
# AHB2APB Bridge Design Report

BY

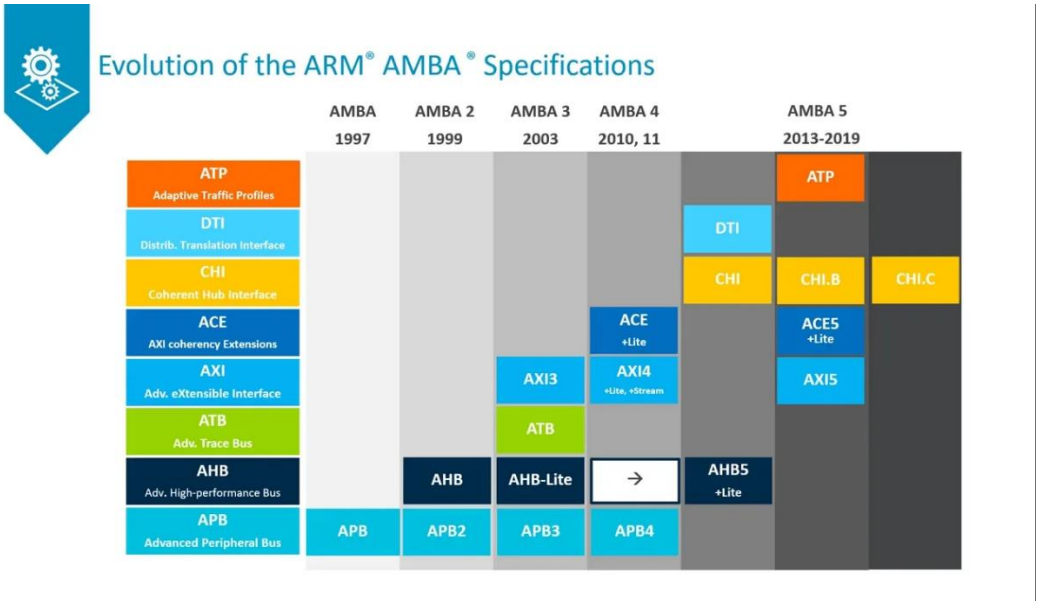
**AKHIL BIJU**

**21BEE1266**

# ARM AMBA:

AMBA (Advanced Microcontroller Bus Architecture) is a freely-available, open standard for the connection and management of functional blocks in a system-on- chip (SoC). It facilitates right-first-time development of multi-processor designs, with large numbers of controllers and peripherals.

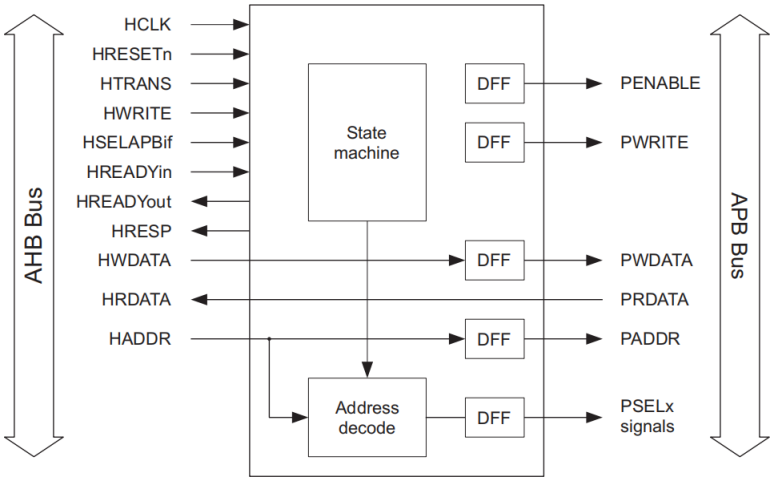
AMBA specifications are royalty-free, platform-independent and can be used with any processor architecture. Due to its widespread adoption, AMBA has a robust ecosystem of partners that ensures compatibility and scalability between IP components from different design teams and vendors.



# AHB to APB Bridge:

The AHB to APB bridge interface is an AHB slave. When accessed (in normal operation or system test) it initiates an access to the APB. APB accesses are of different duration (three HCLK cycles in the EASY for a read, and two cycles for a write). They also have their width fixed to one word, which means it is not possible to write only an 8-bit section of a 32-bit APB register. APB peripherals do not needa PCLK input as the APB access is timed with an enable signal generated by the AHB to APB bridge interface. This makes APB peripherals low power consumption parts, because they are only strobed when accessed.

# Architecture:




The main sections of this module are:

- AHB slave bus interface
- APB transfer state machine, which is independent of the device memory map
- APB output signal generation.

To add new APB peripherals, or alter the system memory map, only the address decodes sections need to be modified.

The base addresses of each of the peripherals (timer, interrupt controller, and remap and pause controller) are defined in the AHB to APB bridge interface, which selects the peripheral according to its base address. The whole APB address range is also defined in the bridge.

These base addresses can be implementation-specific. The peripherals standard specifies only the register offsets (from an unspecified base address), register bit meaning, and minimum supported function

The APB data bus is split into two separate directions:

- read (PRDATA), where data travels from the peripherals to the bridge
- write (PWDATA), where data travels from the bridge to the peripherals.

This simplifies driving the buses because turnaround time between the peripherals and bridge is avoided.

In the default system, because the bridge is the only master on the bus, PWDATA is driven continuously. PRDATA is a multiplexed connection of all peripheral PRDATA outputs on the bus, and is only driven when the slaves are selected by the bridge during APB read transfers.

It is possible to combine these two buses into a single bidirectional bus, but precautions must be taken to ensure that there is no bus clash between the bridge and the peripherals.

# Signal Description:

| Signals | Type | Direction | Description |
|---|---|---|---|
| **HCLK** | Bus clock | Input | This clock times all bus transfers. |
| **HRESETn** | Reset | Input | The bus reset signal is active LOW, and is used to reset the system and the bus. |
| **HADDR[31:0]** | Address bus | Input | The 32-bit system address bus. |
| **HTRANS[1:0]** | Transfer type | Input | This indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY. |
| **HWRITE** | Transfer direction | Input | When HIGH this signal indicates a write transfer, and when LOW, a read transfer. |
| **HWDATA[31:0]** | Write data bus | Input | The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation. |
| **HSELAPBif** | Slave select | Input | Each APB slave has its own slave select signal, and this signal indicates that the current transfer is intended for the selected slave. This signal is a combinatorial decode of the address bus. |
| **HRDATA[31:0]** | Read data bus | Output | The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation. |

| | | | |
|---|---|---|---|
| **HREADYin HREADYout** | Transfer done | Input/output | When HIGH the HREADY signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer. |
| **HRESP[1:0]** | Transfer response | Output | The transfer response provides additional information on the status of a transfer. This module will always generate the OKAY response. |
| **PRDATA[31:0]** | Peripheral read data bus | Input | The peripheral read data bus is driven by the selected peripheral bus slave during read cycles (when PWRITE is LOW). |
| **PWDATA[31:0]** | Peripheral write data bus | Output | The peripheral write data bus is continuously driven by this module, changing during write cycles (when PWRITE is HIGH). |
| **PENABLE** | Peripheral enable | Output | This enable signal is used to time all accesses on the peripheral bus. PENABLE goes HIGH on the second clock rising edge of the transfer, and LOW on the third (last) rising clock edge of the transfer. |
| **PSELx** | Peripheral slave select | Output | There is one of these signals for each APB peripheral present in the system. The signal indicates that the slave device is selected, and that a data transfer is required. It has the same timing as the peripheral address bus. It becomes HIGH at the same time as PADDR, but will be set LOW at the end of the transfer. |

| | | | |
|---|---|---|---|
| **PADDR[31:0]** | Peripheral address bus | Output | This is the APB address bus, which may be up to 32 bits wide and is used by individual peripherals for decoding register accesses to that peripheral. The address becomes valid after the first rising edge of the clock at the start of the transfer. If there is a following APB transfer, then the address will change to the new value, otherwise it will hold its current value until the start of the next APB transfer. |
| **PWRITE** | Peripheral transfer direction | Output | This signal indicates a write to a peripheral when HIGH, and a read from a peripheralwhen LOW.<br>It has the same timing as the peripheral address bus. |

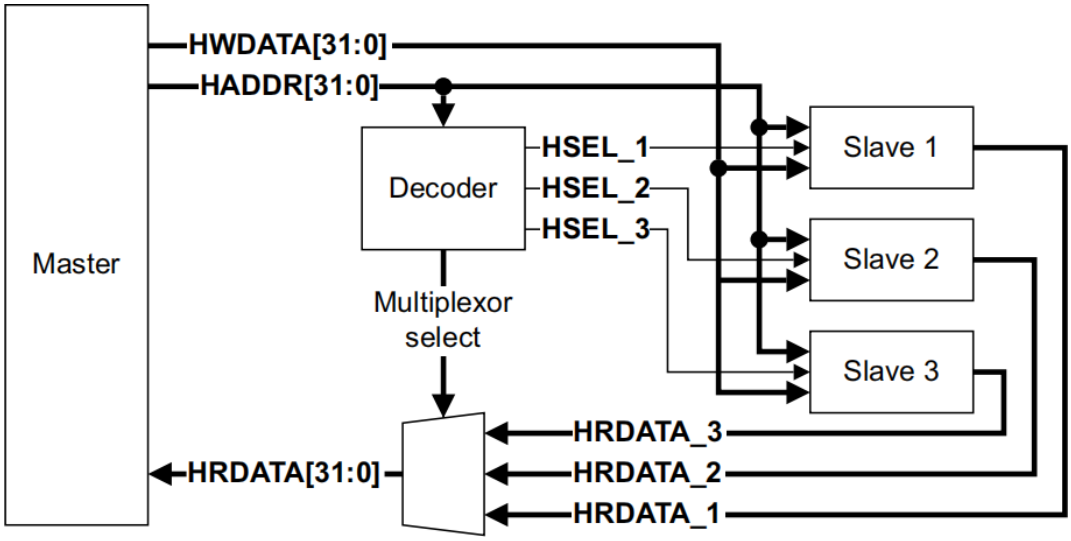
# AHB (AMBA High-performance Bus):

AMBA AHB is a bus interface suitable for high-performance synthesizable designs. It defines the interface between components, such as masters, interconnects, and slaves.

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

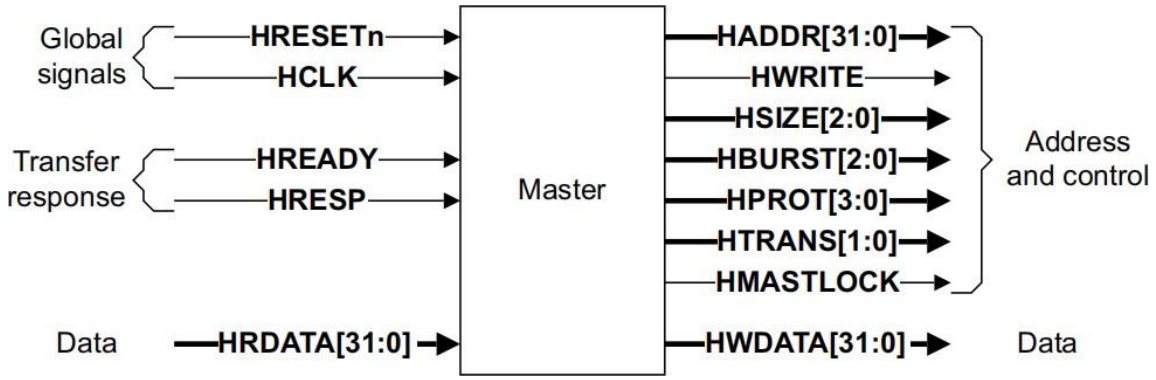
- Burst transfers.
- Single clock-edge operation.
- Non-tristate implementation.
- Wide data bus configurations, 64, 128, 256, 512, and 1024 bits.

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 the higher performance AHB and APB is done using an AHB slave, known as an APB bridge.



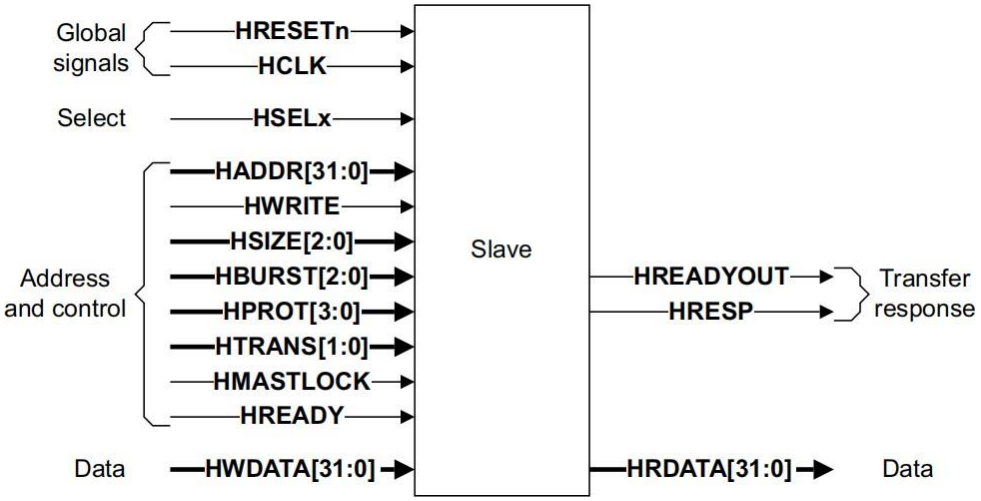
This figure shows a single master AHB system design with the AHB master and three AHB slaves. The bus interconnect logic consists of one address decoder anda slave-to-master multiplexor. The decoder monitors the address from the master so that the appropriate slave is selected and the multiplexor routes the corresponding slave output data back to the master. AHB also supports multi-master designs by the use of an interconnect component that provides arbitration and routing signals from different masters to the appropriate slaves.

## AHB Master Interface




A master provides address and control information to initiate read and write operations. Above figure shows a master interface.

# AHB Slave Interface:




A slave responds to transfers initiated by masters in the system. The slave uses the HSELx select signal from the decoder to control when it responds to a bus transfer.

The slave signals back to the master:

- The completion or extension of the bus transfer.
- The success or failure of the bus transfer. Above figure shows a slave interface.

# APB (Advanced Peripheral Bus):

The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) protocol family. It defines a low-cost interface that is optimized for minimal power consumption and reduced interface complexity.

The APB protocol is not pipelined, use it to connect to low-bandwidth peripheralsthat do not require the high performance of the AXI protocol.

The APB protocol relates a signal transition to the rising edge of the clock, to simplify the integration of APB peripherals into any design flow. Every transfer takes at least two cycles.

The APB can interface with:

- AMBA Advanced High-performance Bus (AHB)
- AMBA Advanced High-performance Bus Lite (AHB-Lite)
- AMBA Advanced Extensible Interface (AXI)
- AMBA Advanced Extensible Interface Lite (AXI4-Lite)

You can use it to access the programmable control registers of peripheral devices

## AHB slave bus interface:

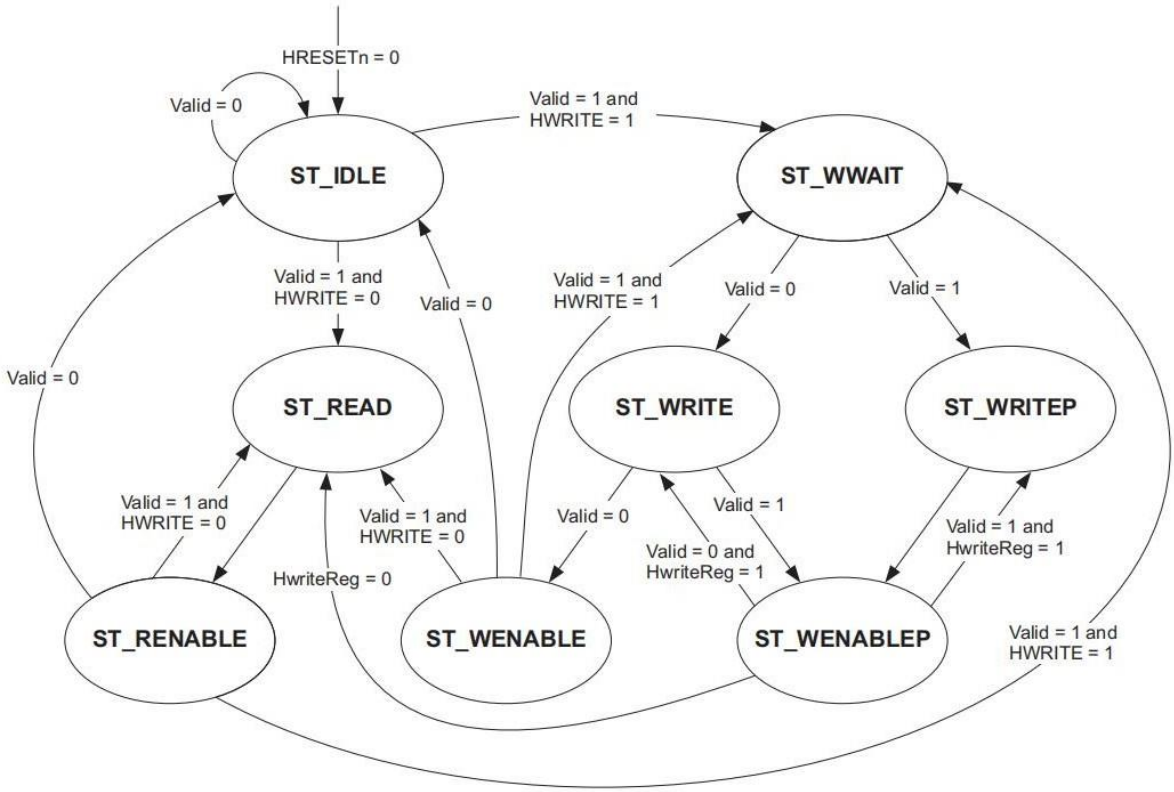
This module uses the standard AHB slave bus interface, which comprises:

• The valid transfer detection logic which is used to determine when a valid transferis accessing the slave.

• The address and control registers, which are used to store the information from theaddress phase of the transfer for use in the data phase.

Due to the different AHB to APB timing of read and write transfers, either the current or the previous address input value is needed to correctly generate the APB transfer. A multiplexor is therefore used to select between the current address input or the registered address, for read and write transfers respectively.

# APB transfer state machine:




The transfer state machine is used to control the application of APB transfers based on the AHB inputs. The state diagram in above figure shows the operation of the statemachine, which is controlled by its current state and the AHB slave interface.

**The individual states of the state machine operation are described in the following sections:**

## ST_IDLE :-

During this state the APB buses and PWRITE are driven with the last values theyhad and PSEL and PENABLE lines are driven LOW.

The ST_IDLE state is entered from:

• reset, when the system is initialized

• ST_RENABLE, ST_WENABLE, or ST_IDLE, when there are no peripheral transfers to perform.

The next state is:

• ST_READ, for a read transfer, when the AHB contains a valid APB read transfer

• ST_WWAIT, for a write transfer, when the AHB contains a valid APB write transfer.

## ST_READ :-

During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven LOW. A wait state is always inserted to ensure that the data phase of the current AHB transfer does not complete until the APB read data has been driven onto HRDATA.

The ST_READ state is entered from ST_IDLE, ST_RENABLE, ST_WENABLE,or ST_WENABLEP during a valid read transfer.

## ST_WWAIT :-

This state is needed due to the pipelined structure of AHB transfers, to allow the AHB side of the write transfer to complete so that the write data becomes available on HWDATA. The APB write transfer is then started in the next clock cycle.

The ST_WWAIT state is entered from ST_IDLE, ST_RENABLE, or ST_WENABLE, during a valid write transfer.

## ST_WRITE :-

During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven HIGH.

A wait state is not inserted, as a single write transfer can complete without affecting the AHB.

The ST_WRITE state is entered from:

• ST_WWAIT, when there are no further peripheral transfers to perform

• ST_WENABLEP, when the currently pending peripheral transfer is a write, and there are no further transfers to perform.

The next state is:

• ST_WENABLE, when there are no further peripheral transfers to perform

• ST_WENABLEP, when there is one further peripheral write transfer to perform.

## ST_WRITEP :-

During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven HIGH. A wait state is always inserted,as there must only ever be one pending transfer between the currently performed APB transfer and the currently driven AHB transfer.

The ST_WRITEP state is entered from:

- ST_WWAIT, when there is a further peripheral transfer to perform.
- ST_WENABLEP, when the currently pending peripheral transfer is a write, and there is a further transfer to perform.

## ST_RENABLE :-

During this state the PENABLE output is driven HIGH, enabling the current APB transfer. All other APB outputs remain the same as the previous cycle.

The ST_RENABLE state is always entered from ST_READ.

The next state is:

- ST_READ, when there is a further peripheral read transfer to perform.
- ST_WWAIT, when there is a further peripheral write transfer to perform.
- ST_IDLE, when there are no further peripheral transfers to perform.

## ST_WENABLE :-

During this state the PENABLE output is driven HIGH, enabling the current APB transfer. All other APB outputs remain the same as the previous cycle.

The ST_WENABLE state is always entered from ST_WRITE.The

next state is:

- ST_READ, when there is a further peripheral read transfer to perform.
- ST_WWAIT, when there is a further peripheral write transfer to perform.
- ST_IDLE, when there are no further peripheral transfers to perform.

## ST_WENABLEP :-

A wait state is inserted if the pending transfer is a read because, when a read follows a write, an extra wait state must be inserted to allow the write transfer to complete on the APB before the read is started.

The ST_WENABLEP state is entered from:

- ST_WRITE, when the currently driven AHB transfer is a peripheral transfer.
- ST_WRITEP, when there is a pending peripheral transfer following the current write.

The next state is:

- ST_READ, when the pending transfer is a read.
- ST_WRITE, when the pending transfer is a write, and there are no further transfersto perform.
- ST_WRITEP, when the pending transfer is a write.

## APB output signal Generation:

The generation of all APB output signals is based on the status of the transfer state machine:

• **PWDATA** is a registered version of the HWDATA input, which is only enabled during a write transfer. As the bridge is the only bus master on the APB, then it can drive PWDATA continuously.

• **PENABLE** is only set HIGH during one of three enable states, in the last cycle ofan APB transfer. A register is used to generate this output from the next state of the transfer state machine.

• **PSELx** outputs are decoded from the current transfer address. They are only valid during the read, write and enable states, and are all driven LOW at all other times sothat no peripherals are selected when no transfers are being performed.

• **PADDR** is a registered version of the currently selected address input (HADDR or the address register) and only changes when the read and write states are entered atthe start of the APB transfer.

• **PWRITE** is set HIGH during a write transfer, and only changes when a new APB transfer is started. A register is used to generate this output from the next state of thetransfer state machine.

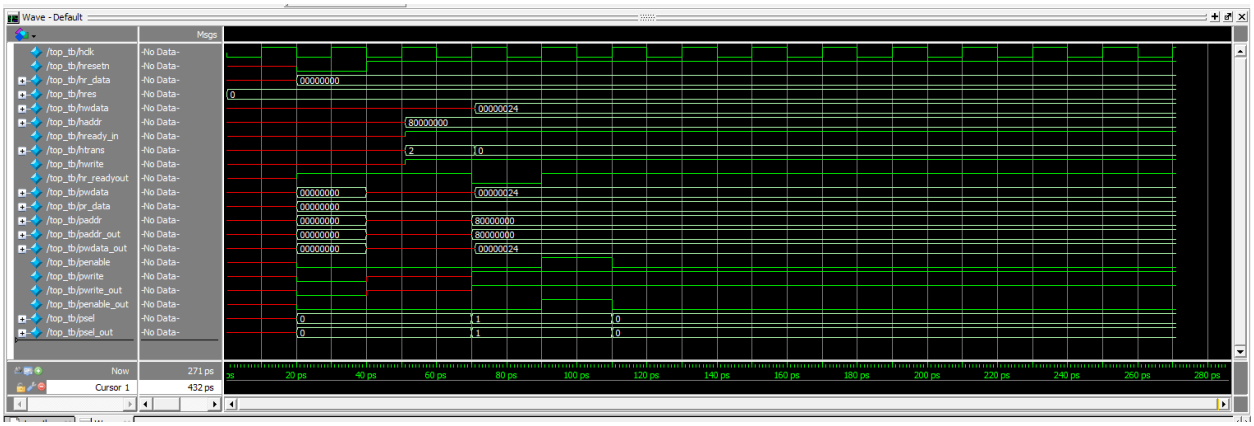
The APB signal is used as an enable on the PSEL, PWRITE and PADDR outputregisters, ensuring that these signals only change when a new APB transfer is started,when the next state is ST_READ, ST_WRITE, or ST_WRITEP.

## AHB output signal Generation:

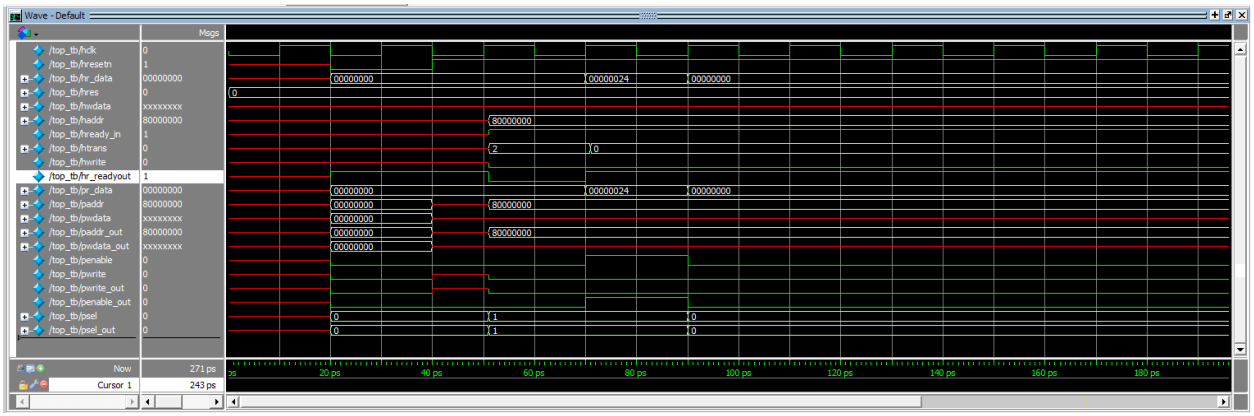
- **HRDATA** is directly driven with the current value of PRDATA. APB slaves only drive read data during the enable phase of the APB transfer, with PRDATA set LOWat all other times, so bus clash is avoided on HRDATA (assuming OR bus connections for both the AHB and APB read data buses).
- **HREADYout** is driven with a registered signal to improve the output timing. Wait states are inserted by the APB bridge during the ST_READ and ST_WRITEP states, and during the ST_WENABLEP state when the next transfer to be performed is a read.
- **HRESP** is continuously held LOW, as the APB bridge does not generate SPLIT, RETRY or ERROR responses.

# Output Waveform:

## AHB.single_write();




## AHB.single_read();




## AHB.brust_4_incr_write();