AndeShape™ ATCTLC2AXI500 Data Sheet

Document Number DS182-10
Date Issued 2020-05-04

Copyright Notice

Contact Information

Should you have any problems with the information contained herein, you may contact Andes Technology Corporation through:

Email — support@andestech.com
Website — https://es.andestech.com/eservice/

Please include the following information in your inquiries:

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of the problem.

General suggestions for improvements are welcome.

Revision History

Rev.	Rev. Date	Revised Content
1.0	2020-05-04	Initial release

Contents

List of Figures

List of Tables

1 Introduction

ATCTLC2AXI500 is a bus bridge, which convert TileLink to AXI. It handles the bus protocol translation between TileLink and AXI4 protocols.

1.1 Features

- Compliant with AXI4 protocols
  - Supports AXI exclusive access
  - Supports A*x*CACHE information
  - Supports A*x*PROT information
- Compliant with TileLink Cached (TL-C)
  - Supports beat interleaving in TileLink channel D
- Configurable clock synchronization
  - asynchronous
  - synchronous N:1 (TileLink clock frequency : AXI clock frequency)
- Configurable 32–64 bits address width
- Configurable 32/64/128/256 bits data width
- Configurable 4–8 ID width
- Supports up to 128 bytes burst size

1.2 Block Diagram

Figure 1 illustrates the top-level block diagram of the ATCTLC2AXI500 design. The upstream TileLink ports are slave interfaces which interact with the connected master interfaces. The downstream TileLink ports are master interfaces which interact with the connected slave interfaces.

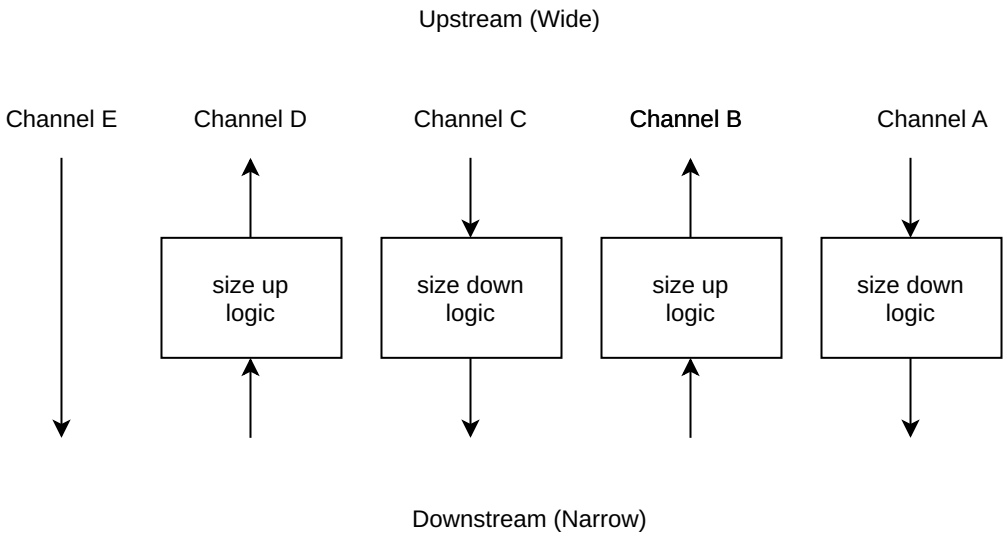

Figure 1: ATCTLC2AXI500 Block Diagram

1.3 Functional Description

1.3.1 TileLink to AXI

ATCTLC2AXI500 translate TileLink to AXI protocol, and support TL-C permission transition request.

- Put and release data in TileLink are translate to AXI write request
- Get and acquire data in TileLink are translate to AXI read request
- Permission transition requests without data transfer is response by ATCTLC2AXI500 without AXI transaction.

1.3.2 Bursts

- Support INCR burst type only.
- All transactions are aligned to bus data width, no multi-beat transfer with narrowed data issued.

1.3.3 Beat Interleave

ATCTLC2AXI500 allows interleaving beats of different messages on Channel D from the slave device through the downstream interface to the master device through the upstream interface. It is, however, forbidden to interleave beats on Channel A, Channel B, and Channel C. For supporting beat interleaving, the connected master device must support the following features when the connected slave device, such as TileLink-to-AXI bridge, responds beats with interleaving.

- Each ID of `a_source` with the request type message of `a_opcode`, such as `PutFullData`, `PutPartialData` and `AcquireBlock`, can only be issued once until the correspond response returned from Channel D.
- Each ID of `c_source` with the request type message of `c_opcode`, such as `Release` and `ReleaseData`, can only be issued once until the correspond response returned from Channel D.
- Each ID of `a_source` and `c_source` with the write request type message can only be issued once until the correspond response returned from Channel D.

2 Hardware Configuration Option

Table 1 lists the configuration parameters of ATCTLC2AXI500. These descriptions can be applied to both upstream side and downstream side unless otherwise stated.

Table 1: Configuration Parameters

Parameter Name	Legal Value	Default Value	Description
ACLK_SYNC	sync, async	sync	Define the axi clock domain, sync(N:1) or async
ADDR_WIDTH	32 - 64	32	Define the bit width of the address field in tilelink and axi channels
DATA_WIDTH	32, 64, 128, 256	64	Define the bit width of the data field in tilelink/axi channels
ID_WIDTH	4 - 8	4	Define the bit width of source/id field in tilelink/axi channels
AXI_AWUSER_WIDTH	1 - 32	1	Define the bit width of AXI write address channel user side-band signals
AXI_ARUSER_WIDTH	1 - 32	1	Define the bit width of AXI read address channel user side-band signals
AXI_WUSER_WIDTH	1 - 32	1	Define the bit width of AXI write data channel user side-band signals
AXI_RUSER_WIDTH	1 - 32	1	Define the bit width of AXI read data channel user side-band signals
AXI_BUSER_WIDTH	1 - 32	1	Define the bit width of AXI write response channel user side-band signals

3 Signal Description

Figure 2 shows the I/O signals of ATCTLC2AXI500, including TileLink signals for upstream and downstream interfaces respectively. Table 2 shows the detailed descriptions of both upstream and downstream interface signals.

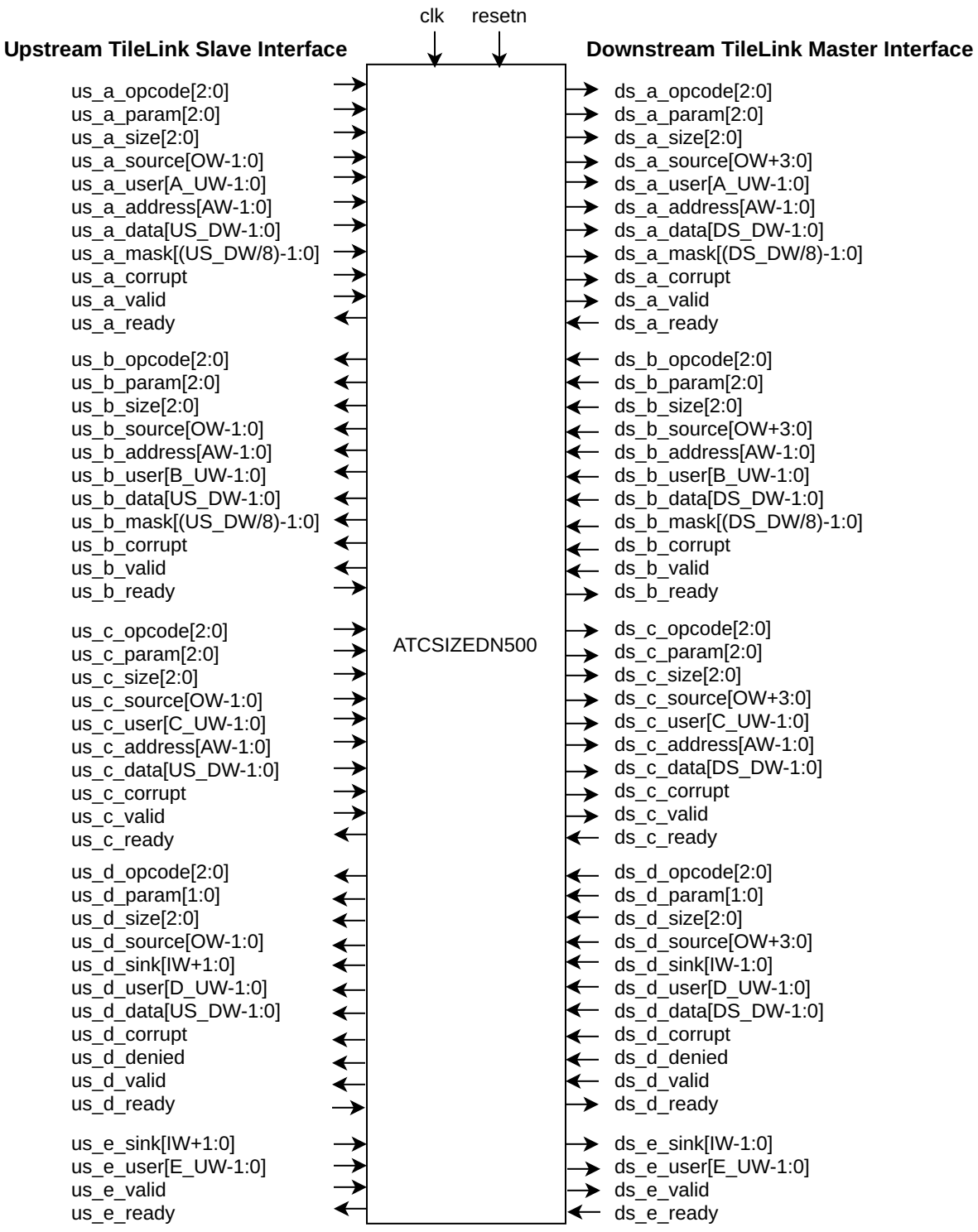

Figure 2: ATCTLC2AXI500 I/O Signals

Table 2: Interface Signal Descriptions

Signal Name	IO Type	Description
clk	I	Bus clock
resetn	I	Bus reset (Active-Low)
aclk	I	AXI clock (async)
aclk_en	I	AXI clock enable (sync)
TileLink Interface		
Channel A		
a_opcode[2:0]	I	Operation code
a_param[2:0]	I	Parameter code
a_size[2:0]	I	Operation size
a_source[ID_WIDTH-1:0]	I	Master source identifier
a_address[ADDR_WIDTH-1:0]	I	Target byte address
a_user[x:0]	I	Sideband for a*x*lock, a*x*cache, a*x*lock
a_data[DATA_WIDTH-1:0]	I	Data payload
a_mask[(DATA_WIDTH/8)-1:0]	I	Byte lane select signal
a_corrupt	I	The data in this beat is corrupt.
a_valid	I	Valid signal.
a_ready	O	Ready signal.
Channel B		
b_opcode[2:0]	O	Operation code
b_param[2:0]	O	Parameter code
b_size[2:0]	O	Operation size
b_source[ID_WIDTH-1:0]	O	Master source identifier
b_address[ADDR_WIDTH-1:0]	O	Target byte address
b_user[B_UW-1:0]	O	User defined side band signals
b_data[DATA_WIDTH-1:0]	O	Data payload
b_mask[(DATA_WIDTH/8)-1:0]	O	Byte lane select signal
b_corrupt	O	The data in this beat is corrupt.
b_valid	O	Valid signal, hard-wired to 0.
b_ready	I	Ready signal.
Channel C		
c_opcode[2:0]	I	Operation code
c_param[2:0]	I	Parameter code
c_size[2:0]	I	Operation size

Continued on next page...

Table 2: (continued)

Signal Name	IO Type	Description
c_source[ID_WIDTH-1:0]	I	Master source identifier
c_user[y:0]	I	Sideband for a*x*lock, a*x*cache, a*x*lock
c_address[ADDR_WIDTH-1:0]	I	Target byte address
c_data[DATA_WIDTH-1:0]	I	Data payload
c_corrupt	I	The data in this beat is corrupt.
c_valid	I	Valid signal.
c_ready	O	Ready signal.
Channel D		
d_opcode[2:0]	O	Operation code
d_param[1:0]	O	Parameter code
d_size[2:0]	O	Operation size
d_source[ID_WIDTH-1:0]	O	Master source identifier
d_sink	O	Slave sink identifier
d_user[z:0]	O	Sideband to transfer rresp and bresp
d_data[DATA_WIDTH-1:0]	O	Data payload
d_denied	O	The slave was unable to service the request
d_corrupt	O	The data in this beat is corrupt.
d_valid	O	Valid signal.
d_ready	I	Ready signal.
Channel E		
e_sink	I	Slave sink identifier
e_user	I	User defined side band signals
e_valid	I	valid signal.
e_ready	O	Ready signal, hard-wired to 1.

AXI Interface		
Write Address Channel		
awid[ID_WIDTH-1:0]	O	Write address ID
awaddr[ADDR_WIDTH-1:0]	O	Write address
awlen[7:0]	O	Write burst length
awsize[2:0]	O	Write burst size
awburst[1:0]	O	Write burst type
awlock	O	Write lock type

Continued on next page...

Table 2: (continued)

Signal Name	IO Type	Description
awcache[3:0]	O	Write memory type
awprot[2:0]	O	Write protection type
awvalid	O	Write address valid
awready	I	Write address ready
Write Data Channel		
wdata[DATA_WIDTH-1:0]	O	Write data
wstrb[(DATA_WIDTH/8)-1:0]	O	Write strobes
wlast	O	Write last
wvalid	O	Write data valid
wready	I	Write data ready
Write Response Channel		
bid[ID_WIDTH-1:0]	I	Write response ID
bresp[1:0]	I	Write response
bvalid	I	Write response valid
bready	O	Write response ready
Read Address Channel		
arid[ID_WIDTH-1:0]	O	Read address ID
araddr[ADDR_WIDTH-1:0]	O	Read address
arlen[7:0]	O	Read burst length
arsize[2:0]	O	Read burst size
arburst[1:0]	O	Read burst type
arlock	O	Read lock type
arcache[3:0]	O	Read memory type
arprot[2:0]	O	Read protection type
arvalid	O	Read address valid
arready	I	Read address ready
Read Data Channel		
rid[ID_WIDTH-1:0]	I	Read response ID
rdata[DATA_WIDTH-1:0]	I	Read data
rresp[1:0]	I	Read response
rlast	I	Read last
rvalid	I	Read data valid
rready	O	Read data ready

Note

- The width of TileLink user field is determined by the width of AXI user side-band. see next section for detail.
- a_user width = MAX((AWUSER+WUSER), ARUSER)+8
- c_user width = AWUSER+WUSER + 8
- d_user = MAX(RUSER, BUSER) + 2

3.1 Side-band Signals

There are two side-band signals: `a_user`, `c_user` and `d_user` for encapsulating AXI signals that cannot be converted to TileLink protocol.

- `a_user` and `c_user` is used to propagate the **A*x*LOCK**, **A*x*CACHE**, and **A*x*PROT** from AXI AR and AW channel.
- `d_user` is used to indicate the **RRESP**, **BRESP** and to the AXI R and B channel.

Table 3: a_user Format for TileLink A Channel Messages Mapped to AXI Write Request

Field	Bit Position	Description
AWPROT	`a_user[2:0]`	AXI AWPROT
AWCACHE	`a_user[6:3]`	AXI AWCACHE
AWLOCK	`a_user[7]`	AXI AWLOCK
AWUSER	`a_user[8+:AWUSER_WIDTH]`	AXI AWUSER
WUSER	`a_user[8+AWUSER_WIDTH+:WUSER_WIDTH]`	AXI WUSER

Table 4: a_user Format for TileLink A Channel Messages Mapped to AXI Read Request

Field	Bit Position	Description
ARPROT	`a_user[2:0]`	AXI ARPROT
ARCACHE	`a_user[6:3]`	AXI ARCACHE
ARLOCK	`a_user[7]`	AXI ARLOCK
ARUSER	`a_user[8+:ARUSER_WIDTH]`	AXI ARUSER

Table 5: c_user Format for TileLink C Channel Messages Mapped to AXI Write Request

Field	Bit Position	Description
AWPROT	c_user[2:0]	AXI AWPROT
AWCACHE	c_user[6:3]	AXI AWCACHE
AWLOCK	c_user[7]	AXI AWLOCK
AWUSER	c_user[8+:AWUSER_WIDTH]	AXI AWUSER
WUSER	c_user[8+AWUSER_WIDTH+:WUSER_WIDTH]	AXI WUSER

Table 6: d_user Format for TileLink D Channel Messages Mapped to AXI Write Response

Field	Bit Position	Description
BRESP	d_user[1:0]	AXI BRESP
BUSER	d_user[2+:BUSER_WIDTH]	AXI BUSER

Table 7: d_user Format for TileLink D Channel Messages Mapped to AXI Read Response

Field	Bit Position	Description
RRESP	d_user[1:0]	AXI RRESP
RUSER	d_user[2+:RUSER_WIDTH]	AXI RUSER

4 Access Latencies

Table 8: Latency between Both Sides for Each Channel

Source Side	Destination Side	Latency
TileLink A channel	AXI AW/AR/W channel	1
TileLink B channel	-	-
TileLink C channel	AXI AW/AR/W channel	1
AXI R/B channel	TileLink D channel	1
TileLink E channel	-	0

5 Message Mapping

The section describes the message mapping form TL-C to AXI. For messages are mapped to AXI read or write transaction, ATCTLC2AXI500 will block the request if there is another outstanding transaction with the same ID of the request.

Note

- For two write with the same ID, the second one will not be issued until all responses of the first one are received.
- For two read with the same ID, the second one will not be issued until all responses of the first one are received.
- For a read and a write with the same ID, the two will not be blocked by each other.

5.1 TileLink A channel

5.1.1 PutFullData

- PutFullData is mapped to AXI write transaction.

5.1.2 PutPartialData

- PutPartialData is mapped to AXI write transaction.

5.1.3 Get

- Get is mapped to AXI read transaction

5.1.4 AcquireBlock

- If a_perm is NtoB or NtoT,
  - AcquireBlock is mapped to AXI read transaction.
  - ATCTLC2AXI500 returns GrantData with toT permission.
- If a_perm is BtoT.
  - AcquireBlock is not mapped to any AXI request.
  - ATCTLC2AXI500 returns Grant with toT permission.

5.1.5 AcquirePerm

- AcquirePerm is not mapped to any AXI request.
- ATCTLC2AXI500 returns Grant with toT permission.

5.2 TileLink C channel

5.2.1 Release

- Release is not mapped to any AXI request.
- ATCTLC2AXI500 returns ReleaseAck with toN permission.

5.2.2 ReleaseData

- ReleaseData is mapped to AXI write transaction.
- ATCTLC2AXI500 returns ReleaseAck with toN permission.

5.3 TileLink D channel

5.3.1 AccessAck

- AccessAck is mapped form AXI write response channel.

5.3.2 AccessAckData

- AccessAckData is mapped form AXI read data channel.

5.3.3 Grant

- Generated by ATCTLC2AXI500, and always have toT permission

5.3.4 GrantData

- GrantData is mapped form AXI read data channel.
- Always have toT permission.

5.3.5 ReleaseAck

- ReleaseAck could be
  - form AXI write response channel (TtoN, TtoB)
  - or generated by ATCTLC2AXI500 (BtoN)
- Always have toN permission.

5.4 TileLink E channel

5.4.1 GrantAck

- e_ready is hard-wired to 1, e_sink is not checked.