AHB generator

v1. 0

20 September 2004

Contents

1 Introduction

An AHB system is made of masters slaves and interconnections. A general approach to include all possible "muxed" implementation of multi layered AHB systems and arbitrated AHB ones can be thought as an acyclic graph where every source node is a master, every destination node is a slave and every internal node is an arbiter; there must be one and only one arc exiting a master and one or more entering a slave (single slave verus multi-slave or arbitrated slave); an arbiter can have as many input and output connections as needed. A bridge is a special node that collapses one or more slave nodes and a master node in a new "complex" node.

An AHB system definition is made by a representation of the graph topology.

2 Elementary blocks

- AHB master

It's a master with the following behaviour:

It supports default master granting, 32-bits bus size and 32-bits transfers.

It can sustain burst accesses of length 1(single), 4/8/16 incr/wrap and incremental of length 1 up to 64k.

On incremental bursts of any length 1k address boundary check is performed: in case of page cross the burst is cut in slices with successive dma requests.

In case of slave split and retry response or arbiter early deasserting request new bus requests are attempted after one or more turnaround cycles.

A master can generate locked or non-locked transfers, according to a programmable bit in a configuration register.

In case a transfer is splitted in more chuncks, the master deasserts bus request (and lock if it's programmed so) and after at least one cycle asserts it/them again.

In case of error the transfer is broken, no recovery dma transfer is attempted.

In any case an end-of-transfer is risen when the complete burst has been transmitted or an error has been detected.

The bursts, once started, continues until end or a special situation occurrs: retry/split from slave, grant deasserted from arbiter or incremental burst crossing 1k addressing boundary.

Whenever a fixed length burst is interrupted for any reason (retry, split, grant deasserted) a new transfer has to be issued: the length can be equal or less than the prefixed, so the original type of transfer is substituted by an incremental burst of unspecified length.
In the case of a wrap type original transfer, a check on wrapping condition is done during the new incremental transfer: the transfer of remaining words will be accordingly split in 2 transfers in order to emulate the requested address wrapping.
For example a wrap4 starting from 4: 4-8-C-0, interrupted after 4, will be issued as:
transfer1-> wrap4 with addr 4;
transfer2-> incr with addresses 8-C;
transfer3-> incr with addr 0

Inside the master a programmable fifo is present to improve performances in both read and write: the length and the flags of "almost full" and "almost empty" are programmable.
N.B.: fifo length MUST be at least 2 locations, and fifo almost full flag MUST be greater or equal than fifo almost empty flag.

Fifo is single-clocked, 1:1 ahb vs master clock frequency ratio is now supported. For different ratioes master block doesn't need any change, instead the master wrapper block has to be changed (in principle could be sufficient to correctly shape the handshake protocol signals "ask" and "get").

The master cannot be directly attached to another fifo interface or "volatile" devices, where it's necessary to read only the exact number of data words (during a write transfer the master generally prefetches more data than strictly required to speed up transactions).

- AHB master wrapper

It couples with a master. It is a memory with a programmable latency and burst capability, both in read and in write direction.
In case the burst capability is turned on, only the first latency access is payed, then a cycly-by-cycle data transfer can be sustained.
If no burst capability is required, any access will take "read/write" latency cycles.

The handshake mechanism implemented could be customized to "asyncronuos" interface, i.e. cores containing memory resources accessible with different time constraints: in that case fixed parameters are not useful, but core side has to generate the ack response with its internal timing.

- AHB slave

It supports same kind of transfers as master.
Protection is implemented as far as bufferable/not bufferable is concerned; other kinds of protection are not treated.
It's possible to choose the behaviour of slave between: 'wait', 'retry' and 'split', corresponding to the 'non ok' answers of ahb transactions.

The slave with ok/wait only answer is the simplest: no buffering is provided.
The transaction is stopped every data, and the new requesting master is waited until the previous transfer is complete.

The others slaves, ok/retry and ok/split types, have an internal fifo to support more efficiently burst transfers.

A retry or split response is issued whenever a higher priority device (for example an associated master) starts its activity or the slave is still busy on previous transfer.
Retry/split/wait slave response follows this scheme: retry or split is issued when the transfer cannot continue or cannot start, wait is the default response in all other cases, when the transfer can continue but not at full speed.

Addresses not word aligned or out of slave configuration address space generate an error response if the slave has been selected by decoder.

The fifo, if present, has programmable length and almost full and almost empty flags, with the same restrictions already seen for masters.

Data width /= 32 bits are not supported.

N.B.: Version 1.0 supports only 'wait' type slaves.

- AHB arbiter, decoder, master mux, slave mux, default slave

AHB arbiter includes all the blocks above, allowing an easy ahb system structure definition and integration.

Arbiter block.
It is the bus request bus grant arbitration block. It can support up to 16 masters.
The implemented algorithms are: fixed priority, modified round robin, random priority, all on a cycle by cycle arbitration.

The cycle by cycle arbitration can be enabled or disabled: in the first case every cycle a new master can be granted (and the previous master grant will be deasserted),
in the latter the grant is mantained until the request is high (as if it were locked by master).

In case no requests are pending, a default master, selected during system configuration, is granted.
In case all masters are currently splitted, a dummy master is granted.

Pipelined grant is supported, with overlay of old master data phase and new granted master address phase.

Decoder block.
It is a simple block that detects valid accesses, i.e. accesses inside slave address spaces mapped during system generation.

When an out of range address is detected the default slave is selected.
It supports up to 16 masters.

Master mux block.
It collects all controls and write data from masters and selects the bus owner.

Slave mux block.
It selects the addressed slave response control and read data among all slaves.
It supports up to 16 slaves including default slave.

Default slave
Whenever a wrong address is issued it answers with the typical 2-cycle error response.

- AHB/AHB bridge

It manages bridging between one or more slaves on AHB-1 bus to an AHB-2 bus master, i.e. it arbiters between several slaves the one that should start master access on the other side of the bridge.

The arbitration policies are fixed, round robin or pseudo-random, and all the requesting transactions that aren't currently selected by internal arbiter are waited (no retry-split capability, because multi slave interface allows the connection of masters from up to 16 AHB-lite layers to the same slave).
The bridge only function is to repeat the ahb-1 type of transfer to ahb-2, but due to impredictable slave behaviour on AHB-1, like change of master, early grant deasserting, errors, the master side of bridge generates always single accesses.

N.B.: Version 1.0 doesn't support ahb/ahb bridge.

- AHB/APB bridge

It manages multi AHB slave to single APB master bridging, i.e. it arbiters between several slave the one that should start master access on the APB bus.

N.B.: Version 1.0 doesn't support ahb/apb bridge.

- APB slave

It's a simple apb memory-like interface.

N.B.: Version 1.0 doesn't support apb slave.

3 Configuration

The AHB system configuration can be read by file, specified in command line (or using "AHB_generate.conf" as default file name) and can be updated, modified, cancelled, checked thanks to a simple Perl/TK GUI.
Using the "-nogui" option only the generation process can be invoked by command line: it will use the specified file as input and will check consistance and finally will generate files or issue errors.

GUI is made by the main menu in which is possible to:

- Reset configuration file (all information entered via GUI or file read is deleted)
- Read configuration file
- Save configuration file
- Check the current configuration and, if it's successful, generate all the .vhd files
- Done to exit the program if last check was passed

It's possible to update present configuration adding the following blocks, each with some parameters to better define the behaviour of the element:

- add master inserts a new master, with the ID automatically incremented with respect to the last inserted (it's not possible to choose ID): the first is number '0'.
  It's allowed to change name: default is ahb_mst + ID
  fifo parameters: length (>=2, power of 2), flags of half empty and full (full>=empty)
  number of address bits: it represents the dimension of master internal memory in term of addressing bits
  Burst capability (both write and read): possibility to concatenate subsequent accesses in burst
  Burst latency (both write and read): number of wait cycles before first data is ready (or every data in case of no burstability).
  UUT base address is for testability purpose: the master will perform some accesses all starting from that base address.

- add slave inserts a new slave, with the ID automatically incremented with respect to the last inserted (it's not possible to choose ID): the first is number '0'.
  It's allowed to change name: default is ahb_slv + ID, and type of answer: 'wait'/'retry'/'split' (v1.0 supports only 'wait').
  Then a new window appears with:
  fifo parameters: length (>=2, power of 2), flags of half empty and full (full>=empty).

Configuration of slave address space: highest and lowest address before and after remap.
number of address bits: it represents the dimension of slave internal memory in term of addressing bits (4 bits means 16 locations 32 bits wide)
Burst capability (both write and read): possibility to concatenate subsequent accesses in burst.
Burst latency (both write and read): number of wait cycles before first data is ready (or every data in case of no burstability).
N.B.: fifo parameters are meaninful only for 'retry' or 'split' slaves, not for 'wait' slaves.
If the slave is used as a part of any bridge also burst capabilities and latencies are meaningles.

- add per. slave inserts a new peripheral slave, i.e. an APB slave, with the ID automatically incremented with respect to the last inserted (it's not possible to choose ID): the first is number '0'.
  It's allowed to change name: default is apb_slv + ID
  Configuration of APB slave address space: highest and lowest address before and after remap.
  number of address bits: it represents the dimension of APB slave internal memory in term of addressing bits (4 bits means 16 locations 32 bits wide)

- add arbiter inserts a new arbiter, with the ID automatically incremented with respect to the last inserted (it's not possible to choose ID): the first is number '0'.
  It's allowed to change name: default is ahb_arb + ID
  Default master is the master granted when none is requesting bus.
  Master list is a spaces separated list of numbers representing ID of masters to be arbitrated.
  Slave list is a spaces separated list of numbers representing ID of slaves to be accessed by the masters declared in master list.
  Arbitration type can be one of fixed, round-robin, pseudo-random algoritms and the same with no cycle by cycle rearbitration (i.e. a grant is not moved from one master to another until the first deasserts request, it's a kind of locking).
  N.B.: in the master list the master using slot number 0 is the rightmost, so in fixed priority the highest priority is the rightmost.

- add ahb bridge inserts a new ahb/ahb bridge, with the ID automatically incremented with respect to the last inserted (it's not possible to choose ID): the first is number '0'.
  It's allowed to change name: default is ahb_brg + ID
  Default slave is simply the slave driving the mux when no access is ongoing.
  Slave list is a spaces separated list of numbers representing ID of slaves to be translated on the second AHB bus.
  Master is the ID of the bridge master on the second AHB bus.

There is no limitation in the number of bridges as in the number of AHB busses.
Arbitration type can be one of fixed, round-robin, pseudo-random algoritms
N.B.: in the slave list the slave using slot number 0 is the rightmost, so this is also the highest priority slave in fixed priority.

- add apb bridge inserts a new apb bridge (in principle it's possible to have more than one apb bridge, even if normally there is only one), with the ID automatically incremented with respect to the last inserted (it's not possible to choose ID): the first is number '0'.
  It's allowed to change name: default is apb_brg + ID.
  Number of addr bits represents the number of address bits decoded by the bridge.
  AHB slave is the slave part that translates accesses from AHB to APB bus.
  APB slave list is a spaces separated list of numbers representing ID of peripheral slaves to be mapped on the APB bus.
  It's normal to declare only one APB bridge and a list of consecutive peripherals, starting from 0.

  N.B.:
  GUI list of masters and slaves must be **space-separated**;
  CONF file list of master and slaves must be **comma-separated**.

4 Matrix generation

When the configuration is ready is necessary to parse it through the check/generate button, that does the following checks:

1) It must exist at least one master, one slave and one arbiter.
2) Every master must be connected to one and only one arbiter.
3) Every slave must be connected to one (and only one) arbiter OR to one (and only one) bridge.
4) Master of ahb bridges should not be declared as real masters, i.e. bridge master number should be higher than the highest master number.
5) If there is an APB bridge there must exist at least a peripheral slave and viceversa

The generate will produce:

- ahb_configure.vhd: a configuration file with address of AHB and APB slaves, before and after remap, and arbiter and bridge matrices;
- ahb_matrix.vhd: a file which instantiate all arbiters and bridges (the AHB complete matrix) with inputs and outputs for the connections with all masters and slaves.

- ahb_system.vhd: the whole ahb system, with all masters and slaves plus the previous components (synthetizable components)
- ahb_tb.vhd: a testbench with the whole ahb system plus a clock and reset generator and one UUT_generator for each master declared (NON synthetizable)

It's strongly suggested in order to get a first time 'pass' to checks to make a scheme of the ahb system under investigation (it reduces connection errors), an insertion of components by GUI (it produces the correct syntax), a SAVE CONF and after all a manual editing of conf file to tune the parameters.

No configuration is saved until a 'SAVE CONF' command is issued.

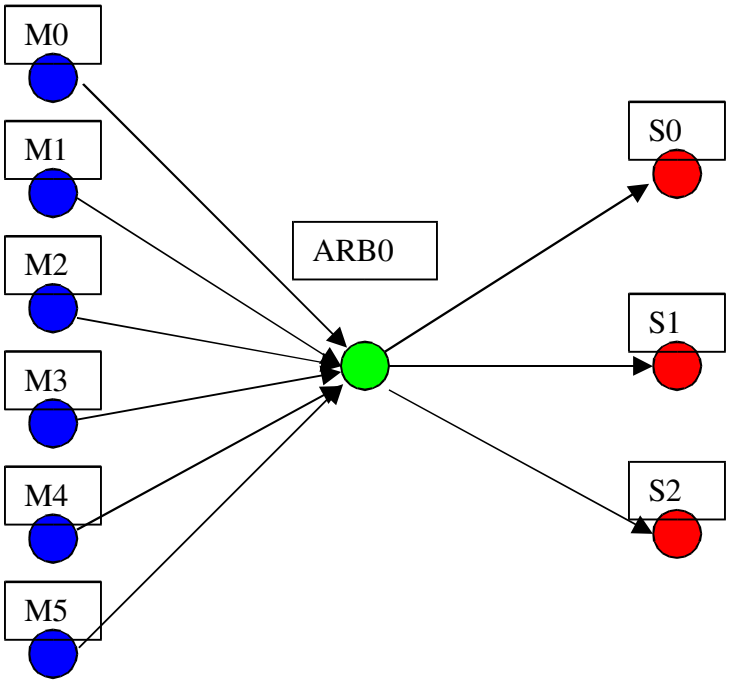
5 AHB system for verification

Last file created, ahb_tb.vhd is a simple testbench including stimulators for masters, with can be programmed to start a sequence of accesses each from a specified base address.
If the user wants to test the functionality of the system plus the connections, he can in a first moment declare a different base address for any master UUT_stimulator and obtain a sequence of accesses to non overlapping memory areas.
In fact the UUT stimuli are patterns that occupy only 32 locations of 32 bits each (the pattern is a write followed by a read of all types of hburst: single, incr,wrap4,incr4,wrap8,incr8,wrap16,incr16 repeated twice and ended by a single data transfer).
If this precaution is taken, every master will present a log file of transactions equal (apart from time occurrencies) to the 'm###.ref file'.
Finally a script file, vdiff.pl, which requires a reference file and the current log file, will check for its correctness.

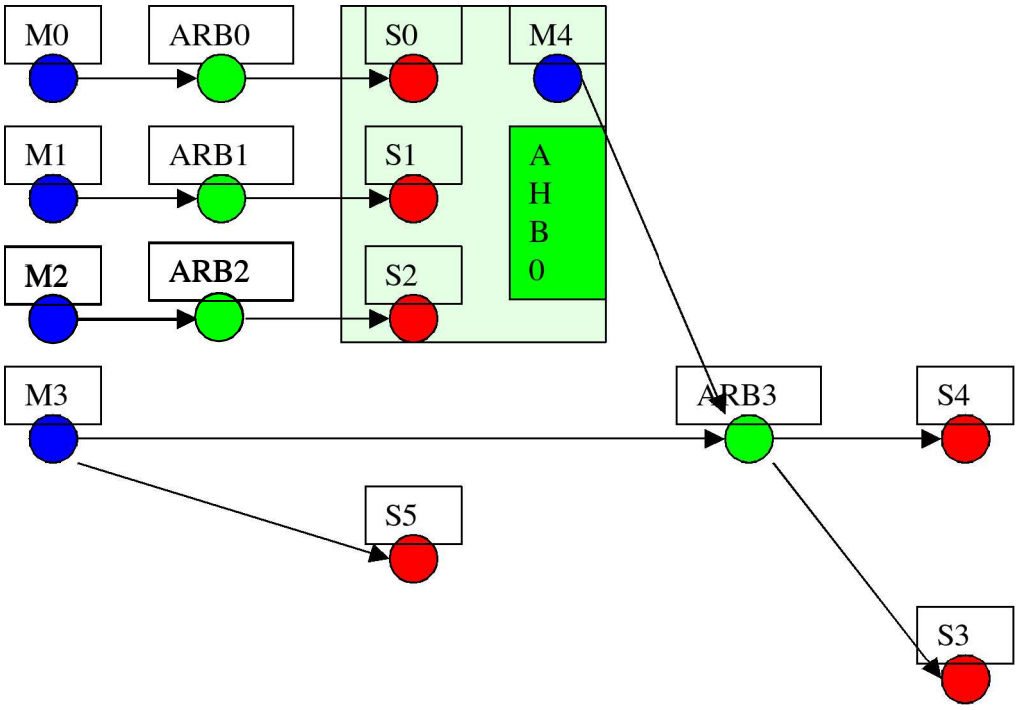
6 Examples

Here are some examples of systems:

ahb_single_arb.conf is a 6 masters, 1 arbiter, 3 slaves



ahb_mult_arb_ahbahb1.conf is a 4 masters, 4 arbiters, 1 ahb bridge and 3 slaves



ahb_complete1.conf is a 4 masters, 4 arbiters, 1 ahb bridge, 1 apb bridge, 2 slaves and 2 apb slaves.