Digital Logic and Electronic Systems Design Company

Verilog AHB Testbench User's Guide

Pulse Logic

www.pulselogic.com.pl

e-mail: info@pulselogic.com.pl

Document version: 1.0

Document date: March 2010

Table of Contents

1. AHB Bus Testbench Introduction

The AHB bus testbench is written in Verilog-2001 HDL language. All sub modules of the testbench except AHBMaster are synthesizable and can be used for testing and also can be implemented to target FPGA or ASIC devices. The testbench is parametrized. It is easy to expand the testbench and add more masters or slaves. Each module has parameters allowing quick expansion. Operations of the AHB master modules can be customized by defining AHB transfers in a text file. A User may add other modules to the testbench or replace existing modules and test them together to check if they correctly respond to different AHB transfers from multiple AHB masters. The testbench was designed to maximally stress the tested units. All the units must respond in the requested time. However, the bus cycles are always compatible with AMBA™ Specification Rev 2.0 and any differences with the specification in timing of the AHB signals should be treated as a bug in the testbench or in a tested unit. The testbench consists of the following files:

File	Type	Description
defs.v	RTL Code	Definition of constants
FileRead_obf.v	Obfuscated behavioral Code	Module for parsing and reading input text file for AHB master
AHBSlaveDummy_obf.v	Obfuscated RTL code	AHB dummy slave
AHBSlave_obf.v	Obfuscated RTL code	AHB slave - synchronous RAM memory
AHBDecoder_obf.v	Obfuscated RTL code	AHB decoder
AHBArbiter_obf.v	Obfuscated RTL code	AHB arbiter
AHBMaster_obf.v	Obfuscated behavioral Code	AHB master bus functional model
AHBMasterToSlave_obf.v	Obfuscated RTL code	AHB master to slave multiplexing unit
AHBSlaveToMaster_obf.v	Obfuscated RTL code	AHB slave to master multiplexing unit
AHBMasterToArbiter_obf.v	Obfuscated RTL code	AHB master to arbiter multiplexing unit
AHBBusSystem.v	RTL Code	RTL Top level module connecting all units
AHBBusSystem_TB.v	Behavioral Code	Top level module driving clocks and resets of all the units
runme.do	Script file	Compilation script file
input_1.txt	Text file	Input file with AHB transfers definition for Master 1.
input_2.txt	Text file	Input file with AHB transfers definition for Master 2.
pattern_1.txt	Text file	Pattern file with responses from Master 1.
pattern_2.txt	Text file	Pattern file with responses from Master 2.

Table 1. AHB Testbench files

Compilation order and simulation initialization is defined in the *runme.do* file. This is Mentor Modelsim™ script file format. It can be easy rewritten to Synopsys VCS™, Cadence NC-Sim™ or Aldec Active-HDL™ compilation script file. The testbench consists of the following units:

- AHBSlaveDummy
- AHBSlave
- AHBDecoder
- AHBArbiter
- AHBMaster
- AHBMasterToSlave
- AHBSlaveToMaster
- AHBMasterToArbiter

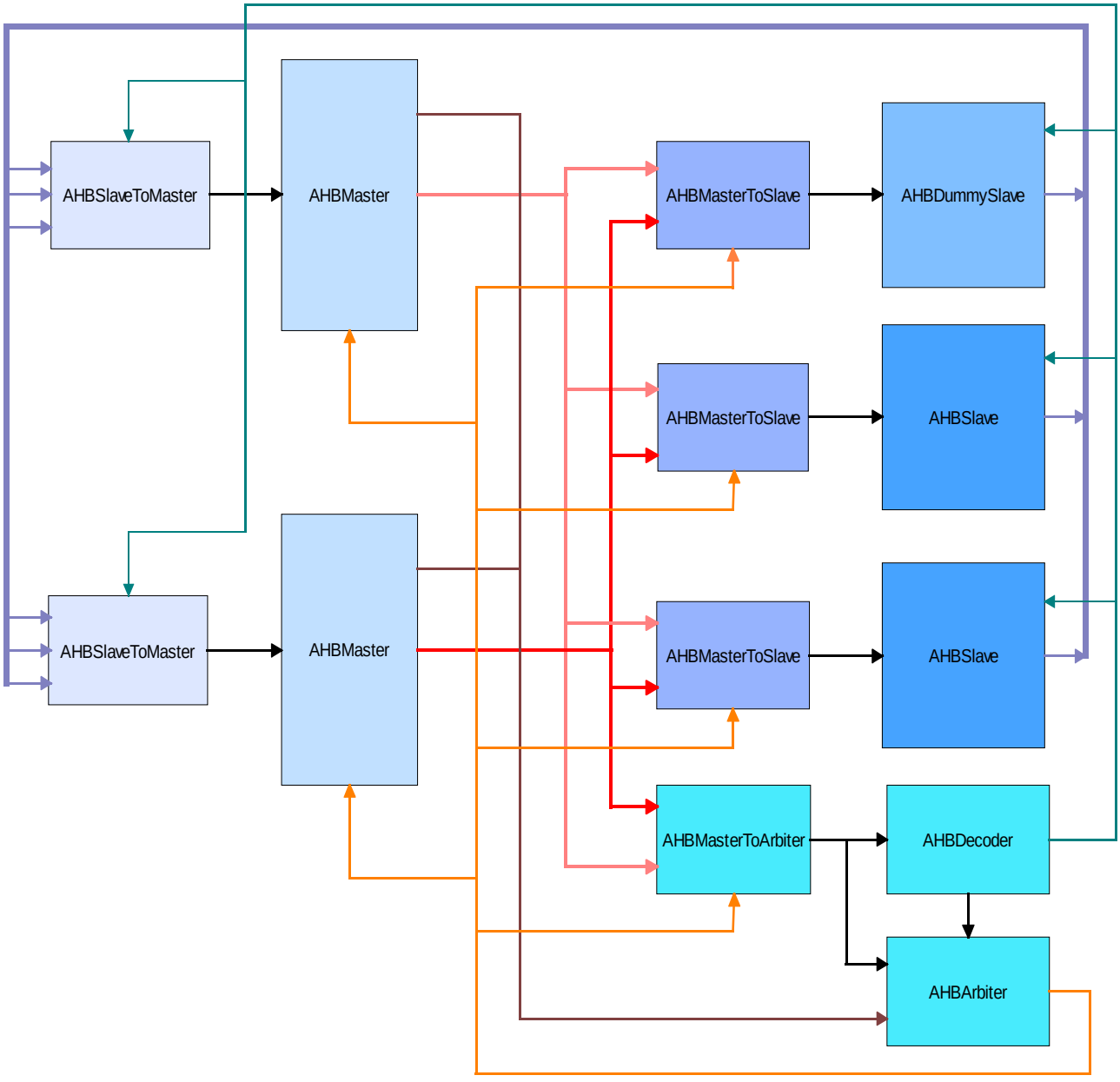

Figure 1. Block diagram of AHB Testbench

The figure above shows interconnections between all the units. The AHB bus does not have any three-state signals and multiplexing units must be used to connect multiple masters and slaves units. Each unit from the figure will be separately described in the following chapters.

2. How To Use This Testbench

The figure above shows an example of AHB bus system. The system consists of two AHB masters and two AHB slaves. AHB arbiter is also needed because two masters may access the same slave at the same time and only one of them can be granted. Similarly, AHB decoder is used to check, which slave is currently addressed by the granted master. Such system is implemented in *AHBBusSystem.v* file. The *AHBBusSystem.v* file contains instances of all the mentioned AHB modules connected like in the figure above. More detailed description of all the modules is available in the following chapters.
You can run simulation of this system to check how all the modules operate in the system. Script files and configuration files for AHB masters are provided as mentioned in the previous chapter. Please, remember that all the multiplexing units are highly optimized to avoid any dead cycles on the bus. The transfers from two masters are parallelized if possible to archive best performance of the system. Analyzing the multiplexing units may be difficult. It is much easier to observer interface of selected slave or master. Each AHB master writes information about performed transfer to a log file. The easiest method of analyzing operations of the AHB master is by reviewing the log file.
Once you are familiar with operation of the testbench and their modules you can start with modification of the testbench architecture. You can do the following:

- Replace one of existing masters or slaves in the testbench with yours module
- Add a new master or slave to the testbench
- Change address locations of the AHB slaves
- Customize input file with transfers definition for AHB masters

The first three items from the above list can be done by modifications of the *AHBBusSystem.v* file. You can add, remove or replace some instances in the file. If you expand the system by adding more masters or slaves you must also configure the multiplexing units including AHB arbiter and decoder. The units have appropriate parameters that needs to be adjusted for given number of masters and slaves in the system. More information about parameters of the units is available in the following chapters.
Similarly, customization of the input file with transfers definition for AHB masters is described in the AHBMaster chapter.

3. How To Download The Testbench Files

The testbench is available for free under modified BSD license. Click the following link to download testbench files:

http://www.pulselogic.com.pl/files/ahbTestbench_obf.zip

4. Limitations Of AHB Testbench

The following functionality was not implemented in the AHB testbench:

- Split transfers
- Retry transfers
- Early burst termination

The testbench is available for free with one limitation. Most of the testbench files are obfuscated. It means that comments and text formating ware removed from the files and most of identifiers ware replaced with automatically generated strings. It makes the files very difficult to read or modify. However, the files are fully functional and should work correctly with any simulation or synthesis tool. The original testbench files are not available for free.

5. AHBMaster Module

The AHBMaster module implements functionality of AHB master device. It is able to initiate transfers on AHB bus and write or read data from AHB slaves. The following tables show interface of the AHBMaster module:

Parameter	Value	Description
P_IN_FILENAME	"input.txt"	Input file name with definition of bus cycles
P_OUT_FILENAME	"output.txt"	Output log file name
P_MASTER_NAME	"master"	Master name used for identifying messages from a master instance
P_STOP_SIM_AT_EOF	1	If not equal to '0' the $stop is called when end of input file is reached.

Table 2. Parameters of AHBMaster module

Port	Direction	Description
HCLK	IN	Clock
HRESETn	IN	Reset
HBUSREQ	OUT	AHB bus request
HLOCK	OUT	AHB locked transfer
HGRANT	IN	AHB bus grant
HTRANS [1 : 0]	OUT	AHB transfer type signal
HBURST [2 : 0]	OUT	AHB burst type signal
HSIZE [2 : 0]	OUT	AHB transfer size
HADDR [31 : 0]	OUT	AHB address bus
HPROT [3 : 0]	OUT	AHB protection control
HWDATA [31 : 0]	OUT	AHB write data bus
HWRITE	OUT	AHB transfer direction signal
HRDATA [31 : 0]	IN	AHB read data bus
HRESP [1 : 0]	IN	AHB transfer response signal
HREADY	IN	AHB transfer done signal

Table 3. Interface signals of AHBMaster module

The AHBMaster module is a typical bus functional model. The AHBMaster reads commands from input text file defined by P_IN_FILENAME parameter. The input file contains definition of AHB bus transfers that should be generated on the AHB bus. The AHBMaster module records all operation in a log file specified by P_OUT_FILENAME parameter. It also prints some messages on simulator console. It is useful to select different name for each master instance using P_MASTER_NAME parameter. It allows easier identification of messages from given master on simulator console. Each master may optionally stop simulation if P_STOP_SIM_AT_EOF parameter is set to one. The AHBMaster module calls $stop task after reaching end of input file

when P_STOP_SIM_AT_EOF parameter equal to one.
The interface of AHBMaster module consists of AHB signals. Please, refer to AMBA™ Specification Rev 2.0 for more details on AHB signals and protocol of operations.

A User does not need to know functionality of the AHBMaster module. A User may customize AHBMaster module operation by providing input file with transfers definition. The input file is a simple text file where AHB transfers are defined using very simple format. The design contains two sample input files *input_1.txt* and *input_2.txt* that can be used as reference. Below is description of the input file format.
There are two possible line types in the input file:

- COMMAND line. Defines AHB transfer that should be generated
- DATA line. Defines data that should be sent over AHB bus when AHBMaster module performs write transfer

The command line starts from COMMAND keyword followed by semicolon and list of parameters defining AHB transfer. The data line starts from DATA keyword followed by semicolon and hexadecimal data word that should be transferred over AHB bus. Below is an example of command and data line:

COMMAND: WRITE BITS32 SINGLE LENGTH: 0 PROT: 0 ADDRESS: 00000000 BREQ_DELAY: 8
DATA: 00000000

In case of burst transfer the command line must be followed by multiple data lines defining all data words that must be transferred during the burst cycle. Below is en example of four beats burst transfer definition:

COMMAND: WRITE BITS32 INCR4 LENGTH: 0 PROT: 0 ADDRESS: 00000080 BREQ_DELAY: 18
DATA: 00000020
DATA: 00000021
DATA: 00000022
DATA: 00000023

The command line need seven parameters defining AHB transfer. The order of parameters is important and must be always the same. The following is the list of the parameters in appropriate order:

- Transfer direction
- Transfer size
- Burst type
- Length of transfer for INCR bursts
- Protection control
- Start address
- Delay of next bus request

Transfer direction parameter specifies direction of the AHB transfer. It has two possible values:

- WRITE
- READ

Please, remember than any transfer with WRITE direction should be followed by appropriate number of DATA lines.
Transfer size parameter specifies size of the AHB transfer. The following is the list of all possible values of the parameter according to AMBA™ Specification Rev 2.0:

- BITS8
- BITS16

- BITS32
- BITS64
- BITS128
- BITS256
- BITS512
- BITS1024

The testbench was tested with 32 bits size of the data bus. The following transfer size parameter values are suitable for 32 bits data bus:

- BITS8 – one byte
- BITS16 – half word
- BITS32 – word

Other values of the parameter should not be used with 32 bits data bus.

The burst type parameter defines eight possible burst transfers

- SINGLE – Single transfer
- INCR – Incrementing burst of unspecified length
- WRAP4 – 4-beat wrapping burst
- INCR4 – 4-beat incrementing burst
- WRAP8 – 8-beat wrapping burst
- INCR8 – 8-beat incrementing burst
- WRAP16 – 16-beat wrapping burst
- INCR16 –16-beat incrementing burst

Please, refer to AMBA™ Specification Rev 2.0 for more details on burst transfers. The INCR transfer is a burst with unspecified length. The length of this transfer should be specified using LENGTH parameter.

The length parameter should be always set to zero except burst transfers of INCR type. Other burst transfers have strictly specified length. The AHBMaster module need to know the transfer length upfront. The LENGTH parameter should be always specified when INCR burst transfer is defined. The length parameter defines length of the transfer.

The protection control parameter can be used to provide additional information about bus access. The parameter indicates if the transfer is:

- an opcode fetch or data access
- a privileged mode access or user mode access.

The parameter is useful for modules that implement some level of protection. Please, refer to AMBA™ Specification Rev 2.0 for more details on protection control.

The start address parameter defines start adders for the transfer. Each read or write transfer starts from the specified start address. In case of burst transfers the adders is incremented for subsequent beats of the burst according to burst type and transfer size parameters. Please, refer to AMBA™ Specification Rev 2.0 for more details on addressing during burst transfers.

The last parameter defines delay of the next bus request. It defines in clock cycles delay between start of the current transfer and start of the bus request for the next transfer. The parameter can be used to postpone the next transfer an force IDLE state on the bus. The AHBMaster starts current

transfer and keeps bus request line inactive until the requested delay elapsed. If the bus transfer is finished before the requested delay elapsed the AHBMaster module drives IDLE state on the AHB bus. The AHBMaster module applies the bus request immediately after the delay period expired. It does not have to wait until current transfer is completed. The AHB bus allows to request access to the bus any time during a transfer.

6. AHBSlave Module

The AHBSlave module is an example of simple AHB slave. This module implements synchronous RAM memory with AHB interface. The following tables show interface of the module.

Parameter	Value	Description
P_ADDR_SIZE	1024	Size of occupied address space
P_WAIT_STATES	5	Number of inserted wait state cycles

Table 4. Parameters of AHBSlave module

Port	Direction	Description
HCLK	IN	Clock
HRESETn	IN	Reset
HSEL	IN	AHB slave select signal
HTRANS [1 : 0]	IN	AHB transfer type signal
HBURST [2 : 0]	IN	AHB burst type signal
HSIZE [2 : 0]	IN	AHB transfer size
HADDR [31 : 0]	IN	AHB address bus
HPROT [3 : 0]	IN	AHB protection control
HWDATA [31 : 0]	IN	AHB write data bus
HWRITE	IN	AHB transfer direction signal
HRDATA [31 : 0]	OUT	AHB read data bus
HRESP [1 : 0]	OUT	AHB transfer response signal
HREADY	OUT	AHB transfer done signal

Table 5. Interface signals of AHBSlave module

The P_ADDR_SIZE parameter defines number of words of the RAM memory. It also determines address space occupied by the RAM memory.
The P_WAIT_STATES parameters defines delay in clock cycles for transfer done signal. The RAM memory may respond with requested delay to test behavior of other modules in AHB testbench.
The AHBSlave module has only AHB signals in its interface. You can find description of the signals in AMBA™ Specification Rev 2.0 document. All the signals should work as stated in the AMBA™ Specification Rev 2.0 document.

7. AHBSlaveDummy Module

The dummy slave is very important for operations of AHB bus. The dummy slave is selected if current address on AHB bus is not assigned to any device. It means, that AHB master is trying to communicate with invalid address. The AHB decoder module selects currently addressed slave by applying active state on its HSEL signal. The AHB decoder has information what address space is assigned to each slave. The address from AHB master is compared to find, which slave is addressed and after that, HSEL signal is applied. If AHB decoder detects an address that is not assigned to any device, it selects the dummy slave. The dummy slave responds always with bus error on any type of transfer from AHB master. It allows the master to complete the transfer and detect that invalid address was selected.
Interface of the AHBSlaveDummy module is shown in the tables below:

Port	Direction	Description
HCLK	IN	Clock
HRESETn	IN	Reset
HSEL	IN	AHB slave select signal
HTRANS [1 : 0]	IN	AHB transfer type signal
HBURST [2 : 0]	IN	AHB burst type signal
HSIZE [2 : 0]	IN	AHB transfer size
HADDR [31 : 0]	IN	AHB address bus
HPROT [3 : 0]	IN	AHB protection control
HWDATA [31 : 0]	IN	AHB write data bus
HWRITE	IN	AHB transfer direction signal
HRDATA [31 : 0]	OUT	AHB read data bus
HRESP [1 : 0]	OUT	AHB transfer response signal
HREADY	OUT	AHB transfer done signal

Table 6. Interface signals of AHBSlaveDummy module

The AHBSlaveDummy module has only AHB signals in its interface. You can find description of the signals in AMBA™ Specification Rev 2.0 document. All the signals should work as stated in the AMBA™ Specification Rev 2.0 document.

8. AHBDecoder Module

The AHBDecoder module is used to decode address from currently granted master and select addressed slave by applying appropriate HSEL line. The AHBDecoder module must be aware what address space is assigned to each slave. The dummy slave can be selected, if the AHB master module applies invalid address that does not refer to any slave in the testbench. The following table shows interface of the AHBDecoder module:

Parameter	Value	Description
P_SLAVES_NUM	2	Number of AHB slaves in the testbench excluding dummy slave
P_ADDR_BASE_LSB	12	LSB address line used for selecting slaves
P_ADDR_BASE_MSB	clog2(P_SLAVES_NUM-1) + P_ADDR_BASE_LSB - 1	MSB address line used for selecting slaves

Table 7. Parameters of AHBDecoder module

Port	Direction	Description
HTRANS [1 : 0]	IN	AHB transfer type signal
HADDR [31 : 0]	IN	AHB address bus
HMASTER_VALID	IN	High state indicates that HMASTER signal is valid
HSEL [P_SLAVES_NUM-1 : 0]	OUT	AHB select signals for all slaves in the testbench
HSEL_DUMMY	OUT	AHB dummy slave select

Table 8. Interface signals of AHBDecoder module

The P_SLAVES_NUM parameter defines number of slaves in the AHB testbench excluding dummy slave. The dummy slave is always present. The P_ADDR_BASE_LSB and P_ADDR_BASE_MSB parameters define slice of the address bus that is used to select the slaves. For example, if P_SLAVES_NUM parameter is set to 4, it means that 4 slaves are connected to the AHB bus plus the dummy slave. The P_ADDR_BASE_LSB is set by default to 12 and P_ADDR_BASE_MSB should be set to 13 to allow addressing the 4 slaves. In such configuration the AHB decoder is checking bit 12 and 13 of the address bus and selects appropriate slave. The following table shows how the slaves are selected:

AHB Address	AHB select signal	Description
HADDR[13 : 12] == 0	HSEL[0] = 1	Slave number 0 selected
HADDR[13 : 12] == 1	HSEL[1] = 1	Slave number 1 selected
HADDR[13 : 12] == 2	HSEL[2] = 1	Slave number 2 selected
HADDR[13 : 12] == 3	HSEL[3] = 1	Slave number 3 selected

Table 9. Selecting slaves algorithm

The AHB decoder monitors HTRANS lines to detect valid address on HADDR bus. If the valid address is detected the AHB decoder activates appropriate HSEL line and applies active high state on HMASTER_VALID signal. The dummy slave has separate HSEL_DUMMY line. The line is selected if current address is not valid or bus is in the idle state.

9. AHBArbiter Module

The AHB arbiter module is the most complicated unit in the testbench. It is responsible for analyzing bus requests from all masters and granting access to the bus for the highest priority master. The arbiter uses rotating priority algorithm to select master with the highest priority. If a grant was given to selected master the lowest priority is assigned to the master. This algorithm is known also as round robin algorithm. There are no devices on the bus that have permanently higher priority than others. All devices requesting the bus have the same priority and the same chances to get access to the bus. The following tables show interface of the AHBArbiter module:

Parameter	Value	Description
P_MASTERS_NUM	2	Number of AHB masters in the testbench
P_SLAVES_NUM	2	Number of AHB slaves in the testbench excluding dummy slave

Table 10. Parameters of AHBArbiter module

Port	Direction	Description
HCLK	IN	Clock
HRESETn	IN	Reset
HBUSREQ [P_MASTERS_NUM - 1 : 0]	IN	AHB bus requests signals from all masters in the testbench
HLOCK [P_MASTERS_NUM - 1 : 0]	IN	AHB locked transfer signals from all masters in the testbench
HGRANT [P_MASTERS_NUM - 1 : 0]	OUT	AHB bus grant signals for all masters in the testbench
HTRANS [1 : 0]	IN	AHB transfer type signal
HBURST [2 : 0]	IN	AHB burst type signal
HADDR [31 : 0]	IN	AHB address bus
HSEL [P_SLAVES_NUM-1 : 0]	IN	AHB select signals for all slaves in the testbench
HREADY_MST [P_MASTERS_NUM-1 : 0]	IN	AHB transfer done signals connected to all masters in the testbench
HREADY_SLV [P_SLAVES_NUM : 0]	IN	AHB transfer done signals from all slaves in the testbench
HMASTER [clog2(P_MASTERS_NUM-1)-1 : 0]	OUT	Number of the granted AHB master
HMASTER_VALID	OUT	High state indicates that HMASTER signal is valid

Table 11. Interface signals of AHBArbiter module

The P_MASTERS_NUM and P_SLAVES_NUM define number of masters and slaves in the testbench.
The HBUSREQ lines are used to request access to the bus. The AHB arbiter module selects master

with the highest priority and responds with active state on the HGRANT lines. Only one bit of the HGRANT bus is active at a time granting access to the bus to the highest priority master.

The AHB arbiter must be aware what type of transfer is currently performed on the bus. It monitors constantly HTRANS and HBURST lines. It must also detect end of current transfer to grant new master at appropriate time. This is why the arbiter monitors HREADY signals. The HMASTER signal is used to switch multiplexing units on the bus. Once a master is granted the HMASTER signal is updated and indicates number of currently granted master. All the multiplexing units can switch and redirect the AHB traffic to the granted master. The multiplexing units are described in the next chapters.

10. AHBMasterToSlave Module

The AHBMasterToSlave module is a multiplexing unit that takes outputs from all AHB masters and passes outputs of currently granted master to connected AHB slave. Each AHB slave has separate AHBMasterToSlave module. It allows to parallelize transfers from different masters to different slaves. The interface of AHBMasterToSlave module is shown in the tables below.

Parameter	Value	Description
P_MASTERS_NUM	2	Number of AHB masters in the testbench

Table 12. Parameters of AHBMasterToSlave module

Port	Direction	Description
HCLK	IN	Clock
HRESETn	IN	Reset
HSEL	IN	AHB slave select signal
HTRANS_IN [1 : 0]	IN	AHB transfer type from currently granted master
HMASTER [clog2(P_MASTERS_NUM-1)-1 : 0]	IN	Number of the granted AHB master
HMASTER_VALID	IN	High state indicates that HMASTER signal is valid
HREADY	IN	AHB transfer done from slave
Outputs of all AHB Masters		
HTRANS_ARR [(P_MASTERS_NUM * 2)-1 : 0]	IN	AHB transfer type signals from all masters
HSIZE_ARR [(P_MASTERS_NUM * 3)-1 : 0]	IN	AHB transfer size signals from all masters
HBURST_ARR [(P_MASTERS_NUM * 3)-1 : 0]	IN	AHB burst type signals from all masters
HPROT_ARR [(P_MASTERS_NUM * 4)-1 : 0]	IN	AHB protection control signals from all masters
HADDR_ARR [(P_MASTERS_NUM * 32)-1 : 0]	IN	AHB address buses from all masters
HWDATA_ARR	IN	AHB write data buses from all masters

[(P_MASTERS_NUM * 32)-1 : 0]		
HWRITE_ARR [P_MASTERS_NUM-1 : 0]	IN	AHB transfer direction signals from all masters
Outputs of currently granted master		
HTRANS [1 : 0]	OUT	AHB transfer type signal for selected slave
HSIZE [2 : 0]	OUT	AHB transfer size signal for selected slave
HBURST [2 : 0]	OUT	AHB burst type signal for selected slave
HPROT [3 : 0]	OUT	AHB protection control signal for selected slave
HADDR [31 : 0]	OUT	AHB address bus signal for selected slave
HWDATA [31 : 0]	OUT	AHB write data bus signal for selected slave
HWRITE	OUT	AHB transfer direction signal for selected slave

Table 13. Interface signals of AHBMasterToSlave module

The P_MASTERS_NUM parameter defines number of AHB masters in the testbench. The AHBMasterToSlave module monitors the following lines to detect when current transfer is completed and new master is granted:

- HSEL – select signal for slave connected to the AHBMasterToSlave module
- HTRANS_IN – transfer type from currently granted master
- HMASTER – Number of the currently granted AHB master
- HMASTER_VALID – HMASTER valid signal
- HREADY – transfer done signal from slave connected to the AHBMasterToSlave module

Once the new master is granted the AHBMasterToSlave module passes its outputs to connected AHB slave. Outputs from all AHB masters are organized as the following vectors:

- HTRANS_ARR
- HSIZE_ARR
- HBURST_ARR
- HPROT_ARR
- HADDR_ARR
- HWDATA_ARR
- HWRITE_ARR

Outputs from AHB master zero should occupy the least significant bits of the vectors. Outputs of the master with the highest number should occupy most significant bits of the vectors. The AHBMasterToSlave module uses HMASTER lines for selecting appropriate bit or slice from the above vectors. Finally, outputs from currently granted master appear on the following signals:

- HTRANS
- HSIZE
- HBURST
- HPROT
- HADDR

- HWDATA
- HWRITE

The signals are connected to one of AHB slaves available in the testbench. Adding a slave to the testbench requires also adding dedicated AHBMasterToSlave module connected with this slave.

11. AHBSlaveToMaster Module

The AHBSlaveToMaster module is a multiplexing unit that takes outputs of all AHB slaves and passes outputs of selected slave to granted master. Each AHB master has dedicated AHBSlaveToMaster module. The interface of AHBSlaveToMaster module is shown in the tables below.

Parameter	Value	Description
P_MASTERS_NUM	2	Number of AHB masters in the testbench
P_SLAVES_NUM	2	Number of AHB slaves in the testbench excluding dummy slave.
P_MASTER_ADDR	0	Address of the master on arbiter's HMASTER bus.

Table 14. Parameters of AHBSlaveToMaster module

Port	Direction	Description
HCLK	IN	Clock
HRESETn	IN	Reset
HTRANS [1 : 0]	IN	AHB transfer type signal
HMASTER [clog2(P_MASTERS_NUM-1)-1 : 0]	IN	AHB master number
HMASTER_VALID	IN	High state indicates that HMASTER signal is valid
HBUSREQ [P_MASTERS_NUM - 1 : 0]	IN	AHB bus requests signals from all masters in the testbench
HGRANT [P_MASTERS_NUM - 1 : 0]	IN	AHB bus grant signals for all masters in the testbench
HSEL [P_SLAVES_NUM-1 : 0]	IN	AHB slave select signals for all slaves in the testbench
HREADY_SLV [P_SLAVES_NUM : 0]	IN	AHB transfer done signals from all slaves in the testbench
HRDATA_ARR [((P_SLAVES_NUM+1) * 32)-1 : 0]	IN	AHB read data buses from all slaves
HRESP_ARR [((P_SLAVES_NUM+1) * 2)-1 : 0]	IN	AHB transfer response signals from all slaves
HREADY_M	OUT	AHB transfer done signal for connected master

HRDATA_M	OUT	AHB read data bus for connected master
HRESP_M	OUT	AHB transfer response signal for connected master

Table 15. Interface signals of AHBSlaveToMaster module

The P_MASTERS_NUM parameter defines number of AHB masters in the testbench. The P_SLAVES_NUM parameter defines number of AHB slaves in the testbench excluding dummy slave. The P_MASTER_ADDR defines number of the master that is connected to the AHBSlaveToMaster module. This parameter is compared with HMASTER signal. If state of the HMASTER signal is equal to P_MASTER_ADDR, the AHBSlaveToMaster module assumes that master connected to its outputs is currently granted. In order to operate correctly the AHBSlaveToMaster module must monitor the bus to get the following information :

- Which AHB master is currently granted
- Which AHB slave is currently selected
- What transfer type is currently performed on the bus

The AHBSlaveToMaster module monitors the following AHB signals to get the required information about performed transaction on the bus:

- HBUSREQ
- HGRANT
- HSEL
- HTRANS
- HMASTER
- HMASTER_VALID

If the AHBSlaveToMaster module detects that its master is currently granted, it checks what slave is accessed by the master and passes outputs from the slave to the master. The following signals from all slaves are multiplexed by AHBSlaveToMaster unit:

- HREADY_SLV - vector of HREADY signals from all slaves
- HRDATA_ARR – vector of HRDATA signals from all slaves
- HRESP_ARR – vector of HRESP signals from all slaves

Outputs of the AHB dummy slave should occupy the last significant bits of the vectors. Next, should be placed the outputs of slave that is selected by HSEL[0] line and in turn slave selected by HSEL[1], ... HSEL[P_SLAVES_NUM – 1]. The same order must be preserved on HSEL and HREADY_SLV, HRDATA_ARR, HRESP_ARR signals because HSEL signal is used to select appropriate slice from the vectors for given slave.
The AHBSlaveToMaster module detects that its master is granted and checks which slave is addressed by monitoring HSEL lines. The outputs of the addressed slave are passed to the master using the following signals connected directly to the master:

- HREADY_M
- HRDATA_M
- HRESP_M

Each slave in the testbench should be connected to all AHBSlaveToMaster units. Adding a master to the testbech requires adding dedicated AHBSlaveToMaster units connected with this master.

12. AHBMasterToArbiter Module

The AHBMasterToArbiter module is a multiplexing unit that takes outputs of all AHB masters and passes outputs of currently granted master to the AHB arbiter. This is asynchronous multiplexing unit. The interface of the AHBMasterToArbiter module is shown in the tables below:

Parameter	Value	Description
P_MASTERS_NUM	2	Number of AHB masters in the testbench

Table 16. Parameters of AHBMasterToArbiter module

Port	Direction	Description
HMASTER [clog2(P_MASTERS_NUM-1)-1 : 0]	IN	Number of the granted AHB master
HTRANS_ARR [(P_MASTERS_NUM * 2) - 1 : 0]	IN	AHB transfer type signals from all masters
HBURST_ARR [(P_MASTERS_NUM * 3) - 1 : 0]	IN	AHB burst type signals from all masters
HADDR_ARR [(P_MASTERS_NUM * 32) - 1 : 0]	IN	AHB address buses from all masters
HTRANS [1 : 0]	OUT	AHB transfer type signal from selected master
HBURST [2 : 0]	OUT	AHB burst type signal from selected master
HADDR [31 : 0]	OUT	AHB address bus from selected master

Table 17. Interface signals of AHBMasterToArbiter module

The P_MASTERS_NUM parameter defines number of masters in the testbench. The AHB arbiter module needs to monitor AHB bus to detect if currently performed transaction is about to complete and bus can be granted to another master. The AHB arbiter needs to monitor only the following lines to get required information about performed transaction:

- HTRANS
- HBURST
- HADDR

The AHBMasterToArbiter module must take the above listed outputs from all masters and pass to the arbiter outputs from currently granted master. The AHB arbiter will analyze state on the lines and grant bus to new master when currently granted master is competing the transfer. The above listed signals are arranged in the following vectors and connected to inputs of AHBMasterToArbiter module:

- HTRANS_ARR
- HBURST_ARR
- HADDR_ARR

Signals from master number zero occupy the last significant bits of the vectors. Next, should be placed signals from the master number one and so on. The HMASTER signal is used to select appropriate slice from the vectors and pass the selected lines to the arbiter. Adding a master to the

testbench requires updating the P_MASTERS_NUM parameters and concatenating its HTRANS, HBURST, HADDR outputs with the same outputs from other masters. The concatenated outputs from all masters should be assigned to HTRANS_ARR, HBURST_ARR, HADDR_ARR inputs of AHBMasterToArbiter module.

13. License

14. Trademarks