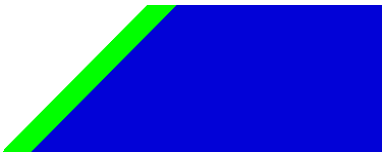


AXI4 Lite APB3/4 Bridge Implementation Document

VERSION 1.0

September 9, 2021

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH LAMPRÓ MÉLLON PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY LAMPRÓ MÉLLON INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN LAMPRÓ MÉLLON'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, LAMPRÓ MÉLLON ASSUMES NO LIABILITY WHATSOEVER, AND LAMPRÓ MÉLLON DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF LAMPRÓ MÉLLON PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER LAMPRÓ MÉLLON INTELLECTUAL PROPERTY RIGHT.
UNLESS OTHERWISE AGREED IN WRITING BY LAMPRÓ MÉLLON, THE LAMPRÓ MÉLLON PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE LAMPRÓ MÉLLON PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.
Lampró Méllon may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Lampró Méllon reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
Contact your local Lampró Méllon office or your distributor to obtain the latest specifications and before placing your product order. This document contains information on products in the design phase of development.
All products, platforms, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice. All dates specified are target dates, are provided for planning purposes only and are subject to change.
This document contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product is available. Verify with your local sales office that you have the latest datasheet before finalizing a design.

Contents

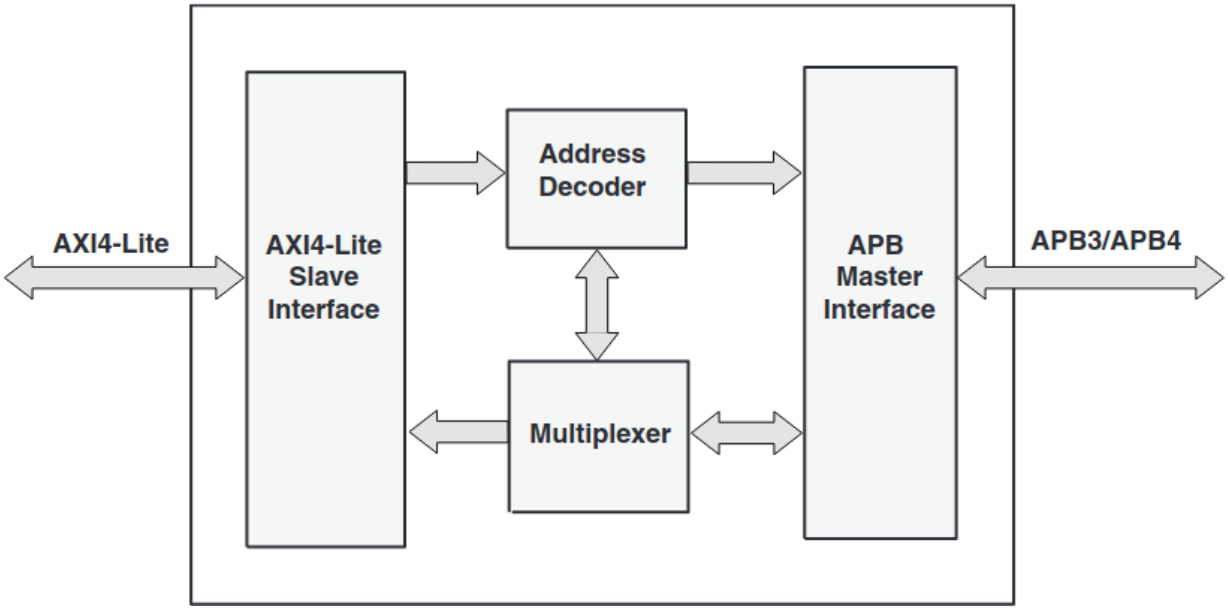
0.1 Introduction

The AXI4 to APB3/4 Bridge is an interfacing logic that interfaces AXI4 Lite Masters with APB3/4 Slaves and Provides an efficient bridging between memory or peripheral devices having APB3/4 as an interfacing protocol with high speed AXI4 Lite Masters.

0.2 Overview

The AXI to APB Bridge facilitates the AXI4 Lite Masters which need to communicate with APB3 or APB4 slaves for memory or peripheral operations and the bridge translates the transactions being generated from AXI4 Lite Master and bridges the transactions to APB slaves. The responses from slaves are also translated back for the AXI4 Master. The Bridge operates on the same clocks as the AXI4 Master.The Clock ratio between the Master and the Bridge is 1:1. Upto Sixteen APB3/4 Slaves can be connected to the Bridge and each can be selected based on the address decoding mechanism and each slave can be accessed based on its memory region, memory operations can be carried out across all slaves sequentially. The AXI4 Lite master is connected to the Side of the Bridge that Acts as the AXI4 Lite Slave which captures the transactions from the AXI4 Master and these transactions are then sent to APB3/4 Master instantiated inside the bridge which drives the Outer ports of the Bridge for the APB3/4 Slaves connected with the bridge.

0.3 Bridge Block Diagram



0.4 Bridge Components

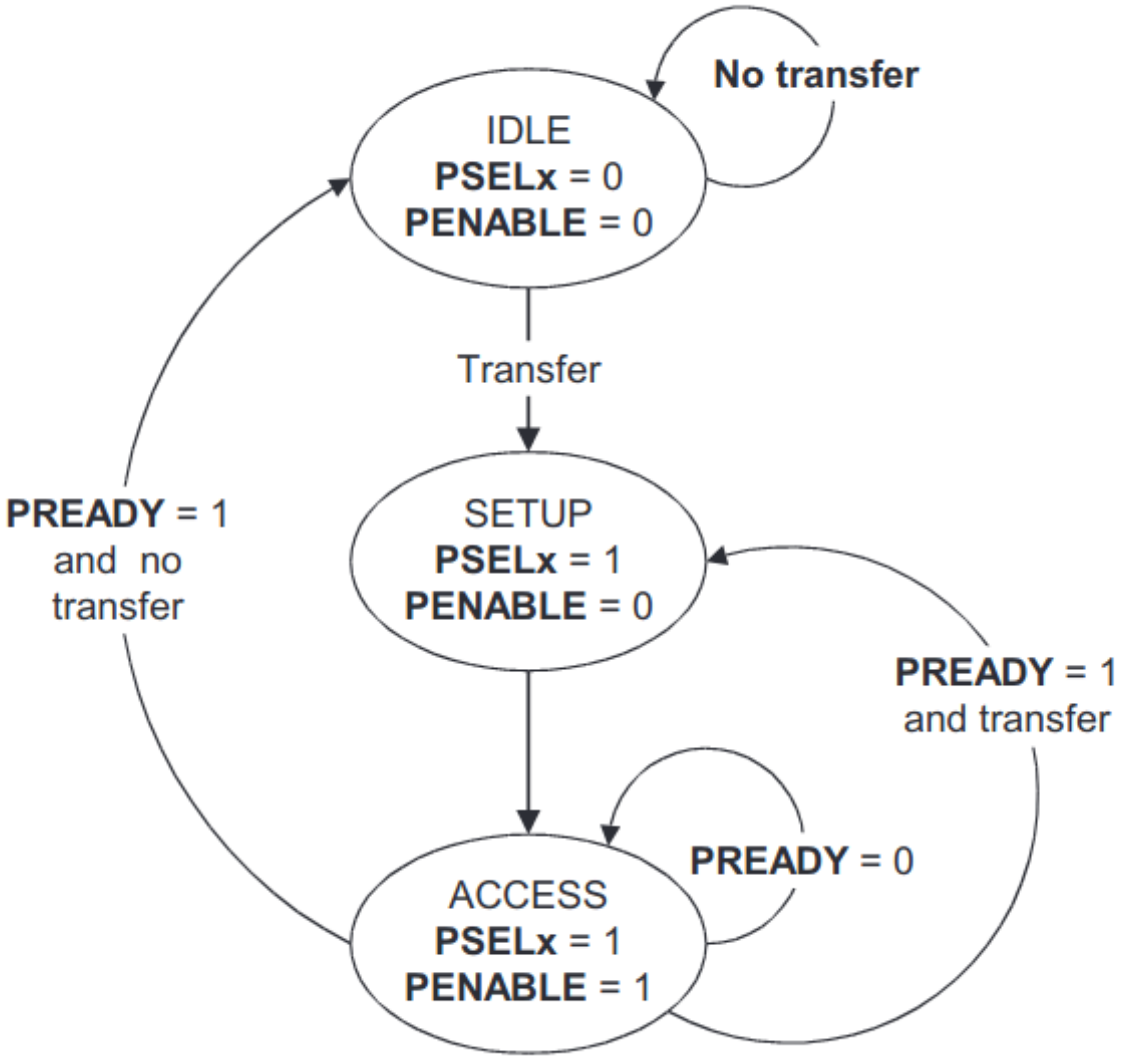
The following components have been the part of the main AXI to APB Bridge Module and have been instantiated in the main module.

- APB3/4 Master Module
- Address Decoder
- Read Data Multiplexer

0.4.1 APB3/4 Master Module

The APB3/4 Master has been implemented as a Moore state machine, the design is made according to the APB Spec 2.0. The APB master has been parametrized to handle multiple slaves and the slaves are selected from one hot address decoding logic and is driven from the bridge.

0.4.1.1 APB3/4 Master State Machine



0.4.1.2 APB Master Interface Description

The APB3/4 master is made with the parametrization support to expand or shrink the port size of the PENABLE signal to connect the Master with a customizable number of slaves. The interface is compliant with the signal descriptions and sizes mentioned in the APB Spec 2.0 .The following table enlists the signals and their descriptions.

Signal	Source	Description
PCLK	Clock source	Clock. The rising edge of **PCLK** times all transfers on the APB.
PRESETn	System bus equivalent	Reset. The APB reset signal is active LOW. This signal is normally connected directly to the system bus reset signal.
PADDR	APB bridge	Address. This is the APB address bus. It can be up to 32 bits wide and is driven by the peripheral bus bridge unit.
PPROT	APB bridge	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
PSELx	APB bridge	Select. The APB bridge unit generates this signal to each peripheral bus slave. It indicates that the slave device is selected and that a data transfer is required. There is a **PSELx** signal for each slave.
PENABLE	APB bridge	Enable. This signal indicates the second and subsequent cycles of an APB transfer.
PWRITE	APB bridge	Direction. This signal indicates an APB write access when HIGH and an APB read access when LOW.
PWDATA	APB bridge	Write data. This bus is driven by the peripheral bus bridge unit during write cycles when **PWRITE** is HIGH. This bus can be up to 32 bits wide.
PSTRB	APB bridge	Write strobes. This signal indicates which byte lanes to update during a write transfer. There is one write strobe for each eight bits of the write data bus. Therefore, **PSTRB[n]** corresponds to **PWDATA[(8n + 7):(8n)]**. Write strobes must not be active during a read transfer.
PREADY	Slave interface	Ready. The slave uses this signal to extend an APB transfer.
PRDATA	Slave interface	Read Data. The selected slave drives this bus during read cycles when **PWRITE** is LOW. This bus can be up to 32-bits wide.
PSLVERR	Slave interface	This signal indicates a transfer failure. APB peripherals are not required to support the **PSLVERR** pin. This is true for both existing and new APB peripheral designs. Where a peripheral does not include this pin then the appropriate input to the APB bridge is tied LOW.

0.4.2 Address Decoder

The address decoder is a dynamically generated selection logic which gets generated dynamically based on the number of slaves, a parameter that is provided while instantiating the bridge. The selection signal's each bit corresponds to the PENABLE signal of each APB3/4 slave connected to the bridge. These bits are assigned independently in a combinational loop where the bits are asserted based on the address which either falls inside the address range of the corresponding slave whose PENABLE bit is connected. The select signal's each bit that is made from the decoding mechanism described above is connected to each slave's PENABLE signal.

0.4.3 Read Data Multiplexer

As the total number of APB3/4 slaves that can be connected to the Bridge are upto 16 the read data channels coming from slaves into the bridge are sixteen and the incoming read data is channeled through a multiplexer to the AXI4 master from the bridge. The read data channel is selected based on the Address decoding mechanism and the selection signal is a one hot signal that is ANDED with the read data lines to select the read data from the correct slave.

0.5 Parametrization Implementation

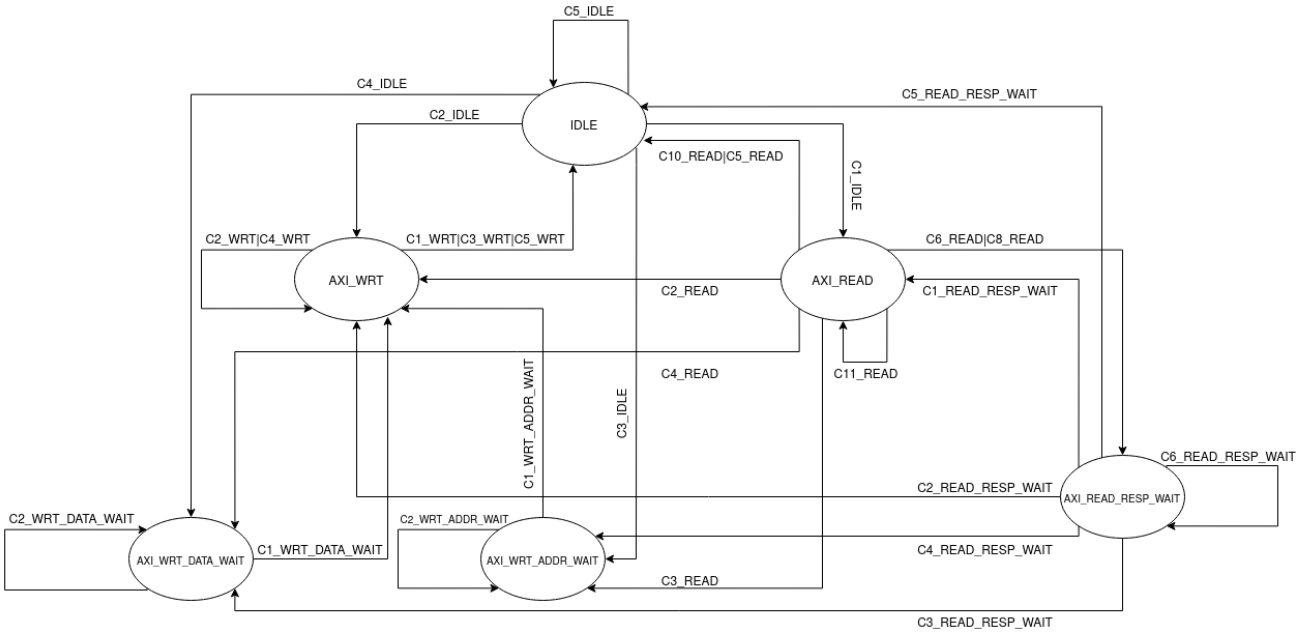
The AXI APB Bridge has been designed with parametrization of specific design features which can be changed while instantiating the Bridge. The following list provides the names of the parameters which are being used in the design to provide dynamic functionality for different designs.

- c-apb-num-slaves
  - This parameter provides the number of slaves going to be connected with the bridge
- memory-regions1
  - This is a 32*c-apb-num-slaves long bit vector that provides the starting addresses of each slave connecting with the bridge.Each slave's starting address is separated from the other by 32 bits as the address space for the design is 32 bits.
- memory-regions2
  - This is a 32*c-apb-num-slaves long bit vector that provides the ending addresses of each slave connecting with the bridge.Each slave's ending address is separated from the other by 32 bits as the address space for the design is 32 bits.
- timeout-val
  - This is an important parameter that provides the val for which the bridge will wait for the APB slaves connected to the bridge and if the slave does not respond in the set clock cycles after the transaction was initiated by the APB master in the bridge then the error response is generated by the bridge and the soft stuck is prevented this way.
- APB-Protocol
  - The parameter provides the number of APB protocols being used in the design. The allowed values are 3 or 4 that correspond to APB3 or APB4.

0.6 Timeout Response Implementation

A timeout counter has been implemented in the design. The counter is loaded with the time-out val on the beginning of each transaction that is sent to the selected APB slave. The counter decrements each cycle as long as the slave does not respond with PREADY response during the Access phase of APB master if the slave does not respond during the set number of clock cycles the bridge responds with an error response on the read response or write response channel depending on which type of transaction was being processed to AXI4 Lite Master and the APB master made to reset and goes into the IDLE state.

0.7 Bridge State Machine



0.7.1 State Transition Conditions

IDLE STATE:

condition1_state_Idle = ((write_happened && s_axi_bready)||(!write_happened));
condition2_state_Idle = (write_happened && !s_axi_bready);

C1_IDLE : (condition1_state_Idle)&(s_axi_arvalid)&&((state==Idle)||(state==Access) : Transition to -> axi_read
C2_IDLE : (condition1_state_Idle)&((s_axi_awvalid)&&(s_axi_wvalid)&&(state==Idle|state==Access)) : Transition to -> axi_write
C3_IDLE : (condition1_state_Idle)&(!(s_axi_awvalid)&&(s_axi_wvalid)&&(state==Idle|state==Access)) : Transition to -> axi_write_address_wait
C4_IDLE : (condition1_state_Idle)&((s_axi_awvalid)&&!(s_axi_wvalid)&&(state==Idle|state==Access)) : Transition to -> axi_write_data_wait
C5_IDLE : ((condition1_state_Idle) & !C1&!C2&!C3&!C4)||(condition2_state_Idle) |!(condition1_state_Idle|condition2_state_Idle) : Transition to -> Bridge_Idle

AXI_READ:

condition1_state_axi_read = ((state == Access) & (|(m_apb_psel & m_apb_pready)));
condition2_state_axi_read = ((state == Access) & !(|(m_apb_psel & m_apb_pready)));

C1_READ : (condition1_state_axi_read)&(s_axi_rready)&(s_axi_arvalid) : Transition to -> axi_read
C2_READ : (condition1_state_axi_read)&(s_axi_rready)&((s_axi_awvalid)&&(s_axi_wvalid) Transition to -> axi_write
C3_READ : (condition1_state_axi_read)&(s_axi_rready)&(!(s_axi_awvalid)&(s_axi_wvalid) Transition to -> axi_write_address_wait
C4_READ : (condition1_state_axi_read)&(s_axi_rready)&((s_axi_awvalid)&&!(s_axi_wvalid) Transition to -> axi_write_data_wait
C5_READ : (condition1_state_axi_read)&(s_axi_rready)&!C4&!C3&!C2&!C1 Transition to -> Bridge_Idle
C6_READ : (condition1_state_axi_read)&(!s_axi_rready) : Transition to -> axi_read_response_wait
C7_READ : (condition2_state_axi_read)&(timeout_counter>32'h00000000) : Transition to -> axi_read
C8_READ : (condition2_state_axi_read)&!(timeout_counter>32'h00000000) : Transition to -> axi_read_response_wait
C9_READ : !(condition1_state_axi_read)&!(condition2_state_axi_read)&(state==Setup) : Transition to -> axi_read
C10_READ : !(condition1_state_axi_read)&!(condition2_state_axi_read)(state==Setup) Transition to -> Bridge_Idle
C11_READ : C1|C7|C9 Transition to -> axi_read

AXI_WRT:

C1_WRT : ((state==Access)&(|(m_apb_psel&m_apb_pready))) : Transition to -> Bridge_Idle
C2_WRT : ((state==Access)&(!|(m_apb_psel&m_apb_pready)))&(timeout_counter>32'h00000000) Transition to -> axi_write
C3_WRT : ((state==Access)&(!|(m_apb_psel&m_apb_pready)))&!(timeout_counter>32'h00000000) Transition to -> Bridge_Idle
C4_WRT : ((state==Setup) Transition to -> axi_write
C5_WRT : ((state==Idle) Transition to -> Bridge_Idle

AXI_WRT_DATA_WAIT:

C1_WRT_DATA_WAIT : (s_axi_wvalid) : Transition to -> axi_write
C2_WRT_DATA_WAIT : !(s_axi_wvalid) : Transition to -> axi_write_data_wait

AXI_WRT_ADDR_WAIT:

C1_WRT_ADDR_WAIT : (s_axi_awvalid) : Transition to -> axi_write
C2_WRT_ADDR_WAIT : !(s_axi_awvalid) : Transition to -> axi_write_addr_wait

AXI_READ_RESP_WAIT:

C1_READ_RESP_WAIT : (s_axi_rready)&&(s_axi_arvalid) : Transition to -> axi_read
C2_READ_RESP_WAIT : (s_axi_rready)&&((s_axi_awvalid)&&(s_axi_wvalid)) Transition to -> axi_write
C3_READ_RESP_WAIT : (s_axi_rready)&&((s_axi_awvalid)&&(!s_axi_wvalid)) Transition to -> axi_write_data_wait
C4_READ_RESP_WAIT : (s_axi_rready)&&((!s_axi_awvalid)&&(s_axi_wvalid)) Transition to -> axi_write_address_wait
C5_READ_RESP_WAIT : (s_axi_rready)&&!C1_READ_RESP&&!C2_READ_RESP&&!C3_READ_RESP&&!C4_READ_RESP Transition to -> Bridge_Idle
C6_READ_RESP_WAIT : !(s_axi_rready) Transition to -> axi_read_response_wait

0.8 Timing Diagram