APB Slave Design and Verification

The Advanced Peripheral Bus (APB) is a protocol that belongs one of the Advanced Microcontroller Bus Architecture (AMBA) protocol family that is arranged for reduced power consumption and interface complexity. It doesn't include pipelined bus interface and can interface with AHB and AXI busses. In the APB protocol, each write and read transfer takes at least 2 clock cycles. The interfacing and overall signals belonging to the APB Protocol is seen in figure 1.

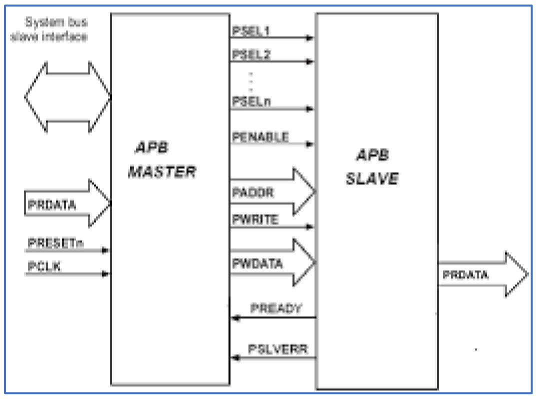


Figure 1. APB Bus Master and Slave Architecture

The descriptions for the corresponding signals can be observed at table 1.

Table 1. APB signal descriptions

Signal	Source	Description
PCLK	Clock source	The rising edge of PCL times all transfers on the APB.
PRESETn	System bus equivalent	The APB reset signal is active LOW. This signal is normally connected directly to the system bus reset signal.
PADDR	APB bridge	This is the APB address bus. It can be up to 32 bits wide and is driven by the peripheral bus bridge unit.
PSELx	APB bridge	The APB bridge unit generates this signal to each peripheral bus slave. It indicates that the slave device is selected and that a data transfer is required. There is a PSELx signal for each slave.
PENABLE	APB bridge	This signal indicates the second and subsequent cycles of an APB transfer.
PWRITE	APB bridge	This signal indicates an APB write access when HIGH and an APB read access when LOW.
PWDATA	APB bridge	This bus is driven by the peripheral bus bridge unit during write cycles when PWRITE is HIGH. This bus can be up to 32 bits wide.
PREADY	Slave interface	The slave uses this signal to extend an APB transfer.
PRDATA	Slave interface	The selected slave drives this bus during read cycles when PWRITE is LOW. This bus can be up to 32-bits wide.
PSLVERR	Slave interface	This signal indicates a transfer failure.

1. Operating States of APB Protocol

There are 3 main states for the APB Protocol which are IDLE, SETUP and ACCESS. **IDLE** is the default state where no transfer happens. When a transfer is required, bus moves to the **SETUP** state where the PSEL signal asserted. The bus always moves to the **ACCESS** state in the next rising edge of the clock. In the **ACCESS** state where the PENABLE signal is asserted, read and write transactions occur. The accomplishment of the write and read transfers is controlled with the PREADY signal.

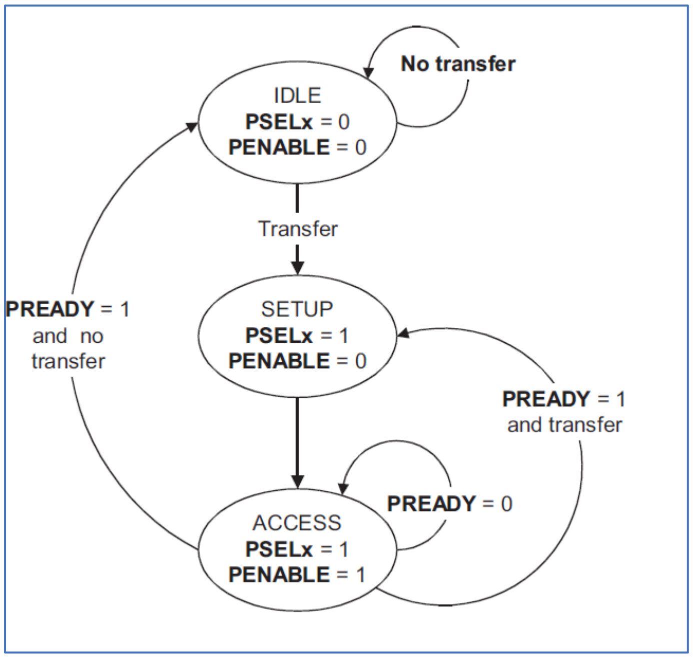


Figure 2. State Diagram of APB Protocol

a. Write Transfer

The write cycle begins with the rising edge of the clock at T1 by registering PADDR, PWDATA, PWRITE, and PSEL which is called the SETUP state. In the next rising edge where the bus is in the ACCESS state, the PENABLE and PREADY are registered. When PENABLE is asserted at T3, it means that there is a transfer pending and if the transfer happens, PREADY implies that the slave can complete the transfer at the next rising edge of the clock.[3]

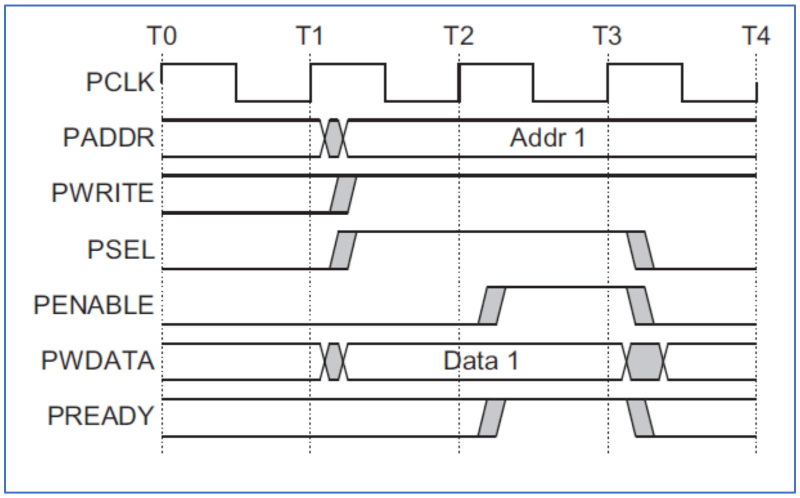


Figure 3. Write transfer with no wait states

The read cycle begins with the rising edge of the clock at T1 similar to the write cycle. The PADDR, PWRITE, and PSEL are registered and this state is called the SETUP state. In the next rising edge where the bus is in the ACCESS state, the PENABLE, PRDATA, and PREADY are registered. When PENABLE is asserted at T3, it means that there is a transfer pending and if the transfer happens, PREADY implies that the slave can complete the transfer at the next rising edge of the clock and the data is read before the end of the transaction, controlled by the slave.[3]

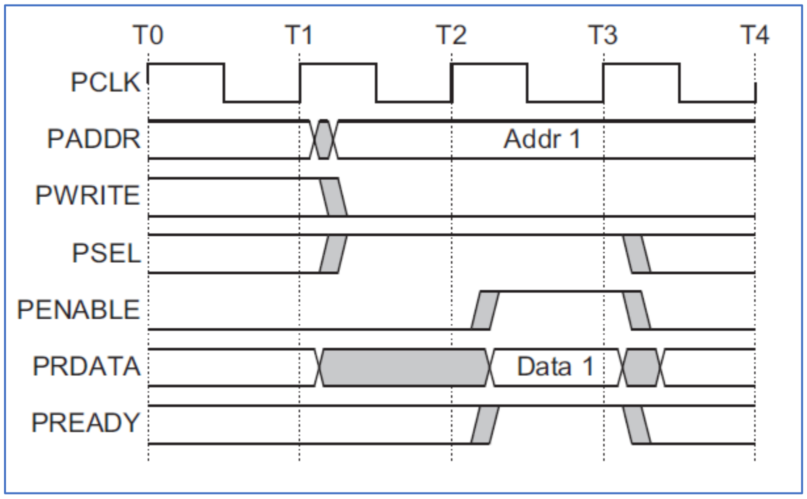


Figure 4. Read transfer with no wait states

Design of an APB Slave Protocol is realized based on these transactions for the read and write cycles and state diagram of the proposed communication protocol. The design file is created using VHDL language and simulation files are created using SystemVerilog language. Further details about the design and testbench files will be given in the coming sections.

2. Verification of the APB Slave Protocol using SystemVerilog Verification

To verify the overall system, SystemVerilog verification methodology will be used. According to this method there are several blocks that arranges the flow of the verification as seen in figure 5. This environment checks whether the functional flow of the main design code under Design Under Test (DUT) is working properly. This environment in general verifies the design by generating stimulus and controlling the design outputs by monitoring the outputs.

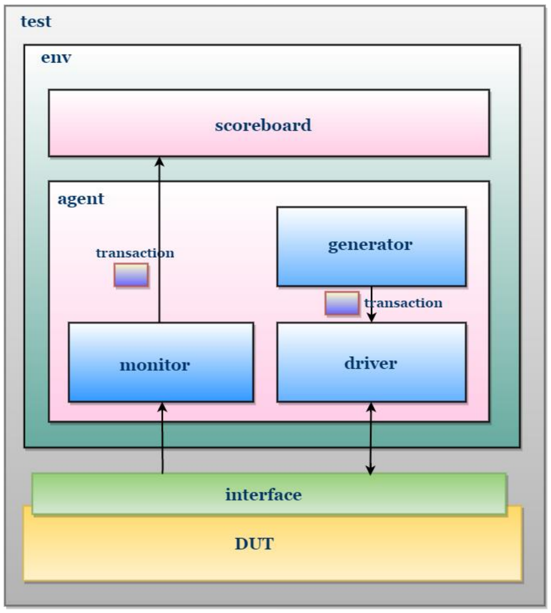


Figure 5. SystemVerilog Verification Flowchart

All these blocks have special duties to verify the system and the descriptions belonging to each block is represented in table 2.[2]

Table 2. SV Verification Testbench Components

Name	**Type**	**Description**
transaction	*class*	Includes packets or frames that will be transferred through the tb environment which generally includes randomized values that will be covered in the further stages.
generator	*class*	Generates random input stimulus that will be monitored by DUT sends them to Driver.
driver	*class*	Gathers the generated stimulus with the help of mailbox and drives them.
monitor	*class*	Monitors the pin level activity on design input and output ports and then send them to scoreboard.
agent	*class*	Contains the class's (generator, driver, and monitor) specific to an interface or protocol.
scoreboard	*class*	Checks the output data items with the expected values.

environment	*class*	Contains all the components including transaction, generator, driver, monitor, and scoreboard.
test	*program*	Configures the tb and initiate the tb components
interface	*interface*	Includes signals that will be driven or monitored.
tbench_top	*class*	Topmost file that connects design with the testbench. It covers all the environment mentioned above

3. Simulation Results of the APB Slave Protocol using QuestaSim Software

After arranging all the blocks desired to apply SV Verification, the system is simulated using QuestaSim Software which is used to verify HDL languages. In below, you can see the simulation results for the proposed design. As seen in figure X, at time t=35 ns, a data (pwdata) is written into an address (x'h3) and the same data is read back from the same address at time t=85 ns, specified with prdata.

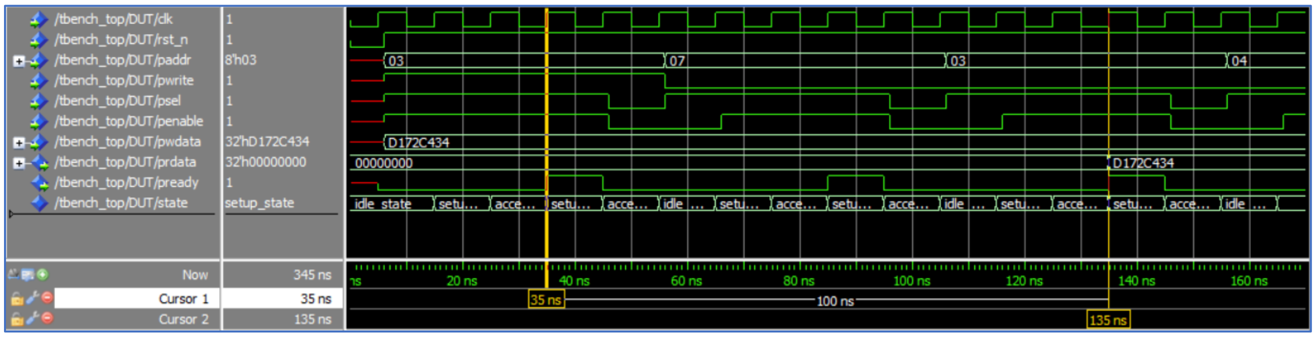
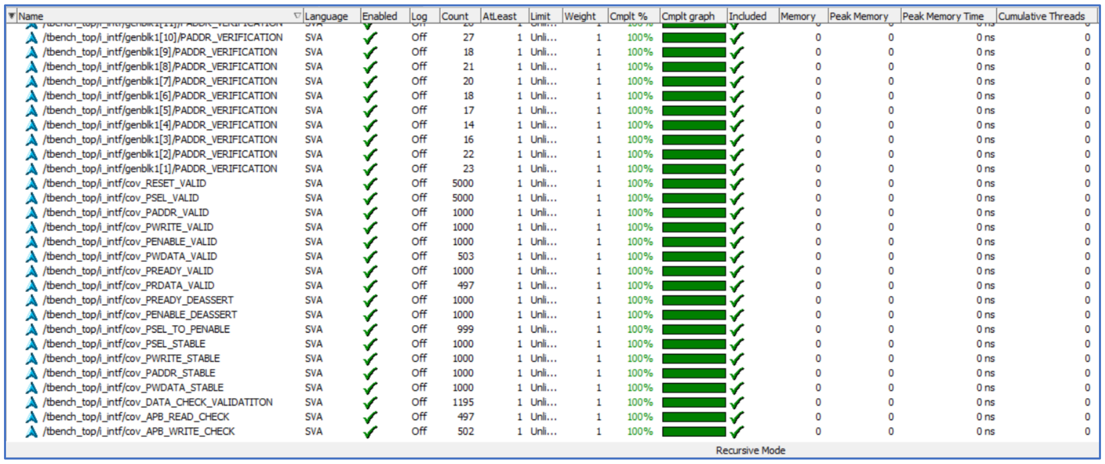


Figure 6.Simulation results of the APB Slave protocol

After simulating the APB Slave protocol, functional coverage and SV Assertions (SVA) are used to check all possible situation for each signal. Functional coverage is used to measure the how much of the scenarios of the design is validated/tested. SVA and coverage are also used to see the behaviour of the design while validating.



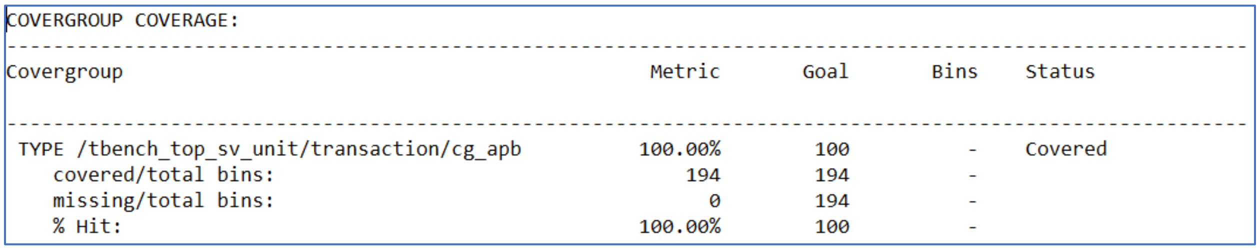
Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/tbench_top/i_intf/genblk1[10]/PADDR_VERIFICATION	SVA	✓	Off	27	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[9]/PADDR_VERIFICATION	SVA	✓	Off	18	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[8]/PADDR_VERIFICATION	SVA	✓	Off	21	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[7]/PADDR_VERIFICATION	SVA	✓	Off	20	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[6]/PADDR_VERIFICATION	SVA	✓	Off	18	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[5]/PADDR_VERIFICATION	SVA	✓	Off	17	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[4]/PADDR_VERIFICATION	SVA	✓	Off	14	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[3]/PADDR_VERIFICATION	SVA	✓	Off	16	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[2]/PADDR_VERIFICATION	SVA	✓	Off	22	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/genblk1[1]/PADDR_VERIFICATION	SVA	✓	Off	23	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_RESET_VALID	SVA	✓	Off	5000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PSEL_VALID	SVA	✓	Off	5000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PADDR_VALID	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PWRITE_VALID	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PENABLE_VALID	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PWDATA_VALID	SVA	✓	Off	503	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PREADY_VALID	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PRDATA_VALID	SVA	✓	Off	497	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PREADY_DEASSERT	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PENABLE_DEASSERT	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PSEL_TO_PENABLE	SVA	✓	Off	999	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PSEL_STABLE	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PWRITE_STABLE	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PADDR_STABLE	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_PWDATA_STABLE	SVA	✓	Off	1000	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_DATA_CHECK_VALIDATITON	SVA	✓	Off	1195	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_APB_READ_CHECK	SVA	✓	Off	497	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/tbench_top/i_intf/cov_APB_WRITE_CHECK	SVA	✓	Off	502	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0

Recursive Mode

Figure 7. SV Cover Directive Results of the APB Slave Protocol

In above, you can see the coverage results of the proposed protocol. Based on this validation method, each state, signals, and the address data relation are observed. For instance, to see check that the written data is found or not in the coming cycles for each address, represented with PADDR_VERIFICATION.

In terms of the functional coverage, an example result is shown in figure X. According to this result, it can be seen that 100% of the total variations that is included inside the functional coverage is covered. In this example, the addr and pwrite is covered individually and then covered using cross coverage.



```
COVERGROUP COVERAGE:
-----------------------------------------------------------------------------------------------------------
Covergroup                                            Metric        Goal       Bins     Status

-----------------------------------------------------------------------------------------------------------
 TYPE /tbench_top_sv_unit/transaction/cg_apb         100.00%        100          -      Covered
    covered/total bins:                                  194        194          -
    missing/total bins:                                    0        194          -
    % Hit:                                           100.00%        100          -
```

After checking all off the possible scenarios, I completed the design and verification of the APB Slave protocol.

References:

[1] *Systemverilog tutorial for Beginners*. Verification Guide. (2020, April 8). Retrieved March 7, 2022, from https://verificationguide.com/systemverilog/systemverilog-tutorial/

[2] *Systemverilog tutorial*. ChipVerify. (n.d.). Retrieved March 7, 2022, from https://www.chipverify.com/systemverilog/systemverilog-tutorial

[3] ARM *"AMBA™ 3 APB Protocol v1.0" ARM IHI 0024B Specification datasheet, Retrieved 2004*