SoC Designer
Version 9.5.0

APB Protocol Bundle
User Guide

Non-Confidential

Arm 101021_0905_00

SoC Designer
APB Protocol Bundle User Guide

Release Information

The following changes have been made to this document.

Issue	Date	Confidentiality	Change
0905-00	November 2017	Non-Confidential	Release with 9.5.0
0902-00	May 2017	Non-Confidential	Release with 9.2.0

Non-Confidential Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

http://www.arm.com

Table of Contents

1 Introduction

This is the user guide for the SoC Designer APB Protocol Bundle. This protocol bundle contains SoC Designer components and probes for the Arm® AMBA® 3 APB transaction protocol. For more information, refer to the *Arm AMBA APB Protocol Specification* (Arm IHI0024).

2 Requirements

The APB protocol bundle requires the following:

- SoC Designer v9.0.0 or later
- Compilation tools as described in the *SoC Designer Installation Guide* (Arm 100975)

3 APB Mixed-Version Compatibility Rules

APB transactors are described in the *Cycle Model Studio User Manual* (Arm 101108). If your system uses transactors for multiple versions of the APB protocol, be aware of the following restrictions:

- APB4_Master can only connect to APB4_Slave, not to APB2 or APB3 slaves.
- APB4_Slave can connect to any master except APB2 masters generated with SoC Designer Plus version 7.18.0 or later.
- APB3_Slave can not connect to APB2 masters generated with SoC Designer Plus version 7.18.0 or later
- APB2_Master connection to APB3_Slave is not recommended due to potential data integrity issues. APB3 supports a ready signal (PREADY) and a transfer failure signal (PSLVERR). If an APB3 slave port connected to an APB2 master port uses these signals to delay the transfer or indicate data error, data integrity issues could result. When both the APB3 slave and APB2 master are generated with SoC Designer Plus version 7.18.0 the connection is disallowed.

4 Protocol Bundle Contents

The APB protocol bundle contains the following components:

- APB Models - Generic APB components are included in this protocol bundle.
- APB Probes - Probes provide visibility into transactions between two components. APB-specific probes are included in this protocol bundle.
- APB Transaction Definition - APB transaction definition header files are included in this protocol bundle.

5 Models

Table 4-1 lists the APB components included in this protocol bundle.

Component	*Description*
APBIntCtrl	This is a generic APB interrupt controller.
APBMerger	APB 16x1 merger. Merges 16 APB slots to a single APB slot occupying a 64KB region.

Table 5-1 APB Components

APBIntCtrl

APBIntCtrl is an interrupt controller model. It provides a basic interrupt mechanism for Arm-based systems. Memory mapped registers program the device and are accessed over an APB bus.

Registers

Table 5-2 describes the registers for the APBIntCtrl controller model.

Name	*Offset*	*Description*
IRQ Status	0x0	Read-only register used to mask the interrupt input sources. Bit value of 1 indicates the interrupt is enabled, 0 indicates the interrupt is disabled. All interrupts are disabled on reset.
IRQ Source	0x4	Read-only register showing the interrupt sources prior to masking.
IRQ Enable	0x8	Write-only register used to enable specific interrupts. Bit value of 0 has no effect, value 1 enables the interrupt.
IRQ Clear	0xC	Write-only register used to clear specific interrupts. Bit value of 0 has no effect, value 1 disables the interrupt.
IRQ Soft	0x10	Write-only register to set/clear a programmed interrupt. Bit 1 set high generates a programmed interrupt, bit 1 set low clears the interrupt.
FIQ Status	0x20	Read-only 1-bit register indicating the FIQ status. Bit value 1 indicates FIQ is enabled, value 0 indicates it is disabled.
FIQ Source	0x24	Read-only register showing the FIQ source status prior to masking.
FIQ Enable	0x28	Write-only register to enable FIQ. 1 enables FIQ, 0 has no effect.
FIQ Clear	0x2C	Write-only register to disable FIQ. 1 clears FIQ. 0 has no effect.

Table 5-2 APBIntCtrl registers

Parameters

Table 5-3 describes the APBIntCtrl model parameters.

Name	*Description*
APB Base	Base address of the APB interrupt controller. Memory region size is fixed at 0x1000.

Table 5-3 APBIntCtrl parameters

Signal Interface

isrc Signal Slave

Interrupt sources are driven onto the isrc port of `APBIntCtrl` by the `CASI driveSignal()` method.

```
driveSignal( uint32_t value, uint32_t *extValue );
```

Here, *`value`* indicates the interrupt line and *`extValue`* indicates the interrupt signal value.

"fiq" and "irq" Signal Masters

These ports drive out the FIQ and IRQ interrupts.

```
driveSignal( uint32_t value, uint32_t * extValue);
```

Here, *`value`* holds the signal value. *`extValue`* is not used; it is set to NULL.

APBMerger

APBMerger is an APB 16x1 component that merges 16 APB slots to a single APB slot occupying a 64 KB region. Figure 5-2 shows the component ports: apbm signal master and apbs_[0 – 15] signal slave.

Parameters

Table 5-4 describes the APBMerger model parameters.

Name	*Description*
APB Base	Base address of the APB interrupt controller. Memory region size is fixed at 0x1000.
Enable Debug Messages	Boolean value. When set to True, the model debug messages are displayed as output. Set to False by default.

Table 5-4 APBMerger parameters

6 Probes

Table 6-1 describes the simulation probes included in the APB Protocol Bundle. Because the interface for transaction breakpoints is the same as the CASI *Mx* transaction interface, breakpoint probes are not included in the APB protocol bundle. These probes are included with the Mx protocol bundle.

Name	*Description*
APB Monitor	Shows APB transaction details (see Section 6.1).
APB Tracer	Enables tracing of signals on an APB connection. Traced signals can be viewed in the SoC Designer Simulator waveform window (see Section 6.2).

Table 6-1 APB probes

Monitor Probe

To insert a transaction monitor, right-click on an APB connection and select **Insert/Remove Monitor** from the context menu. This displays the monitor window shown in Figure 6-1.

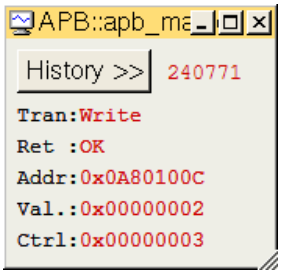

Figure 6-1 APB Monitor

Click the **History** button to display transactions and transaction details (Figure 6-2).

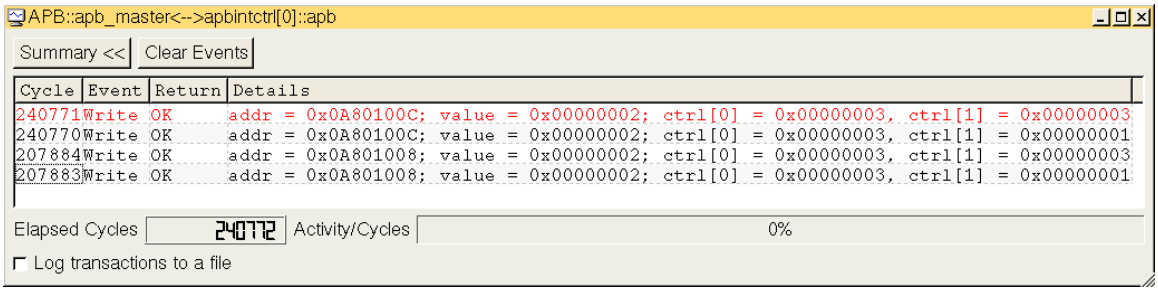

Figure 6-2 APB Monitor History view

Use the **Log transaction to a file** checkbox to dump the monitor contents to a file (Figure 6-3). Specify the output file using the **Browse** button.

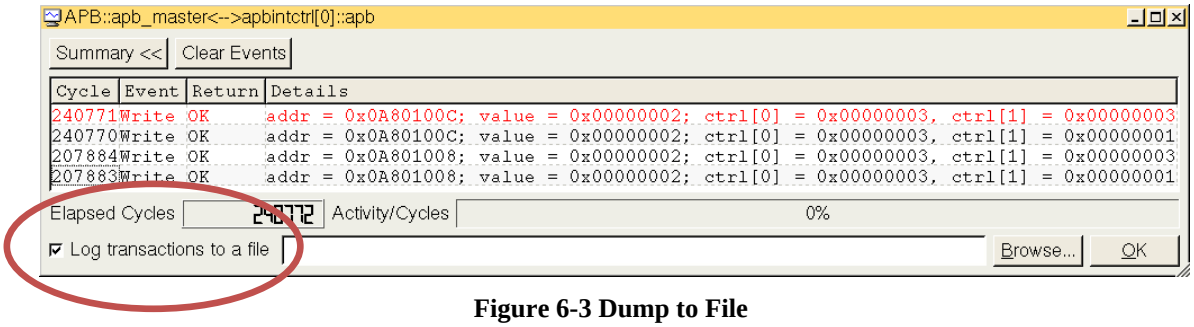

Figure 6-3 Dump to File

Tracer Probe

This probe allows tracing of APB signals. Use the SoC Designer Waveform window to see the traced signals. Refer to the *SoC Designer Plus User Guide* (Arm 100996) for instructions on adding a tracer probe and displaying the **Tracer Properties** dialog (Figure 6-4).

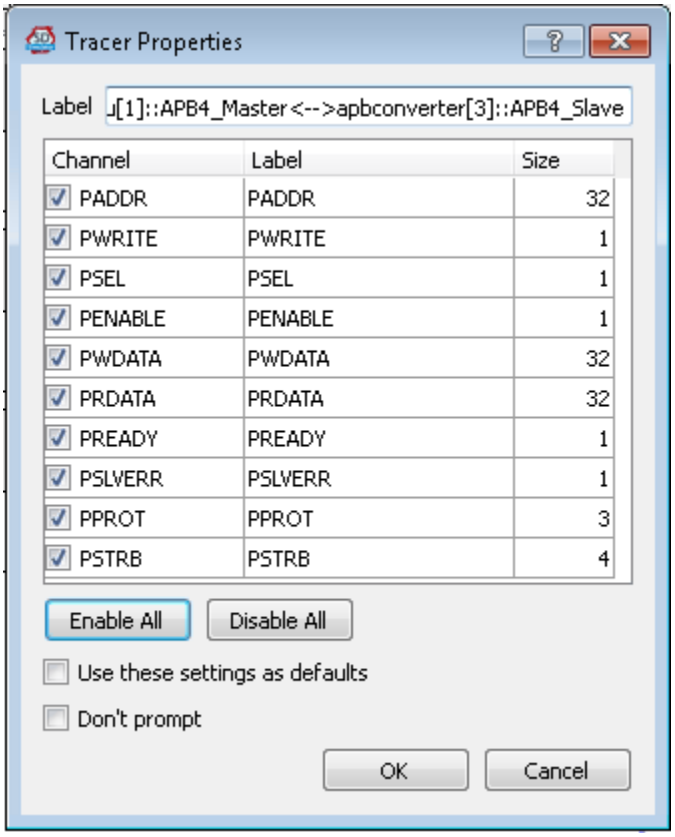

Figure 6-4 Tracer Properties

By default, all signals are traced. To disable tracing of certain signals, use the checkboxes located on the left side of the signal.

Note: AXI4 tracer properties show all AMBA4 AXI signals including the ACE channel signals regardless of whether or not the additional signals are used by the components. Disable unused signals or channels by deselecting the check-boxes next to the Channel/Signal name in the **Tracer Properties** dialog.

7 CASI APB Transaction Protocol

To view the type defines for CASI APB transactions, open `APB_Transaction.h` in the SoC Designer installation directory (`$MAXSIM_PROTOCOLS/APB/include/`).

Master/Slave Interface Implementation

APB transactions use the synchronous read/write methods of `CASITransactionIF` as follows:

```
CASIStatus read( uint64_t addr, uint32_t *value, uint32_t *ctrl );
CASIStatus write( uint64_t addr, uint32_t *value, uint32_t * ctrl);
```

where *`ctrl`* is an array that embeds the APB phases. You can use `APB_CTRL_IDX` (see the APB generic transaction definition below) to index into `ctrl`. For example:

- `ctrl [APB_IDX_CYCLE]` is used to access the APB phase, which is either:
  - `APB_CYCLE_ADDR` (equivalent to APB SETUP phase), or
  - `APB_CYCLE_DATA` (equivalent to APB ACCESS phase)
- `ctrl [APB_IDX_PSTRB]` = APB4 pstrb value (ignored for APB2, APB3)
- `ctrl [APB_IDX_PPROT]` = APB4 pprot value (ignored for APB2, APB3)

The SoC Designer generic transactor implements the APB master/slave interface using `CASITransactionIF` as follows:

```
   APB read setup phase():
     read() with
        ctrl[APB_IDX_CYCLE] == APB_CYCLE_ADDR
        ctrl [APB_IDX_PSTRB] = <pstrb>;
        ctrl [APB_IDX_PPROT] = <pprot>;

   APB write setup phase():
     write() with
        ctrl[APB_IDX_CYCLE] == APB_CYCLE_ADDR
        ctrl [APB_IDX_PSTRB] = <pstrb>;
        ctrl [APB_IDX_PPROT] = <pprot>;

   APB read access phase():
     read() with
        ctrl[APB_IDX_CYCLE] == APB_CYCLE_DATA
        addr => read address
        *value => used to return read data
        return value:
          CASI_STATUS_WAIT  -> APB3 PREADY=0 delay
          CASI_STATUS_ERROR -> APB3 PSLVERR=1 (read transfer failed)
          CASI_STATUS_OK    -> read transfer completed (with read data
returned)

   APB write access phase():
     write() with
        ctrl[APB_IDX_CYCLE] == APB_CYCLE_DATA
        addr => write address
        *value => write data passed from APB master
        return value:
          CASI_STATUS_WAIT  -> APB3 PREADY=0 delay
          CASI_STATUS_ERROR -> APB3 PSLVERR=1 (write transfer failed)
          CASI_STATUS_OK    -> write transfer completed
```

The return value of the `read` and `write` methods indicates the transaction status; these are described in Table 7-1.

Status	*Description*
`CASI_STATUS_WAIT`	Use this return status to extend a transaction in the APB_ENABLE cycle. This is applicable for AMBA3 APB components. Analogous to setting AMBA3 APB PREADY low.
`CASI_STATUS_ERROR`	An error response. Analogous to AMBA3 APB PSLVERR.
`CASI_STATUS_OK`	Request completed without errors.

Table 7-1 APB transaction status