International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 06 | June 2021 www.irjet.net p-ISSN: 2395-0072

Verification of Advanced Peripheral Bus Protocol (APB V2.0)

Meghana Jain H K[1], Dr. Punith Kumar M B[2]

[1]*Student, Dept of Electronics and Communication Engineering, P.E.S College of Engineering, Karnataka, India*
[2]*Professor, Dept of Electronics and Communication Engineering, P.E.S College of Engineering, Karnataka, India*

Abstract - *Likewise with the improvement of Very Large Scale Integration Circuit (VLSI), billions of transistors being incorporated on a single chip turned out to be better known. System on Chip (SoC) is an Integrated Circuit(IC) that will incorporate all parts of computer or some other electronic components in it. System on Chip utilizes Advanced RISC Machines (ARM) based Advanced Microcontroller Bus Architecture (AMBA) transport for all the interconnections of the bocks present on the chip. AMBA comprises of three ports for communication. They are Advanced High Performance Bus (AHB), Advanced Peripheral Bus (APB) and Advanced System Bus(ASB). AHB and ASB are superior buses used to interface components which are high performance in nature. APB, which is essentially used to interface lower performance components with lower data transfer capacity like Timer, I/O peripherals, Universal Asynchronous Receiver and Transmitter (UART). Back in 1990's there was just a single essential language which was utilized to verify the designs. Verilog was in use to check the designs which were not complex. As the intricacy increased there was need for better Verification environment. System Verilog and Universal Verification Methodology (UVM) give better platform to carry out the verification. This paper focuses on verifying APB v2.0 Protocol using System Verilog and UVM.*

Key Words: **System on Chip, Universal Verification Methodology, System Verilog, Advanced Peripheral Bus, Advanced Microcontroller Bus Architecture.**

1. INTRODUCTION

Advanced Microcontroller Bus Architecture (AMBA) protocol is an on chip communication standard for designing high performance embedded microcontroller. AMBA was originally designed for Advanced RISC Machines (ARM). In order to achieve good performance with a processor and its components it is very important to have a very good interconnect which will connect all the components. AMBA protocol which consists of both high performance and low performance buses is thus advantageous. These protocols mainly define how these components will mainly communicate with each other. AMBA is also widely used in Internet of Things (IOT) sub systems, smart phones, networking the System on Chip (SoC's) etc... They provide IP re-use which will always reduce SoC development cost and the time scale. Hence thousands of SoCs mainly make use of AMBA as it provides flexibility, range of interface specification for different requirements. The AMBA architecture is shown in the Fig 1. The architecture as shown is always used to connect one or more microcontrollers along with several components like Digital Signal Processor (DSP), Direct Memory Access (DMA), memory and other peripheral components like timer, keypad etc. The AMBA architecture basically consists of two components in it. They are Advanced System Bus (ASB) OR Advanced High Performance Bus (AHB) and one more is APB. Some of the components like high bandwidth on chip Random Access Memory(RAM), Processor, Memory require high performance bus and hence are connected to ASP or AHB. And some of the low performance and lower bandwidth components like Timer, keypad will be interconnected using the APB bus. The bridge shown will connect the high performance AHB or ASB with the low performance ASP. Hence here the bridge acts as APB master and all other devices/components connected to it act as APB slave.

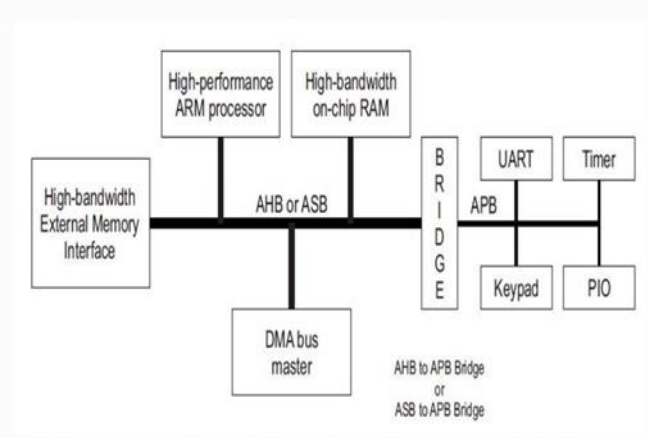

Fig -1: AMBA Block Diagram

Verilog which is an example of Hardware Description Language is used to describe the hardware behavior so that the devices can be digitally designed. To have better verification and to support testing of more complex circuits System Verilog and Universal Verification Methodology can be used.

1.1 Block Diagram of APB Protocol

The Advanced Peripheral Bus (APB) is one of the buses of the Advanced Microcontroller Bus Architecture. APB mainly defines a low-cost interface and is optimized for minimal power usage and is also having reduced interface complexity. APB Protocol is note pipelined. In APB, signal transitions are mainly represented at the rising edge of the clock. Every transfer in APB takes a minimum of two clock cycles. Fig 2 shows the block diagram of APB.

There is only single bus master in the APB and there is no need for any arbiter. APB Block Diagram is represented in

Fig 2. AMBA-APB usually consists of a single APB Bridge/master and can have many slaves. In addition, the APB Bridge will also behave as a slave on the high level system bus.

APB Master Description is described below:

- APB Bridge will convert the data and address from System bus transfer to APB.
- APB Master will latch the address and holds it valid throughout the transfer.
- APB Master will decode the address and will generate a peripheral select, PSELx. And only one select signal can be active during a particular transfer.
- APB Master will drive the data during write transfer.
- APB Master will drive the data on to system bus using read transfer.

APB Slave is having a simple but yet a flexible interface and it can be used to interface many slaves. The select signal that is PSELx, the address PADDR and the write signal represented as PWRITE can be combined to determine which register should be updated during the write transfers. During the read transfers the data can be driven on to the data bus and read operation can be carried when PWRITE is LOW and both the signals PSELx and PENABLE are high. PADDR is used to determine which register should be read during the transfer.

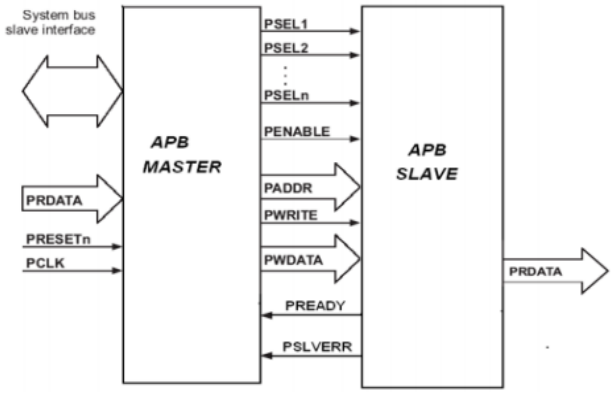

Fig-2: APB Block Diagram

2. SIGNAL SPECIFICATION OF AMBA APB2.0

The signals which are included in APB are explained as below:

- **PCLK:** The rising edge of PCLK will refer all transfers on the APB.
- **PRESETn:** Reset is system bus equivalent. It is a signal which is active when it is low. This signal will be normally connected directly to the system bus reset signal.
- **PADDR:** This is the address of APB Bridge. This represents the APB address bus. This can be up to 32 bits wide and will be driven by the peripheral bus bridge unit.
- **PPROT**: This is APB bridge Protection type. This signal indicates three conditions. They are the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access can also be dipicted.
- **PSELx:** APB Bridge Select. The APB master unit will generate this signal to one among the slaves by passing it to the bus of the slave. It indicates that the slave device is selected and that a transfer is required. There will be a PSELx signal for each slave.
- **PENABLE:** This is a signal of APB Bridge. This signal indicates the second and all subsequent cycles of an APB Transfer.
- **PWRITE:** Signal from APB bridge will give the direction.. This signal indicates an APB write access when it is asserted HIGH and an APB read access when it is asserted LOW.
- **PWDATA:** This is write data signal. This is asserted during write cycles when PWRITE is equal to one. This bus can be up to 32 bits wide in size.
- **PSTRB:** This signal represents APB bridge Write strobes. This signal will indicate which byte lanes to update during a write transfer. There is one write strobe for each eight bits of the write data bus signal. Hence PSTRB[n] is equal to PWDATA[(8n + 7):(8n)]. Write strobe must not be active when there is a read transfer.
- **PREADY:** This represents Slave interface is Ready. The slave will use this signal to extend an APB transfer.
- **PRDATA**: Slave interface Read Data. The selected slave will drive this bus during read cycles that is when PWRITE is LOW. This bus can be up to 32-bits wide in size.
- **PSLVERR**: This is signal of Slave interface This signal indicates that a transfer is failed. OPERATING STATES.

Do not use abbreviations in the title or heads unless they are unavoidable.

2.1 Operating States

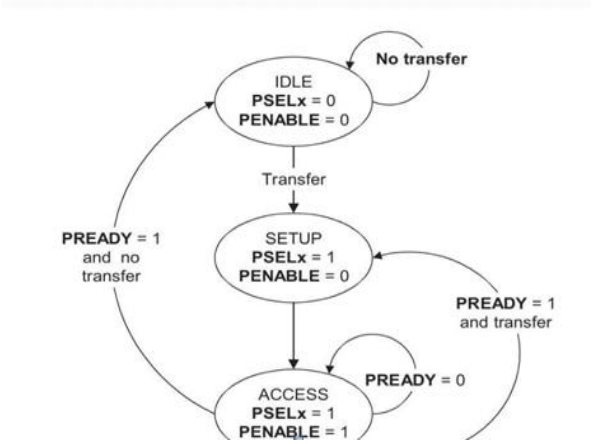

Fig -3: Operating States in APB

APB mainly consists of three operating states for its operation. This is depicted in Fig 3. The operating states are named IDLE, SETUP and ACCESS states.

- Idle is the normal state in case of APB protocol.
- When any transfer is necessary the bus will relocate into the Setup state, where select signal, Pselx, will be made high. The bus will wait in the SETUP state for only one clock cycle and always it will move to the ACCESS states on the next rising edge of the clock cycle to enable write or read transfer.
- ACCESS State will enable signal, PENABLE signal will be asserted in the ACCESS state. The write signal, write data signals, select signal, and address line all these signals must remain stable during the transition from the SETUP to ACCESS state during the operation. If the PREADY signal is held LOW by the slave then the bus must remain in the ACCESS state until PREADY is driven HIGH by the slave. Then the ACCESS state will be exited and the bus returns to the IDLE state if no more transfers are required and will continue from the same clock cycle when required.

3. TRANSFERS IN APB

APB includes various write and read operations which are mentioned with both wait and without wait states. In the newly revised specification of APB Protocol it includes both PPROT and PSTRB signals.

3.1 Write Transfer without Wait States

The first cycle the write transfer will be initiated when the address bus, write data signal, write signal and the select signal all change after the rising edge of the clock as shown in Fig 4. After SETUP State in the following clock cycle enable signal will be asserted to 1. This will start the ACCESS State. The address signal, the data signal and the control signal will be active until the completion of ACCESS State and still the transfer is complete. PENABLE signal will be disabled at the end of the transfer when the whole transfer will be complete. The select signal PSELx will also be disabled until another transfer is required.

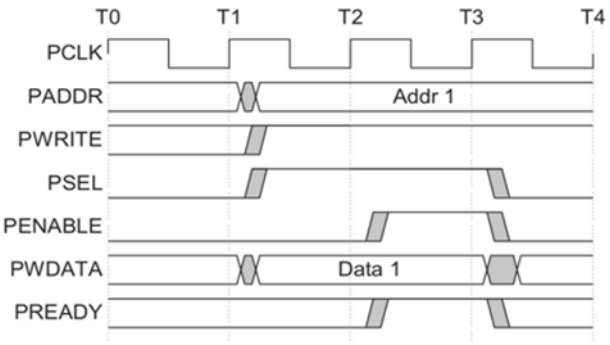

Fig -4: Write Transfers without Wait States

3.2 Write Transfers with Wait States

Fig 5 shows how PREADY signal can be delayed during a transfer and how we can use PENABLE Signal to extend the transfer for one more clock cycle by keeping PENABLE Signal high in its state.

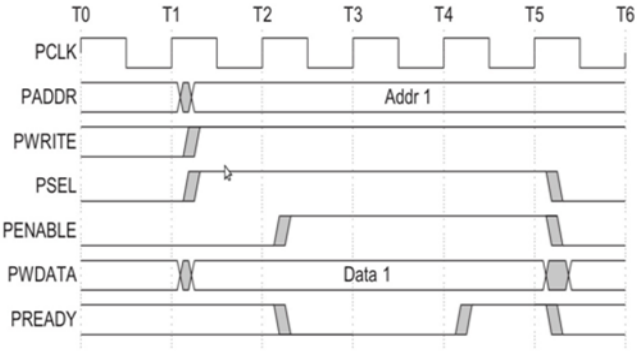

Fig-5: Write Transfers with Wait States

3.3 Read Transfer without Wait States

Fig-6 shows a read transfer without wait states. The read transfer will take place whenever PWRITE Signal will be asserted low. All the other conditions remain same as that of Write Transfer. During the setup state PWRITE Signal will be made low. PSELx signal will be high whereas PENABLE Signal will be low. It enters the ACCESS Phase in the next clock cycle when PENABLE Signal will be asserted high and also the PREADY Signal is high.

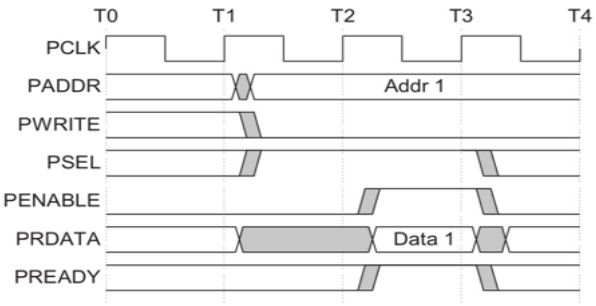

Fig-6: Read transfer without wait states

3.4 Read Transfer with Wait States

Fig-7 shows how the read transfer with wait states. The operation is similar to that WRITE operation with wait states.

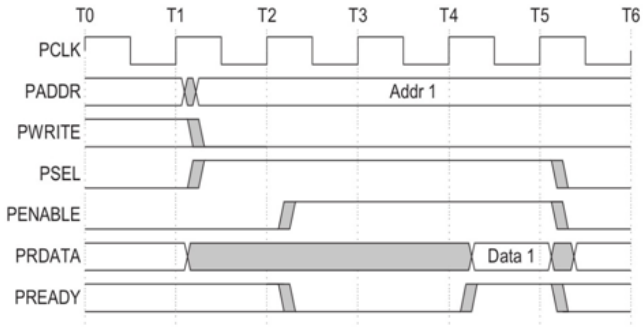

Fig-7: Read Transfers with Wait States

3.5 Write Strobes

The write strobe signals, represented as PSTRB, enable small amount of data transfer on the write data bus. Each write Strobe signal will correspond to one byte of the write data on the write bus. When it is enabled, a write strobe signal will indicate that the corresponding byte of the write data bus data is containing valid information. There is write strobe signal for each of eight bits of the write data bus, so PSTRB[n] will corresponds to PWDATA[(8n + 7):(8n)].

3.6 Error Response

PSLVERR is used to indicate an error condition on an APB transfer. Error conditions can occur in both write and read transactions. PSLVERR will be considered valid during the last cycle of an APB transfer, that is in the access phase when PSEL, PENABLE, and PREADY signals will be HIGH. PSALVERR can be used in order to determine any of the invalid data being written or being read during the transfers.

3.7 Protection Unit Support

To support any complex system designs, it is often necessary to provide protection against any illegal transactions. In case of APB interface, protection is provided by the PPROT[2:0] signals. It is a three bit signal. There are three levels of protection access which are explained below.

Normal or privileged, PPROT[0]

• LOW Signal will indicate a normal access

• HIGH Signal will indicate a privileged access.

This is used by some of the masters inorder to indicate their processing mode.

Secure or non-secure, PPROT[1]

• LOW signal will indicate a secure access

• HIGH signal will indicate a non-secure access.

This is used in systems where it requires a higher degree of differentiation between processing modes.

Data or Instruction, PPROT[2]

• LOW signal will indicate a data access

• HIGH signal will indicate an instruction access.

This signal will differentiate between data and instruction.

4. SYSTEM VERILOG AND UVM

The Protocol is being verified in both system Verilog and UVM

4.1 System Verilog

It is an expansion of the popular Verilog language, conveying a more elevated amount of abstraction to design and verification. System Verilog gives a complete verification environment, utilizing Constraint Random Generation, Assertion-Based Verification and Coverage Driven Verification. These methods improve drastically the verification process. Now that SV incorporates OOPs, interposes communication, and dynamic threads, it can be used for system design. The System Verilog verification methodology depends on 3 building blocks, shown in figure, which can be utilized independently, or all together.

•Stimuli: the design utilizing consequently generated random scenarios - constrained-random (CR) test generation.

•Check: the behavior of the design (assertions) and the yield data (scoreboard) to confirm the rightness of activity.

•Measure: the functional coverage metrics to give feedback to the generation and analyze the advancement of verification.

Regularly in the functional verification, the disappointment mode analysis and fault robustness are performed for the design. For overcoming the slag in the functional verification process, thus the reusability of a large number of IP cores in the unpredictable design makes the functional verification process so significant (for example as it includes 70% percent of the time range as compared to the 30% of the time span for the design process) In order to overcome this difficulty and large time span the verification check engineers proposed a methodology for confirming the usefulness of the chip utilizing inbuilt verification environment called as Verification IP (VIP).

4.2 Universal Verification Methodology

Almost 70% of the efforts go for design efforts that we require for verification because we have to find the bugs in the design and rectify it to avoid product failure. There are many Verification methodologies to test any protocol or IP core but Universal Verification Methodology is best among all. The advantages provided by UVM are listed below.

- UVM does not interface with DUT so it provides high reusability.
- To reduce the time spent in verifying the design. UVM has constraints based test benches.
- Verification time is less and maximum coverage using UVM.
- Configurable, flexible and reusable test benches which can verify specific bit lines or signal

UVM allows us to achieve coverage-driven verification (CDV). CDV is combination of automatic test generation, self-checking test benches, and coverage metrics to significant

5. VERIFICATION ENVIRONMENTS IN DETAIL

System Verilog language is an extension of Verilog language and is getting more broadly adopted in the industry. It is an IEEE standard and supports a lot of improvements from Verilog for design constructs. It is an expansion of the popular Verilog language, conveying a more elevated amount of abstraction to design and verification. System Verilog gives a complete verification environment, utilizing Constraint Random Generation, Assertion-Based Verification and Coverage Driven Verification. These methods improve drastically the verification process. Now that SV incorporates OOPs, interposes communication, and dynamic threads, it can be used for system design. Fig-8 shows the System Verilog Environment. The operations of each of the components in System Verilog are as follows:

- Generator generates the transactions [Write/Read packets] and sends them to driver.
- For every interface [Write & Read], drivers and monitors are created.
- Driver will collect the transactions and drives them as binary values to DUT as per the DUT interface protocol [Write/Read protocol].
- Monitors will collect the binary values from the DUT pins and convert them back into transactions [Write/Read packets].
- All the monitors are connected with the scoreboard which compares the DUT outputs with the expected values.
- Scoreboard has reference model and comparison logic.
- Reference model will produce the value expected and the logic for comparison compares the DUT outputs with the values expected.
- Monitors post the transactions into the scoreboard.
- Upon successful comparison of the scoreboard even the functional coverage analysis can be carried out.
- Verification environment creates separate objects of all the transactions, generator, driver, monitor and scoreboard.

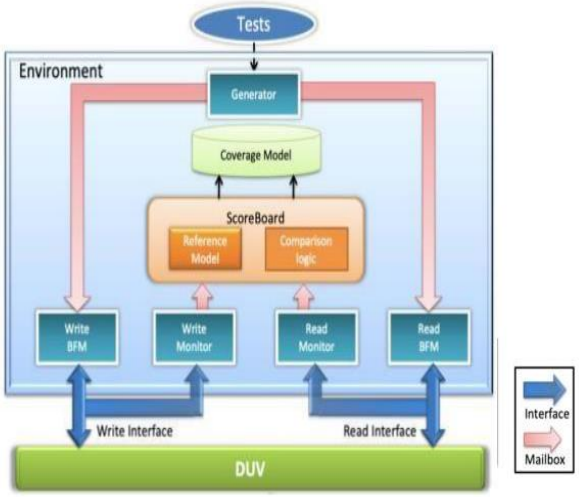

Fig -8: System Verilog Environment

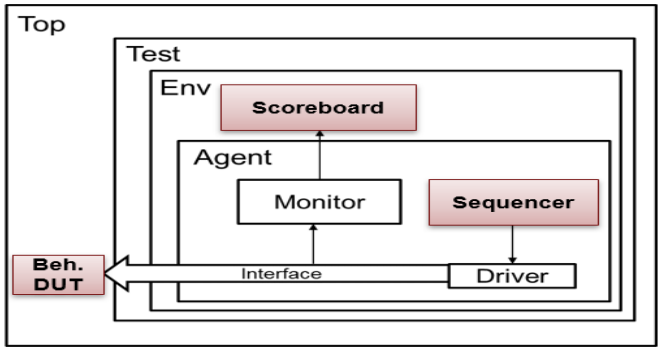

Fig-9: Universal Verification Methodology

The Universal Verification Methodology (UVM) is a normalized approach for checking coordinated circuit plans. UVM is gotten chiefly from the Open Verification Methodology which was, to a huge part, in light of the Reuse Methodology for the Verification Language created by Verisity Design in 2001. The UVM class library carries a lot of computerization to the SystemVerilog language, for example, arrangements and information robotization highlights and so on, and dissimilar to the past procedures grew freely by the test system merchants, is an Accellera standard with help from different sellers: Aldec, Cadence, Mentor Graphics, Synopsys, Xilinx Simulator(XSIM).

Several Components in the UVM Environment are explained below:

Factory:
We use Factory when we want to intantiate any other Object.
Sequencer: It is having 3 main functions:

- It will put the DUT into state of initialisation
- The DUT and the verification environment will be configured together.
- The entire DUT will be initiated.

Initialisation:
Here the conditions will be initialised before beginning the simulation. It involves the following:

- It involves initialising and loading the memory.
- The setting of the registers which cannot be altered during the time of simulation will be taken care of.
- Settings of the verification environment which cannot be done during simulation will be taken care.

Score Board:

Implementation of Scoreboard can be done in many ways. The score board will take the inputs and outputs everything from DUT. This will also decide relationship of the input and output and check if the design is acting accordingly.

Agent:

In any DUT there may be many number of interfaces. These interfaces will have unique objects which are associated with them. All these various objects will be initialised with the help of the agent.

Driver:

Driver will take responsibility of all the sequence items that comes under it and will also give appropriate stimulus.

6 SIMULATION OUTCOMES

The following Fig 10 shows write and read transfer outcome without wait states for 5 write and read transfers without wait states.

Fig-10: Write and Read Transfers without Wait States

The following Fig-11 shows write and read transfers with wait states for 10 write and read transfers. It also shows error response for a particular transfer along with PPROT and PSTRB signals which is being given for each states.

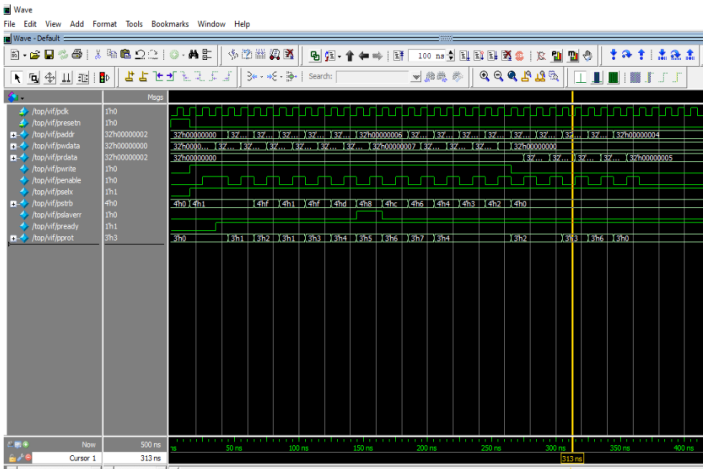

Fig -11: Write and Read Transfers with Wait States

Fig-12 shows the score board analysis of the write and read operations as displayed in the transcript window.

```
# stroing write transactioon paddr=        10 pwdata=        11
# t=                 265
# paddr=         10
# pwdata=         11
# pwrite=1
# pselx=0
# penable=0
# pstrb= 2
# pprot=4
# prdata=          0
# pslaverr=0
# pready=1
# SCOREBOARD PASS....exp=          1  act=          1
# t=                 285
# paddr=          0
# pwdata=4034530006
# pwrite=0
# pselx=0
# penable=0
# pstrb= 0
# pprot=1
# prdata=          1
# pslaverr=0
# pready=1
# SCOREBOARD PASS....exp=          2  act=          2
# t=                 305
# paddr=          1
# pwdata= 194348458
# pwrite=0
# pselx=0
# penable=0
# pstrb= 0
# pprot=6
```

Fig-12: Transcript Window Output for Score Board Analysis

Functional Coverage tells if all the functionalities which is given in the specification are being verified or not. 100% Function Coverage is achieved in the verification process and is displayed in Fig 13.

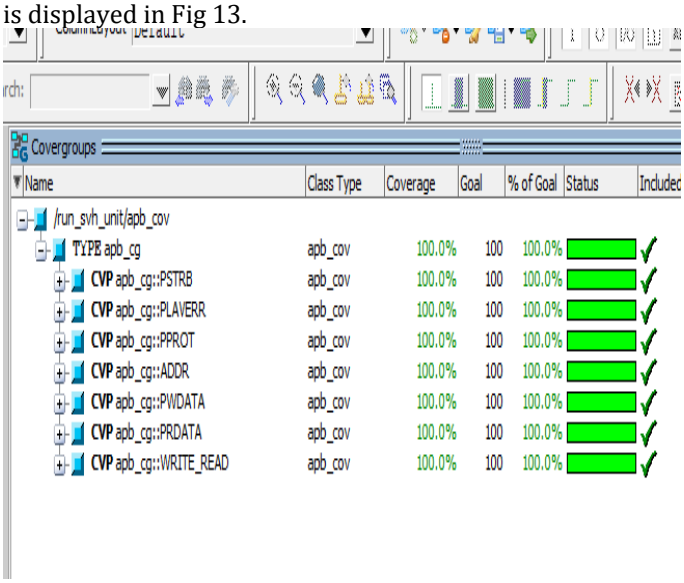

Fig-13: Functional Coverage Analysis

Assertion is a procedure where a property is being asserted and is being checked if the DUT is working according to the asserted property. 100% assertion is being achieved as shown in Fig 14.

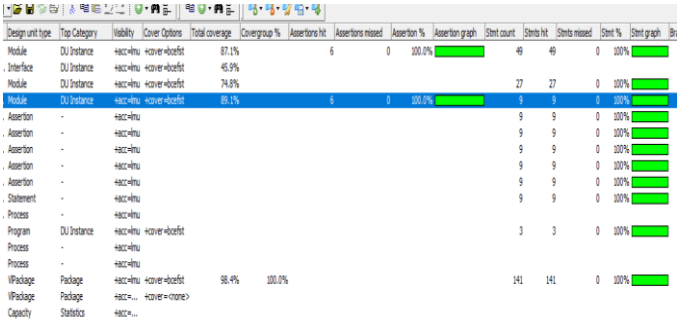

Fig -14: Output of Assertion

Code Coverage Analysis tells if all the lines which is written in the code of design or the verification is not being left unexecuted. Fig 15 below shows that 100% Code Coverage is being achieved.

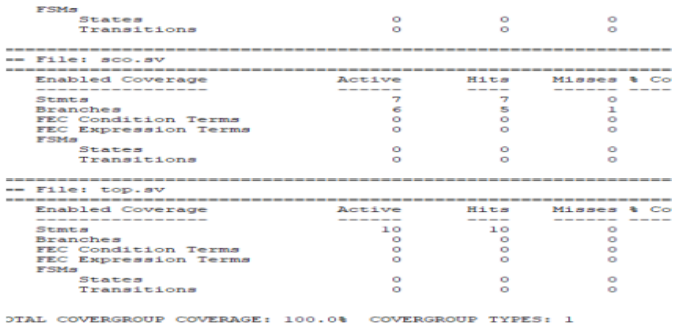

Fig-15: 100% Code Coverage

APB Protocol supports only a single master but multiple slaves can be connected to the same single master. Verification was carried out where a single master was connected to two slaves and the write and the read operations for the same were verified. Fig-16 shows the outcomes achieved.

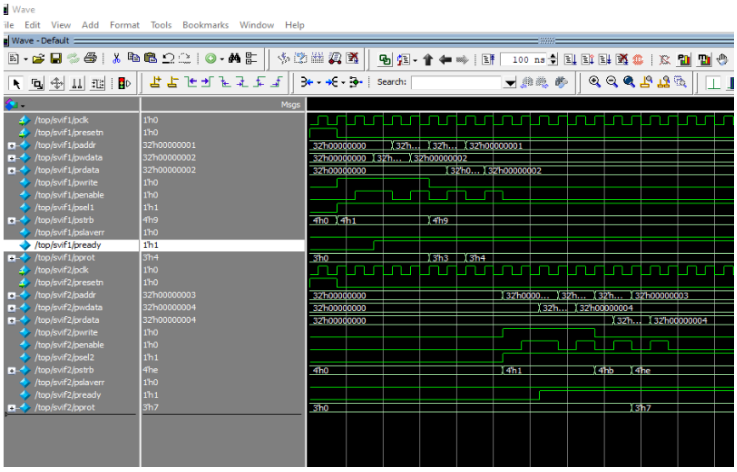

Fig-16: Single Master Multiple Slave Write and Read Transfers.

Verification was also carried in UVM and the same protocol Register Transfer Level code was verified. Fig 17 shows the write and read transfers achieved with wait states.

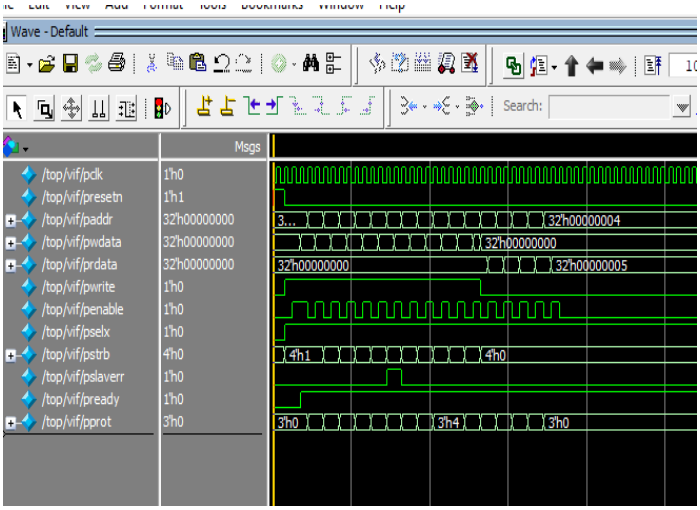

Fig -17: Write and Read Transfers

Scoreboard analysis was also carried out in UVM. Fig 18 shows the transcript window output in UVM for the score board analysis. And in all the test cases the scoreboard was pass.

```
req                             apb_tx    -     @781
  paddr                         integral  32    'h0
  pwdata                        integral  32    'h1
  pwrite                        integral  1     'h1
  pselx                         integral  1     'h0
  penable                       integral  1     'h0
  pstrb                         integral  4     'h1
  pprot                         integral  3     'h0
  prdata                        integral  32    'h0
  pslaverr                      integral  1     'h0
  pready                        integral  1     'h0
  begin_time                    time      64    15
  depth                         int       32    'd2
  parent sequence (name)        string    3     seq
  parent sequence (full name)   string    30    uvm_test_top.env.agent.sqr.seq
  sequencer                     string    26    uvm_test_top.env.agent.sqr
--------------------------------------------------------------------------------
SCOREBOARD PASS....exp.pwdata=        1  act.pwdata=        1   exp.prdata=        0  act.prdata=        0
assertion pass..paddr=        1 pwdata=        1..              45
SCOREBOARD PASS....exp.pwdata=        2  act.pwdata=        2   exp.prdata=        0  act.prdata=        0
--------------------------------------------------------------------------------
Name                            Type      Size  Value
--------------------------------------------------------------------------------
req                             apb_tx    -     @846
  paddr                         integral  32    'h1
  pwdata                        integral  32    'h2
  pwrite                        integral  1     'h1
  pselx                         integral  1     'h0
  penable                       integral  1     'h1
  pstrb                         integral  4     'h1
  pprot                         integral  3     'h1
  prdata                        integral  32    'h0
  pslaverr                      integral  1     'h0
  pready                        integral  1     'h1
  begin_time                    time      64    45
  depth                         int       32    'd2
  parent sequence (name)        string    3     seq
  parent sequence (full name)   string    30    uvm_test_top.env.agent.sqr.seq
  sequencer                     string    26    uvm_test_top.env.agent.sqr
--------------------------------------------------------------------------------
SCOREBOARD PASS....exp.pwdata=        2  act.pwdata=        2   exp.prdata=        0  act.prdata=        0
assertion pass..paddr=        2 pwdata=        2..              65
SCOREBOARD PASS....exp.pwdata=        3  act.pwdata=        3   exp.prdata=        0  act.prdata=        0
```

Fig-18: Score Board analysis in UVM

Fig 19 shows UVM output of assertion where 100% assertion was achieved.

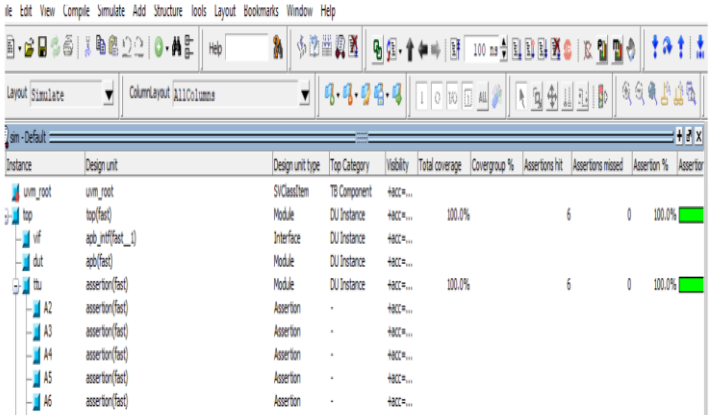

Fig -19: Assertion output of UVM

Functional Coverage was also carried out in UVM and all the functionalities given in the specifications were covered. Fig 20 shows the output of 100% Functional Coverage.

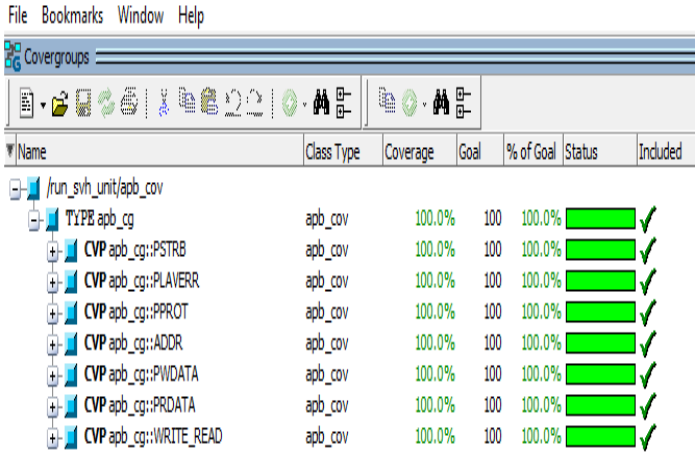

Fig-20: Output of Functional Coverage

7. CONCLUSIONS

An RTL Code of APB Protocol was chosen and was being verified both in System Verilog ad UVM. The verification is being done to achieve the waveform of write and read transfers without wait states and also with wait states. Scoreboard analysis was carried out to achieve passing of score board in all the test cases taken. 100% Functional Coverage, 100% Assertion and 100% code coverage was also achieved.

DUT was also being checked for the operation where a single master is being connected to multiple slaves. The wave form achieved here is also accurate and meets all the specification requirements.

Code is also verified using Universal Verification Methodology. Score board analysis is carried out where all

the test cases are passing with 100% Functional coverage and 100% Assertion are also being achieved.

The whole design is being checked under all the possible scenarios with respect to its specification to verify the exact working and functionality of the design.

REFERENCES

[1] Lingling Chai, Zheng Xie, Xin'an Wang "A Verification Methodology for Reusable Test Cases and Coverage Based on System Verilog" IEEE International Conference on electron devices and solid state circuits, IEEE, 2014, pp. 1-4.
[2] Han Ke, Deng Zhongliang, Shu Qiong "Verification of AMBA Bus Model Using SystemVerilog" The Eighth International Conference on Electronic Measurement and Instruments, 2007, pp. 776-780.
[3] Ni, Wei, and Jichun Zhang. "Research of Reusability Based on UVM Verification" IEEE 11th International Conference on ASIC (ASICON), IEEE, 2015, pp. 1-4.
[4] Khalifa, Khaled, and Khaled Salah "An RTL Power Optimization Technique Based on System Verilog Assertions" IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, 2016, pp. 1-4
[5] Kommirisetti Bheema Raju., Bala Krishna Konda "Design And Verification of AMBA APB Protocol" Int. Journal of Engineering Research and Application, ISSN: 2248-9622, Vol. 7, Issue 1, (Part -1) January 2017, pp. 87-90
[6] Swetha Reddy., Punna Soujanya., D. Kanthi Sudha "ASIC Design and Verification of AMBA APB Protocol Using UVM", International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9 Issue-9, July 2020, pp. 636-640.
[7] Shilpa Rao and Arati S. Phadke "Implementation of AMBA Compliant Memory Controller on a FPGA", International Journal of Emerging Trends in Electrical and Electronics (IJETEE) Vol. 2, Issue. 2, April-2013, pp. 21-23 [8] Hitanshu Saluja, Dr. Naresh Grover "Multiple Master Communication in AHB IP Using Arbiter", I.J. Engineering and Manufacturing, 2020, Published Online February 2020 in MECS (http://www.mecs-press.net) pp. 30-39
[9] Samir Palnitkar, "Verilog HDL: A guide to Digital Design and Synthesis", (2nd Edition), Pearson, 200