AMBA AHB Bus Protocol Checker

[1]Sidhartha Velpula, student, ECE Department, KL University, India,

[2]Vivek Obilineni, student, ECE Department, KL University, India

[3]Syed Inthiyaz, Asst.Professor, ECE Department, KL University, India

[4]Siva Kumar Munuswamy, Asst.Professor, ECE Department, KL University, India

Abstract :

The communication between the integrated IP's is enabled by a common, shared on chip bus. The behavior of the bus becomes vital to the working of soc. We propose an efficient rule based bus protocol debugging mechanism. The scope of this paper is to develop basic components of an AMBA AHB bus namely master, slave, arbiter and decoder. The models are developed in verilog language, simulated on modelsim simulator, synthesized with XILINX tool.

Key words: Integrated IP's ,SOC, AMBA AHB bus, Model sim,XILINX.

1.Introduction:

The Advanced microcontroller bus architecture (AMBA) protocol is an open standard, on-chip bus specification that details a strategy for the interconnection and management of functional blocks that makes up a System-on-chip (SOC). It facilitates "right-first-time" development of embedded processors with one or more CPU/signal processors and multiple peripherals. The AMBA protocol enhances a reusable design methodology by defining a common backbone for SOC modules. Originally developed in 1995, the AMBA Specification has been refined and extended with additional support to provide the capabilities required for soc design, culminating in the current AMBA 3 specification. IP re-use is an essential component in reducing SOC development costs and timescales.

1.1Introduction to AMBA busses:

The *Advanced microcontroller bus architecture* (AMBA) specification defines on chip communications standard for designing high-performance embedded microcontrollers. Three distinct bussed are defined within the AMBA specification. They are

The *Advanced High-performance Bus* (AHB)

The *Advanced System Bus* (ASB)

The *Advanced Peripheral Bus* (APB)

A test methodology is included with AMBA specification which provides an infrastructure for modular macrocell test and diagnostic access.

1.2Advanced High-performance Bus (AHB):

The AMBA AHB is for high-performance, high clock frequency modules. The AHB acts as the high-performance system *backbone* bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions.

1.3Advanced System Bus (ASB):

The AMBA ASB is for high-performance system modules. AMBA ASB is an alternative system bus suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions.

1.4 Advanced Peripheral Bus (APB):

The AMBA APB is for low-power peripherals. AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB

can be used in conjunction with either version of the system bus.

1.5 Typical AMBA System:

AMBA AHB:

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation.

2.Design methodology:

This includes the features of AMBA AHB protocol and its components like master, slave, arbiter, decoder and its interconnections.

2.1 AMBA AHB protocol features:

AMBA AHB implements the features required for high-performance, high clock frequency systems including:

- Burst transfers
- Split transactions
- Single-cycle bus master handover
- Single-clock edge operation
- Non-tristate implementation
- Wider data bus configurations(64/128 bits).

Bridging between this higher level of bus and the current ASB/APB can be done efficiently to ensure that any existing designs can be easily integrated.

An AMBA AHB design may contain one or more bus masters, typically a system would contain at least the processor and test interface. However, it would also be common for a *Direct Memory Access*(DMA) or *Digital Signal Processor*(DSP) to be included as bus masters. The external memory interface, APB bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave.

A typical AMBA AHB system contains the following components:

AHB master A AHB master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any time.

AHB slave A bus slave responds to a write or read operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

AHB arbiter The bus arbiter ensures that only one master at a time is allowed to initiate data transfers. An AHB would include only one arbiter, although this would be trivial in single bus master systems.

AHB decoder The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A single centralized decoder is required in all AHB Implementations.

2.2 Bus interconnection:

The AMBA AHB bus protocol is designed to be used with a central multiplexer interconnection scheme. A central decoder is also required to control the read data and response signal multiplexer, which selects the appropriate signals from the slave that is involved in the transfer.

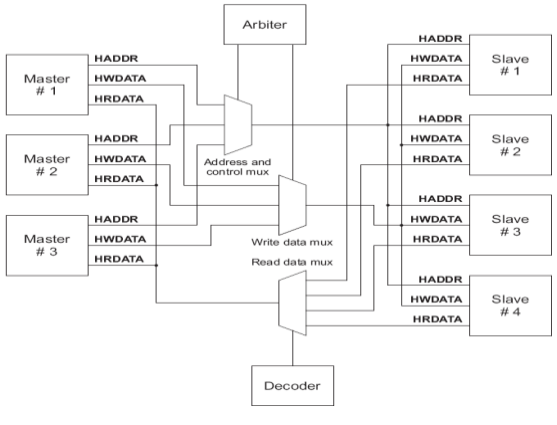


Figure 2-1 Multiplexer interconnection

3.Design analysis:

3.1 Basic transfer:

An AHB transfer consists of two distinct sections:

The address phase, which lasts only a single cycle

The data phase, which may require several cycles. This is achieved using HREADY signal.

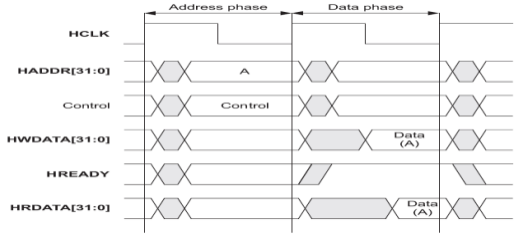


Figure 3-1 Simple transfer

In a simple transfer with no wait states:

The master drives the address and control signals onto the bus after the rising edge of HCLK.

The slave then samples the address and control information on the next rising edge of the clock.

After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

3.2 Transfer with wait states:

The address phase of any transfer occurs during the data phase of previous transfer. This overlapping of address and data is fundamental to the pipelined nature of the bus and allows for high performance operation, while still providing adequate time for slave to provide the response to a transfer. A slave may insert wait states in to any transfer as shown in the figure 3.2 which extends the transfer allowing additional time for completion.

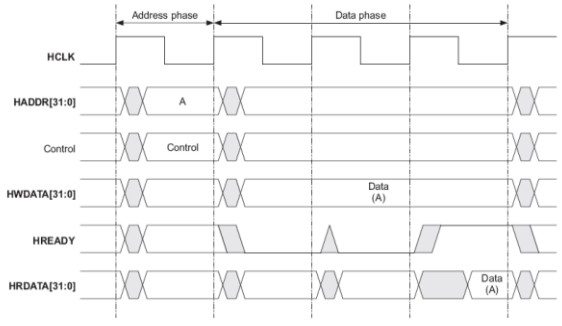


Figure 3-2 Transfer with wait states

For write operations the bus master will hold the data stable throughout the extended cycles. For read transfers the slave does not have to provide valid data until the transfer is about to complete.

When a transfer is extended in this way it will have the side-effect of extending the address phase of the following transfer. This is illustrated in figure 3.3 which shows three transfers to unrelated addresses, A, B &C.

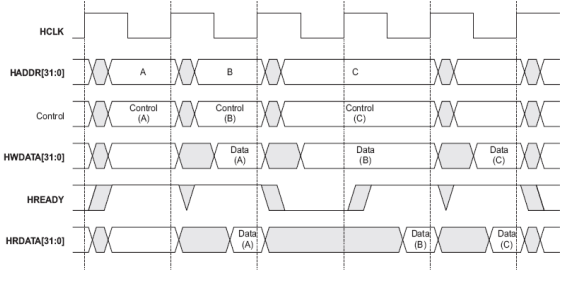


Figure 3-3 Multiple transfers

In figure 3.3,

The transfers to addresses A and C are both zero wait state

The transfer to address B is one state

Extending the data phase of the transfer to address B has the effect of extending the address phase of the transfer to address C.

3.3Burst operation:

Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as undefined-length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol:

Incrementing bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address.

For wrapping bursts, if the start address of the transfer is not aligned to the total number of bytes in the burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached.

3.4Master Interface diagram:

The interface diagram of an AHB bus master shows the main signal groups.

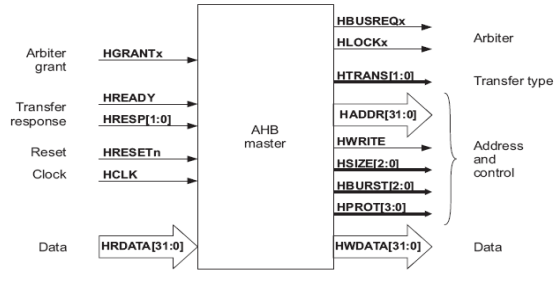


Figure 3-4 AHB bus master interface diagram

3.4.1Bus Master Timing diagrams:

The following diagrams shows the timing parameters related to an AHB bus master operating in an AMBA system:

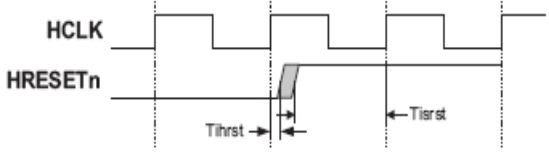


Figure 3.5 shows the AHB master reset timing parameters.

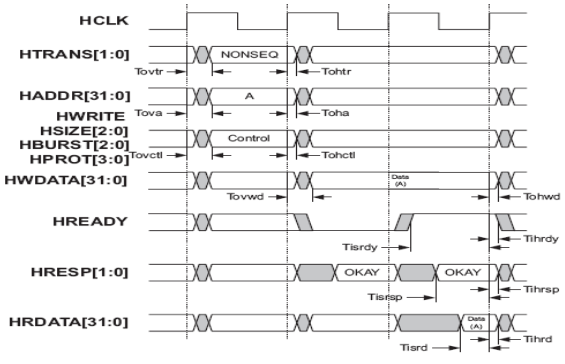


Figure 3.6 shows the AHB master transfer timing parameters.

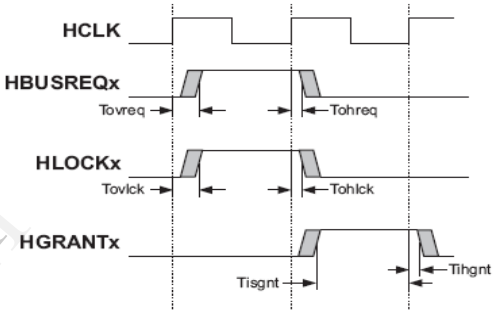


Figure 3.7 shows the AHB master arbitration timing parameters.

3.5AHB bus slave:

An AHB bus slave responds to transfers initiated by bus masters within the system. The slave uses HSELx select signal from the decoder to determine when it should respond to a bus transfer. All other signals required for the transfer, such as the address and control information, will be generated by bus master.

3.5.1Interface diagram:

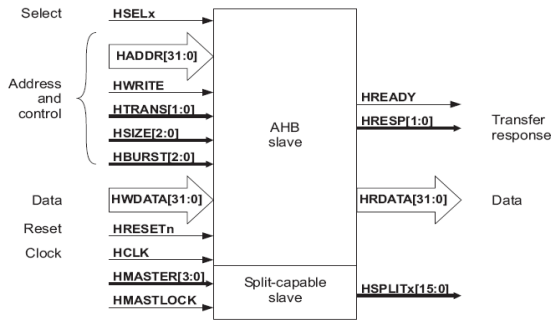


Figure 3. 8 shows an AHB bus slave interface.

3.5.2Timing diagrams:

The following diagrams show the timing parameters related to an access to an AHB bus slave operating in an AMBA system:

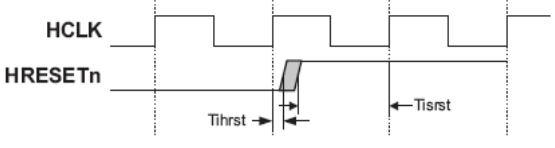


Figure 3.9 shows the AHB slave reset timing parameters.

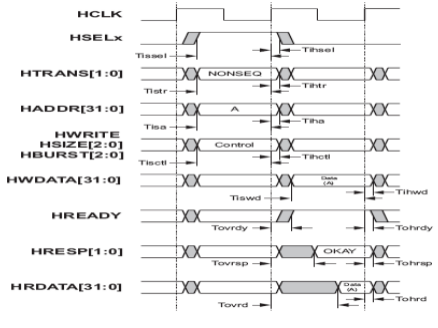
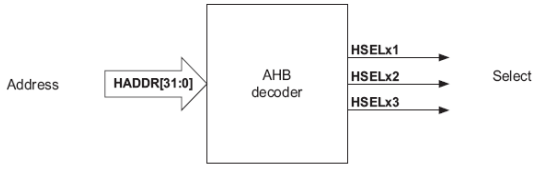


Figure 3. 10shows the main AHB slave timing parameters.

3.6 AHB Decoder:

The decoder in an AMBA system is used to perform a centralized address decoding function, which improves the portability of peripherals, by making them independent of the system memory map.

3.6.1Interface diagram:



3.6.2Timing diagram:

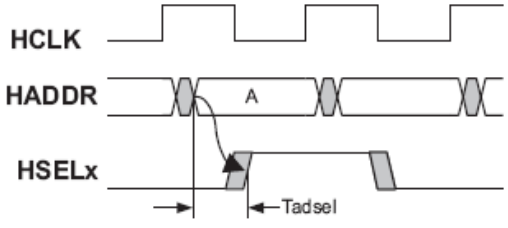


Figure 3-11 AHB decoder timing parameter.

3.7AHB arbiter:

The role of an arbiter in an AMBA system is to control which master has access to the bus. Every bus master has a REQUEST/GRANT interface to the arbiter and the arbiter uses a prioritization scheme to decide which bus master is currently the highest priority master requesting the bus. Each master also generates an HCLOCKx signal which is used to indicate that the master requires exclusive access to the bus. The detail of the priority scheme is not specified and defined for each application. It is acceptable for the arbiter to use other signals, either AMBA or non-AMBA, to influence the priority scheme that is in use.

3.7.1Interface diagram:

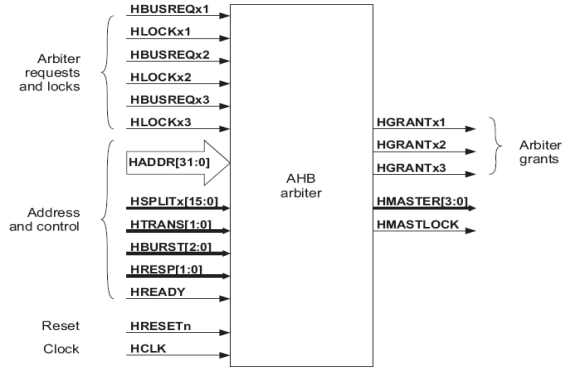


Figure 3.12 signal interface of an AHB arbiter.

3.7.2Timing diagrams:

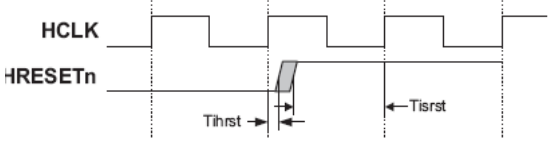


Figure 3.13 shows the AHB reset timing parameters

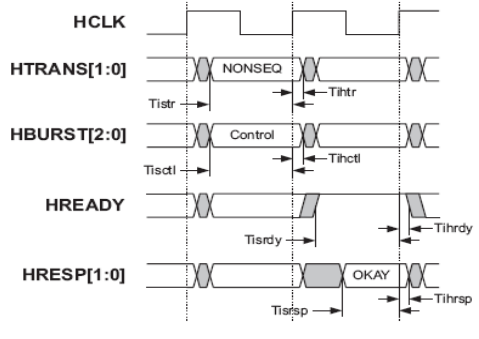


Figure 3.14 AHB transfer timing parameters

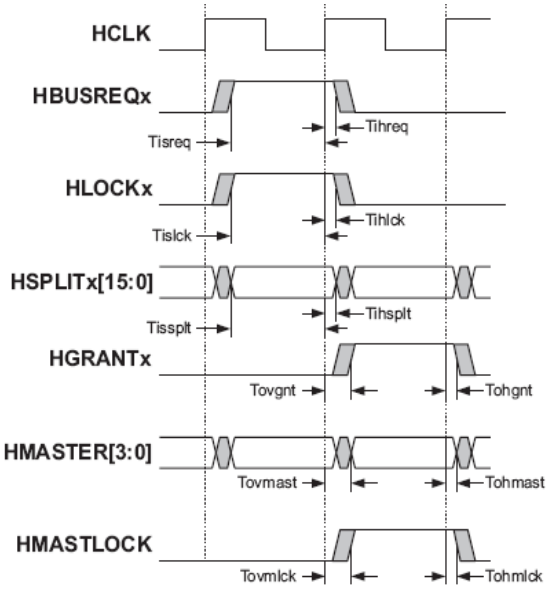


Figure 3.15 AHB split timing parameters.

3.8AHB Protocol checker:

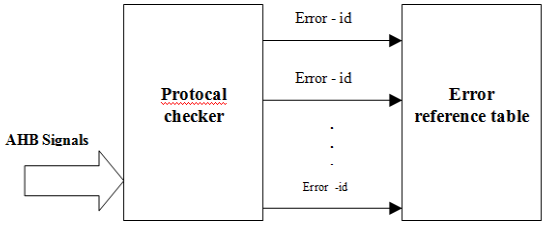


Figure 3-16 Protocal checker

Figure 3.16 shows AHB protocol checker (HP Checker) architecture, which contains two main function blocks: Protocol checker, Error Reference Table.

3.9Protocol Checker:

HPChecker is a rule-based protocol checker, thus how to establish a set of well-defined rules is very important. Besides, according to our design experiences, we add new rules to increase our error finding ability. In conclusion, our protocol checker has rules, including master-related rules,

slave-related rules, reset-related rules, and bus components-related rules. Bus components include arbiter and decoder. Protocol checker is the main core of HP Checker, the inputs are all AHB bus signals, and the outputs are error signals and corresponding master and slave ID's. The Master/Slave ID is formed by HMASTER signal. If an error occurs, the HPChecker will output the corresponded master ID number or slave ID number to indicate which master or slave violates the AHB protocol.

3.10 Error Reference Table(ERT):

Traditional protocol checkers will assert error signals or a printout error message for every error occurs,but this way is very inefficiently for early debugging in long simulation period. In early debugging period, there may have many errors, especially when test bench or real case application ahs very long execution time. Such large amount of log messages or error signals waveform could help designer to debug limitedly, because that will be hard to read or analyze. Moreover, we do not care about the timing information when error occurred in most case. Thus we provide error reference table that can summarize what errors have been occurred.

4.Results:

4.1 Master results:

4.1.1Master single transfer results:

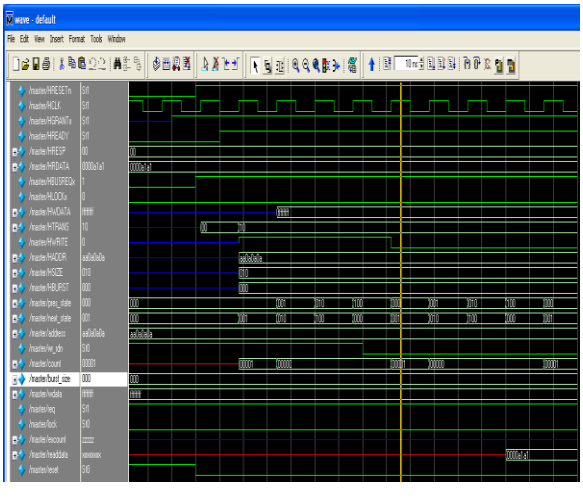


Figure 4-1 master single transfer results

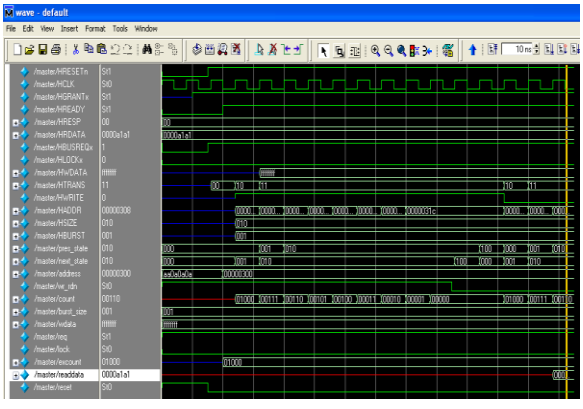


Figure 4-2 Master incrementing burst transfer results

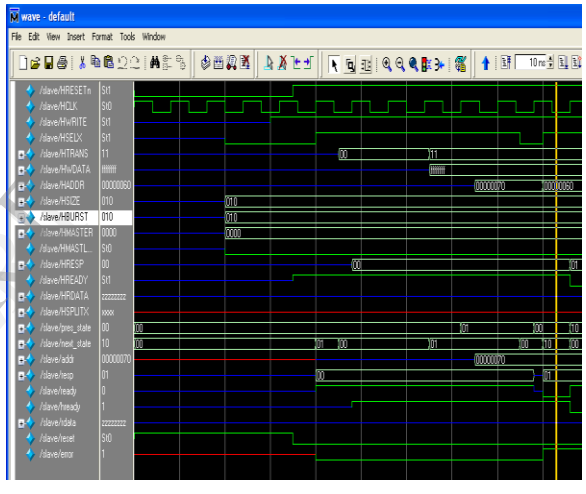


Figure4-3 slave results

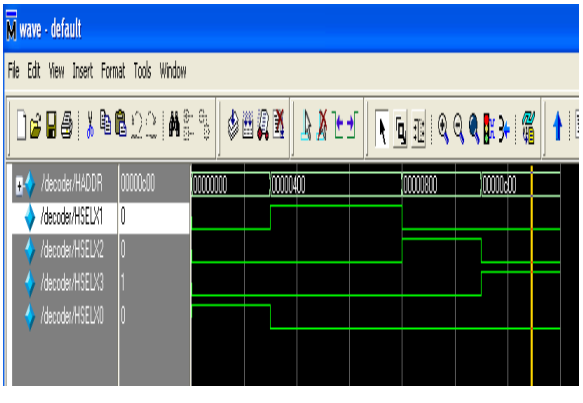


Figure 4-4decoder results

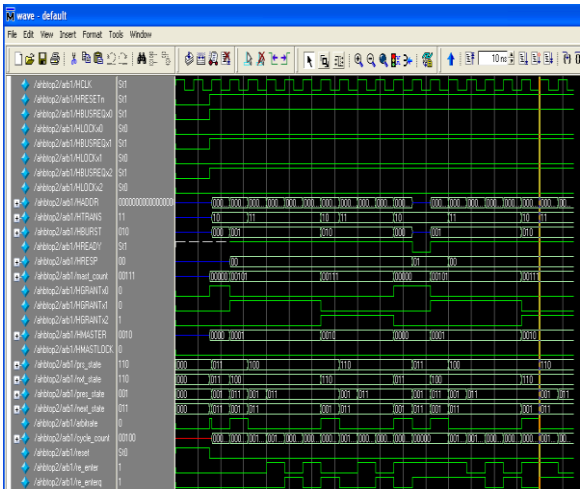


Figure4-5 arbiter results

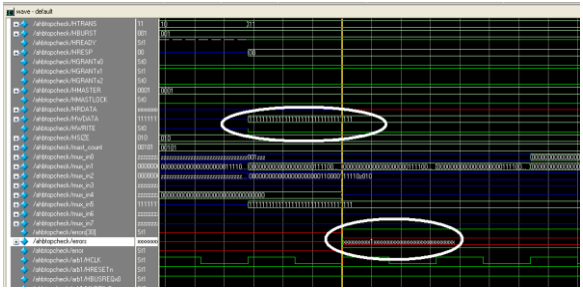


Figure4-6 error results

5.Future scope

The future scope of the project protocol checker is , to include some additional features like split capable slave, protection mode and window trace buffer. SPLIT transfers improve the overall utilization of the bus by separating (or splitting) the operation of the master providing the address to a slave from the operation of the slave responding with the appropriate data.

6.References

[1]. AMBA™ Specification (Rev 2.0)

[2]. **IEEE xplore 2008** AMBA AHB Bus Protocol Checker with Efficient Debugging Mechanism Yi-Ting Lin, Chien-Chou Wang, and Ing-Jer Huang department of Computer Science and Engineering National Sun Yat-sen University Kaohsiung, 80424 , Taiwan

[3]. AHB Example AMBA System Technical Reference Manual ARM

[4]. IEEE Standard Verilog® Hardware Description Language

(LRM)IEEE Std 1364-2001 (Revision of IEEE Std 1364-1995)

[5]. Verilog HDL A guide to Digital Design and Synthesis Samir

Palnitkar SunSoft Press 1996.

[6]. Unix course material - Tata Elxsi.doc