Design of Efficient AMBA AHB Arbiter

*Vennapoosa Hemanth Kumar Reddy[1], Harika Chaparala[2], Prof. K. Malakondaiah[3]

[1]*MS in VLSI Engineering, Veda IIT, Guntur, v.h.k.reddy522@gmail.com*

[2]*SMTS, Invecas, Guntur, harika.chaparala@invecas.com*

[3]*Director, HOD, Dept of VLSI, Veda IIT, Guntur, malakondaiah.kakarlapudi@vedaiit.com*

Abstract: Many separate IP cores are coupled together with a complicated on-chip bus communication architecture in a typical System-on-Chip (SOC) design. In complicated SOC design, the on-chip bus communication architecture is a major predictor of total performance. AMBA, Wishbone, Core Connect, Avalon, and other connection buses are often used in the industry. AMBA is the most popular of the three because it has a hierarchy of buses, with AHB (Advanced High-Performance Bus) for high-performance peripherals and APB (Advanced Peripheral Bus) for low-performance peripherals. When dealing with several masters seeking to access a single data bus, resolution is a major difficulty in SOC. At such point, an arbiter is crucial. The goal of this research is to develop RTL code for AMBA AHB arbiter and perform STA to improve the timing aspects. The RTL is developed for a generic number of masters, allowing us to add and remove them as needed. For Synthesis and STA (Static timing analysis) of the design, I have considered mylib.db as TL (Technology library). And finally, I generated the netlist. The design architecture is written using Verilog HDL code and STA is done using DC (Design compiler from Synopsys) tools. The architecture is modelled and synthesized using RTL (Register Transfer Level) abstraction.

Keywords: AMBA, AHB, Arbiter, SOC, IP, RTL, STA, Synthesis, TL, Verilog HDL.

1. Introduction

The manner in which the functional units exchange and synchronise their data is determined by the on-chip bus communication architecture, which has a significant impact on the system's performance. As more IP cores are introduced into the design platform, the bandwidth of communication between them grows, and thus becomes a source of performance bottlenecks. One of the most important components of a SOC platform is the on-chip bus communication architecture. The interface behaviour of each IP block integrated into the complicated SOC must be satisfied by an efficient on-chip communication system. There are a variety of commercially defined communication architectures available on the market, each with its own bus protocol. For example, OMI's PI-Bus, ARM's AMBA bus, Mentor Graphics' FISP bus, IBM's Core-Connect, Sonics' Silicon backplane, and Silicore's Wishbone, among others.

The Advanced Microcontroller Bus Architecture (AMBA) is an open standard on-chip interconnect architecture for connecting and managing functional blocks in a system-on-chip (SoC) AMBA aids in the construction of multiprocessor designs with a large number of controllers and peripherals that are done perfectly the. The Advanced Microcontroller Bus Architecture (AMBA) has the capability of reusing designs, which means it can reuse IP. In today's technology, IP re-use is a critical aspect in lowering development costs and timelines for System-on-Chip (SoC). AMBA is a standard

interface specification that ensures that IP components from different design teams or manufacturers are compatible.

This research focuses on AMBA AHB, specifically AHB arbiter. With the growing number of system components in SOC architecture, an efficient arbiter becomes one of the most crucial variables in ensuring the system's high performance. Bus with Advanced High-Performance Technology (AHB) The AMBA AHB is a high-performance system module with a high clock frequency [1].

2. Overview of AMBA Buses

The Advanced High-Performance Bus (AHB), Advanced System Bus (ASB), and Advance Peripheral Bus (APB) are the three buses defined in the AMBA 2.0 specification (APB) [1].

The AHB serves as the system's high-performance backbone bus. Processors, on-chip memories, and off-chip external memory interfaces may all be connected efficiently with AHB's low-power peripheral macro cell features. AHB is also designed to be simple to utilise in a design flow that includes synthesis and automated testing. ASM (Advanced System Management) (ASB). The AMBA ASB is a system module for high-performance systems. AMBA ASB is a system bus that can be used in situations where the high-performance capabilities of AHB aren't required. ASB also enables low-power peripheral macro cell functionalities to connect CPUs, on-chip memories, and off-chip external memory interfaces. APB (Advanced Peripheral Bus) is a type of peripheral bus (APB) Low-power peripherals are the focus of the AMBA APB. To support peripheral functions, AMBA APB has been tuned for low power consumption and decreased interface complexity. The APB protocol can be used with either version of the system bus [4].

3. Design Flow

ASIC design flow is now a highly developed and sophisticated process. Until date, the overall ASIC design flow, as well as the many processes inside it, have shown to be both practical and robust in multi-million dollar ASIC designs. Let's have a look at these processes in the design process in general. Figure 2.1 shows the VLSI design flow [5].

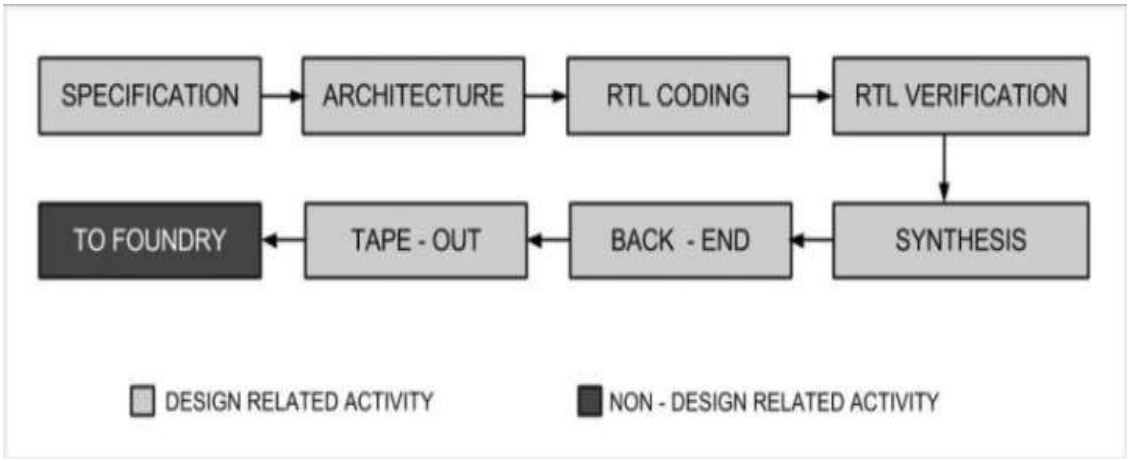

Figure 2.1 VLSI Design Flow

- **Specification**

A lot of work is done upfront, from gathering market requirements to deciding on technical aspects. This is the most important step because it will determine the

product's future. Vendors may wish to seek feedback on what potential customers are looking for here. After that, a final specification sheet is created with all of the technical specifics and handed over to the next team.

- **Architecture**

Here is when the real work begins. The architecture of the target IC is determined using the specification sheet, and a layout is developed using EDA tools by design engineers. The architecture is then developed and tested using programming languages and tools in the next step.

- **RTL Coding**

The acronym RTL stands for register transfer level. This means that the architecture-based VHDL code explains how data is altered as it moves from register to register.

- **RTL Verification**

Simulation and verification of the Register Transfer Level (RTL) is a key step. This assures that the design is logically valid and free of significant timing flaws. This step is beneficial to execute, especially in the early stages of the design. RTL verification can be done with Synopsys simulation tools. For verification, a test bench file might be used.

- **Synthesis**

This is when the design starts getting physical. Logic synthesis is the process of translating desired circuit behaviour, such as Register Transistor Level, into a logic gate design that drives the circuit or architecture. FPGA/CPLD/ASIC hardware tools are used to do this. The IDEs provided by certain vendors can be used to access these target boards.

The RTL design description is converted to a gate-level "netlist" or "list of wires" by logic synthesis tools. A gate-level netlist is a list of logic gates and their connections that describes the complete chip. Synthesis tools verify that the gate-level netlist complies with timing, area, and power requirements.

- **Backend**

After synthesis, the final tested design is handed to the IC manufacturer. An Automatic Place and Route tool uses the gate-level netlist to construct a layout. After that, the layout is confirmed before being produced on a chip. The gate level netlist is transformed into a comprehensive physical geometric representation here.

- **Tapeout**

Tape out is a backend-only procedure in which the manufacturer receives the FRONTEND (first five steps) final output in the form of a photomask. The manufacturer then goes through wafer processing, packaging, testing, and sample delivery to test the physical IC. The layout's output file is the GDSII (GDS2) file, which is the file utilised by the foundry to build the silicon. The arrangement should adhere to the silicon foundry's design guidelines.

- **Foundry**

The design is sent to mass production when the sample has been tested and all requirements have been met.

The following are the important points to remember from the preceding steps:

- The first step is to create specifications, which outline the functionality, interface, and architecture of the digital IC circuit to be created.
- The design is subsequently analysed in terms of functionality, performance, compliance with provided standards, and other specifications using a behavioural description.
- HDLs are used to describe RTL. To test functionality, this RTL description is emulated. From here on out, we'll need the assistance of EDA tools.

- Logic synthesis tools are then used to transform the RTL description to a gate-level netlist. A gate-level netlist is a representation of the circuit in terms of gates and connections between them, designed to match the timing, power, and area requirements.

After that, a physical layout is created, which is then confirmed before being sent to manufacture.

The majority of digital design activity in today's world is focused on manually optimising the RTL description of the circuit. The only manual step in the entire flow is the frontend. Following the freezing of the RTL description, a variety of EDA tools from various vendors are available to aid the designer in subsequent procedures.

4. Logical Synthesis

The process of converting and mapping RTL code written in HDL (such as Verilog or VHDL) into technology-specific gate level representation is referred to as logic synthesis.

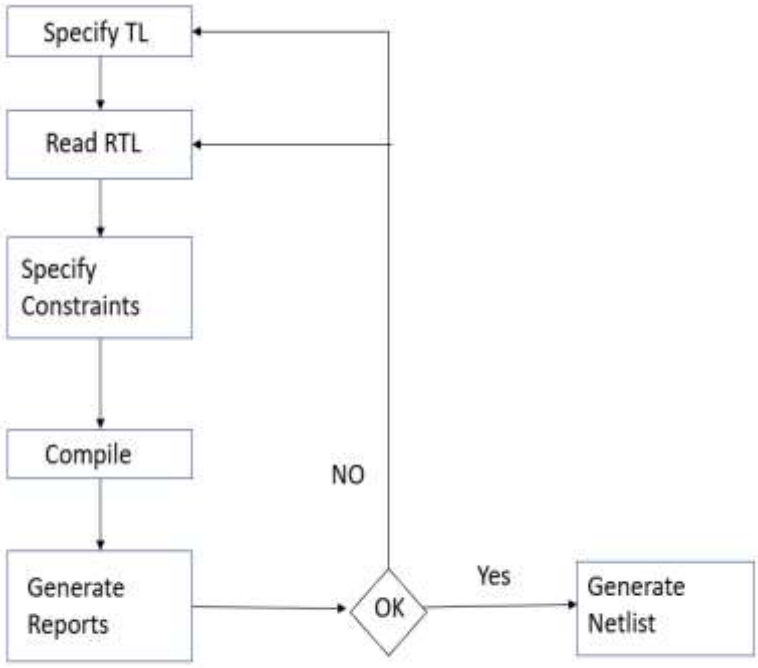

Figure 4.2 Front End Synthesis Flow

Figure 4.2 shows the VLSI Front End synthesis flow.

- **Specify TL**

Technology Library (TL) is collection of cells along with characteristic information (like power, area and time). It is provided by fabrication houses like TSMC, UMC etc.

- **Read RTL**

In this stage the following actions will be done

- Design Compiler (DC) checks the syntax errors and reports if there are any syntax errors.
- File list is compiled and stored in .db format.

The RTL code is converted into technology independent netlist by using GTECH library. The transformed logic is in the form of a Boolean equation.

- **Constraints**

There are 3 types of constraints.

- Environmental constraints

- Design Rule Check (DRC) Constraints
- Optimization constraints

➢ **Compile**

The following actions are done in this stage:

- **Optimization:** Boolean equation is optimized using SoP or PoS optimization methods.
- **Technology mapping:** Based on design restrictions and a library of available technology gates, technology independent Boolean logic equations are translated to technology dependent library logic gates. This results in an optimal gate level representation, which is commonly expressed in Verilog. The produced gate level circuit is then rationally tuned to fulfil the user-specified criteria or goals. The clock frequency objective is the most important goal that the synthesis operation must achieve.

5. Simulation Results

➢ **Reset Operation**

The Reset signal is active Low and is used to reset the system and bus. This reset has been asserted and deasserted synchronously after the rising edge of Clock. In the below Fig 6.1 we can see all the outputs become zero after applying the reset irrespective of the bus requests.

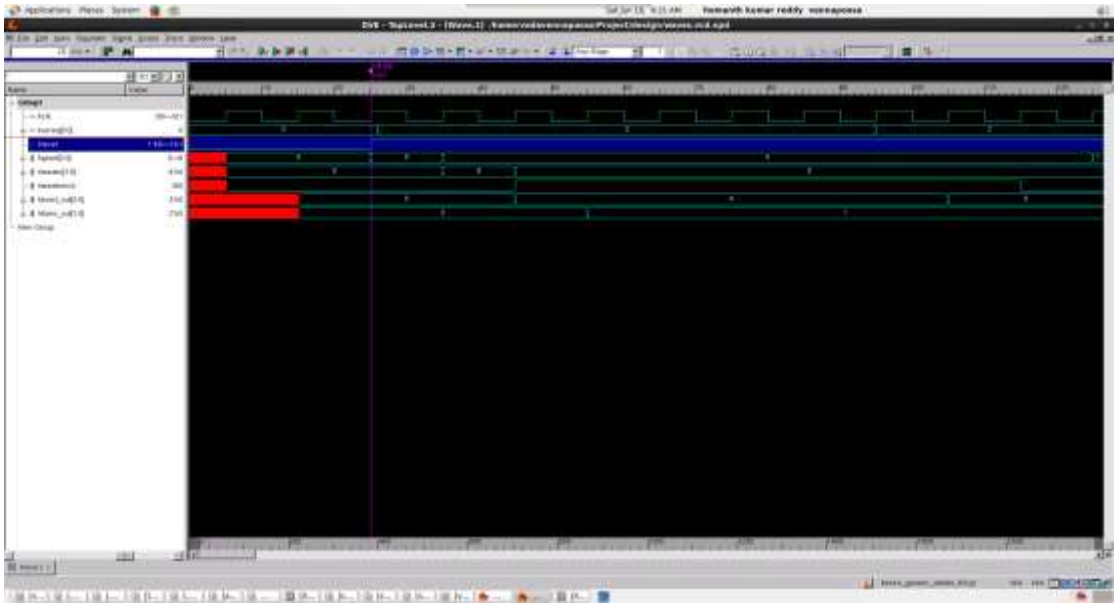

Figure 6.1 Reset Operation

➢ **Default Master**

Initially the bus is with master-3 as we can see in the Figure 6.2. So, the master-3 will become the least priority master as per the Round Robin Arbitration scheme. When mastr-3 is having bus master-8 requesting for the bus. So the Arbiter will give the bus to the master-8 after master-3 completes its transactions. And after master-8 there are no other requests to the arbiter for bus access. So, the arbiter will grant the bus to the default master i.e., master-3. Since master-3 is the least priority device. From the Figure 6.2 we can see at 5^{th} posedge there are no other requests and after master-8 completes its transaction at 8^{th} posedge and bus is hand out to the default master master-3.

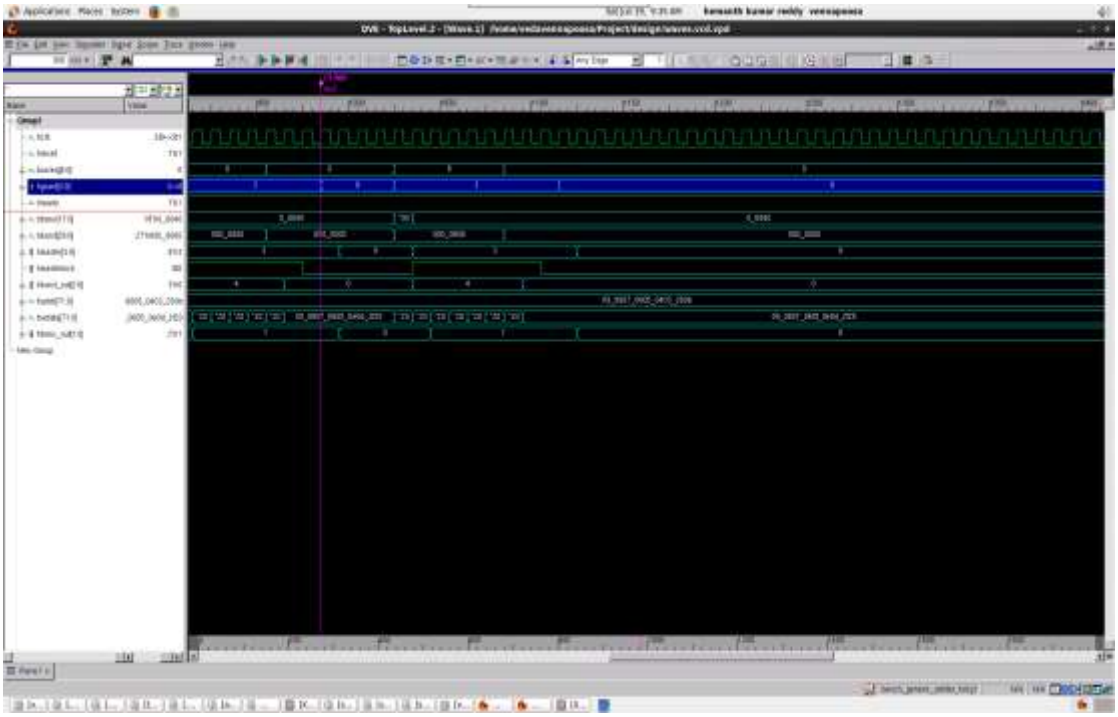

Figure 6.2 Default master

- **Master Switchover**

Initially the bus is with master-1 and master-2 is requesting for the bus at the same time. So, the arbiter will give access to master-2 after master-1 completes its transaction. From the Figure 6.3 we can see at 2nd posedge master-2 requests and at 11th posedge master-1 completes its transaction and bus is given to the master-2.

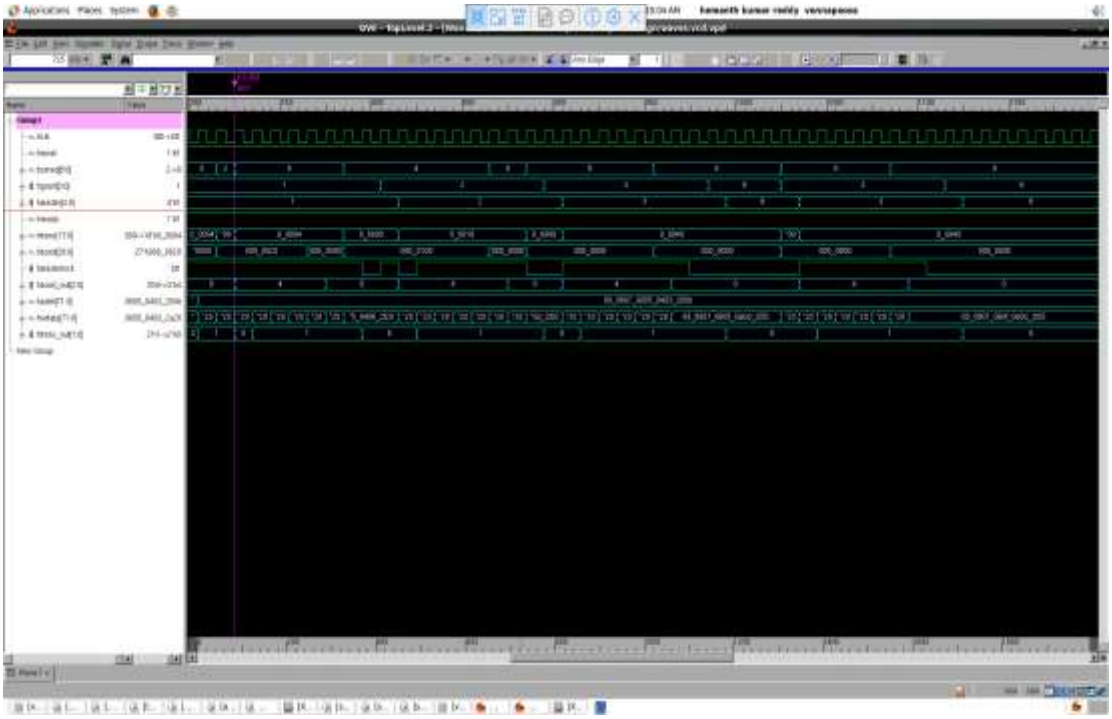

Figure 6.3 Master switchover

- **Wrap Cycle with Wait State and Idle State**

This version of the design supports wait and idle state. For both wait and idle state the data is not sampled by the slave and it can be seen in the Figure 6.4.

In fig 6, at 4th and 5th positive clock edge the idle signal comes and hence no address or data phase occurs in these two cycles and on the 6th positive clock edge the address and the data phase of the master starts as htrans is now sequential.

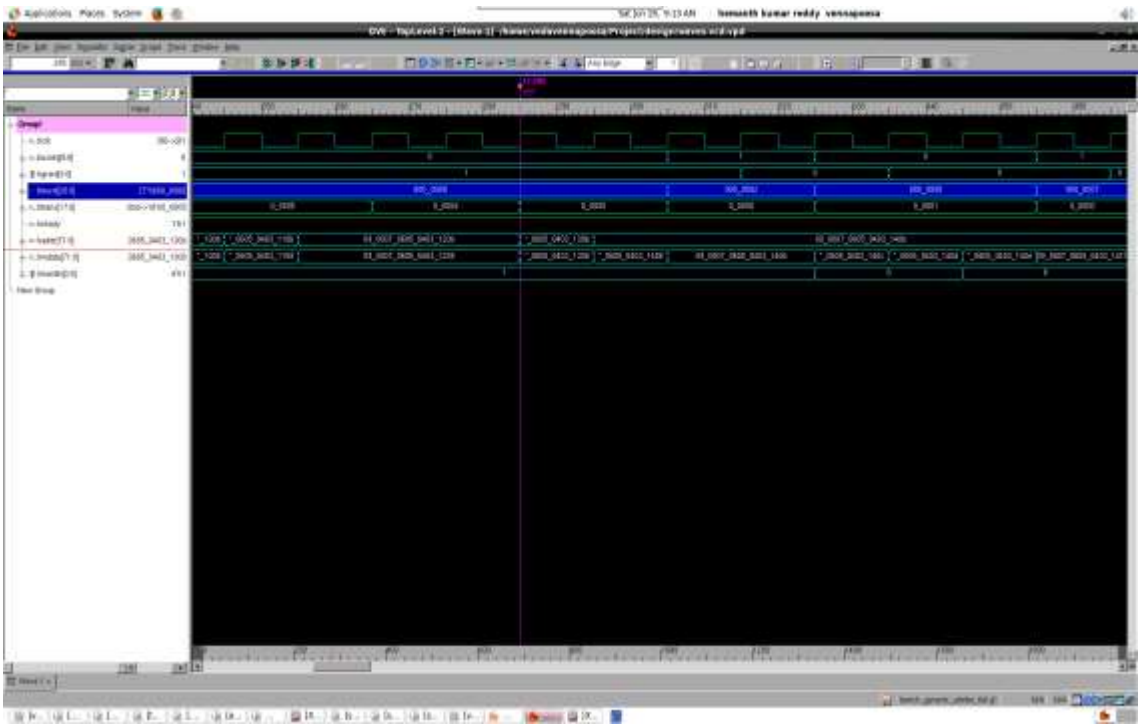

Figure 6.4 Wrap cycle with wait state and idle state

6. Static timing analysis (STA) Reports

Design Compiler (DC) tool from Synopsys was used for carrying out the STA. Netlist and the timing reports are generated.

- **Setup Report**

Setup slack is the margin by which a timing path meets setup check requirement. If setup slack is positive, it means the timing path meets setup requirement. On the other hand, a negative setup slack means setup violating timing path. If, by chance, a fabricated design is found to have a setup violation, you can still run the design at less frequency than specified. From Figure 6.5 we can see that this design don't have negative setup slack.

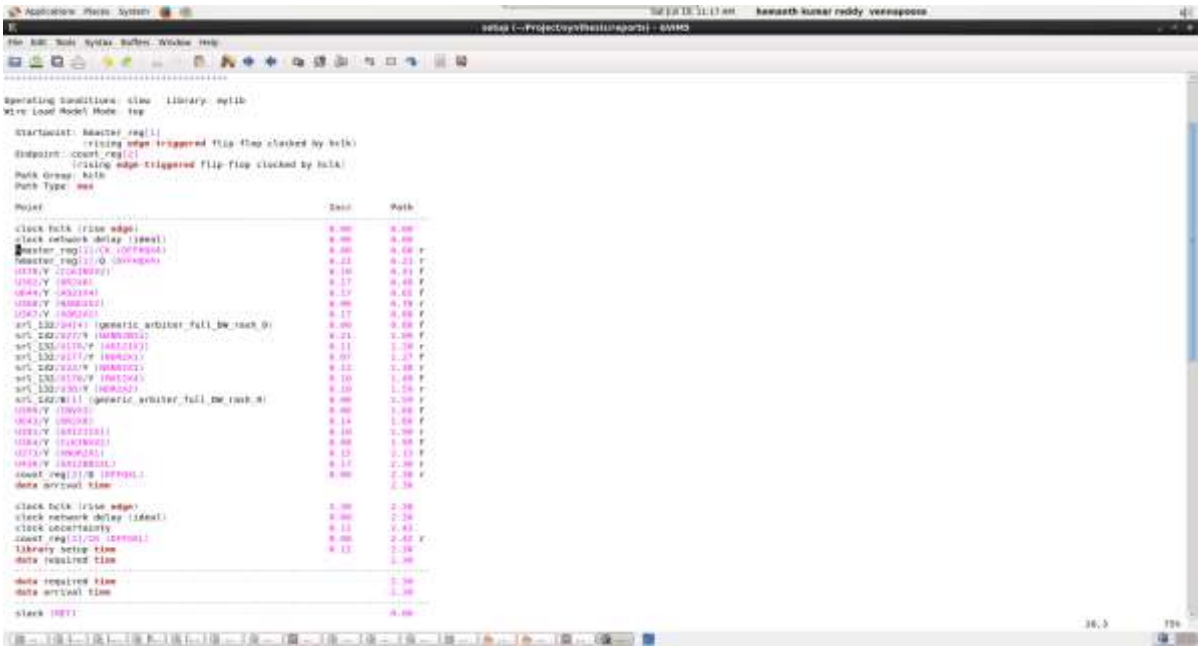

Figure 6.5 Setup report

- **Hold Report**

Similar to setup slack, the presence and magnitude of hold violation is governed by a parameter called as hold slack. If hold slack is positive, it means there is still some margin available before it will start violating for hold. A negative hold slack means the path is violating hold timing check by the amount represented by hold slack. From Figure 6.6 we can see that this design don't have negative hold slack.

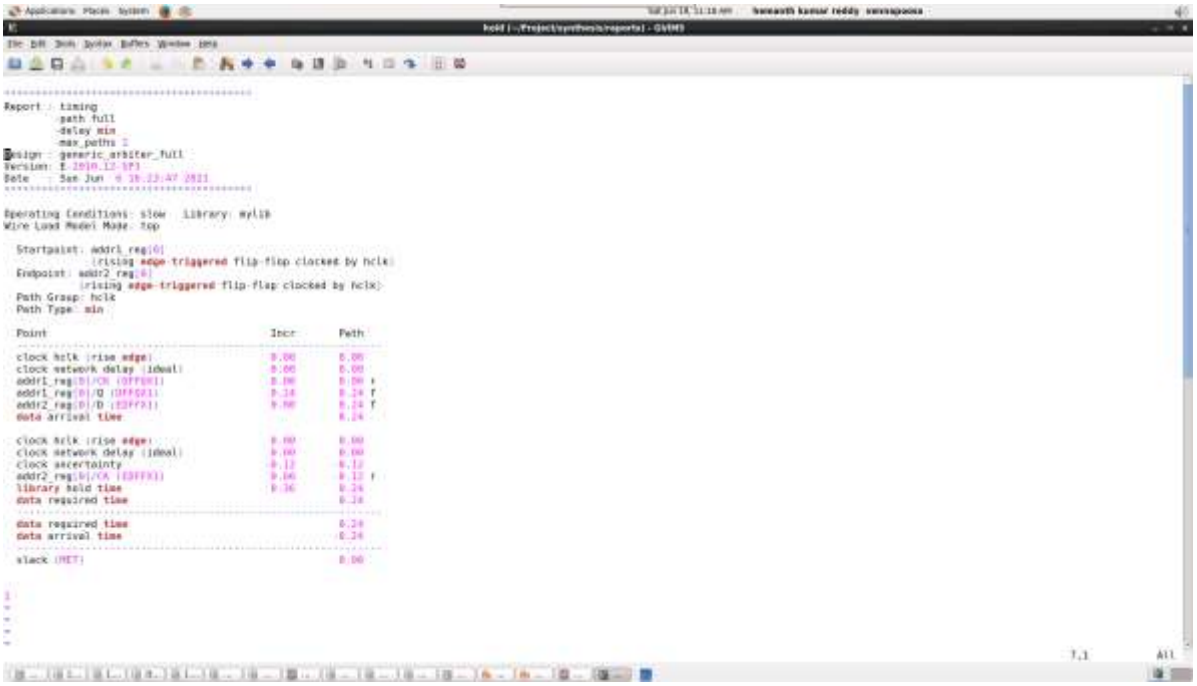

Figure 6.6 Hold report

7. Conclusion

The design of Efficient AMBA AHB arbiter includes generic number of masters i.e., we can add or remove the masters in the system depending upon the requirement. And it is designed using Verilog HDL (Hardware Description Language). Round Robin (RR) arbitration scheme is used in this design. In RR arbitration scheme the priority of each master is keep on changing dynamically. So, the implementation of Default Master is very challenging. Apart from the design Static Timing Analysis (STA) is done to the design by using Design Compiler (DC), which is from Synopsys. One of the research papers as considered as reference to improve clock frequency and slack values over it. In STA fixing setup and hold violations is very challenging task. There are some techniques to fix these violations. But as this design is little large, fixing in one path may affect the other paths. Finally, all these violations are fixed and also improved the clock frequency and slack values over the referred design and generated the Gate level Netlist.

8. Acknowledgement

I express my sincere thanks to Mrs. Harika Chaparala, Senior Member of Technical Staff, Invecas, Guntur, for her continuous support and guidance and valuable inputs and also for her congenial cooperation which helped me in carrying out this Project Work. I convey my deep sense of gratitude to Prof. K. Malakondaiah, Director, VEDAIIT Amaravati, for providing me with an opportunity for carrying out this Project Work under his esteemed guidance, and for his support in providing all the essentials. I also thank Mrs. Madhuri Nallapaneni Director, INVECAS Amaravati, for providing me an opportunity for internship in INVECAS and support offered towards the Project Work.

9. References

[1] AMBA | Specifications – Arm Developer

[2] Pravin S. Shete1 , Dr. Shruti Oza2,"Design of an AMBA AHB Reconfigurable Arbiter for On-chip Bus Architecture", International Journal of Application or Innovation in Engineering & Management (IJAIEM)

[3] YJ Huang, YH chen, CK yang, and SJ Lin, "Design and Implementation of a Reconfigurable Arbiter", 7th WSEAS International conference Signal, Speech and Image Processing, China, 2007.

[4] https://www.arm.com/

[5] https://asic4u.wordpress.com/2018/01/06/asic-design-flow-outline-part-1/

[6] What is Static Timing Analysis (STA)? – Overview | Synopsys

[7] http://www.vlsi-expert.com/2011/04/static-timing-analysis-sta-basic-part3b.html

[8] https://www.eetimes.com/a-look-inside-behavioral-synthesis/

[9] https://www.synopsys.com/glossary/what-is-static-timing-analysis.html