GEM_AMBA: AMBA AXI, AHB, and APB Generator

Version 0 Revision 3

July 10, 2021 (March 1, 2016)

Ando Ki, Ph.D.
andoki@gmail.com / adki@future-ds.com

Copyright notice

License notice

Abstract

This document describes a set of programs and API that generate AMBA bus systems, which include AMBA AXI, AHB, APB, and bus bridges.

Table of contents

1 Introduction

AMBA (Advanced Microcontroller Bus Architecture) bus family includes several protocols as follows:

- APB: Advanced Peripheral Bus - Minimal gate count for peripherals
- ASB: Advanced System Bus - the main system bus, but no longer supported
- AHB: Advanced High-Performance Bus - the main system bus in microcontroller usage for a few bus masters and slaves in a SoC's
- AXI: Advanced eXtensible Interface – the high performance system bus to connect many masters and slaves in a complex SoC's
- AXI-Stream:
- ACE: AXI Coherency Extensions – an extension to support caches in a complex SoC's
- CHI: Coherent Hub Interface for high data rate applications

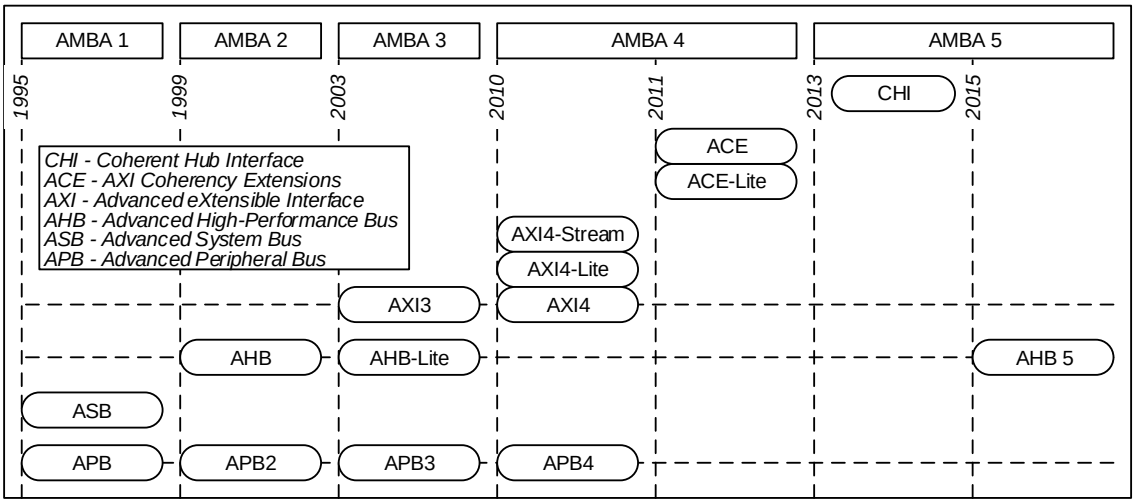

Figure 1: Evolution of AMBA standards

This package provides you with automatic generation of following AMBA Buses.

- APB with AHB bridge supporting APB3 and APB4
- APB with AXI bus bridges supporting APB3 and APB4
- AHB-Lite for single-master
- AHB for multi-master
- AXI for multi-master including AXI3 and AXI4

1.1 Getting started

In order to generate AXI with 3 masters and 4 slaves, do as follows.

```
$ gen_amba_axi --master=3 --slave=4 --output=amba_axi_m3s4.v
```

In order to generate AXI-to-APB for 3 ports, do as follows.

```
$ gen_amba_apb --axi --slave=3 --output=axi_to_apb_s3.v
```

In order to generate AHB with 3 masters and 4 slaves, do as follows.

```
$ gen_amba_ahb --master=3 --slave=4 --output=amba_ahb_m3s4.v
```

In order to generate AHB-to-APB for 3 ports, do as follows.

```
$ gen_amba_apb --ahb --slave=3 --output=ahb_to_apb_s3.v
```

1.2 API convention

In addition to generating program, this package can be used as a library that contains various API (Application Programming Interface). Each API returns 0 when completes successfully. Otherwise, it returns non-zero number and error number is stored in internal variable.

2 AMBA APB

AMBA APB is designed to support low-speed peripherals. APB bus devices are slave and do not initiate bus request. APB is synchronous bus and all bus operations are based on the rising edge of PCLK signal. APB only supports a single access, i.e., APB doesn't support burst transaction.

```
$ gen_amba_apb [options]
```

Options:

- --axi|ahb: generates AXI-to-APB or AHB-to-APB.
- --slave=num: specify number of APB ports
  - It should be 2 or higher. If not given, 2 by default.
- --moduleule=string: specify module name
  - ‘axi_to_apb_sX’ or ‘ahb_to_apb_sY’ if not given, where ‘X’ and ‘Y’ is the number of APB ports specified by corresponding option.
- --prefix=string: specify prefix of sub-module name
  - If not given, no prefix is used.
- --output=string: specify output file name.
  - If no given, standard output is used by default.
- -h: help

1.3 AHB-to-APB

Following command generates ‘ahb_to_apb_s2.v’ file that is AHB-to-APB bus bridge with two APB ports.

```
$ gen_amba_apb --ahb --slave=2 --output=ahb_to_apb_s2.v
```

1.4 AXI-to-APB

Following command generates ‘axi_to_apb_s2.v’ file that is AXI-to-APB bus bridge with two APB ports.

```
$ gen_amba_apb --axi --slave=2 --output=axi_to_apb_s2.v
```

1.5 API

1.5.1 gen_ahb2apb

Function prototype:

```
int gen_ahb2apb ( unsigned int num, char* module, char* prefix, FILE* fo );
```

Argument:

- num: number of APB ports
  - It should be equal to or larger than 2.
- module: pointer to the string to be used as name of module
  - It should not be null pointer.
- prefix: pointer to the string to be used as prefix of all sub-module
  - It should not be null pointer and should contain “\0” if prefix is not used.
- fo: file pointer

Return:

- 0: successful completion
- !=0: on failure

Synopsis: It generates AHB-to-APB containing AHB-to-APB bridge and APB decoder and MUX. By default, ‘ahb_to_apb_sX’ module, where ‘X’ is the number of APB ports.

1.5.2 gen_axi2apb

Function prototype:

```
int gen_axi2apb ( unsigned int num, char* module, char* prefix, FILE* fo );
```

Argument:

- num: number of APB ports
  - It should be equal to or larger than 2.
- module: pointer to the string to be used as name of module
  - It should not be null pointer.

- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AXI-to-APB containing AHB-to-APB bridge and APB decoder and MUX. By default, ‘axi_to_apb_sX’ module, where ‘X’ is the number of APB ports.

1.5.3 gen_ahb2apb_bridge

Function prototype:

```
int gen_ahb2apb_bridge( char *prefix, FILE* fo );
```

Argument:
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates bus interface converting AHB to APB protocol. By default, it generates ‘ahb_to_apb_bridge’ module.

1.5.4 gen_axi2apb_bridge

Function prototype:

```
int gen_axi2apb_bridge( char *prefix, FILE* fo );
```

Argument:
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates bus interface converting AXI to APB protocol. By default, it generates ‘axi_to_apb_bridge’ module.

1.5.5 gen_apb_amba

Function prototype:

```
int gen_apb_amba(unsigned int numS, char *module, char *prefix, int bridge, FILE* fo);
```

Argument:

- ✧ numS: number of APB ports
  - o It should be equal to or larger than 2.
- ✧ module: pointer to the string to be used as name of module
  - o It should not be null pointer.
  - o ‘amba_apb_sX’ by default.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ bridge: If it is 1, the resultant module will be instantiated in upper module. This means that ‘gen_ahb2apb()’ or ‘gen_axi2apb()’ is called before.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It calls ‘gen_apb_amba_core()’ that generates AMBA APB bus including decoder and MUX.

1.5.6 gen_apb_amba_core

Function prototype:

```
int gen_apb_amba_core ( unsigned int num
                      , char *module
                      , char* prefix
                      , int bridge
                      , FILE* fo );
```

Argument:

- ✧ numS: number of APB ports
  - o It should be equal to or larger than 2.
- ✧ module: pointer to the string to be used as name of module
  - o It should not be null pointer.
  - o ‘amba_apb_sX’ by default.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.

- ✧ bridge: If it is 1, the resultant module will be instantiated in upper module. This means that ‘gen_ahb2apb()’ or ‘gen_axi2apb()’ is called before.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA APB bus including decoder and MUX.

1.5.7 gen_apb_decoder

Function prototype:

```
int gen_apb_decoder   ( unsigned int num, char* prefix, FILE* fo );
```

Argument:
- ✧ num: number of APB ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA APB decoder.

1.5.8 gen_apb_mux

Function prototype:

```
int gen_apb_mux ( unsigned int num, char* prefix, FILE* fo );
```

Argument:
- ✧ num: number of APB ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA APB MUX.

3 AMBA AHB

AMBA AHB is designed to support high-performance blocks. AHB is synchronous bus and all bus operations are based on the rising edge of HCLK signal.

```
$ gen_amba_ahb [options]
```

Options:

- --lite: force to generate AMBA AHB-Lite
- --master=num: specify number of AHB master ports[1]
  - It should be 2 or higher. If not given, 2 by default.
- --slave=num: specify number of AHB slave ports[2]
  - It should be 2 or higher. If not given, 2 by default.
- --module=string: specify module name
  - 'amba_ahb_mXsY' if not given, where 'X' and 'Y' is the number of master and slave ports specified by corresponding option.
- --prefix=string: specify prefix of sub-module name
  - If not given, no prefix is used.
- --output=string: specify output file name.
  - If no given, standard output is used by default.
- -h: help

1.6 AHB

Following command generates 'amba_ahb_m3s4.v' file.

```
$ gen_amba_ahb --master=3 --slave=4 --output=amba_ahb_m3s4.v
```

1.7 API

1.7.1 gen_ahb_amba

Function prototype:

```
int gen_ahb_amba( unsigned int numM
                , unsigned int numS
                , char *module
                , char *prefix
                , FILE *fo);
```

[1] Master port will be connected to AHB master module such as processor.
[2] Slave port will be connected to AHB slave module such as memory.

Argument:

- numM: number of AHB master ports
  - It should be equal to or larger than 2.
- numS: number of AHB slave ports
  - It should be equal to or larger than 2.
- module: pointer to the string to be used as name of module
  - It should not be null pointer.
- prefix: pointer to the string to be used as prefix of all sub-module
  - It should not be null pointer and should contain “\0” if prefix is not used.
- fo: file pointer

Return:

- 0: successful completion
- !=0: on failure

Synopsis: It calls ‘gen_ahb_amba_core()’ that generates AMBA AHB bus.

1.7.2 gen_ahb_amba_core

Function prototype:

```
int gen_ahb_amba_core( unsigned int numM
                     , unsigned int numS
                     , char *module
                     , char *prefix
                     , FILE *fo);
```

Argument:

- numM: number of AHB master ports
  - It should be equal to or larger than 2.
- numS: number of AHB slave ports
  - It should be equal to or larger than 2.
- module: pointer to the string to be used as name of module
  - It should not be null pointer.
- prefix: pointer to the string to be used as prefix of all sub-module
  - It should not be null pointer and should contain “\0” if prefix is not used.
- fo: file pointer

Return:

- 0: successful completion
- !=0: on failure

Synopsis: It generates AMBA AHB bus.

1.7.3 gen_ahb_arbiter

Function prototype:

```
int gen_ahb_arbiter  ( unsigned int numM
                     , unsigned int numS
                     , char* prefix
                     , FILE *fo);
```

Argument:

- ✧ numM: number of AHB master ports
  - o It should be equal to or larger than 2.
- ✧ numS: number of AHB slave ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AHB arbiter.

1.7.4 gen_ahb_m2s

Function prototype:

```
int gen_ahb_m2s      ( unsigned int numM, char* prefix, FILE *fo);
```

Argument:

- ✧ numM: number of AHB master ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AHB master to slave MUX.

1.7.5 gen_ahb_lite

Function prototype:

```
int gen_ahb_lite     ( unsigned int lite
```

```
                    , unsigned int numS
                    , char* module
                    , char* prefix
                    , FILE *fo);
```

Argument:

- ✧ lite: force to generate AHB-Lite only, when non zero.
- ✧ numM: number of AHB master ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AHB master to slave MUX.

1.7.6 gen_ahb_decoder

Function prototype:

```
int gen_ahb_decoder  ( unsigned int num, char* prefix, FILE *fo);
```

Argument:

- ✧ num: number of AHB slave ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AHB decoder.

1.7.7 gen_ahb_s2m

Function prototype:

```
int gen_ahb_s2m  ( unsigned int num, char* prefix, FILE *fo);
```

Argument:

- ✧ num: number of AHB slave ports

- It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AHB slave to master MUX.

1.7.8 gen_ahb_s2m

Function prototype:

```
int gen_ahb_default_slave( char* prefix, FILE *fo);
```

Argument:
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AHB default slave.

4 AMBA AXI

AMBA AHB is designed to support high-performance blocks. AXI is synchronous bus and all bus operations are based on the rising edge of ACLK signal.

```
$ gen_amba_axi [options]
```

Options:
- ✧ --master=num: specify number of AHB master ports[3]
  - It should be 2 or higher. If not given, 2 by default.
- ✧ --slave=num: specify number of AHB slave ports[4]
  - It should be 2 or higher. If not given, 2 by default.
- ✧ --module=string: specify module name

[3] Master port will be connected to AXI master module such as processor.
[4] Slave port will be connected to AXI slave module such as memory.

- ‘amba_ahb_mXsY’ if not given, where ‘X’ and ‘Y’ is the number of master and slave ports specified by corresponding option.

✧ --prefix=string: specify prefix of sub-module name
- If not given, no prefix is used.

✧ --output=string: specify output file name.
- If no given, standard output is used by default.

✧ --axi3: force to generate AMBA AXI3[5]
- If not given, AMBA AXI4 is generated.

✧ --verbose=num: verbose level
- If not given, verbose level is 0.

✧ --version: print version information

✧ --license: print license terms

✧ --help: print help message

Figure 2 shows a rough structure of AMBA AXI that supports multiple masters and slaves.

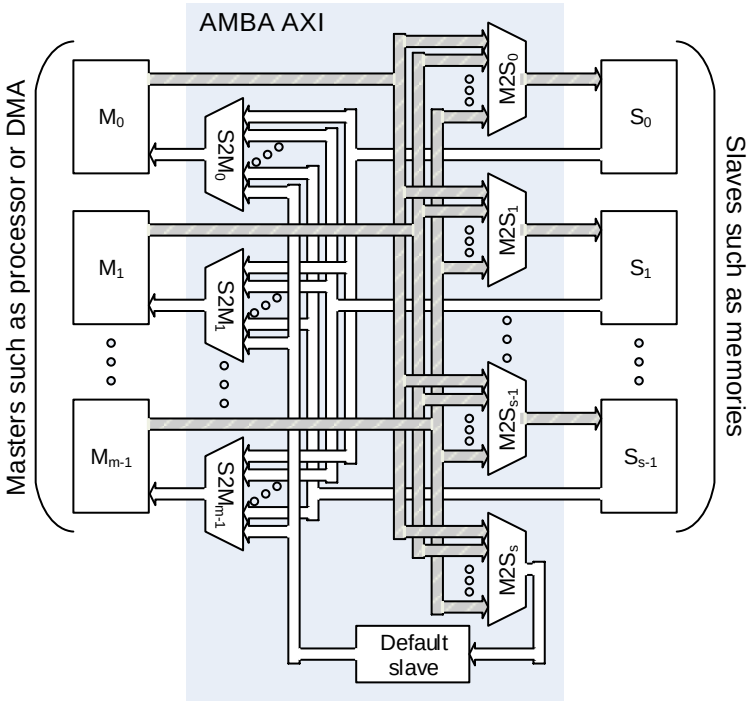

Figure 2: AMBA AXI internal structure

1.8 AXI

Following command generates ‘amba_axi_m2s3.v’ file, which support 2 masters and 2 slaves.

[5] AXI3: burst length up to 16, lock and exclusive, WID[…].

```
$ gen_amba_axi --master=2 --slav --output=amba_axi_m2s3.v
```

As shown in Code 1, parameters and macros are used to configure details of bus structure.

- Parameters
  - ✧ WIDTH_ID: bit-width of transaction id for AxID, WID, BID, RID.
  - ✧ WIDTH_AD: bit-width of address
  - ✧ WIDTH_DA: bit-width of data
  - ✧ SLAVE_ENn and ADDR_LENGn: starting address and its depth for each slave
- Macros
  - ✧ AMBA_AXI_ARUSER, AMBA_AXI_AWUSER, AMBA_AXI_WUSER, AMBA_AXI_BUSER, AMBA_AXI_RUSER: define this macro to use user-defined bus
  - ✧ AMBA_AXI_CACHE: define this to use AxCACHE port
  - ✧ AMBA_AXI_PROT: define this to use AxPROT
  - ✧ AMBA_AXI_QOS: define this to use AxQOS and AxREGION

Master port name starts with ‘Mn_’ prefix, where ‘n’ start from 0 to (number of master ports – 1). Slave port name starts with ‘Sn_’ prefix.

Code 1: AMBA AXI RTL code example

```
module amba_axi_m2s3
    #(parameter NUM_MASTER  = 2  // should not be changed
            , NUM_SLAVE   = 3  // should not be changed
            , WIDTH_CID   = clogb2(NUM_MASTER) // Channel ID width in bits
            , WIDTH_ID    = 4 // ID width in bits
            , WIDTH_AD    =32 // address width
            , WIDTH_DA    =32 // data width
            , WIDTH_DS    =(WIDTH_DA/8)  // data strobe width
            , WIDTH_SID   =(WIDTH_CID+WIDTH_ID)// ID for slave
            `ifdef AMBA_AXI_AWUSER
            , WIDTH_AWUSER= 1 // Write-address user path
            `endif
            `ifdef AMBA_AXI_WUSER
            , WIDTH_WUSER = 1 // Write-data user path
            `endif
            `ifdef AMBA_AXI_BUSER
            , WIDTH_BUSER = 1 // Write-response user path
            `endif
            `ifdef AMBA_AXI_ARUSER
            , WIDTH_ARUSER= 1 // read-address user path
            `endif
            `ifdef AMBA_AXI_RUSER
            , WIDTH_RUSER = 1 // read-data user path
            `endif
            , SLAVE_EN0   = 1 , ADDR_BASE0  =32'h00000000 , ADDR_LENGTH0=12
            , SLAVE_EN1   = 1 , ADDR_BASE1  =32'h00002000 , ADDR_LENGTH1=12
            , SLAVE_EN2   = 1 , ADDR_BASE2  =32'h00004000 , ADDR_LENGTH2=12
```

```
      )
(
      input   wire                  ARESETn
    , input   wire                  ACLK
    , input   wire  [WIDTH_ID-1:0]      M0_AWID
    , input   wire  [WIDTH_AD-1:0]      M0_AWADDR
    ….
    , output  wire  [WIDTH_SID-1:0]     S0_AWID
    , output  wire  [WIDTH_AD-1:0]      S0_AWADDR
    ….
);
codes are not shown
endmodule
```

1.9 API

1.9.1 gen_axi_amba

Function prototype:

```
int gen_axi_amba( unsigned int numM
                , unsigned int numS
                , char *module
                , char *prefix
                , int axi4
                , FILE *fo);
```

Argument:

- ✧ numM: number of AHB master ports
  - ○ It should be equal to or larger than 2.
- ✧ numS: number of AHB slave ports
  - ○ It should be equal to or larger than 2.
- ✧ module: pointer to the string to be used as name of module
  - ○ It should not be null pointer.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - ○ It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It calls ‘gen_axi_amba_core()’ that generates AMBA AXI bus.

1.9.2 gen_axi_amba_core

Function prototype:

```
int gen_axi_amba_core( unsigned int numM
                     , unsigned int numS
                     , char *module
                     , char *prefix
                     , int axi4
                     , FILE *fo);
```

Argument:

- ✧ numM: number of AHB master ports
  - o It should be equal to or larger than 2.
- ✧ numS: number of AHB slave ports
  - o It should be equal to or larger than 2.
- ✧ module: pointer to the string to be used as name of module
  - o It should not be null pointer.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AXI bus.

1.9.3 gen_axi_mtos

Function prototype:

```
int gen_axi_mtos( unsigned int num, char *prefix, int axi4, FILE *fo);
```

Argument:

- ✧ num: number of AHB master ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AXI master to slave MUX.

1.9.4 gen_axi_stom

Function prototype:

```
int gen_axi_stom( unsigned int num, char *prefix, FILE *fo);
```

Argument:

- ✧ num: number of AHB slave ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AXI slave to master MUX.

1.9.5 gen_axi_arbiter_mtos

Function prototype:

```
int gen_axi_arbiter_mtos( unsigned int num, char *prefix, FILE *fo);
```

Argument:

- ✧ num: number of AHB master ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module
  - o It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:

- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AXI master to slave arbiter.

1.9.6 gen_axi_arbiter_mtos

Function prototype:

```
int gen_axi_arbiter_stom( unsigned int num, char *prefix, FILE *fo);
```

Argument:

- ✧ num: number of AHB slave ports
  - o It should be equal to or larger than 2.
- ✧ prefix: pointer to the string to be used as prefix of all sub-module

- It should not be null pointer and should contain “\0” if prefix is not used.
- ✧ fo: file pointer

Return:
- ✧ 0: successful completion
- ✧ !=0: on failure

Synopsis: It generates AMBA AXI slave to master arbiter.

Wish list

- AXI4-Lite
- AHB5
- Bus bridges: ahb-to-ahb, axi-to-axi, axi-to-ahb, ahb-to-axi

References

[1] ARM, AMBA Specification, Chapter 5 AMBA APB, ARM IHI 0011A, 1999.

[2] ARM, AMBA 3 APB Protocol Specification, v1.0, ARM IHI 0024B, 2003-2004.

[3] ARM, AMBA APB Protocol Specification, v2.0, ARM IHI 0024C, 2003-2010.

[4] ARM, AMBA3 AHB-Lite Protocol Specification, v1.0, ARM IHI 0033A, 2006.

[5] ARM, AMBA 5 AHB Protocol Specification, AHB5, AHB-Lite, ARM IHI 0033B, 2015.

[6] ARM, AMBA AXI Protocol Specification, v1.0, ARM IHI 0022B, 2003-2004.

[7] ARM, AMBA AXI and ACE Protocol Specification, ARM IHI 0022E, 2003-2013.

The 2-Clause BSD License

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Revision history

- ☐ 2021.07.10: options changed; AMBA AXI4 feature updated.
- ☐ 2016.04.13: Started by Ando Ki (andoki@gmail.com or adki@future-ds.com).