

Application Note 169

Using the L210 Cache Controller with ARM7 and ARM9 Cores

Released on: 2nd June, 2006



Application Note 169

Using the L210 Cache Controller with ARM7 and ARM9 Cores

Copyright © 2006. All rights reserved.

Release Information

Table 1 Change history

Date	Issue	Change
June 2006	A	First release
June 2007	B	Updated release to clarify some sections

Proprietary Notice

Words and logos marked with © and ™ are registered trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

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>

1 Overview

This Application Note describes some compatibility considerations when using the cache controller with ARM7 cores and ARM9 cores. It contains the following sections:

- Connecting the L210 to the ARM720T rev4 processor core in chapter 2
- Connecting the L210 to ARM9 processor cores in chapter 3
- Connecting the L210 to a v5 memory system in chapter 4

For information regarding connection to any ARM7 or ARM9 core not detailed in this document, please contact ARM Support.

2 Using the L210 with an ARM720T rev4 core

This chapter describes how to use the L210 with an ARM720T rev 4 processor core.

2.1 General Connections

The ARM720T has a single AHB port. This should be connected to slave 1 of the L210. Therefore you should:

Tie off L210 Slave Port 0 and 2 pins

- HBSTRBSx[7:0] must be tied HIGH
- HPROTSx[4] must be tied to HPROTSx[3]
- HPROTSx[5] must be tied LOW
- HUNALIGNsx must be tied LOW
- HRESPMx[2] must be tied LOW.

Tie off unused pins on slave ports 1

- Read data is on HRDATA[31:0], and HRDATA[63:32] is always 0
- Write data must be duplicated on HWDATA[63:32] and HWDATA[31:0].

Tie off unused pins on master port 1

- Write data is on HWDATA[31:0], and HWDATA[63:32] is always 0.
- Read data must be duplicated on HRDATA[63:32] and HRDATA[31:0].

2.2 Special Considerations

The way ARM720T rev4 cores drive the **HPROT** signals on the AHB imposes some restrictions when using the cache controller with ARM720T rev4 cores.

To ensure memory consistency when WT memory accesses are used, the cache controller must be forced to treat all cacheable memory accesses as WT, allocate on read miss.

- To do this, write to the cache controller Debug Control Register, register 15, and set bit 1 to 1'b1. This forces the cache controller to treat all cacheable accesses as WT, allocate on read miss. See *L210 Debug Control Register* on page 2-22 of the L210 Technical Reference Manual.
- If the above approach is not acceptable, it can be done in hardware by implementing a piece of logic between the ARM720T rev4 core and the cache controller to detect HPROT=WB and map it to HPROT=WT at L2. The following is an example:

```

if (HPROT[3:2] == 2'b11)
    HPROTSn[3:2] = 2'b10; // WB mapped to WT
else
    HPROTSn[3:2] = HPROT[3:2]; // WT, NCNB, NCB unaltered

```

————— Note —————

This approach still enables the write allocate override bit to be set in the Auxiliary Control Register. See *L210 Auxiliary Control Register* on page 2-9 of the L210 Technical Reference Manual.

This may cause problems if legacy software assumes WT with read allocate behavior, when WT with read-allocate and write allocate is possible.

3 Using the L210 with ARM9 Processor Cores

This chapter describes how to use the L210 with an ARM9 processor cores.

3.1 ARM920T and ARM922T cores

This section describes how to use the L210 with the ARM920T and ARM922T processor cores.

General Connections

The ARM920 and ARM922T processor cores have a single ASB. These must be converted to AHB by using an ARM920/ARM922T AHB wrapper. The resulting single AHB interface should be connected to slave 1 of the L210. Therefore you should:

Tie off L210 Slave Port 0 and 2 pins

- HBSTRBSx[7:0] must be tied HIGH
- HPROTSx[4] must be tied to HPROTSx[3]
- HPROTSx[5] must be tied LOW
- HUNALIGNsx must be tied LOW
- HRESPMx[2] must be tied LOW.

Tie off unused pins on slave ports 1

- Read data is on HRDATA[31:0], and HRDATA[63:32] is always 0
- Write data must be duplicated on HWDATA[63:32] and HWDATA[31:0].

Tie off unused pins on master port 1

- Write data is on HWDATA[31:0], and HWDATA[63:32] is always 0.
- Read data must be duplicated on HRDATA[63:32] and HRDATA[31:0].

Special Considerations

When using the cache controller with ARM920 or ARM922 cores the AHB wrapper for the ARM920 or ARM922 core maps some cacheable stores to being non-cacheable. Re-mapping logic is required between the AHB wrapper and the cache controller:

```
// Maps all cacheable reads and all bufferable writes
// to WT
HPROTSn[3] = HPROT[3] or HPROT[2]
HPROTSn[2] = 1'b0
```

In addition you must write to L210 Debug Control Register, register 15, and set bit 1 to 1'b1. This forces the L210 to treat all cacheable accesses as WT, allocate on read miss. See *L210 Debug Control Register* on page 2-22 of the L210 Technical Reference Manual.

3.2 ARM926EJ cores (revisions r0p0-r0p5)

This section describes how to use the L210 with the ARM926EJ-S processor core and derivatives.

General Connections

The ARM926EJ-S has dual AHB ports. These should be connected to slaves 0 and 1 of the L210. Therefore you should:

Tie off L210 Slave Port 2 pins

- HBSTRBSx[7:0] must be tied HIGH
- HPROTSx[4] must be tied to HPROTSx[3]
- HPROTSx[5] must be tied LOW
- HUNALIGNSx must be tied LOW
- HRESPMx[2] must be tied LOW.

Tie off unused pins on slave ports 0 and 1

- Read data is on HRDATA[31:0], and HRDATA[63:32] is always 0
- Write data must be duplicated on HWDATA[63:32] and HWDATA[31:0].

Tie off unused pins on master ports 0 and 1

- Write data is on HWDATA[31:0], and HWDATA[63:32] is always 0.
- Read data must be duplicated on HRDATA[63:32] and HRDATA[31:0].

Special Considerations

The way ARM926EJ cores drive the HPROT signals on the AHB imposes some restrictions when using cache controller with ARM926EJ cores.

When using the cache controller with ARM926EJ cores:

- Before disabling the L1 cache, first clean and invalidate, and disable the cache controller cache.
- All page tables must reside in a WT memory region.
- ARM926EJ does not distinguish between WT and WB accesses at L2. By default the cache controller treats all cacheable accesses by ARM926EJ as WB. To ensure memory consistency when WT memory accesses are used, the cache controller must be forced to treat all cacheable memory accesses as WT.

— To do this write to L210 Debug Control Register, register 15, and set bit 1 to 1'b1. This forces the cache controller to treat all cacheable accesses as WT, allocate on read miss. See *L210 Debug Control Register* on page 2-22 of the L210 Technical Reference Manual.

If the above approach is not acceptable, it can be done in hardware by implementing a piece of logic between the ARM926EJ core and the cache controller to detect HPROT = WB and map it to HPROT = WT at L2. The following is an example:

```
if (HPROT[3:2] == 2'b11)
    HPROTSn[3:2] = 2'b10; // WB mapped to WT
else
    HPROTSn[3:2] = HPROT[3:2]; // WT, NCNB, NCB unaltered
```

ARM926EJ-S ITCM and L210 incompatibility

A separate errata document deals with ARM926EJ-S ITCM and L210 Cache Controller incompatibility:

*ARM926EJ-S DHPROT incorrect when the instruction TCM is accessed,
ARM926EJ-GENC-003141*

3.3 ARM946E cores

This section describes how to use the L210 with the ARM946E-S processor core and derivatives.

General Connections

The ARM946E-S has a single AHB port. This should be connected to slave 1 of the L210. Therefore you should:

Tie off L210 Slave Port 0 and 2 pins

- HBSTRBSx[7:0] must be tied HIGH
- HPROTSx[4] must be tied to HPROTSx[3]
- HPROTSx[5] must be tied LOW
- HUNALIGNSx must be tied LOW
- HRESPMx[2] must be tied LOW.

Tie off unused pins on slave ports 1

- Read data is on HRDATA[31:0], and HRDATA[63:32] is always 0
- Write data must be duplicated on HWDATA[63:32] and HWDATA[31:0].

Tie off unused pins on master port 1

- Write data is on HWDATA[31:0], and HWDATA[63:32] is always 0.
- Read data must be duplicated on HRDATA[63:32] and HRDATA[31:0].

Special Considerations

When using the cache controller with ARM946 cores:

- Before disabling the L1 cache, first clean and invalidate, and disable the cache controller cache.
- Write to L210 Debug Control Register, register 15, and set bit 1 to 1'b1. This forces the cache controller to treat all cacheable accesses as WT, allocate on read miss. See *L210 Debug Control Register* on page 2-22 of the L210 Technical Reference Manual.

4 Connecting the L210 to a v5 Memory System

There are two possible types of Memory system that you can implement your L2CC in:

- v5 Memory System - This is the common name given to the AMBA AHB 2 protocol
- v6 Memory System - This is the common name given to the AMBA AHB 2 with ARM11 extensions. For more information about this type of system, please see the ARM11 documentation.

The L2CC was designed to interface with a v6 Memory System. For most effective implementation, you should strongly consider the use of the ARM v6 AMBA Extensions within your memory system connected to the master ports of the L2CC.

It is possible to implement the L2CC in a v5 (straight AMBA AHB 2 protocol) Memory System using a Byte Lane Strobe Converter module, with some loss of L2CC performance.

4.1 Recommended Approach: Use a v6 Memory System

Slaves within the v6 Memory System must be able to correctly interpret the v6 AHB Extensions. In most cases, slaves can very easily be adapted to ensure that they are v6 compliant.

HUNALIGN - this signal indicates that the access is unaligned or sparse (where some bytes are missing from the access). With the ARM7 and ARM9 processor cores connected to the L2CC slave ports, only writes will assert this signal, and when it is not asserted the slave can behave as normal (v5). When this signal is asserted, the slave must monitor other v6 signals to determine what is required.

HBSTRB - this bus indicates which of the byte lanes is valid. Since the L2CC will only assert HUNALIGN on buffered writes when connected to the ARM7 and ARM9 cores, this bus indicates which of the byte lanes contain real data to be written, and which do not (and thus, the byte in memory should not be updated).

4.2 Using the L2CC with a Byte Lane Strobe Converter and v5 Memory System

When connected to ARM7 and ARM9 processor cores, the L2CC will perform reads and non-buffered writes as straight AHB 2 accesses (HUNALIGN = 1'b0).

However, the write buffer in the L2CC is a merging buffer. Data from any number of writes on the same line (within an 8 word boundary) will be merged together into a single write from the L2CC. Because of the likelihood of sparse data within the write buffer, the L2CC performs all write buffer drains as unaligned (v6 AHB Extensions).

There is no way to switch off the merging capability of the write buffer within the L2CC.

As has already been mentioned, the L2CC can be used in an existing v5 Memory System by using a Byte Lane Strobe Converter. This block takes the v6 style outputs of the L2CC and converts the signals and accesses into a v5 style AHB.

What is the Byte Lane Strobe converter?

As supplied by ARM Ltd., the BLS Converter is an AHB master gasket module (i.e. it is not a bridge - it acts as a transparent converter), which converts v6 type AHB accesses to v5 type AHB accesses.

The BLS Converter appears to be transparent when the L2CC is performing standard v5 accesses (that is, when HUNALIGN is 1'b0), but when HUNALIGN is asserted intervenes and converts the possibly unaligned or sparse accesses to byte accesses. In a 32-bit system every full word access that is marked with HUNALIGN is converted to 4 byte accesses and hence the transfers using the BLS Converter decreases down to about ¼ speed.

Because the BLS has no visibility of the future, and because the L2CC can perform a seemingly v5 burst (aligned and complete) from its write buffer with HUNALIGN set high, using the L2CC with the BLS Converter and a v5 Memory System can reduce the efficiency of the system dramatically.

4.3 AHB Slave Capabilities

It is worth noting at this point that the L2CC performs different types of accesses than the connected ARM processor core under some conditions, including extensive use of the BUSY transfer type and INCR burst types. In order for your existing slaves to work with the L2CC they must support the full AHB 2 specification.

The L2CC will produce BUSY transfer types when it needs to hold off the master from producing any further access so that the response from the slave can be correctly asserted to the master in time for the address that caused it.

There is no way of removing these BUSY transfers since if they are removed then any ERROR being returned from the slave would not be back to the master in time to indicate the ERROR to the correct address. The result of this would produce software errors (where the software is being told by the system that a particular address caused an error, when in fact it was a different address).

4.4 Placement of the Event Monitor

There are no special requirements on the placement of the Event Monitor block - it need not go on a 64-bit master, nor after the BLS Converter block (if one exists in your system).

You should remember that the Event Monitor needs to be in a NCNB 4kB region, which is not the same region as the L2CC configuration region.

4.5 Other Connection Points

The master ports on the L2CC are AHB-Lite implementations, as the intention was for the L2CC to be used in a Multi-Layer AHB system. If you wish to place the L2CC in a system with other masters you will need to incorporate an AHB-Lite to AHB converter module.