ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Assertion Based Verification Environment Development Using System Verilog

N.Karthik
M.Tech VLSI,
CMR Institute of Technology,
Kandlakoya Village,
Medchal Road Hyderabad.

M.Gurunadha Babu
Professor & H.O.D,
CMR Institute of Technology,
Kandlakoya Village,
Medchal Road Hyderabad.

Ms. Muni Praveena Rela
Associate Professor,
CMR Institute of Technology,
Kandlakoya Village,
Medchal Road Hyderabad.

Abstract:

Writing assertions concurrently with the RTL design and keeping these assertions closely tied to the RTL code has been found to bring significant benefits in both the design and verification processes for digital hardware. The primary benefit is that assertions help to detect more functional bugs, detect them earlier in the process and detect them closer to their original cause. A secondary benefit is that the very act of formulating and writing assertions can give the designer a better understanding of the design, and hence uncover bugs in the specification or else avoid introducing bugs into the design in the first place. This paper describes assertion based system Verilog verification environment with a robust and widely used AMBA AHB bus protocol master-slave architecture.

Keywords:

AMBA, AXI, Verification, System Verilog, Coverage Driven Verification, Functional Verification, Assertion, AHB, Functional Coverage.

1) Introduction:

Assertions bring the possibility of increased metrication to the verification process. Assertions directly increase observability of the state of the design during verification. By measuring and controlling the density of assertions and logging assertion passes as well as failures, it is possible to bring some science to the task of knowing when functional verification is complete.
This approach has following advantages.
• Tracks the behavioral aspects of the design under verification.

oCycle accuracy checks.

oHorizontal Coverage.

• Test-Bench Enhancements

oUse of Assertions.

oUse of Verification Metrics to check completeness.

oIndustry Standard Methodology.

• Simple and Robust Bus Architecture

oAHB - Advanced High Performance Bus

Assertion-based verification (ABV) is a methodology in which designers use assertions to capture specific design intent and, either through simulation, formal verification, or emulation of these assertions, verify that the design correctly implements that intent. Assertion-based verification (ABV) is a technique that aims to speed one of the most rapidly expanding parts of the design flow. It can also be used in simulation, emulation and silicon debug.

Research has suggested that verification can take up 70% of the time and cost of a full design cycle and that, within that, functional verification can take up more than half of the verification time. A number of studies have concluded the use of ABV reduces functional verification dramatically compared with traditional methods.

In assertion-based verification, RTL assertions are used to capture design intent in a verifiable form as the design is created, providing portable monitors that check for correct behavior. During simulation, assertions improve observability coverage, making the source of an error evident. Simulation debug time is greatly reduced. As targets for formal verification, assertions improve controllability coverage.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

When verifying assertions, formal verification algorithms explore the equivalent of billions and billions of input patterns without requiring test vector creation. And ideally, the same assertions are used for both simulation and formal verification so there is no need to repeatedly specify the same assertions multiplied different ways for different tools.

2) Assertion flow and steps:

It is a 4 step iterative process.

- Write
- Verify
- Debug -> Check Metrics
- Fix Bugs

This results in faster turnaround time.

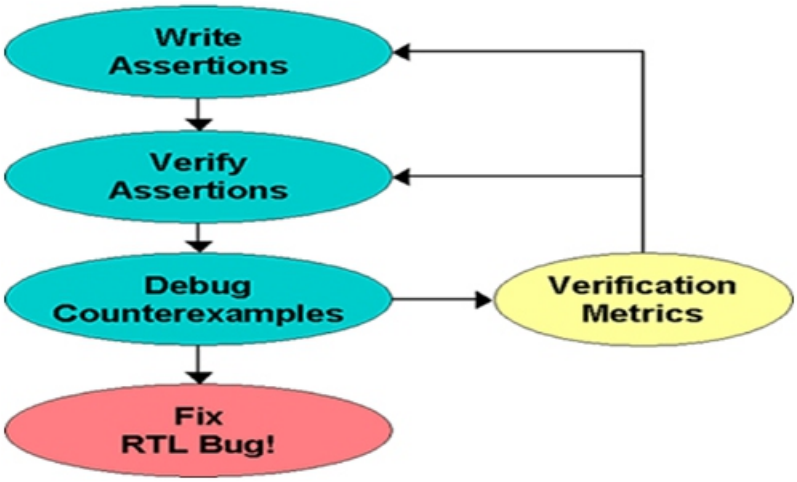

Figure 1: Assertion Flow

Benefits:

- Assertions actively monitor a design or test-bench to ensure correct functional behavior

- Assertions detect design errors at their source, greatly increasing observability and decreasing debug time

- With assertion languages like SVA part of SystemVerilog and PSL standardized, the ABV methodology today is supported by a wide array of tools and work with Verilog, SystemVerilog, and VHDL designs

- The availability of assertion libraries, for example Mentor's Questa and Questa Formal Verification product both include a rich assertion library, ABV is easy to adopt

- Assertions have a high ROI in that they can be used across simulation, formal verification, and even emulation

We must also consider that implementing assertions presents the following challenges.

- Implementation Need to be consistent.
- Coding and debug Need proper and clear coding guidelines
- Customization Flexibility of off-the-shelf assertion
- Metrics Defining and tracking the coverage

A distinction must be drawn between the use of assertions and their integration within a coherent and effectively implemented methodology.

3) AMBA AHB Architecture :

AMBA AHB addresses the requirements of high-performance synthesizable designs. It is a bus interface that supports a single bus master and provides high-bandwidth operation.

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

- Burst transfers
- Single-clock edge operation
- Non-tristate implementation
- Wide data bus configurations viz. 64, 128, 256, 512, and 1024 bits

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

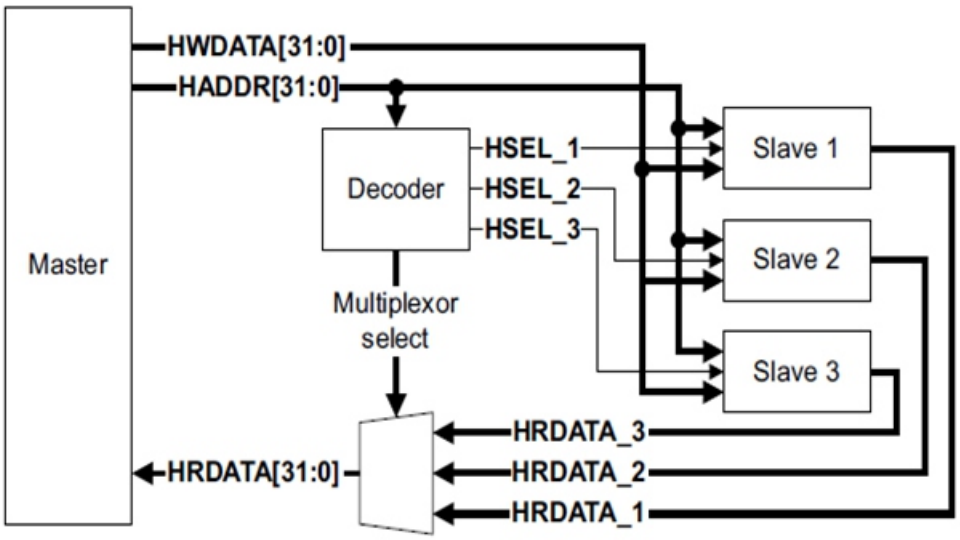

Figure 2: AHB Master-Slave Architecture

The most common AHB slaves are internal memory devices, external memory interfaces, and high bandwidth peripherals. Although low-bandwidth peripherals can be included as AHB slaves, for system performance reasons they typically reside on the AMBA Advanced Peripheral Bus (APB). Bridging between this higher level of bus and APB is done using a AHB slave, known as an APB bridge. The bus interconnect logic consists of one address decoder and a slave-to-master multiplexor. The decoder monitors the address from the master so that the appropriate slave is selected and the multiplexor routes the corresponding slave output data back to the master.

The main component types of an AHB system are:

- Master
- Slave
- Decoder
- Multiplexor

3.1 AHB Bus Structure and Signals:

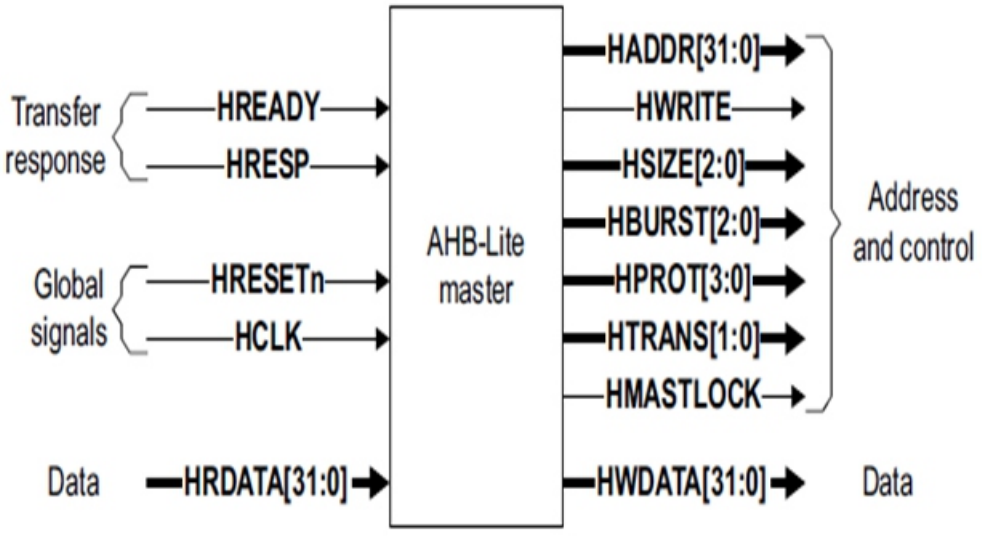

Figure 3: AHB Bus Structure

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

• HCLK
o The clock is an input to all elements in an AHB system and is assumed to come from some external clock generator.
• HRESETn
oThis active low signal is an input to all elements in AHB.
• HADDR
oA 32 bit bus, output from the master, which indicates the address to be used for a transfer.
• HSIZE
oThis master output indicates the size of the read or write transfer to be performed.
• HTRANS
oAn output from the master (input to slave) which shows the type of transfer to be done.
• HWRITE
o An output from the master which shows the direction of data transfer.
• HBURST
oThe master generates this bus to tell the system about the kind of burst it is performing.
• HPROT
o This allows the master to tell the slave characteristics of the transfer.
• HWDATA
oThe master output which contains the data it wants to write to an address in a slave.
• HRDATA
oThe slave output (input to the master) which contains data that is read from a slave.
• HSELx
oA set of system-specific decoder outputs, generated combinatorially from the decoder's HADDR input
• HRESP
oAn output from the slave (input to the master) which indicates whether the transfer was successful
• HREADY
oThis is an output from each slave which shows when the transfer is completed.
• HLOCK/HMASTLOCK
o Indicates a locked transfer.
• HMASTER
oAn arbiter output, in a full AHB system, showing the number of the currently selected master.
• HSPLIT
oAn output from split-capable slaves, to show the arbiter which masters are currently waiting a response from that

3.2 Verification Environment of AHB

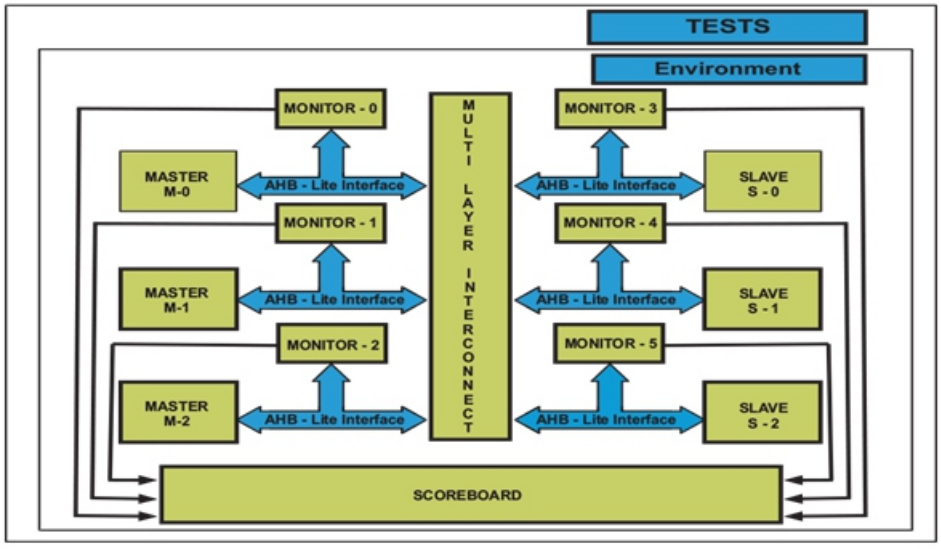

Figure 4: AHB Verification Environment

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

3.3 Verification Environment Components:

•Verification Plan
oContains list of all features and test-cases to verify them.

•Assertion Matrix
oContains all the assertions definition which are to be implemented. Each assertion captures specific design intent or a design check.

•Test Plan
oIt captures the coverage points to be hit and the metrics for coverage closure.

•AHB Master Agent
oThis contains the Master behavior as to how to drive the signals on the DUT as per the design protocol.

•Assertions Monitor
oThe module which implements all the assertions defined in the Assertion Matrix.

•AHB Scoreboard
oThe class which compares the expected and actual DUT responses and validates the system behavour.

•Tests
oThe SV class instantiates the verification environment and initiates the stimulus on the design.

•Sequences
oThe actual stimulus given to the DUT, which wiggles the design pins.

Tools Used:

•Modelsim
•Questasim

3.4 AHB Write and Read Transfers

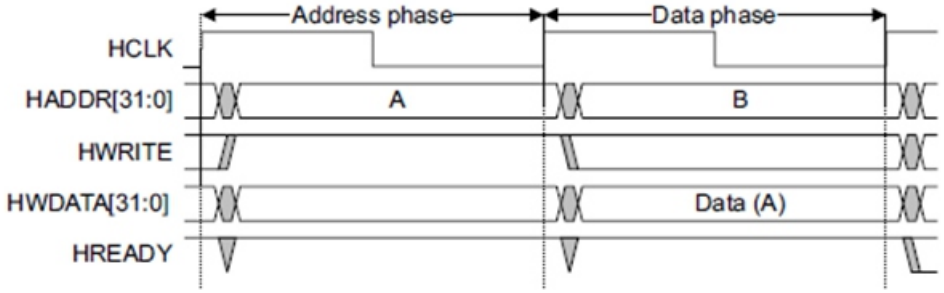

Figure 5: AHB Write Transfer

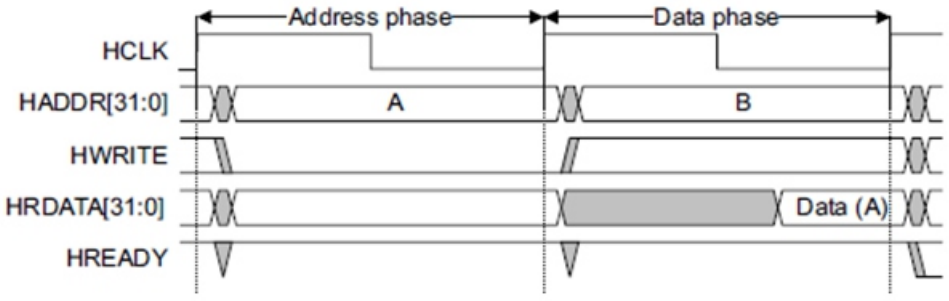

Figure 6: AHB Read Transfer

An AHB Transfer consists of 2 Phases.
•Address Phase
o1 single HCLK cycle.
•Data Phase
oSeveral HCLK cycles.
Direction of Data Transfer is controlled by HWRITE
•1 -- Write
•0 -- Read

HREADY indicates whether slave can accept the transfer or notThe simplest transfer is one with no wait states, so the transfer consists of one address cycle and one data cycle.

In this case:

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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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

• After the slave has sampled the address and control it can start to drive the appropriate HREADY response. This response is sampled by the master on the third rising edge of HCLK.

If we have wait cycles slave pulls HREADY low until it is ready to receive again.For write operations the master holds 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 has the side-effect of extending the address phase of the next transfer.

3.5 AHB Transfers and other features:

AHB Transfer types can be classified into 4 types based on data transfers:

• IDLE: Indicates that no data transfer is required

• BUSY: The BUSY transfer type enables masters to insert idle cycles in the middle of a burst

• NONSEQ: Indicates a single transfer or the first transfer of a burst.

• SEQ: The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer.

AHB Transfer types can be classified into 2 types based on privileged access:

• Normal: Any Master can access this region.

• Protected: Only a specific master can access this space and it has locked it.

AHB Bursts can be classified into 3 main types:

• Single: A single location is accessed in a transfer.

• Incremental: A set of locations are accessed continuously.

• Wrapping: After set of locations are accessed continuously and when transfer boundary is reached address goes back to initial starting address.

AHB transfers can be classified into 3 main types based on protection control:

The below terms follow standard definitions.

• Bufferable

• Cacheable

• Privileged

• Opcode Transfer

• Data Transfer

4) CONCLUSIONSL:

Assertions can be used to capture the information about various levels such as conceptual where we can verify system level properties which are more architectural level and also at design level where unit level properties are expressed.

Deploying assertions has several advantages which can be summarized as below:

• Improving Observability.

• Reduces the debug time.

• Bugs can be found earlier and are more isolated.

• Controllable severity level.

• Can interact with C functions.

• Describe the Documentation and Specification of the design.

• Coverage information tells us where we are in terms of project completeness.

AHB protocol combines the advantages of less complex design and verification architectures with high performance and bandwidth with multiple transfer and protection features.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

5) FUTURE SCOPE:

AMBA AHB also can provide parallel communications with multi master bus management, high clock frequency, high performance systems for data transfer operation from the memory interfaced with the master or slave peripheral devices.AMBA AHB can support on chip communications standard for designing high-performance embedded microcontrollers. AHB can employ multilayer advanced high-performance bus matrix which has slave-side arbitration. Slave-side arbitration is different from master-side arbitration in terms of request and grant signals since, in the former, the master merely starts a burst transaction and waits for the slave response to proceed to the next transfer. Therefore, in the former, the unit of arbitration can be a transaction or a transfer. However, the bus matrix of ARM offers only transfer- based fixed-priority and round-robin arbitration schemes.SV Assertions can be used to increase the visibility within assertion circuits, and also enhance the coverage information provided by the checkers. Other forms of debug information, such as signal dependencies, can also be sent to the front-end applications.

6) References:

[1] Assertion-Based Verification using SystemVerilog - Verilab www.verilab.com/files/svug_2007_abv_litterick.pdf

[2] SOC Bus Transaction Verification Using AMBA ... - JSTS

[3] www.jsts.org/html/journal/journal_files/.../6/2020_vol2_no2_132.pdf

[4] http://www.cadence.com/products/fv/Pages/abv_flow.aspx

[5]http://www.techdesignforums.com/practice/guides/guide-to-assertion-based-verification/

[6]https://verificationacademy.com/courses/assertion-based-verification

[7]http://www.mentor.com/products/fv/methodologies/abv/