (hwdata) is transferred to the APB side (pwdata).

Conversely, during a read operation, when hwrite is low, data from the APB side (pr_data) is read and transferred back to the AHB (hr_data). Control signals such as hread_in indicate the readiness of the AHB bus for new transactions, while penable and psel manage

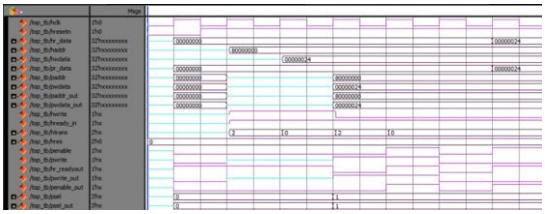

Fig. 6. Write/Read Transfer

the enablement and selection of peripherals on the APB bus. The transitions and timing of these sig- nals ensure the correct execution of data transfers, showcasing the bridge's functionality in managing communication between the two buses.

B. Verification using SystemVerilog Environment

The fig. 7 presents the write transfer operation in an AHB to APB bridge, as observed in a SystemVerilog testbench environment. Initially, at cycle 25, the generator initiates the write operation by setting the Hwdata to a3 and asserting the Hwrite signal to 1, indicating a write transaction. The Hreadyin signal is also set to 1, showing that the AHB bus is ready for the next transaction, while the Htrans signal remains 0, indicating an idle state.

```
-----GENERATOR writing-----25
Hwdata=a3, Hwrite=1, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=x
Penable=x, Pwrite=x, Hreadyout=x, Htresp=x,Pselx=x,Paddr=x,Pwdata=x,Hrdata=x
-----DRIVER WITH OUTPUT-----50
Hwdata=a3, Hwrite=1, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=x
Penable=x, Pwrite=x, Hreadyout=x, Htresp=x,Pselx=x,Paddr=x,Pwdata=x,Hrdata=x
-----COLLECTED IN MONITOR-----70
Hwdata=a3, Hwrite=1, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=x
Penable=0, Pwrite=x, Hreadyout=0, Htresp=x,Pselx=x,Paddr=x,Pwdata=a3,Hrdata=x
-----scb-----70
Hwdata=a3, Hwrite=1, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=x
Penable=0, Pwrite=x, Hreadyout=0, Htresp=x,Pselx=x,Paddr=x,Pwdata=a3,Hrdata=x
```

Fig. 7. Write Transfer in SystemVerilog Testbench

The address for the operation, Haddr, is set to 80000001. At cycle 50, the driver captures these values, ensuring they are correctly set according to the generator's commands, although signals related to the APB side, such as Paddr and Pwdata, are still undefined at this point. By cycle 70, the monitor captures the transaction, confirming that the write data a3 has been correctly driven onto the APB bus (Pwdata). Finally, the scoreboard (scb) verifies the transaction's accuracy, ensuring the data (a3) and address (80000001) match the expected values, thereby validating the write operation's correctness.

```
-----GENERATOR reading-----70
Hwdata=x, Hwrite=0, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=a3
Penable=x, Pwrite=x, Hreadyout=x, Htresp=x,Pselx=x,Paddr=x,Pwdata=x,Hrdata=x
-----DRIVER WITH OUTPUT-----70
Hwdata=x, Hwrite=0, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=a3
Penable=x, Pwrite=x, Hreadyout=x, Htresp=x,Pselx=x,Paddr=x,Pwdata=x,Hrdata=x
-----COLLECTED IN MONITOR-----90
Hwdata=x, Hwrite=0, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=a3
Penable=0, Pwrite=x, Hreadyout=0, Htresp=x,Pselx=0,Paddr=x,Pwdata=x,Hrdata=a3
-----scb-----90
Hwdata=x, Hwrite=0, Hreadyin=1, Htrans=0,Haddr=80000001,Prdata=a3
Penable=0, Pwrite=x, Hreadyout=0, Htresp=x,Pselx=0,Paddr=x,Pwdata=x,Hrdata=a3
```

Fig. 8. Read Transfer

The Fig. 8. presents Read Transfer initially, the generator initiates a read operation from the ad- dress 0x80000001, with the relevant signals such as Hwrite set to 0 (indicating a read operation), Hreadyin set to 1 (indicating the master is ready), and Haddr set to 0x80000001. The peripheral's read data is indicated by Prdata=a3. The driver then echoes this read operation, reflecting the same signal values as those generated by the generator. As the signals propagate through the driver, the monitor collects them, showing the Pselx signal set to 0, indicating the correct peripheral is selected. At this point, the read data (Hrdata) is captured as a3, which matches the expected value.

Finally, the scoreboard verifies the correctness of the operation by confirming that the read data from the AHB (Hrdata) is indeed a3, matching the peripheral's data. This sequence confirms the suc- cessful execution of the read transfer from the AHB to the APB bridge, ensuring that the bridge and data transfer mechanisms are functioning correctly

V. PNR FLOW

The Fig. 9. floorplan represents the physical layout of the AHB to APB bridge on the silicon die, showcasing the spatial organization and inter- connections of various components. The core logic, including master and slave interfaces, control logic, and data paths, is strategically placed to optimize performance and minimize signal delays. Power dis- tribution networks ensure stable and efficient power delivery, maintaining the low power consumption characteristic of both AHB and APB buses.

Fig. 9. Floorplan layout

The Fig. 10. shows the power rail layout for the AHB to APB bridge, highlighting the crucial aspect of stable and efficient power distribution across the integrated circuit. The grid-like structure of the power rails, consisting of vertical and hor- izontal lines representing VDD (power) and VSS (ground), ensures that all parts of the bridge receive adequate power supply, minimizing voltage drops and enhancing performance and reliability.

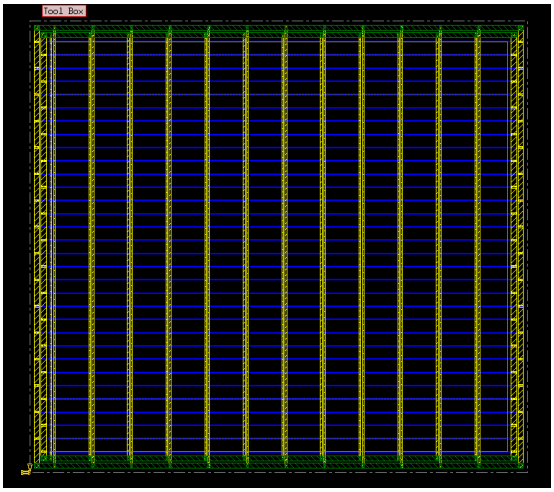

Fig. 10. Power Rails Layout

The Fig. 11. shows the placement output for an AHB to APB bridge, essential in system-on-chip (SoC) designs for connecting high-speed and low- speed components. It includes standard cells like registers and multiplexers, with interconnections represented by colored lines indicating different metal layers. The perimeter features pins for ex- ternal connections, managing data, address, control signals, and clocks. The grid structure ensures op- timized cell alignment for performance and power efficiency.

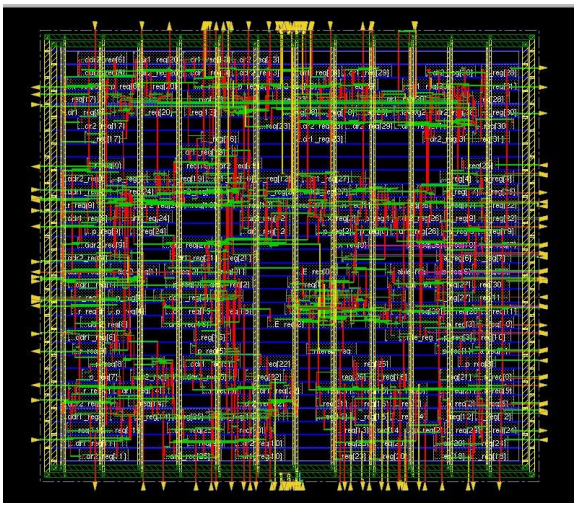

Fig. 11. Standard Cell Placement

Fig.12 illustrates the Clock Tree Synthesis (CTS) process for an AHB to APB bridge, a crucial step in the physical design of digital integrated circuits. CTS is essential in distributing the clock signal efficiently across the circuit, ensuring that it reaches all registers or flip-flops (endpoints) simultaneously or with minimal skew. In the figure, the root of the clock tree, where the clock signal originates, is represented at the topmost point in yellow. This signal is then distributed throughout the circuit via a series of clock buffers or inverters, depicted by yellow lines.

The Fig. 13. displays the routing output for an AHB to APB bridge. It illustrates the intricate interconnections between the standard cells, with yellow, green, and red lines indicating different

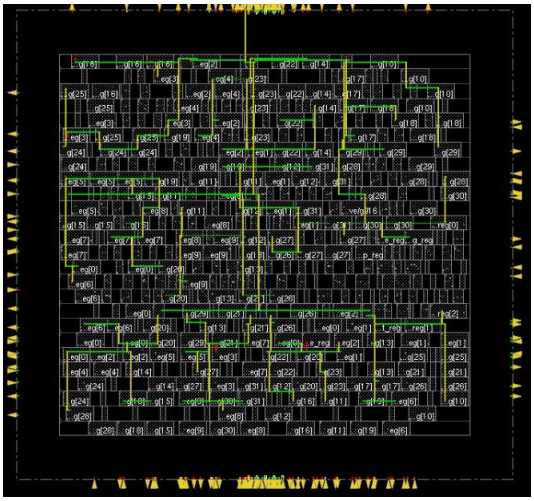

Fig. 12. Clock Tree Synthesis

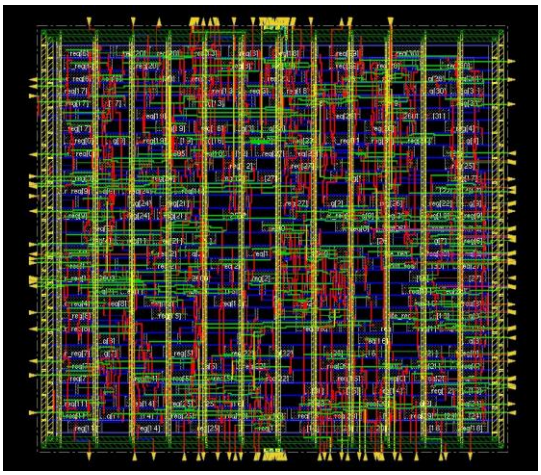

Fig. 13. Routing

metal layers used for routing. These lines repre- sent the paths for data, control signals, power, and ground, ensuring efficient and organized signal flow within the chip. The dense network of routes is optimized to minimize signal delay and power con- sumption while maximizing performance and area efficiency. This routing output, crucial for achieving the desired timing and functionality, exemplifies the careful planning and execution required in SoC design, providing a comprehensive overview of the signal pathways that interconnect the AHB and APB domains.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution GENUS15.
  Generated on:           Aug 13 2024  03:50:33 pm
  Module:                 Bridge_Top
  Technology libraries:   slow_vdd1v0 1.0
                          abstract_models
  Operating conditions:   PVT_0P9V_125C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                    Leakage   Dynamic    Total
  Instance   Cells Power(nW) Power(nW) Power(nW)
------------------------------------------------------------
Bridge_Top     456    27.075  9888.351  9915.426
  APBControl   317    17.275  3998.069  4015.343
  AHBSlave     139     9.800  5261.018  5270.819
```

Fig. 14. Power Report

The Fig. 14. presents the power report for an AHB to APB bridge, a key component in system- on-chip (SoC) designs. The table summarizes the power consumption of different modules within the bridge, including Bridge_Top, APBControl, and AHBSlave. For each module, the table lists the number of cells, leakage power, dynamic power, and total power consumption in nanowatts (nW). Bridge_Top has 456 cells, consuming a total of 9915.426 nW, with 27.07 nW leakage and 9888.35 nW dynamic power. APBControl comprises 317 cells with a total power consumption of 4015.34 nW, while AHBSlave has 139 cells with 5270.8 nW total power consumption.

```
#
innovus 3>  *** Starting Verify Geometry (MEM: 1108.6) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1920
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macr

  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 1.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.
```

Fig. 15. Geometry Verification

The Fig. 15. presents the geometry verification output for the AHP to APB bridge design shows

a successful validation with no detected violations. The process initializes by allocating memory and setting up sub-areas for detailed analysis. A warning about the unused PG pin of macro cells is noted but does not indicate a violation. The verification checks various categories such as cells, same-net connections, wiring, antenna effects, shorts, and overlaps, all of which reported zero violations.

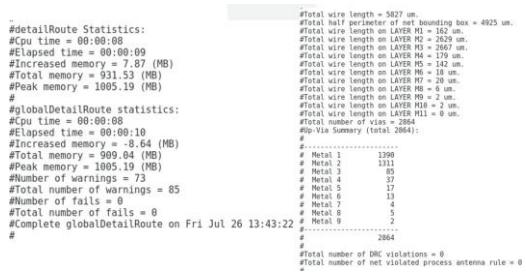

Fig. 16. Routing Statistics and DRC violation

The Fig. 16 presents routing statistics and DRC (Design Rule Check) violation output for the AHP to APB bridge design indicate a successful routing process with no errors. Despite 85 warnings during the routing process, no DRC violations or process antenna rule violations were found, confirming the design's adherence to all routing and fabrication standards.

The Table presents a concise summary of the key performance parameters for the designed circuit. The design was implemented using a 45nm tech- nology process node, which signifies the advanced miniaturization of the circuit components, allowing for improved performance and power efficiency. Operating at a clock frequency of 20MHz, the circuit is designed to handle data at a rate of 20 million cycles per second, balancing speed with power consumption. With a utilization rate of 70%, the design effectively uses the available resources, ensuring a good balance between performance and efficiency.

In terms of power consumption, the circuit ex- hibits a leakage power of 27.0nW, which reflects the power drawn when the circuit is idle, and a dynamic power of 9888.3nW, indicating the power consumed during active operation. The total power consumption stands at 9915.4nW, a critical figure that showcases the overall energy efficiency of the design. The circuit also features 456 nets, highlighting a moderately complex interconnection structure that supports the circuit's functionality. These parameters together provide a comprehensive overview of the circuit's performance and efficiency, making it suitable for its intended application.

TABLE IV FINAL RESULTS

Parameter	**Value**
Technology Process node	45nm
Clock frequency	20MHZ
Utilization	0.70
Leakage Power	27.0nW
Dynamic Power	9888.3nW
Total Power	9915.4nW
Number of nets	456

VI. CONCLUSION

In this research, we designed and tested an efficient AHB-to-APB bridge for read and write operations using the QuestaSim tool, following the AMBA bus architecture's data transfer protocols. The verification of the AHB-to-APB bridge, which was implemented in a Single Master-Single Slave configuration, was carried out by creating a test-bench environment using SystemVerilog. After the verification, we successfully completed the RTL-to-GDSII flow using the Cadence Virtuoso tool. The final results indicate a technology process node of 45nm with a clock frequency of 20MHz and 70% utilization. The power analysis revealed a leakage power of 27.0nW and dynamic power of 9888.3nW, leading to a total power consumption of 9915.4nW.

Additionally, the design included 456 nets. These results validate the efficiency and performance of our bridge design.

Acknowledgment

I would like to express my deepest gratitude to my guide, Dr. Kiran V, for their invaluable guidance, continuous support, and encouragement throughout the course of this research. Their exper- tise and insightful feedback greatly contributed to the successful completion of this work. I also extend my thanks to all those who assisted me during the research process.

References

[1] S. Wu, K. Zhao, X. Wang, S. He and D. Guo, ”An UVM-Based Verification Platform for Hardware and Soft- ware Co-Design,” 2023 IEEE 17th International Con- ference on Anti-counterfeiting, Security, and Identifi- cation (ASID), Xiamen, China, 2023, pp. 21-24, doi: 10.1109/ASID60355.2023.10426039.

[2] N. Deshpande and R. Sadakale, ”AMBA AHB to APB Bridge Protocol Verification Using System Ver- ilog,” 2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI), Tiruchengode, India, 2023, pp. 1-3, doi: 10.1109/ICAEECI58247.2023.10370951.

[3] M. B. Savadatti, V. R, S. S. M. L, S. V and
T. J. Reddy, ”Data Transfer Using AMBA Bus: An Empirical Approach,” 2024 2nd International Confer- ence on Artificial Intelligence and Machine Learn- ing Applications Theme: Healthcare and Internet of Things (AIMLA), Namakkal, India, 2024, pp. 1-6, doi: 10.1109/AIMLA59606.2024.10531613.

[4] S. A. Edidi, R. Marada, T. A. Khan and C. E, ”Improving Data Integrity with Reversible Logic-based Error Detec- tion and Correction Module on AHB-APB Bridge,” 2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computa- tional Intelligence (RAEEUCCI), Chennai, India, 2023, pp. 1-9, doi: 10.1109/RAEEUCCI57140.2023.10134491.

[5] D. Krishnegowda, ”Developing a Bus Functional Model for APB slave using Universal Verification Methodology,” 2021 Second International Conference on Smart Tech- nologies in Computing, Electrical and Electronics (IC- STCEE), Bengaluru, India, 2021, pp. 1-5, doi: 10.1109/IC-STCEE54422.2021.9708572.

[6] B. H. Niharika and S. Ramesh, ”The Configuration and Verification Analysis of AMBA-Based AHB2APB Bridge,” 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICM- NWC), Tumkur, Karnataka, India, 2022, pp. 1-5, doi: 10.1109/ICMNWC56175.2022.10031928.

[7] M. Ismael, A. Hroub and A. Abu-Issa, ”AUTG: An Automatic UVM-based TestBench Generator for VLSI Chip Design Verification,” 2023 International Conference on Microelectronics (ICM), Abu Dhabi, United Arab Emirates, 2023, pp. 162-167, doi: 10.1109/ICM60448.2023.10378885.

[8] P. Bhatt, D. Joshi and S. Jadhav, ”RTL to GDS Im- plementation and Verification of UART using UVM and OpenROAD,” 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2024, pp. 0713-0720, doi: 10.1109/CCWC60891.2024.10427771.

[9] P. Gurha and R. R. Khandelwal, ”SystemVerilog Assertion Based Verification of AMBA-AHB,” 2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), Ghaziabad, India, 2016, pp. 641- 645, doi: 10.1109/ICMETE.2016.67.

[10] D. Acharya and U. S. Mehta, ”Performance Analysis of RTL to GDS-II Flow in Opensource Tool Qflow and Commercial Tool Cadence Encounter for Syn- chronous FIFO,” 2022 IEEE International Conference of Electron Devices Society Kolkata Chapter (EDKCON), Kolkata, India, 2022, pp. 199-204, doi: 10.1109/ED-KCON56221.2022.10032906.

[11] P. Giridhar and P. Choudhury, ”Design and Verifica- tion of AMBA AHB,” 2019 1st International Confer- ence on Advanced

Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), Bangalore, India, 2019, pp. 310-315, doi: 10.1109/ICATIECE45860.2019.9063856.

[12] H. Duan, L. Yu, H. Zhou, Y. Du and Y. Ren, ”An On-Chip AHB Bus Tracer for Non-intrusive Debugging,” 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 322-326, doi: 10.1109/IM- CEC46724.2019.8984151.

[13] I. H. Shanavas, P. A. Reddy, M. M. Baburaj, N. C. Krishna and N. S, ”Design and Analysis of APB and AHB Lite Protocol,” 2023 7th International Conference on Design Innovation for 3 Cs Compute Communicate Control (ICDI3C), Karnataka, India, 2023, pp. 1-6, doi: 10.1109/ICDI3C61568.2023.00009.

[14] L. Deeksha and B. R. Shivakumar, ”Effective Design and Implementation of AMBA AHB Bus Protocol using Verilog,” 2019 International Conference on Intelligent Sus- tainable Systems (ICISS), Palladam, India, 2019, pp. 1-5, doi: 10.1109/ISS1.2019.8907975.

[15] C. Sharma and D. K. Chauhan, ”High performance low power AHB DMA controller with FSM decomposi- tion technique,” 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineer- ing (ICPCSI), Chennai, India, 2017, pp. 456-461, doi: 10.1109/ICPCSI.2017.8392337.

[16] International conference: Kiran. V, Vinila Nagaraj’ “De- signing, prototyping & verification of I2C master/slave controller with APB interface” In International confer- ence on Data Engineering and Communication system 2011(ICDECS-20011) 0n 31 December 2011 at RNSIT Bangalore