

AndeShape™ ATCBMC300 Design Specification

|  |  |
| --- | --- |
| Document Number | DSP-00092 |
| Date Issued | 2016-04-13 |
| Status | Final release |



**CONFIDENTIAL**

Revision History

| Rev. | Revision Date | Revised Chapter-Section | Revised Content | Author |
| --- | --- | --- | --- | --- |
| 1.0 | 2016-04-13 | 4.3.3 | 1. Update slave size/base address configuration restriction | Joshua |
| 0.4 | 2016-02-25 | 3,4,5,6 | 1. Remove deadlock detection logic and related configuration 2. Describe the response ordering policy 3. Describe slave size/base address configuration restriction clearly 4. Move priority register to 0x10 offset address 5. Move the base/size registers to be started from 0x100 6. Describe the address mapping scheme and address request arbitration | Joshua |
| 0.3 | 2016-02-04 | 3,4,5,6 | 1. Remove error status register 2. Revise deadlock condition 3. Add signal table 4. Change data width configuration 5. Add slave device respond ordering configuration 6. Describe the slave size/base address configuration restriction 7. Remove default slave 8. Change to single priority register for all slaves | Joshua |
| 0.2 | 2016-01-12 | 4,5 | 1. Single ID bit width configuration to all master 2. Change the “in order only” feature from pin assignment to configurable 3. Rename define macro 4. Change response control register for all master to each response register per master 5. Change the single priority register for all slave to each priority register per slave 6. Add default slave and internal slave design | Joshua |
| 0.1 | 2015-12-25 | All | Initial Draft | Joshua |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Review History** | | | | |
| **Rev.** | **Review Date** | **Reviewed Chapter-Section** | **Comments** | **Reviewer** |
| **0.4** | 2016-02-25 | all | 1. Remove deadlock detection logic and related configuration 2. Describe the response ordering policy 3. Describe slave size/base address configuration restriction clearly 4. Move priority register to 0x10 offset address 5. Move the base/size registers to be started from 0x100 6. Describe the address mapping scheme and address request arbitration | Jonathan, Alexander, Frank |
| **0.3** | 2016-02-04 | all | 1. Remove deadlock detection logic and related configuration 2. Remove ATCBMC300\_SLVy\_RESP\_OUT\_OF\_ORDER\_SUPPORT define macro 3. Describe Slave size/base address configuration restriction clearly 4. Move priority register to 0x10 offset address 5. Move the base/size registers to be started from 0x100 | Jonathan, Alexander, Frank |
| **0.2** | 2016-01-12 | all | 1. Remove error status register 2. Change single priority register for all slaves | Jonathan, Eric, Frank |
| **0.1** | 2015-12-25 | all | 1. Single ID bit width configuration to all master 2. Change the “in order only” feature from pin assignment to configurable 3. Proper define macro renaming 4. Change the single priority register for all slave to each priority register per slave 5. Evaluate the “in-order only” and “re-orderable” PPA difference | Jonathan, Eric, Wolfson. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Sign-off List** | | | |
| **Department** | **Name** | **Date** | **Comment** |
| VLSI | Jonathan | 2016-04-13 | Accepted for final release |

Table of Contents

[Revision History ii](#_Toc438822183)

[List of Tables vi](#_Toc438822184)

[List of Figures vii](#_Toc438822185)

[1. Overview 1](#_Toc438822186)

[1.1. Introduction 1](#_Toc438822187)

[1.2. Features 1](#_Toc438822188)

[1.3. Limitation and Restrictions 2](#_Toc438822189)

[1.4. Block Diagram 3](#_Toc438822190)

[2. Signal Description 4](#_Toc438822191)

[3. Functions 6](#_Toc438822192)

[4. Design Configuration 7](#_Toc438822193)

[4.1. Address Width 7](#_Toc438822194)

[4.2. Data Width 7](#_Toc438822195)

[4.3. Number of Master Ports 7](#_Toc438822196)

[4.4. Number of Slave Ports 7](#_Toc438822197)

[4.5. Master and Slave Ports Connectivity 7](#_Toc438822198)

[4.6. Base Address 8](#_Toc438822199)

[4.7. Space Size 8](#_Toc438822200)

[5. Programming Model 10](#_Toc438822201)

[5.1. Summary of Registers 10](#_Toc438822202)

[5.2. Register Description 10](#_Toc438822203)

[5.2.1. ID and revision register (0x00) 10](#_Toc438822204)

[5.2.2. Control register (0x10) 10](#_Toc438822205)

[5.2.3. Interrupt status register (0x18) 11](#_Toc438822206)

[5.2.4. Base/Size Register of AXI Slave m (0x40+(m\*8)) 13](#_Toc438822207)

[6. Micro-Architecture 15](#_Toc438822208)

[6.1. Address Channel Data Path 15](#_Toc438822209)

[6.2. Write Data Channel Data Path 16](#_Toc438822210)

[6.3. Read Data/BRESP Channel Data Path 17](#_Toc438822211)

List of Tables

[Table 3. Mapping table of space size 8](#_Toc438822212)

[Table 4. ATCBMC300 register summary 10](#_Toc438822213)

[Table 5. ID and revision register 10](#_Toc438822214)

[Table 7. Control register 11](#_Toc438822215)

[Table 8. Interrupt status register 11](#_Toc438822216)

[Table 9. Base/Size registers of AXI slave *m* 14](#_Toc438822217)

List of Figures

[Figure 1. Block Diagram 2](#_Toc426029749)

[Figure 2. The Interface Signals of ATCAXIDMAC100 with single AXI Interface 4](#_Toc426029750)

**Typographical Convention Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document Element | Font | Font Style | Size | Color |
| Normal | Georgia | Normal | 12 | Black |
| Code | Lucida Console | Normal | 11 | Indigo |
| USER VARIABLE | Lucida Console | ALL-CAPS | 11 | INDIGO |
| Note/Warning | Georgia | Normal | 12 | Red |
| Hyperlink | Georgia | Bold + Underlined | 12 | Blue |

# Overview

## Introduction

AndeShape™ ATCBMC300 is designed to connect AXI master interface (MI) slots and slaves interface (SI) slots.

## Features

* Compliant with AMBA®4 AXI4™,
* Up to 16 master interface slots
* Up to 32 slave interface slots
* Support 24-64 bit unified address width
* Support 32/64/128 bit unified data width
* Configurable connectivity between masters and slaves
* Programmable two level priority arbitration scheme
* Configurable ID width to all master interface

## Limitation and Restrictions

None

## Block Diagram

Upstream Port 0

Upstream Port 1

Downstream Port 1

Downstream Port 2

Internal Slave

Master 0

Master 1

Slave 1

Slave 2

ATCBMC300

# Signal Description

us***x***\_awid[ID\_WIDTH-1:0]

us***x***\_awaddr[ADDR\_WIDTH-1:0]

us***x***\_awlen[7:0]

us***x***\_awsize[2:0]

us***x***\_awburst[1:0]

us***x***\_awlock

us***x***\_awcache[3:0]

us***x***\_awprot[2:0]

us***x***\_awvalid

us***x***\_awready

us***x***\_wdata[DATA\_WIDTH-1:0]

us***x***\_wstrb[DATA\_WIDTH/8-1:0]

us***x***\_wlast

us***x***\_wvalid

us***x***\_wreadyus***x***\_bid[ID\_WIDTH-1:0]

us***x***\_bresp[1:0]

us***x***\_bvalid

us***x***\_bready

us***x***\_arid[ID\_WIDTH-1:0]

us***x***\_araddr[ADDR\_WIDTH-1:0]

us***x***\_arlen[7:0]

us***x***\_arsize[2:0]

us***x***\_arburst[1:0]

us***x***\_arlock

us***x***\_arcache[3:0]

us***x***\_arprot[2:0]

us***x***\_arvalid

us***x***\_arready

us***x***\_rdata[DATA\_WIDTH-1:0]

us***x***\_rid[ID\_WIDTH-1:0]

us***x***\_rresp[1:0]

us***x***\_rlast

us***x***\_rvalid

us***x***\_rready

AXI4 Slave Port

AXI4 Master Port

**ATCBMC300**

aclk aresetn

ds***y***\_awid[ID\_WIDTH+3:0]

ds***y***\_awaddr[ADDR\_WIDTH-1:0]

ds***y***\_awlen[7:0]

ds***y***\_awsize[2:0]

ds***y***\_awburst[1:0]

ds***y***\_awlock

ds***y***\_awcache[3:0]

ds***y***\_awprot[2:0]

ds***y***\_awvalid

ds***y***\_awready

ds***y***\_wdata[DATA\_WIDTH-1:0]

ds***y***\_wstrb[DATA\_WIDTH/8-1:0]

ds***y***\_wlast

ds***y***\_wvalid

ds***y***\_wready

ds***y***\_bid[ID\_WIDTH+3:0]

ds***y***\_bresp[1:0]

ds***y***\_bvalid

ds***y***\_bready

ds***y***\_arid[ID\_WIDTH+3:0]

ds***y***\_araddr[ADDR\_WIDTH-1:0]

ds***y***\_arlen[7:0]

ds***y***\_arsize[2:0]

ds***y***\_arburst[1:0]

ds***y***\_arlock

ds***y***\_arcache[3:0]

ds***y***\_arprot[2:0]

ds***y***\_arvalid

ds***y***\_arready

ds***y***\_rdata[DATA\_WIDTH-1:0]

ds***y***\_rid[ID\_WIDTH+3:0]

ds***y***\_rresp[1:0]

ds***y***\_rlast

ds***y***\_rvalid

ds***y***\_rready

| **Signal Name** | | | **I/O Type** | **Description** | |
| --- | --- | --- | --- | --- | --- |
| **AXI global signals** | | | | | |
| aclk | | I | | AXI bus clock. | |
| aresetn | | I | | AXI bus reset. | |
| **Master AXI write address channel signals** | | | | | |
| us*x*\_awid[ID\_MSB:0] | I | | | | Write address ID tag | |
| us*x*\_awaddr[ADDR\_MSB:0] | I | | | | Write Address | |
| us*x*\_awlen[7:0] | I | | | | Write Length | |
| us*x*\_awsize[2:0] | I | | | | Write burst size | |
| us*x*\_awburst[1:0] | I | | | | Write burst type | |
| us*x*\_awlock | I | | | | Write atomic access | |
| us*x*\_awcache[3:0] | I | | | | Cache type | |
| us*x* \_awprot[2:0] | I | | | | Protection type | |
| us*x*\_awvalid | I | | | | Write address valid | |
| us*x*\_awready | O | | | | Write address ready | |
| **Master AXI write data channel signals** | | | | | |
| us*x*\_wdata[DATA\_MSB:0] | I | | | | Write data bus | |
| us*x*\_wstrb[WSTRB\_MSB:0] | I | | | | Write enable | |
| us*x*\_wlast | I | | | | Last write in a burst | |
| us*x*\_wvalid | I | | | | Write data valid | |
| us*x*\_wready | O | | | | Write data ready | |
| **Master AXI write response channel signals** | | | | | |
| us*x*\_bid[ID\_MSB:0] | O | | | | Write Response ID tag | |
| us*x*\_bresp[1:0] | O | | | | Write response status | |
| us*x*\_bvalid | O | | | | Write response valid | |
| us*x*\_bready | I | | | | Write response ready | |
| **Master AXI read address channel signals** | | | | | |
| us*x* \_arid[ID\_MSB:0] | I | | | | Read address ID tag | |
| us*x*\_araddr[ADDR\_MSB:0] | I | | | | Read address | |
| us*x*\_arlen[7:0] | I | | | | Read length | |
| us*x*\_arsize[2:0] | I | | | | Read burst size | |
| us*x*\_arburst[1:0] | I | | | | Read burst type | |
| us*x*\_arlock | I | | | | Read atomic access | |
| us*x*\_arcache[3:0] | I | | | | Cache type | |
| us*x*\_arprot[2:0] | I | | | | Protection type | |
| us*x*\_arvalid | I | | | | Read address valid | |
| us*x*\_arready | O | | | | Read address ready | |
| **Master AXI read data channel signals** | | | | | |
| us*x*\_rid[ID\_MSB:0] | I | | | | Read ID tag | |
| us*x*\_rdata[DATA\_MSB:0] | I | | | | Read data bus | |
| us*x*\_rresp[1:0] | I | | | | Read response | |
| us*x*\_rlast | I | | | | Last read in a burst | |
| us*x*\_rvalid | I | | | | Read data valid | |
| us*x*\_rready | O | | | | Read data ready | |
| **Slave AXI write address channel signals** | | | | | |
| ds*y*\_awid[ID\_MSB+4:0] | O | | | | Write address ID tag | |
| ds*y*\_awaddr[ADDR\_MSB:0] | O | | | | Write Address | |
| ds*y*\_awlen[7:0] | O | | | | Write Length | |
| ds*y*\_awsize[2:0] | O | | | | Write burst size | |
| ds*y*\_awburst[1:0] | O | | | | Write burst type | |
| ds*y*\_awlock | O | | | | Write atomic access | |
| ds*y*\_awcache[3:0] | O | | | | Cache type | |
| ds*y*\_awprot[2:0] | O | | | | Protection type | |
| ds*y*\_awvalid | O | | | | Write address valid | |
| ds*y*\_awready | I | | | | Write address ready | |
| **Slave AXI write data channel signals** | | | | | |
| ds*y*\_wdata[DATA\_MSB:0] | O | | | | Write data bus | |
| ds*y*\_wstrb[WSTRB\_MSB:0] | O | | | | Write enable | |
| ds*y*\_wlast | O | | | | Last write in a burst | |
| ds*y*\_wvalid | O | | | | Write data valid | |
| ds*y*\_wready | I | | | | Write data ready | |
| **Slave AXI write response channel signals** | | | | | |
| ds*y*\_bid[ID\_MSB+4:0] | I | | | | Write Response ID tag | |
| ds*y*\_bresp[1:0] | I | | | | Write response status | |
| ds*y*\_bvalid | I | | | | Write response valid | |
| ds*y*\_bready | O | | | | Write response ready | |
| **Slave AXI read address channel signals** | | | | | |
| ds*y*\_arid[ID\_MSB+4:0] | O | | | | Read address ID tag | |
| ds*y*\_araddr[ADDR\_MSB:0] | O | | | | Read address | |
| ds*y*\_arlen[7:0] | O | | | | Read length | |
| ds*y*\_arsize[2:0] | O | | | | Read burst size | |
| ds*y*\_arburst[1:0] | O | | | | Read burst type | |
| ds*y*\_arlock | O | | | | Read atomic access | |
| ds*y*\_arcache[3:0] | O | | | | Cache type | |
| ds*y*\_arprot[2:0] | O | | | | Protection type | |
| ds*y*\_arvalid | I | | | | Read address valid | |
| ds*y*\_arready | I | | | | Read address ready | |
| **Slave AXI read data channel signals** | | | | | |
| ds*y*\_rid[ID\_MSB+4:0] | O | | | | Read ID tag | |
| ds*y*\_rdata[DATA\_MSB:0] | O | | | | Read data bus | |
| ds*y*\_rresp[1:0] | O | | | | Read response | |
| ds*y*\_rlast | O | | | | Last read in a burst | |
| ds*y*\_rvalid | O | | | | Read data valid | |
| ds*y*\_rready | I | | | | Read data ready | |

# Functions

* The response ordering policy
  + If the response of downstream port connected slave device is in-order, the ATCBMC300\_SLV***y***\_FIFO\_DEPTH can be any legal value for highest performance.
  + If the response of downstream port connected slave device is out-of-order, the ATCBMC300\_SLV***y***\_FIFO\_DEPTH must be 1 to prevent deadlock.
  + The upstream port returns responses in-order

# Design Configuration

## Global Configuration

### Address Width

Define the following macro to set the address width. The address width range is from 24 to 64. Default address width is 32-bit.

`define ATCBMC300\_ADDR\_WIDTH 32

### Data Width

Define the following macro in atcbmc300\_config.vh to set the data width to 32-bit, 64-bit or 128-bit. Default data width is 32-bit.

`define ATCBMC300\_DATA\_WIDTH 32

### ID Width

Define the following macro to set the upstream ID port width. The downstream ID port width is `ATCBMC300\_ID\_WIDTH + 4 to hint the upstream port source. Default upstream ID port width is 4-bit.

`define ATCBMC300\_ID\_WIDTH 4

## Upstream Port Configuration

### Enabling Upstream Port Configuration

Define ATCBMC300\_MST***x***\_SUPPORT to enable the ***x***th AXI upstream port, where ***x***=0~15. For example,

`define ATCBMC300\_MST0\_SUPPORT

`define ATCBMC300\_MST4\_SUPPORT

:

### Outstanding Depth Configuration

Each master has the ATCBMC300\_MST***x***\_OUTSTANDING\_DEPTH macro to define the maximum number of outstanding request acceptable. The legal value can be 2,4,8,…,2***n***(where ***n***>=1). For example,

`define ATCBMC300\_MST0\_OUTSTANDING\_DEPTH 4

`define ATCBMC300\_MST4\_OUTSTANDING\_DEPTH 4

### Reset Priority Value Configuration

Each master has ATCBMC300\_MST***x***\_DEFAULT\_PRIORITY\_RELOAD define macro to predefine the ownership **PReload** bit register content after reset pin release. The ATCBMC300\_MST0\_DEFAULT\_HIGH\_PRIORITY define macro predefine the **PHigh0** register content.

`define ATCBMC300\_MST0\_DEFAULT\_HIGH\_PRIORITY 0

`define ATCBMC300\_MST0\_DEFAULT\_PRIORITY\_RELOAD 1

`define ATCBMC300\_MST1\_DEFAULT\_PRIORITY\_RELOAD 1

## Downstream Port Configuration

### Enabling Downstream Port Configuration

Define ATCBMC300\_SLV***y***\_SUPPORT to enable the ***y***th AXI downstream port, where ***y***=1~31. For example,

`define ATCBMC300\_SLV1\_SUPPORT

`define ATCBMC300\_SLV5\_SUPPORT

:

### Downstream Port FIFO Depth Configuration

The ATCBMC300\_SLV***y***\_FIFO\_DEPTH macro value indicates the maximum number of outstanding request can be issued by the ***y***th AXI downstream port, where ***y***=1~31. The legal value can be 1,2,4,8,…,2***n***(where ***n***>=0). If the response of connected slave device is in-order, this macro value can be more than 1 for higher performance. Otherwise, if the response of connected slave device can be out-of-order, this macro value should be 1 to prevent deadlock. For example,

`define ATCBMC300\_SLV1\_FIFO\_DEPTH 4

`define ATCBMC300\_SLV5\_FIFO\_DEPTH 4

### Base Address and Space Size

This parameter defines the base address and space size of AXI slave ***y*** by defining ATCBMC300\_AXI\_SLV***y***\_BASE\_ADDR and ATCBMC300\_SLV***y***\_SIZE, where ***y*** is ranging from 1 to 31. If the Slave ***y*** is non-instanced, it does not need to set the corresponding ATCBMC300\_AXI\_SLV***y***\_BASE\_ADDR and ATCBMC300\_SLV***y***\_SIZE define macro.

There are some restrictions:

* Avoid to let multiple slaves map same address
* Table 9 shows the ATCBMC300\_SLV***y***\_SIZE number mapped address size. The setting range is “(`ATCBMC300\_ADDR\_WIDTH defined full address size)/2 >= ATCBMC300\_SLV***y***\_SIZE > 0”.
* The ATCBMC300\_SLV***y***\_BASE\_ADDR valid bit field range are:
  + The bit width of ATCBMC300\_SLV***y***\_BASE\_ADDR must equal to the address width (`ATCBMC300\_ADDR\_WIDTH).
  + ATCBMC300\_SLV***y***\_BASE\_ADDR must be aligned to the size of slave (indicated by `ATCBMC300\_SLV***y***\_SIZE).
  + Otherwise, the outside bit of ATCBMC300\_SLV***y***\_BASE\_ADDR valid bit field will be ignored.

For example, if ATCBMC300\_ADDR\_WIDTH is 32, the full address size is 4GB. The largest slave address size is 2GB. If the SLV***y*** size is 16MB, the valid bit field of ATCBMC300\_SLV***y***\_BASE\_ADDR is from bit24 to bit31.

The following shows an example to define a 16MB space to AXI slave 5 at the 32’h9030\_0000 base address:

`define ATCBMC300\_SLV5\_BASE\_ADDR `ATCBMC300\_ADDR\_WIDTH’h9030\_0000

`define ATCBMC300\_SLV5\_SIZE 5

## Upstream and Downstream Ports Connectivity

Define ATCBMC300\_MST***x\_***SLV***y*** macro to enable the connection between AXI master ***x*** and slave ***y***. For example, master0 (MST0) is connecting to slave0 (SLV0), slave1 (SLV1), slave2 (SLV2); master1 (MST1) is connecting to slave0 (SLV0), slave2 (SLV2); master3 (MST3) is connecting to slave1 (SLV1), slave2 (SLV2); the following shall be defined:

`define ATCBMC300\_MST0\_SLV0

`define ATCBMC300\_MST0\_SLV1

`define ATCBMC300\_MST0\_SLV2

`define ATCBMC300\_MST1\_SLV0

`define ATCBMC300\_MST1\_SLV2

`define ATCBMC300\_MST3\_SLV1

`define ATCBMC300\_MST3\_SLV2

:

## Product ID and Revision

`define ATCBMC300\_PRODUCT\_ID 24’h0000\_30

`define ATCBMC300\_REV\_MAJOR 4’h0

`define ATCBMC300\_REV\_MINOR 4’h0

# Programming Model

## Summary of Registers

Table 4 shows the address offset, type, and its description of ATCBMC300 programming registers.

Table . ATCBMC300 register summary

| Address | Type | Description |
| --- | --- | --- |
| +0x00 | R | ID and revision register |
| +0x04~0x0c | - | Reserved |
| +0x10 | R/W | priority register |
| +0x14 ~ 0x0fc | - | Reserved |
| +0x100+(*y*\*0x8) | R | AXI slave *y* base/size register (*y*=0~31) |

## Register Description

### ID and Revision Register (0x00)

Table . ID and revision register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Bit | Type | Description | Reset |
| ID | 31:8 | R | ID number for ATCBMC300 | 0x000030 |
| RevMajor | 7:4 | R | Major revision number | ATCBMC300\_REV\_MAJOR |
| RevMinor | 3:0 | R | Minor revision number | ATCBMC300\_REV\_MINOR |

### Priority (0x10)

ATCBMC300 supports two-level priority scheme to arbitrate master requests. Three modes are provided by setting **PReload** field. The initial value of **PReload** is 0xFFFF which indicates using round-robin mode by default. Set **PHigh0** to indicate that master 0 always has the highest priority. ATCBMC300 internal priority status register will be reloaded to **PReload** value as soon as **PReload** is updated. See section 4 for two-level priority scheme details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Bit | Type | Description | Reset |
| PHigh0 | 31 | R/W | Master 0 is the highest priority master  0: Disable  1: Enable | Depend on the configuration |
| - | 30:16 | - | Reserved | - |
| PReload | 15:0 | R/W | PReload value while PStatus meets the reload condition, default value 0xFFFF indicates arbitration in round-robin mode | Depend on the configuration |

### Base/Size Register of AXI Slave *y* (0x100+(*y*\*0x8))

This register configures the base address/space size of AXI slave ***y***. The values of the Base/Size register of all the slaves that have the same format and same definition as the fields. Either the “Base” or “Size” fields are READ ONLY of AXI slave ***y***. Table 9 shows the bit assignment.

Table . Base/Size registers of AXI slave *y*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Bit | Type | Description | Reset |
| Base High Part Register(0x104+(*y*\*0x8)) | | | | |
| Base\_High | 31:0 | R | Base address [63:32] | Depend on the configuration |
| Base Low Part/Size Register(0x100+(*y*\*0x8)) | | | | |
| Base\_Low | 31:20 | R | Base address [31:20] | Depend on the configuration |
| - | 19:8 | - | Reserved | - |
| Size | 7:0 | R | 0x00: No slave  0x01~0x32: See Table9  0x33~0xFF: Reserved | Depend on the configuration |

Table . Size register value mapped the size of AXI slave*y*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Size Unit  Size Num. | MB(2^20) | GB(2^30) | TB(2^40) | PB(2^50) | EB(2^60) |
| 1 | 1 | 11 | 21 | 31 | 41 |
| 2 | 2 | 12 | 22 | 32 | 42 |
| 4 | 3 | 13 | 23 | 33 | 43 |
| 8 | 4 | 14 | 24 | 34 | 44 |
| 16 | 5 | 15 | 25 | 35 |  |
| 32 | 6 | 16 | 26 | 36 |  |
| 64 | 7 | 17 | 27 | 37 |  |
| 128 | 8 | 18 | 28 | 38 |  |
| 256 | 9 | 19 | 29 | 39 |  |
| 512 | 10 | 20 | 30 | 40 |  |

# Micro-Architecture

## Address Channel Data Path

Pending Buffer

AR/AW

slv***m***\_avalid

Comparator

Slave 0base address

:

Slave ***y***base address

us\_araddr/us\_awaddr

AR/AW

ARID/AWID\_mid

Arbiter

m0\_avalid….m***x***\_avalid

m0\_addr…….m***x***\_addr

Master\_ID

us\_addr\_ctrl

ds\_addr\_ctrl

Slave\_ID

### Upstream Address Channel

The upstream address channel controller does 4 stuffs.

1. Receiving the read/write address from AXI master.
2. Comparing address with all connected slave base address and size define macro
3. Sending the mapped address request to downstream port.
4. If no slave can be mapped, the DECERR response will be responded directly

The address comparing logic is:

localparam SLV0\_SIZE = `ATCBMC300\_SLV0\_SIZE;

assign slv0\_addr\_mask = {{(65-(`ATCBMC300\_SLV0\_SIZE+19)){1'b1}},{(`ATCBMC300\_SLV0\_SIZE+19){1'b0}}} & {{(65-`ATCBMC300\_ADDR\_WIDTH){1'b0}},{`ATCBMC300\_ADDR\_WIDTH{1'b1}}};

assign slv0\_masked\_base\_addr = {{(65-`ATCBMC300\_ADDR\_WIDTH){1'b0}},`ATCBMC300\_SLV0\_BASE\_ADDR} & slv0\_addr\_mask;

assign slave\_hit[0] = master\_arvalid & slv0\_connect & (slv0\_masked\_base\_addr==({{65-`ATCBMC300\_ADDR\_WIDTH{1'b0}},master\_araddr} & slv0\_addr\_mask));

### Downstream Address Channel

Each address channel of downstream port does 3 stuffs.

1. Receiving the address requests from upstream port.
2. Arbitrating one address requests
3. Sending the selected address to AXI slave.

The address request selection has 2 phase arbitration

1. **Priority level request selection**

ATCBMC300 has **PHigh0** and **PReload** register to rule the address request priority level. The priority level rule is:

1. The address requests with the value 0 **PReload** are the lowest priority level.
2. The address requests with the value 1 **PReload** are the third high priority level. At this moment, the not granted address requests with the value 1 **PReload** will be logged as the pending reloaded address requests.
3. The pending reloaded address requests are second high priority level
4. The MST0 address request with value 1 **PHigh0** is the highest priority level.

always@(posedge aclk or negedge aresetn)

if(~aresetn)

mst0\_priority\_avalid <= 1'b0;

else

mst0\_priority\_avalid <= mst0\_connect &

~((arb\_mid==4'd0) & slv\_aready) &

((~pending\_priority\_avalid & mst0\_avalid & reg\_priority\_reload[0])

| mst0\_priority\_avalid);

priority\_avalid[0] = mst0\_priority\_avalid;

assign pending\_priority\_avalid = priority\_avalid[15:0]!=16'h0;

assign arb\_avalid = (reg\_mst0\_high\_priority & mst\_avalid[0]) ? 16'b1 :

pending\_priority\_avalid ? priority\_avalid :

((mst\_avalid & reg\_priority\_reload)!=16'h0) ? (mst\_avalid & reg\_priority\_reload) : mst\_avalid;

1. **Priority encoder**

After priority level selection, all selected address requests will be encoded by fix priority. The encoding priority is “MST0 (highest)>MST1>….>MST15 (lowest)”.

assign arb\_mid[3] = (~|arb\_avalid[7:0]);

assign arb\_mid[2] = arb\_mid[3] ? (~|arb\_avalid[11:08]) : (~|arb\_avalid[03:00]);

assign arb\_mid[1] =

(arb\_mid[3:2]==2'h3 & (~|arb\_avalid[13:12]))|

(arb\_mid[3:2]==2'h2 & (~|arb\_avalid[09:08]))|

(arb\_mid[3:2]==2'h1 & (~|arb\_avalid[05:04]))|

(arb\_mid[3:2]==2'h0 & (~|arb\_avalid[01:00]));

assign arb\_mid[0] =

(arb\_mid[3:1]==3'h7 & ~arb\_avalid[14]) |

(arb\_mid[3:1]==3'h6 & ~arb\_avalid[12]) |

(arb\_mid[3:1]==3'h5 & ~arb\_avalid[10]) |

(arb\_mid[3:1]==3'h4 & ~arb\_avalid[08]) |

(arb\_mid[3:1]==3'h3 & ~arb\_avalid[06]) |

(arb\_mid[3:1]==3'h2 & ~arb\_avalid[04]) |

(arb\_mid[3:1]==3'h1 & ~arb\_avalid[02]) |

(arb\_mid[3:1]==3'h0 & ~arb\_avalid[00]) ;

## Write Data Channel Data Path

us\_wdata\_ctrl

Pending Buffer

slv0\_wready….slv***y***\_ready mst\_wdata

ds\_wdata

FIFO

m0\_wvalid….m***x***\_wvalid

m0\_wdata…m***x***\_wdata

Master\_ID

ds\_wdata\_bresp\_ctrl

ds\_wvalid

wmid

us\_wdata

pending\_buffer

m0\_bready….m***x***\_bready slv\_bresp

Slave\_ID

FIFO

ds\_bresp

bmid

## Read Data/BRESP Channel Data Path

FIFO

Slave\_ID

slv0\_rd/br….slv***y***\_rd/br

m0\_rready….m***x***\_rready slv\_rdata

ds\_rdata

ds\_rdata\_ctrl

Pending Buffer

us\_resp\_ctrl

us\_resp\_data

rmid

## Internal Slave

The internal slave is for accessing register file. The internal slave has the following features

* The internal slave is always enabled
* The base address of internal slave is configurable
* The size of internal slave is always 1MB
  + It means that the accessing response of this address space accessing is always OK.
* The internal slave does not support exclusive access
* The connection between internal slave and each master is configurable
  + If the internal slave disconnect to all masters, no master can access the internal slave content.

awid,awaddr

wmid

Arbiter

m0\_awvld m***n***\_awvld

m0\_awaddr m***n***\_awaddr

m0\_wvalid, m***n***\_wvalid,

m0\_wlast, m***n***\_wlast,

m0\_wdata ……. m***n***\_wdata

m0\_wstrb m***n***\_wstrb

mo\_bready m***n***\_bready

bid slv0\_awready slv0\_wready slv0\_bvalid bresp

2’b00

registers

slv0\_wready

slv0\_bvalid

arid,arddr

rmid

Arbiter

m***n***\_arvld m***n***\_awvld

m0\_araddr m***n***\_araddr

mo\_rready m***n***\_rready

slv0\_rvalid

rid rrdata rresp arready rvalid rlast

2’b00

rcnt

m0\_arlen m***n***\_arlen

registers