VeriFlow Technologies India (P) Ltd

AHB Monitor VIP

Version 0.3, Dec 05, 2008

Prabuddha Khare

VeriFlow Technologies India (P) Ltd

Rev. #	Designer	Description	Date Released
0.1	Prabuddha Khare	Initial Draft	May 29, 2008
0.2	Prabuddha Khare	Added more sections and TOC	July 22, 2008
0.3	Prabuddha Khare	Added missing features	Dec 05, 2008

VeriFlow Technologies India (P) Ltd

Table of Contents

VeriFlow Technologies India (P) Ltd

1 Introduction

This is a user-cum-specification document for the AHB monitor VIP. The document captures the usage-intent of the module and provides details of all the supported features. There is a section on usage where particulars about instantiation and interconnections are given. There is also a section showing sample messages generated by the VIP for normal transaction logging as well as for violations and abnormal bus behavior.

1.1 Intended Audience

The AHB monitor VIP is targeted for use by ASIC verification teams working on projects that use the AHB bus as one of the main communication trunk line. Usually this involves several masters and slaves with one or more CPUs to form a system-on-chip or SOC. During development and debugging, therefore, the AHB monitor will prove to be an invaluable help to ensure transactions on AHB are following the standard protocol.

1.2 Intended Usage

The AHB monitor VIP serves as verification component running in the background during simulation runs. By default it logs the AHB transactions onto standard output. It also logs violations as and when encountered. These logs can be selectively disabled. The monitor module should be instantiated and connected on the test-bench. In most SOC configurations, the port connections of the monitor will have to be hierarchically connected to the AHB bus, it being internal to the SOC.

1.3 Delivery Overview

The VIP is delivery consists of the following two files. The first is the main module definition and the second is an include file containing constant definitions.

AHBMonitor.sv
AHBDefines.svh

2 AHB Monitor and the AHB system-bus

The AHB monitor is a verification IP to be used in system level simulation environments which have multi-master multi-slave configurations with an arbiter/decoder/multiplexer system controller module. The diagram below shows the placement of the monitor with respect to the AHB bus.

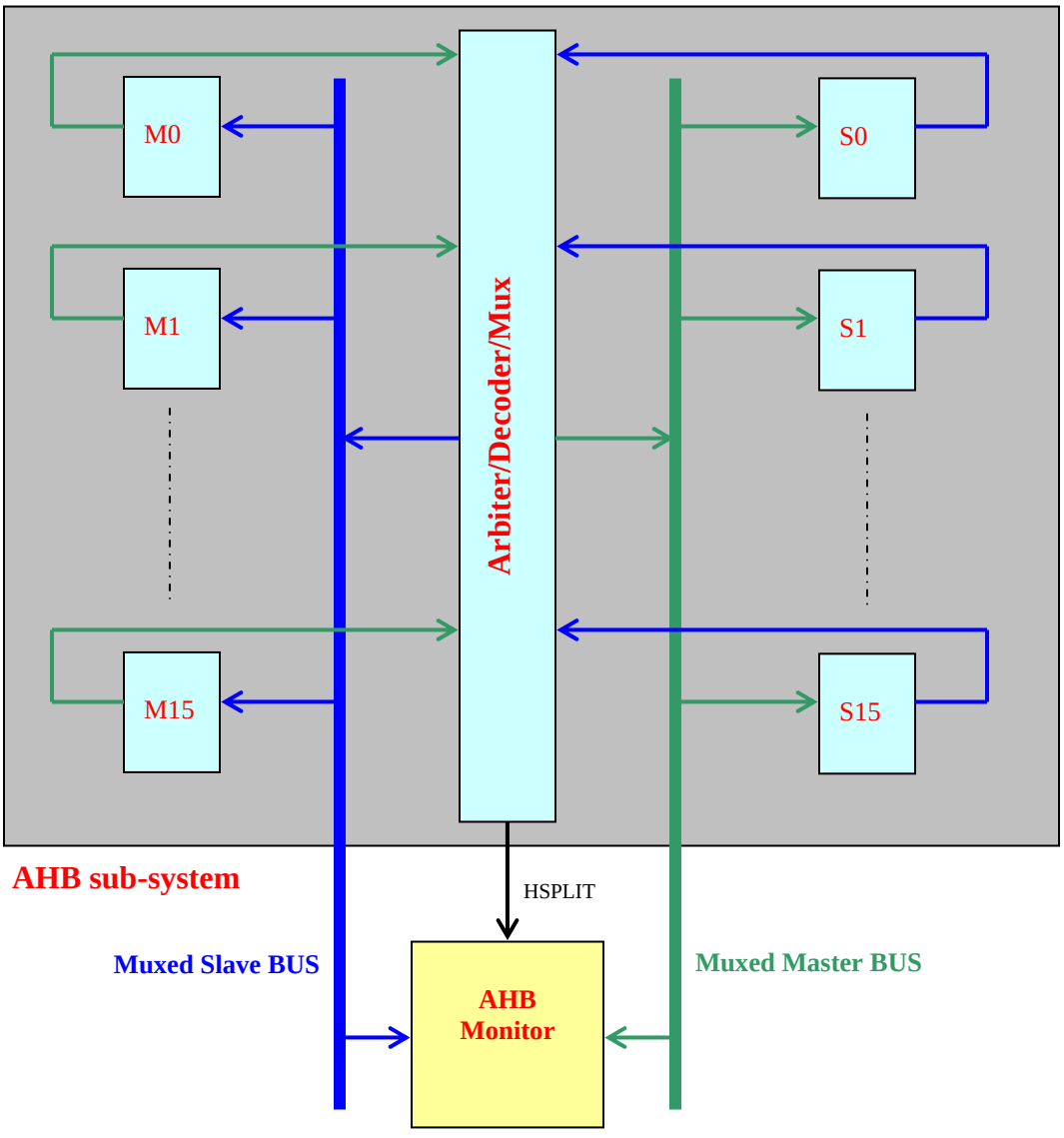

Figure-1 – Placement of AHB Monitor in the AHB sub-system

As can be seen, the monitor must be connected to the muxed AHB buses and not to the point to point connections of masters and slaves to the Arbiter/decoder/mux module.

Apart from the bus connections the monitor must also be connected to certain signals from the arbiter/decoder/mux module. These connections have not been shown in the above diagram. For more details please refer to the signal list section of the monitor in this document. The example instantiation code may also be referred to.

3 Legal transactions monitored and reported

By default the monitor shall report the on-going transactions in the following single line format:

MONITOR-> STime: Master-ID-> Slave-ID Type-Command-Size A=Address D=Data-> Resp at ETime

Where: <Type> is Single/INCR/INCR4/INCR8/INCR16/WRAP4/WRAP8/WRAP16
<Command> is READ or WRITE
<Size> is WORD/HALF-WORD/BYTE
<Resp> is the slave response OK/RETRY/SPLIT/ERROR
<STime> is the start time of transaction
<ETime> is the end time of transaction

The logging of such legal transactions may be turned off as described later.

4 Violations monitored and reported

The following is a list of AHB bus protocol related violations that the monitor shall check for and report. These may be selectively turned on/off. Since the AHB protocol specification is unclear on several points leaving them open to interpretation, the monitor has been designed to be configurable for some of these points and where such flexibility could not be built-in, the behavior has been clearly described to avoid any ambiguity in its usage by the end-user. When a violation is reported, the single line format shown above shall be logged as a prelude to the violation information with as much information as possible to report at that juncture.

- Multiple grant violation
- Address misalignment.
- Address increment violation for bursts
- 1-k Boundary crossing for bursts
- WRAP burst violation
- Address decoding violation (Multiple HSEL)
- Command sequence violation on HTRANS
- Command-bus violation for bursts (HSIZE, HBURST, HWRITE)
- Burst sequence violation on HTRANS
- Normal response violation

- 2-cycle response violation
- BUSY insertion violation for bursts
- HREADY violation
- Bus Grant to split-master violation
- Split/un-split timing violations
- Splitting the correct master on a split response
- Un-splitting by inappropriate slave
- HMASTLOCK timing violation
- Pre-emption of a locked transfer violation

The violations are logged in the following format:

MONITOR_VIOLATION-> STime: Master-ID-> Slave-ID Type-Command-Size A=Address D=Data-> Resp <Violation Details> at ETime

Where: <Violation Details> are specific to the violation detected.
<STime> is the time start time of transaction
<ETime> is the time at which the violation was detected

The monitor maintains a count of the total violations detected that can be queried anytime by a task call. The detailed description for these violations is described hereunder.

4.1 Multiple grant violation

In the AHB sub-system, only one master is allowed to own the bus. If the arbiter asserts grants to multiple masters it is a violation. This feature can only be checked if the point to point connections between the masters and the arbiter for the signals HBUSREQ and HGRANT are also made to the monitor. The diagram in Figure-1 does not show this connectivity but the current release of the monitor has the necessary ports and supports this feature.

Violation messages issued:
"… Grant issued to multiple masters …"

4.2 Address misalignment

The monitor shall decode the HSIZE and check whether the address is aligned for that transfer size. For word transfer, the address bits 0 and 1 must be 2'b00. For half-word transfer, address bit-0 must be 0. For byte transfer the address can be any value.

Violation messages issued:
"… Misaligned address at …"

4.3 Address increment violation for bursts

When a burst transfer is seen, the monitor shall check whether the address increments are commensurate with HSIZE. For word transfers, the increments must be by 4. For half-word transfers the increment must be by 2. For byte transfers the increment must be by 1. For wrap bursts also the increments are monitored.

Violation messages issued:

"… Bad INCR address at …"

4.4 1-k Boundary crossing for bursts

When a burst is seen on the AHB bus, the monitor shall check whether it is a fixed length burst or not. If it is, then it shall check that the length of the burst does not cross 1-k boundary based on the HSIZE, HBURST and the starting address. It will be able to report the violation at the start of the transaction itself. For incremental burst, the monitor will have to wait till the actual boundary is reached before being able to decide whether the last beat violated the 1-K boundary crossing rule or not. For wrap bursts, this check is not necessary as once the boundary is reached, wrap logic should generate an address within the same 1k range. In case this is violated, it will be caught byte the '**wrap burst violation**' case mentioned below.

Violation messages issued:

"… Transfer length crosses 1-k boundary at …"

4.5 WRAP burst violation

For wrap bursts, the monitor shall check that the address wraps at the wrap boundary as decided by HBURST and HSIZE. For wrap-16 the wrap size is 64 bytes for word transfers, 32 bytes for half-word transfers and 16 bytes for byte transfers. For wrap-8, the wrap size is 32 bytes for word transfer, 16 bytes for half-word transfers and 8 bytes for byte transfers. For wrap-4, the wrap size is 16 bytes for word transfer, 8 bytes for half-word transfer and 4 bytes for byte transfer.

Thus for word transfer of wrap-16, the wrap blocks are:
0x0000_0000 to 0x0000_003f
0x0000_0040 to 0x0000_007f,
and so on.

For half-word transfer of wrap-16, the wrap blocks are:
0x0000_0000 to 0x0000_001f
0x0000_0020 to 0x0000_003f,
and so on.

Thus for byte transfer of wrap-16, the wrap blocks are:

0x0000_0000 to 0x0000_000f
0x0000_0010 to 0x0000_001f,
and so on.

Similar algorithm for other wrap sizes is used. When the starting address falls within these blocks, the wrap should happen when the end of the block is reached.

Since the 1-K boundary is always aligned with wrap boundary, the 1-K boundary crossing for wrap transfers will be detected automatically as a wrap violation.

Violation messages issued:

"… WRAP address out of wrap-block …"

4.6 Address decoding violation (Multiple HSEL)

The monitor shall check that only one slave is selected at any one time since AHB does not support overlapped slave addressing. AHB transactions being pipelined, it is possible that a slave gets de-selected during its data phase.

Violation messages issued:

"… Multiple HSEL active at …"

4.7 Command sequence violation on HTRANS

A new transaction can begin only with a NON-SEQ command. If the monitor detects a SEQ command after IDLE, it shall report it as a violation. It shall also report a violation if a SEQ command is detected immediately following a burst transfer.

Violation messages issued:

"… Unexpected SEQ command after IDLE at …"
"… Unexpected SEQ command beyond burst at …"

4.8 Command-bus violation for bursts (HSIZE, HBURST, HWRITE)

During a burst, the AHB command bus signals namely HSIZE, HBURST and HWRITE are not allowed to change. The monitor shall check that this condition is satisfied.

Violation messages issued:

"… Command signals changed within a burst at …"

4.9 Burst sequence violation on HTRANS

Burst sequence cannot have IDLEs inserted during the burst. Nor can there be a NON-SEQ command in the middle of a fixed burst length transfer. BUSY may be inserted during a burst.

Violation messages issued:

"… IDLE within a BURST not allowed at …"
"… Unexpected NON-SEQ command within a burst at …"

4.10 Normal response violation

The monitor shall check that the slaves generate OK response with HREADY high. Any other type of response with HREADY high is considered a violation.

Violation messages issued:

"… Invalid response. Expected OK HRESP at …"

4.11 2-cycle response violation

When a slave generates RETRY, SPLIT or ERROR responses, it should drive HRESP for two clocks with HREADY asserted low for the first clock and de-asserted on the 2nd clock. The monitor shall check for any violation in this behavior.

The monitor will also report it as a violation if the HRESP changes on the 2nd clock of the two-cycle response.

Violation messages issued:

"… Two-cycle RESP has HREADY low for more than 1-clock at …"
"… HRESP has changed in the 2nd clock of two-cycle response at …"

4.12 BUSY insertion violation for bursts

During a burst BUSY may be inserted anywhere except before the last beat or after it. Violation of this requirement shall be reported by the monitor. Busy insertion before NON-SEQ is also allowed.

Violation messages issued:

"… BUSY insertion not allowed on last beat at …"
"… BUSY insertion not allowed after end of burst at …"

4.13 HREADY violation

Although there is no explicit rule about how log the HREADY may be held asserted, the monitor shall report a violation if it is held asserted low for more than a configurable value (default 16).

It can also report which slave is driving wait states for so long, provided the HREADY signal from each slave is connected to the monitor. The diagram in Figure-1 does not show this configuration. The current release of the monitor does not report the slave ID.

Another violation related to HREADY reported by the monitor is when it is driven LOW during IDLE command.

Violation messages issued:

"… WAIT state with HREADY low exceeds nn clocks at …"

"… Unexpected HREADY for IDLE command at …"

4.14 Bus Grant to split-master violation

When a master is split, its request must not be honored by the arbiter until it is unsplit. In fact, the arbiter should remove the HGRANT immediately when a SPLIT response is detected. The monitor shall report a violation for both these conditions.

Violation messages issued:

"… Grant issued to split master …"

"… Split master's hgrant not removed in time at …"

4.15 Split/un-split timing violations

The AHB protocol says that a master cannot be unsplit at the same clock as the clock on which it is split. This means that there must be at least one clock delay between a split generation and an unsplit occurring. The monitor shall check that this requirement is satisfied by the system being monitored.

Violation messages issued:

"… Un-split happened in less than 1 clock after split for master …"

4.16 Splitting the correct master on a split response

When a slave splits a master, it is its responsibility to unsplit it at the appropriate time. If the slave performs an unsplit of a master that it has not already split, it is considered a violation and the monitor shall report it as such. If a slave has split several masters then it can unsplit all of them together or unsplit them one by one but it must not unsplit a master more than once.

Violation messages issued:

"… Un-split issued for not-split master …"

4.17 Un-splitting by inappropriate slave

If a master is split by one slave but gets unsplit by another, it is a violation. This condition cannot be checked by monitoring only the muxed HSPLIT bus. To support this feature the HSPLIT bus from all the slaves must be connected to the monitor. The diagram in Figure-1 does not show this configuration. The current release of the monitor does not support this feature.

4.18 HMASTLOCK timing violation

The HMASTLOCK is supposed to follow the HLOCK signal during a burst and must remain asserted throughout the burst if HLOCK was asserted at the start of transaction. The monitor shall report a violation if either of these requirements are not met.

Violation messages issued:

"… HMASTLOCK inconsistent with HLOCK at start of transaction …"
"… HMASTLOCK changed in middle of transaction …"

4.19 Pre-emption of locked transfer violation

A locked transfer cannot be pre-empted by the arbiter. Moreover, if a slave generates a RETRY response, then also the arbiter may not switch the grant even if it was the last beat. The arbiter is supposed to maintain the grant with the locked master for an extra cycle so that the response of the last beat can be handled without causing a pre-emption of the locked burst. The monitor checks for both these conditions and reports violation appropriately.

Violation messages issued:

"… Locked burst got pre-empted at …"
"… Locked transfer with RETRY got pre-empted at …"

5 User tasks for configuration and status

The following tasks are provided for configuring the monitor and extracting useful status information about the transactions at any time.

5.1 Task: set_wait_limit (integer limit);

Sets the upper limit for the wait-state monitoring. When the monitor detects wait states exceeding the given limit, it shall log it as a violation. Default value is 16.
Usage: set_wait_limit (20); --> Sets the wait limit to 20.

5.2 Function: get_total_violations();

This function returns the total violations detected by the monitor thus far.
Usage: vcount = get_total_violations ();

5.3 Task: transaction_logging (bit enable_disable);

This task may be used to enable or disable the logging of regular transactions. It does not affect the logging of violations. When the input to the task 'enable_disable' is '1', the logging is enabled. When the input is '0', the logging is disabled. Default is enabled.
Usage: transaction_logging (0); --> Transaction logging is disabled.

5.4 Task: violation_logging (bit enable_disable);

This task may be used to enable or disable the logging of violations. It does not affect the logging of regular transactions. When violation logging is disabled, the violation counting is also suspended. The counter is not reset so that when logging is re-enabled subsequently the counting will resume from that point. When the input to the task 'enable_disable' is '1', the logging is enabled. When the input is '0', the logging is disabled. Default is enabled.
Usage: violation_logging (0); --> Violation logging is disabled. Counting is suspended.

6 Using the Monitor

This section gives details about the Monitor entity with port connection details, instantiation and task usage examples. Sample log messages are also shown.

6.1 Monitor port connections

The following figure shows the port connections and module definition for the behavioral model of the monitor VIP.

```
module AHBMonitor (
        hclk,
        hresetn,
        hready,
```

```
            hresp,
            hrdata,
            hsplit,
            haddr,
            hwrite,
            hsize,
            htrans,
            hburst,
            hwdata,
            hsel,
            hmaster,
            hmastlock,
            hbusreq,
            hlock,
            hgrant
);   // END module AHBMonitor

input            hclk;
input            hresetn;
input            hready;
input [1:0]      hresp;
input [31:0]     hrdata;
input [15:0]     hsplit;
input [31:0]     haddr;
input            hwrite;
input [2:0]      hsize;
input [1:0]      htrans;
input [2:0]      hburst;
input [31:0]     hwdata;
input [15:0]     hsel;
input [3:0]      hmaster;
input            hmastlock;
input [15:0]     hbusreq;
input [15:0]     hlock;
input [15:0]     hgrant;
```

6.2 Instantiation and task usage

The monitor is instantiated in the test-bench as show in the example below:

```
AHBMonitor u_ahbmonitor (
            .hclk       (hclk),
            .hresetn    (hresetn),
            .hready     (hready),
```

```
        .hresp      (hresp),
        .hrdata     (hrdata),
        .hsplit     (hsplit_arb),
        .haddr      (haddr),
        .hwrite     (hwrite),
        .hsize      (hsize),
        .htrans     (htrans),
        .hburst     (hburst),
        .hwdata     (hwdata),
        .hsel       ({12'h000, hsel}),
        .hmaster    (hmaster),
        .hmastlock (hmastlock)
        .hbusreq    (hbusreq),
        .hlock      (hlock)
        .hgrant     (hgrant),
      );
```

The following is a typical usage of the monitor from a user-test:

```
initial begin
  // Wait for reset to complete
   …
  // Configure monitor to allow wait-insertion for 20 clocks
  u_ahbmonitor. set_wait_limit (20);

  // Perform the test sequence
  …
  …
  // At the end of the test sequence (or at anytime) get violation count from monitor
  violation_cnt = u_ahbmonitor.get_total_violations ();
  if (violation_cnt != 0) begin
    $display ("================================================");
    $display ("ERROR: Test FAILED with %d AHB violations", violation_cnt);
    $display ("================================================");
  end
  $finish();
end  // End test case
```

The following are sample log messages for **legal** transactions:

```
MONITOR->   255000ps: M3-> S0 INCR-WRITE-HALFWORD A=0e9c259e D=b5340000-> OK at   265000ps

MONITOR->   345000ps: M1-> S0 INCR4-WRITE-WORD A=0000224a-> ERROR at   385000ps

MONITOR-> 1645000ps: M0-> S3 INCR-WRITE-BYTE A=300025c4-> RETRY at  1675000ps
```

VeriFlow Technologies India (P) Ltd

The following are sample log messages for transactions that **violated** the AHB protocol:

MONITOR_VIOLATION-> 92305000ps: M0-> S0 INCR-WRITE-WORD A=00001adc NA=00001ac0 EX=00001ae0
Bad INCR address at 92315000ps

MONITOR_VIOLATION-> 285625000ps: M0-> S0 INCR-READ-WORD A=00001adc -> WAIT state with
HREADY low exceeds 20 clocks at 285845000ps