Document Number：UM026

Issued：September 2009

Revision History

Revision Number	Revision Date	Author	Revised Chapters/Sections	Description
V 1.4	9/25/2009	CH/Wilson	Section 7.2, 9.3.1, 9.3.5, 9.7.5, 9.7.7, 9.9.2.2	1. Sync to internal v6.4.
V 1.3	5/13/2009	CH/Wilson	Section 9.2.7, 9.8.2, 9.10	1. Added Audio and FPU-related system registers.
V 1.2	12/31/2008	CH/Wilson	Section 2.11, 4.1, 9.2.6	1. Extended COREID field up to 7 bits. 2. Added next-precise exception type for data watchpoint. 3. Added mixed endian support and new device register endian control mechanism descriptions.
V1.1	6/13/2008	CH/Wilson	Section 2.2.2.1, 2.2.2.2, 6.1, 9.10	1. Added system register existence summary table. 2. Added a new local memory base address alignment scheme. 3. Added MMU version 2. 4. Added NTM to Non-translated attribute.
V 1.0RB	4/2/2008	CH/Wilson	Section 9.8	1. Enhanced descriptions for several sections. 2. Added PRUSR_ACC_CTL register.
V 1.0	7/1/2007	CH/Wilson		First version

Table of Contents

List of Tables

List of Figures

Preface

This preface describe the contents of this manual

About this manual

This manual provide the information about AndeStar™ system privileged architecture and all information contains in this document is subject to change without notice

Version of AndeStar™ SPA manual

This manual is version 1.4.

Contact information

Please contact Andes Technology Corporation by email at support@andestech.com or on the Internet at www.andestech.com for support.

Copyright notice

Chapter 1
Introduction

This document describes the privileged system architecture resources and mechanisms the AndesCore™ will provide to the system software for flexible and efficient management of user processes and system processes themselves.

It covers the following topics:

- Memory Management Unit (Section Chapter 2)
- Memory Protection Unit (Section Chapter 3)
- Interruption Architecture (Section Chapter 4)
- Performance Monitoring mechanism (Section Chapter 5)
- Local Memory support (Section Chapter 6)
- Local Memory DMA support (Section Chapter 7)
- High Speed Memory Port support (Section Chapter 8)

And all the system registers which control the behaviors of the above mentioned mechanisms are summarized in the System Register Definitions (Section Chapter 9) of this document.

Chapter 2

Memory Management Unit with Non-Fully-Associative TLB

This chapter describes memory management unit and TLB operations and contains the following sections

2.1. Introduction

Andes Memory Management Unit (MMU) is responsible for managing physical memory resources using virtual page to physical page address translations. It provides address space protection among different processes (running programs) and provides memory page attributes such as cacheability, coherency, etc. for system performance enhancement.

The Andes MMU's main component is a Translation Look-aside Buffer (TLB) structure. It can be completely managed by software or a hardware-assisted page table walker (HPTWK) can be enabled to improve the TLB fill performance and/or the TLB capacity.

The exact organization of the TLB structure is implementation-dependent. A specific TLB implementation will be provided in Appendix A for reference. However, from the software's point of view, the Andes TLB structure consists of two parts: a software-visible part and a software-invisible part. The software-visible part of the TLB is a non-fully-associative cache memory with N sets and K ways, thus N*K=M entries. All the *M* entries can be managed by software using targeted TLB read/write, probe, and flush operations. Another TLB random write operation is also provided to insert a page translation into a hardware-determined entry to relieve software from the responsibility of managing TLB indexes. The software-invisible part of the TLB is implementation-dependent. And once it exists, software needs to manage it using probe and flush operations, but not the TLB read/write operations.

The relationship between the software-visible and software-invisible part of the TLB is implementation-dependent. The software-invisible part of the TLB could contain a subset of translation information of the software-visible part of the TLB. Or the software-invisible part of the TLB could contain a completely different set of translation information from the software-visible part of the TLB.

The relationship between the HPTWK and the two parts of the TLB structure is implementation-dependent as well. The HPTWK can be implemented to insert translation information into only the software-invisible part of the TLB or it can be implemented to

insert the information into the software-visible part of the TLB. The HPTWK can be enabled by setting a bit in the MMU L1_PPTB register. In order to use the HPTWK, the operating system has to prepare a physical page table (PPT) in a fixed format in order for the HPTWK to find the VA to PA translation (i.e. Page Table Entry) from the page table. In the current architecture, the page table format supports only a single small page size and a large page size which is an integer multiple of the small page size.

2.2. Address spaces in Andes MMU

There are three addressing spaces in an Andes memory system: the virtual addresses (VA), the physical addresses (PA), and the system address, which correspond to the three major operating modes for the machine: user mode, superuser mode and hypervisor mode, respectively.

Since this document describes a MMU architecture without hypervisor support, the physical address space is equal to the system address space. And the physical address space is the address that shows up on the system bus throughout in this document.

The characteristics of the virtual and physical address spaces are as follows:

1. The virtual address space is the address space used by a process when fetching instructions (program counter value) and accessing memory using load/store instructions. The virtual address is 32-bit wide, thus the virtual address space is a 4GB space. The user and superuser processes will both generate virtual addresses.
2. The physical address space is the address space managed/controlled by superuser mode processes. Any superuser mode process manages the physical address space by controlling the VA to PA translation policies for user mode processes and sometimes for itself. It can also disable the translation process for superuser and user mode processes respectively. If the translation process is disabled, the virtual address is equivalent to physical address.
3. Typically, the physical address space is the universe the operating system operates. The superuser mode is a protected privilege mode designed for the operating system software to multiplex resources amongst applications. The operating system is responsible for managing the translation of virtual addresses into the physical addresses.

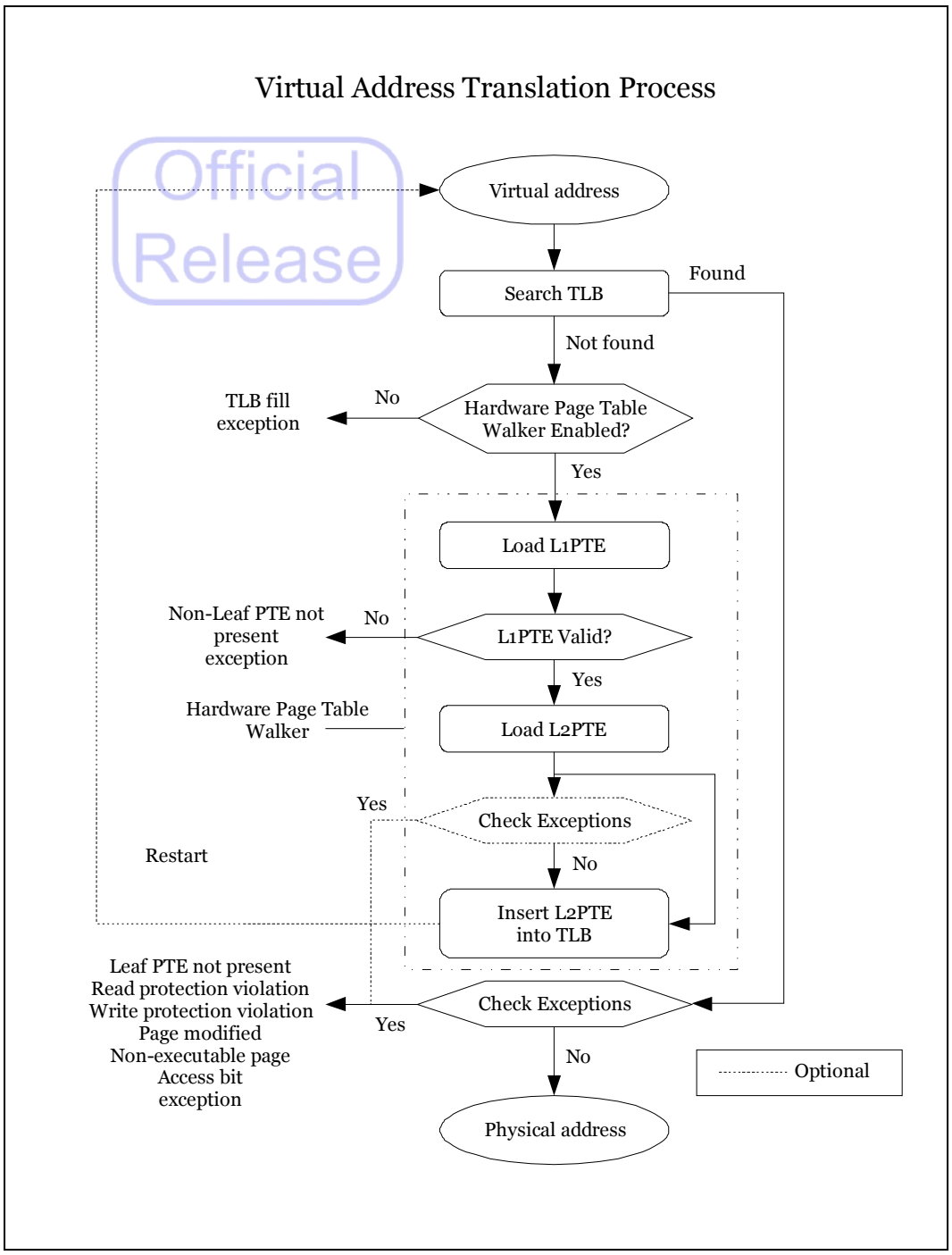
2.2.1. Address translation process

A virtual address must be translated into a physical address before the data could be accessed if the translation process for user mode or superuser mode is enabled.

The virtual address translation starts when the pipeline sends a virtual address to the TLB structure. If a translation is found in the TLB structure, a physical address is generated from the translation information and used to access memory. If the lookup fails, the virtual address is subsequently processed by the HPTWK or by the TLB fill exception handler.

Figure 1 illustrates the complete address translation process.

Figure 1. Complete Virtual Address Translation Process.



2.2.2. Address Space Attributes

Attributes are defined for virtual/physical address spaces for system performance

enhancement, access control, and sharing of program or data. Based on the translation processes, the attributes are defined differently.

2.2.2.1. Attributes for Translated Virtual/Physical Address

For virtual address that translates into physical address, the attributes are defined in the page table entry for each page. And since it is page-based, more fine grained access control can be added to each page. The page attributes are listed in the following tables along with their bit numbers in the page table entry (32 bits). Table 1 lists the attribute definitions for Andes MMU version 1 and Table 2 lists the attribute definitions for Andes MMU version 2. The differences between version 1 and version 2 are in the definitions of the X and the M[3:1] attribute fields. Version 2 allows earlier permission violation detection on the user code executing non-accessible superuser page than that of version 1.

Table 1. Translated Address Space Attributes for MMU version 1.

Field	Bits	Description ("-" means Reserved)
PPN	31:12	Physical Page Number of the physical memory page.
LP	11	Large Page hint for Hardware Page Table Walker (HPTWK). Setting of this bit allows the HPTWK to fill a large page PTE into the TLB from searching a small-page Page Table structure. The size of the large page is implementation-dependent based on the large page support of the TLB structure and the HPTWK's size choice for the large page insertion function.
C	10:8	Cacheability: 3'd0: device space 3'd1: device space, write bufferable/coalescable 3'd2: non-cacheable memory 3'd3: - (Reserved) 3'd4: cacheable, write-back, write-allocate memory (shared) 3'd5: cacheable, write-through, no-write-allocate memory (shared) 3'd6: cacheable, non-shared, write-back, write allocate memory 3'd7: cacheable, non-shared, write-through, no-write-allocate memory Using of Reserved values (3) will generate UNPREDICTABLE

<table>
<tr><th>Field</th><th>Bits</th><th>Description (“-“ means Reserved)</th></tr>
<tr><td></td><td></td><td>result for the following implementation:
● N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
● N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)
But will generate Reserved PTE Attribute exception (Instruction or Data) for all other implementations.
Note: for dual-cores sharing a second level cache, the implementation may choose to cache “shared” data only in L2 while caching “non-shared” data in both L1/L2. Please see implementation documents for details.</td></tr>
<tr><td>G</td><td>7</td><td>Global bit: the translation is to be shared across contexts</td></tr>
<tr><td>A</td><td>6</td><td>Accessed bit: Access Bit exception if access to this page and this bit is set.</td></tr>
<tr><td>X</td><td>5</td><td>Executable bit: exception when fetching a page without executable bit set.</td></tr>
<tr><td>D</td><td>4</td><td>Dirty bit: Page modified exception if a store to this page without this bit set.</td></tr>
<tr><td>M</td><td>3:1</td><td>Read/Write Access mode:
<table>
<tr><th>M[3:1]</th><th>User mode</th><th>Superuser mode</th></tr>
<tr><td>3’b000</td><td>-</td><td>-</td></tr>
<tr><td>3’b001</td><td>Read only</td><td>Read only</td></tr>
<tr><td>3’b010</td><td>Read only</td><td>Read/Write</td></tr>
<tr><td>3’b011</td><td>Read/Write</td><td>Read/Write</td></tr>
<tr><td>3’b100</td><td>-</td><td>-</td></tr>
<tr><td>3’b101</td><td>No Read/Write access</td><td>Read only</td></tr>
<tr><td>3’b110</td><td>-</td><td>-</td></tr>
<tr><td>3’b111</td><td>No Read/Write access</td><td>Read/Write</td></tr>
</table>
Note that this Access mode only governs the access permission of using load/store instructions to access the page. It does not govern the access permission of an instruction fetching and execution.
Using of Reserved values (0, 4, 6) will generate UNPREDICTABLE result for the following implementations:
● N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
● N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)
But will generate Reserved PTE Attribute exception (Data) for all</td></tr>
</table>

Field	Bits	Description (“-“ means Reserved)
		other implementations.
V	0	This Page Table Entry is valid and present.

Table 2. Translated Address Space Attributes for MMU version 2.

Field	Bits	Description (“-“ means Reserved)
PPN	31:12	Physical Page Number of the physical memory page.
LP	11	Large Page hint for Hardware Page Table Walker (HPTWK). Setting of this bit allows the HPTWK to fill a large page PTE into the TLB from searching a small-page Page Table structure. The size of the large page is implementation-dependent based on the large page support of the TLB structure and the HPTWK’s size choice for the large page insertion function.
C	10:8	Cacheability: 3’d0: device space 3’d1: device space, write bufferable/coalescable 3’d2: non-cacheable memory 3’d3: - (Reserved) 3’d4: cacheable, write-back, write-allocate memory (shared) 3’d5: cacheable, write-through, no-write-allocate memory (shared) 3’d6: cacheable, non-shared, write-back, write allocate memory 3’d7: cacheable, non-shared, write-through, no-write-allocate memory Using of Reserved values (3) will generate Reserved PTE Attribute exception (Instruction or Data). Note: for dual-cores sharing a second level cache, the implementation may choose to cache “shared” data only in L2 while caching “non-shared” data in both L1/L2. Please see implementation documents for details.
G	7	Global bit: the translation is to be shared across contexts
A	6	Accessed bit: Access Bit exception if access to this page and this bit is set.
X	5	Executable bit. ● In superuser mode, a non-executable page exception will be

<table>
<tr><th>Field</th><th>Bits</th><th>Description (“-“ means Reserved)</th></tr>
<tr><td></td><td></td><td>generated when fetching a page without this bit set.
● In user mode and when M[3:1] != 0b101 and M[3:1] != 0b111, a non-executable page exception will be generated when fetching a page without this bit set.
● In user mode and when M[3:1] == 0b101 or M[3:1] == 0b111, a non-executable page exception will always be generated. The X bit is don’t-care in this case.</td></tr>
<tr><td>D</td><td>4</td><td>Dirty bit: Page modified exception if a store to this page without this bit set.</td></tr>
<tr><td>M</td><td>3:1</td><td>Read/Write Access mode + user mode fetching restriction on superuser mode page.
<table>
<tr><th>M[3:1]</th><th>User mode</th><th>Superuser mode</th></tr>
<tr><td>3’b000 X==0</td><td>-</td><td>-</td></tr>
<tr><td>3’b000 X==1</td><td>No Read/Write</td><td>Read/Write</td></tr>
<tr><td>3’b001</td><td>Read only</td><td>Read only</td></tr>
<tr><td>3’b010</td><td>Read only</td><td>Read/Write</td></tr>
<tr><td>3’b011</td><td>Read/Write</td><td>Read/Write</td></tr>
<tr><td>3’b100</td><td>-</td><td>-</td></tr>
<tr><td>3’b101</td><td>No Read/Write/Fetch access</td><td>Read only</td></tr>
<tr><td>3’b110</td><td>-</td><td>-</td></tr>
<tr><td>3’b111</td><td>No Read/Write/Fetch access</td><td>Read/Write</td></tr>
</table>
Note that this Access mode governs the access permission of using load/store instructions to access the page. And it governs the user mode instruction fetching/execution permission when M[3:1] == 0b101 or M[3:1] == 0b111.
Using of Reserved values 0 (when X is 0), 4, and 6 will generate Reserved PTE Attribute exception for load/store instruction (Data) and instruction fetching (Instruction).</td></tr>
<tr><td>V</td><td>0</td><td>This Page Table Entry is valid and present.</td></tr>
</table>

2.2.2.2. Attributes for None-Translated Virtual/Physical Address

For virtual to physical address mapping that does not go through the page table

translation process, the address space attributes is defined in either two or four fields of the non-translated attribute field of the MMU control register. The non-translated attribute field is indexed using either the top or the top two bits of the virtual address (i.e. VA(31) or VA(31,30)). Since this is a very coarse partition of the address space (2 or 4 segments) for system design, only cacheability attribute is defined for each segment. And the cacheability defined here is lumped into four different types in the following table instead of seven types in the translated address space attributes.

Table 3. Non-translated Address Space Attribute

Non-translated Cacheability (NTC)	Meaning
0	Non-cacheable/non-coalesable
1	Non-cacheable/coalesable
2	Cacheable/write-back
3	Cacheable/write-through

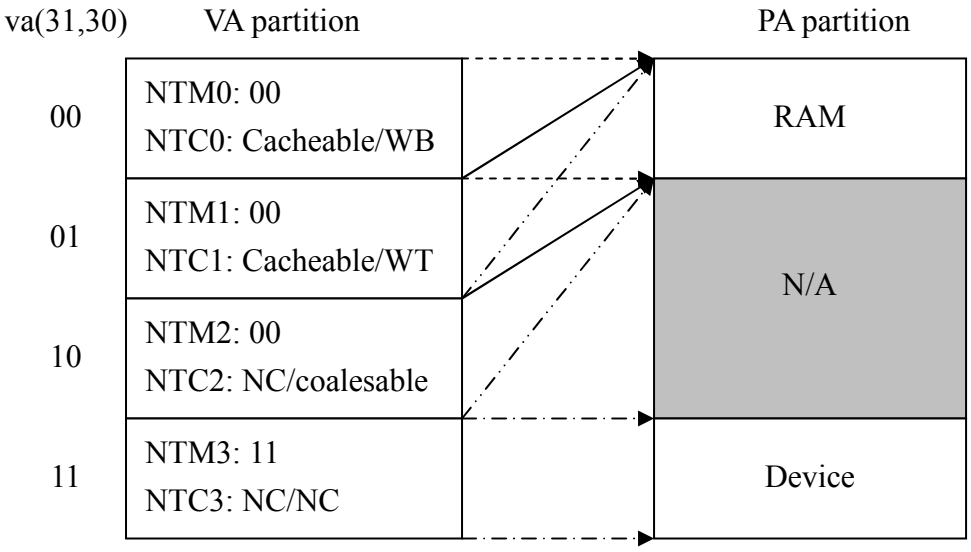
To improve memory access performance on a small system without MMU or MPU, another partition-level VA to PA mapping attribute (NTM) is defined such that different cacheability attributes can be mapped to the same PA partition through different VA partitions without frequently changing the content of the NTC field. This will improve the software efficiency of small systems.

An example usage of NTM partition mapping is illustrated in the following figure (Figure 2). In this example, VA partitions 0-2 are all mapped to the same PA partition 0 where the real memory is resided. In this small system, a programmer can use the following C code sequences to define the cacheability attribute for each variable. Note that changing a variable from cacheable to non-cacheable requires a cache invalidation operation to guarantee correct behaviors.

```
#define CacheWT  (1 << 30)
#define NonCache (1 << 31)

int a, b, c, d;
int *p, *q, *r, *s;
p = &a | CacheWT;  // make "a" cacheable/WT
q = &b | NonCache; // make "b" non-cacheable
```


Figure 2. Example usage of NTM partition mapping



2.2.3. Address Space Operation Ordering Requirement

In Andes memory architecture, loads and stores to different addresses from an AndesCore™ are, in most part, not ordered except for regions characterized by certain device-related address space attributes. For un-ordered loads and stores, if there is a need to enforce certain ordering between these memory operations, MSYNC instruction must be used to achieve this goal.

The following table lists the address space operation ordering requirement based on the translated address space attributes. The symbols used in this table are defined as:

- < : This is a "BEFORE" relation between two operations in time or program sequence. For example, "A < B" in a program means instruction A comes earlier than instruction B in a program sequence.
- u : This is an "UN-ORDERED" relation between two operations in time of performing their operations.

Table 4. Address Space Operation Ordering Requirement

A < B in program order	**B**	Device		Bufferable device		Uncacheable memory		WT cacheable memory		WB cacheable memory	
A		read	write	read	write	read	write	read	write	read	write
Device	read	<	<	<	<	u	u	u	u	u	u
	write	<	<	<	<	u	u	u	u	u	u
Bufferable device	read	<	<	<	<	u	u	u	u	u	u
	write	<	<	<	u	u	u	u	u	u	u
Uncacheable memory	read	u	u	u	u	u	u	u	u	u	u
	write	u	u	u	u	u	u	u	u	u	u
WT cacheable memory	read	u	u	u	u	u	u	u	u	u	u
	write	u	u	u	u	u	u	u	u	u	u
WB cacheable memory	read	u	u	u	u	u	u	u	u	u	u
	write	u	u	u	u	u	u	u	u	u	u

For non-translated memory attributes, they can be mapped into the ordering requirement of the translated memory attributes using the following mapping:

Non-translated Attribute	Mapped Translated Attribute
Non-cacheable/non-coalesable	Device
Non-cacheable/coalesable	Bufferable Device
Cacheable/write-back	WB Cacheable Memory
Cacheable/write-through	WT Cacheable Memory

2.2.4. Reduced Address Space for Small System

To reduce cost for a small system, a reduced 24-bit address space configuration is provided by some AndesCore implementations. The existence of this configuration is recorded in the ADR24 field of the MSC_CFG system register.

In this reduced 24-bit address space configuration, the non-translated attribute fields (NTC and NTM) are indexed using either the top or the top two bits of the 24-bit virtual address (i.e. VA(23) or VA(23,22)).

2.3. Large page and multiple page size support

The Andes MMU TLB structure provides multiple page size support in the software-visible part of the TLB. It may support ten different page sizes: 4KB, 8KB, 16KB, 64KB, 256KB, 1MB, 4MB, 16MB, 64MB, and 256MB. The exact number of page size supported is implementation-dependent. And the HPTWK may support only one or two predetermined page sizes to reduce hardware cost in the HPTWK itself and the software-invisible part of the TLB structure. For page sizes not supported by the HPTWK, software is required to insert the page translation into the Andes TLB structure.

2.4. TLB locking support

It is sometimes desirable to lock some particular entries inside the TLB to avoid triggering page faults that cause recursions, race conditions, or simply for performance concerns. The Andes MMU TLB structure provides locking support in the software-visible part of the TLB.

Recall that the software-visible part of the TLB is an N sets, K ways set-associative (or non-fully-associative) cache memory. To provide TLB locking support, a TLB “fill with lock” instruction will be provided to lock the TLB entry from TLB random replacement. However, since the TLB is not a fully-associate, but a K way set-associate cache,

software has to make sure at most only K-1 ways can be locked in each set to allow at least one entry in a set to be used as the replacement entry for a TLB random insertion or HPTWK TLB insertion operation. If this is violated such that all K ways of a set are locked and a TLB insertion operation (from either a TLBOP Random Write instruction or a HPTWK TLB fill action) is performed on the set, then depending on the setting of the TLBALCK field of the MMU Control register, either a “Machine Error” exception will be generated or a hardware random replacement which overwrites a locked entry will be performed. Please see section 9.4.1 MMU Control Register for more detail on the TLBACLK field.

2.5. TLB Implementation Constraint

The Andes Instruction Set Architecture is a mixed 16-bit/32-bit instruction set architecture. So it is natural in an Andes program to have a 32-bit instruction straddling across two pages. To fetch that instruction successfully, in a worst case scenario, two TLB fill operations can be invoked successively during the fetching process. And if the successive TLB fill operations thrash with each other, then the fetching operation can never complete, resulting in a live-lock condition. For a non-fully-associative TLB, this is really not a problem since the two pages which contain the instruction will have different, but consecutive, TLB indexes when filling their PTEs into the TLB. However, any extra implementation-dependent fully-associative TLB that cannot be controlled by software (e.g. a micro TLB) needs to follow the following constraint:

"The TLB replace algorithm has to guarantee that a TLB fill operation will not replace the TLB entry inserted/filled last time".

2.6. MMU bootstrapping

There are two working modes for MMU: “no translation”, and “VA translation” modes. After system reset, the memory management system is entered into the “no translation” mode and the processor boots up in superuser mode. The operating system enables the

"VA translation" mode after it initializes the page table to map itself and later transfer control to a user mode application.

2.7. MMU related registers

Table 5 lists all the MMU related registers. All these registers require superuser mode privilege to access. Please see Section 9.4 MMU System Registers for their detailed definitions.

Table 5. MMU related system registers

Name	Type	Updater	Section
MMU control	R/W	Inst	9.4.1
Exception VA	R	Hardware	9.3.5
L1 PPT Base	R/W	Inst	9.4.2
TLB Access Misc	R/W	Inst	9.4.5
TLB Access VPN	R/W	Inst/Hardware	9.4.3
TLB Access Data	R/W	Inst	9.4.4
Virtual Linear PT Index	R/W	Inst/Hardware	9.4.6

The "Reserved" field in a register is reserved for future extension. It must be written with 0. The processor behavior is "UNPREDICTABLE" if 0 is not used to write the "Reserved" field. The reserved field is read as 0.

The "Reserved" value in a register field is reserved for future extension. If a reserved value is written into a register field, the processor behavior will be "UNPREDICTABLE".

The "Ignored" field in a register is ignored by the processor when it is written. And it will be read as 0.

Implementation Note:
Making "Reserved" field behaved as "UNPREDICTABLE" when written with a non-zero value alleviates the hardware from spending logics checking the incoming input

written value since any software has to make sure only “PREDICTABLE” behavior is performed from a software’s instructions. The only downside is that a software bug unnoticed by software which a non-zero value is written may show up in a later implementation which changes a “Reserved” field in the previous implementation into a “meaningful” field.

2.8. TLB Management Instructions

Nine types of TLB management instructions are defined to manage the Andes TLB structure. They are summarized in the following table and described in detail in the following sections.

Table 6. MMU management instructions

Instruction	Operation
TLBOP Rx, TargetRead (TRD)	Read targeted TLB entry
TLBOP Rx, TargetWrite (TWR)	Write targeted TLB entry
TLBOP Rx, RWrite (RWR)	Write PTE into a TLB entry
TLBOP Rx, RWriteLock (RWLK)	Write PTE into a TLB entry and lock
TLBOP Rx, Unlock (UNLK)	Unlock a TLB entry
TLBOP Rt, Rx, Probe (PB)	Probe TLB entry
TLBOP FlushAll (FLUA)	Flush all TLB entries except locked entries
TLBOP Rx, Invalidate (INV)	Invalidate TLB entries containing VA stored in Rx (except locked entries)
(Optional) LW_VLPT	Load VLPT page table which always goes through data TLB translation. On TLB miss, generate Double TLB miss exception

2.8.1. TLB Target Read

The TLB Target Read operation reads a specified entry in the software-visible

portion of the TLB structure. The specified entry is in the Rx register. The read result is placed in the TLB _VPN, TLB _DATA, and TLB _MISC registers.

The TLB Target Read operation is performed using the following instruction:

"TLBOP Rx, TargetRead" or "TLBOP Rx, TRD"

The normal instruction sequence of performing the TLB Target Read operation is as follows:


```
movi   Rx, TLB_rd_entry  // prepare read entry number
tlbop  Rx, TRD           // read TLB
dsb                      // data serialize barrier
mfsr Ry, TLB_VPN  // move read out to reg
```

The "TLB_rd_entry" in the code represents the entry number of a TLB entry you want to read out. The TLB entry number for the non-fully-associative N sets K ways TLB cache is as follows:

31 log2(N*K)	Log2(N*K)-1 log2(N)	Log2(N)-1 0
Ignored	Way number	Set number

2.8.2. TLB Target Write

The TLB Target Write operation writes a specified entry in the software-visible portion of the TLB structure. The specified entry is in the Rx register. The other write operands are in the in TLB _VPN, TLB _DATA, and TLB _MISC registers. The TLB indexed write operation is performed using the following instruction:

"TLBOP Rx, TargetWrite" or "TLBOP Rx, TWR"

The normal instruction sequence of performing the TLB Target Write operation is as follows:

```
mtsr    Ra, TLB_VPN     // prepare VPN (may not needed if HW
                        // already put in the correct value)
mtsr    Rb, TLB_DATA    // prepare PPN, etc.
mtsr    Rc, TLB_MISC    // prepare CID… (may not needed if the
                        // correct value is already there)
dsb                             // data serialize barrier
                                // may not needed (imp-dep)
movi    Rx, TLB_wr_entry        // prepare write index
tlbop   Rx, TWR                 // idx TLB write
isb                             // insn serialize barrier
```


The TLB entry number for the non-fully-associative N sets K ways TLB cache is as follows:

31 log2(N*K)	Log2(N*K)-1 log2(N)	Log2(N)-1 0
Ignored	Way number	Set number

If the selected target entry is locked, this instruction will overwrite the locked entry and clear the lock flag.

2.8.3. TLB Random Write (hardware-determined way in a set)

The TLB_Random_Write operation writes a hardware-determined random TLB way in a set determined by the VA (in TLB_VPN) and page size (in TLB_MISC) in the software-visible portion of the TLB structure. The Rx operand contains the data that is going to write into the TLB_DATA portion of the TLB structure. The other write operands are in the TLB_VPN, and TLB_MISC register. The TLB_Random_Write operation is performed using the following instruction:

"TLBOP Rx, RWrite" or "TLBOP Rx, RWR"

The normal instruction sequence of performing the TLB_random_insert operation is as follows:

```
mtsr    Rb, TLB_MISC   // prepare CID (may not needed if the
                       // correct value is already there)
dsb                    // data serialize barrier
                       // may not needed (imp-dep)
tlbop   Rx, RWR        // Rx contains PPN, etc.
isb                    // inst serialize barrier
```


This instruction will generate an imprecise “Data Machine Error” exception if all the ways in the determined set is all locked.

2.8.4. TLB Random Write and Lock

TLB Random Write and Lock operation is similar to TLB Random Write Operation to write a PTE into a hardware determined TLB entry. In addition to the write operation, this instruction also locks the TLB entry.

The TLB Random Write and Lock operation is performed using the following instruction:

“TLBOP Rx, RWriteLock” or “TLBOP Rx, RWLK”

This instruction will generate an imprecise “Data Machine Error” exception if all the ways in the determined set is all locked before this instruction locks an entry.

2.8.5. TLB Unlock

TLB Unlock operation unlocks a TLB entry if the VA in the Rx matches the VPN of an entry in a set determined by the VA (in Rx) and the page size (in TLB_MISC).

The TLB Unlock operation is performed using the following instruction:

“TLBOP Rx, Unlock” or “TLBOP Rx, UNLK”

2.8.6. TLB probe

The TLB_probe operation searches all TLB structures (software-visible and software-invisible) for a specified VA. And the search result is written into the general register Rt. The search VA is specified in the Rx portion of the TLB probe instruction. The result that stored in Rt has the following format:

31	30	29	28 ... n	n-1 ... 0
NF	HW	SW	Reserved	Entry #

It contains the entry number to the TLB entry (software-visible TLB) which contains the VA search key in the general source register of the TLB probe instruction. If the VA can be found in the software-visible part of the TLB, the “sw” bit will be set. If the VA can be found in the software-invisible part of the TLB, the “hw” bit will be set. And if the VA cannot be found in either the software-visible or software-invisible part of the TLB, the “nf” bit will be set.

The TLB entry number for the non-fully-associative N sets K ways TLB cache is as follows:

Log2(N*K)-1 ... log2(N)	Log2(N)-1 ... 0
Way number	Set number

Field Name	Bits	Description	Type	Reset
TLB entry number	n (n-1,0)	Indicates the entry number of a PTE entry which contains the VA specified in the Rx of a TLB probe instruction. The TLB entry number for a non-fully-associative N sets K ways TLB cache is as follows: Log2(N*K)-1 ... log2(N) | Log2(N)-1 ... 0 Way number | Set number	RO	0
Reserved	28-n+1 (28,n)	RAZWI	RAZWI	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>SW</td><td>1
(29)</td><td>Search flag for the software-visible part of TLB.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The page containing VA is not found in the software-visible part of TLB.</td></tr><tr><td>1</td><td>The page containing VA is found in the software-visible part of TLB.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>HW</td><td>1
(30)</td><td>Search flag for the software-invisible and non-subset part of TLB.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The page containing VA is not found in the software-invisible and non-subset part of TLB.</td></tr><tr><td>1</td><td>The page containing VA is found in the software-invisible and non-subset part of TLB.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>NF</td><td>1
(31)</td><td>Search result.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The page containing VA is found in TLB.</td></tr><tr><td>1</td><td>The page containing VA is not found in TLB.</td></tr></table></td><td>RO</td><td>0</td></tr>
</table>

The TLB_probe operation is performed using the following instruction:

"TLBOP Rt, Rx, Probe" or "TLBOP Rt, Rx, PB"

The normal instruction sequence of performing the TLB probe operation is as follows:

```
tlbop   Rt, Rx, PB              // VA is in Rx
<Examine> Rt
```


2.8.7. TLB flush all

The TLB_flush_all operation flushes all TLB entries (software-visible and software-invisible) except the locked TLB entries.

The TLB_flush_all operation is performed by using the following instruction:

"TLBOP FlushAll" or "TLBOP FLUA"

The normal instruction sequence of doing the TLB_flush_all operation is as follows:

```
tlpop   FLUA          // move to system reg
isb                   // Inst serialize barrier
```

2.8.8. TLB Invalidate VA

The TLB_Invalidate_VA operation flushes the TLB entry that contains the VA in the Rx register (software-visible and software-invisible) and the page size specified in the TLB_MISC register..

The TLB_Invalidate_VA operation is performed by using the following instruction:

"TLBOP Rx, Invalidate" or "TLBOP Rx, INV"

The normal instruction sequence of doing the TLB_invalidate_VA operation is as follows:

```
// prepare VA in Rx
…
tlbop  Rx, INV   // Invalidate TLB entries containing VA
isb              // inst serialize barrier
```

Note that this TLB invalidate operation may flush more pages than the exact number of pages which contain this VA, up to flushing the entire TLB structure. And this is implementation-dependent.

2.8.9. Load VLPT page table (Optional)

The LW_VLPT instruction will be used to do a memory load from the virtual linear page table. Its difference from the normal load instruction is that (1) it is a load that will always use translation without regarding to the DT bit in the PSW register, and (2) on TLB miss, instead of a TLB fill exception, it will generate a Double TLB miss exception.

Note that if we can get to this instruction in the TLB fill exception handler, the HPTWK is definitely not enabled to do TLB fill operations.

AndesCore™ Info: N12 does not have this feature implemented.

2.9. Page Table Formats

The Andes hardware page table format is implementation-dependent. However, this section describes a 4KB and an 8KB page tables which are the most likely implementations for the first few Andes MMU implementations.

2.9.1. 4KB Page Table

The Andes 4KB page hardware page tables are organized as two level hierarchical tables. The entries in the first level of the page tables are indexed by VA(31,22). The entries in the second level of the page tables are indexed by VA(21,12). The entries in the first level tables are pointers that points to the second level tables. Both tables have 2^10 = 1024 entries, each entry is 32-bit (4-byte) wide and therefore each table is 4KB in size.

2.9.2. 8KB Page Table

The Andes 8KB page hardware page tables are organized as two level hierarchical tables. The entries in the first level of the page tables are indexed by VA(31,24). The entries in the second level of the page tables are indexed by VA(23,13). The entries in the first level tables are pointers that points to the second level tables. The first level table has 256 4-byte page table entries, thus occupy 1/8 of an 8KB page. And the second level table has 2048 4-byte page table entries, thus occupy an 8KB page.

2.9.3. Page Table Coherence Requirements

Since the Hardware Page Table Walker is logically a separate DMA agent to the memory system from AndesCore™'s first-level cache system and the HPTWK is not required to walk the page table through the first-level cache, to maintain a consistent and coherent view of memory data between the HPTWK and the AndesCore™ which maintains and manipulates the page table data, software needs to use extra cache management and memory access ordering instructions to enforce the coherence of the PTE data after any PTE update operation. Software has to follow Rule A and Rule B after any PTE update operation; otherwise the HPTWK is not guaranteed to get the updated page table content.

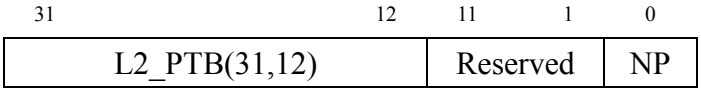
- **Rule A**
  - **A1:** if the page table has a "non-cacheable" memory attribute, then no cache management instruction is needed.
  - **A2:** if the page table has a "cacheable/write-through" memory attribute, then no cache management instruction is needed.
  - **A3:** if the page table has a "cacheable/write-back" memory attribute, then a "CCTL Ra, L1D_VA_WB" instruction is needed after a PTE update instruction (i.e. store) to write back the data from the first-level Dcache to memory.
- **Rule B**
  - Disregard of the cacheability attribute, a "MSYNC store" instruction is needed to force all PTE updates visible in memory before any load/store instruction which triggers HPTWK loading of the updated PTE. The "MSYNC" instruction has to be after any cache management instruction.

The following example code sequence illustrates the above requirements.

```
Lw  R1, [R2]          // load PTE
...                   // PTE manipulation
Sw  R1, [R2]          // store PTE
cctl R2, L1D_VA_WB  // assuming "cacheable/write-back" attribute
msync store           // drain L1D write buffer to memory
iret or isb           // serialize next fetch/HPTWK read
```

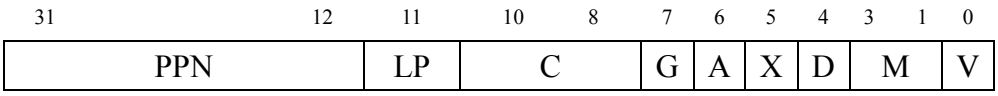

To reduce these extra overheads, software can consider preparing more PTEs than necessary at each PTE update occasion.

2.9.4. L1_PHYSICAL_PAGE_TABLE_ENTRY



L2_PTB is a 4KB-aligned physical address that points to the level-2 page table containing the page translation for the accessed VA. If the "NP" field is set, the level-2 page table does not exist and a L1_PTE_NOT_PRESENT exception will be signaled by hardware.

2.9.5. L2_PHYSICAL_PAGE_TABLE_ENTRY



PPN is the Physical Page Number the translated Virtual Page maps to. Once this entry is found by the HPTWK, it will insert this entry into the ANDES TLB structure, if no other exception is generated.

For detailed descriptions of the different fields in this page table format, please see Table 1 in section 2.2.2.1 Attributes for Translated Virtual/Physical Address.

The "LP" field is added to indicate that this PTE is part of a large page whose size is determined by a processor core implementation. The Hardware Page Table Walker is

required to use this bit to determine that if this PTE is for a large page size or not. If this PTE is for a large page size, then the HPTWK is required to insert the PTE into the TLB as a large page PTE entry. Note that this large page is built on the page table structure of a small page size. So an OS needs to prepare duplicated small page PTE entries to cover all the page table search space for the large page. Using a 4KB/1MB page size combination as an example, 256 entries of 4KB PTE needs to be prepared when the 1MB large page is being allocated.

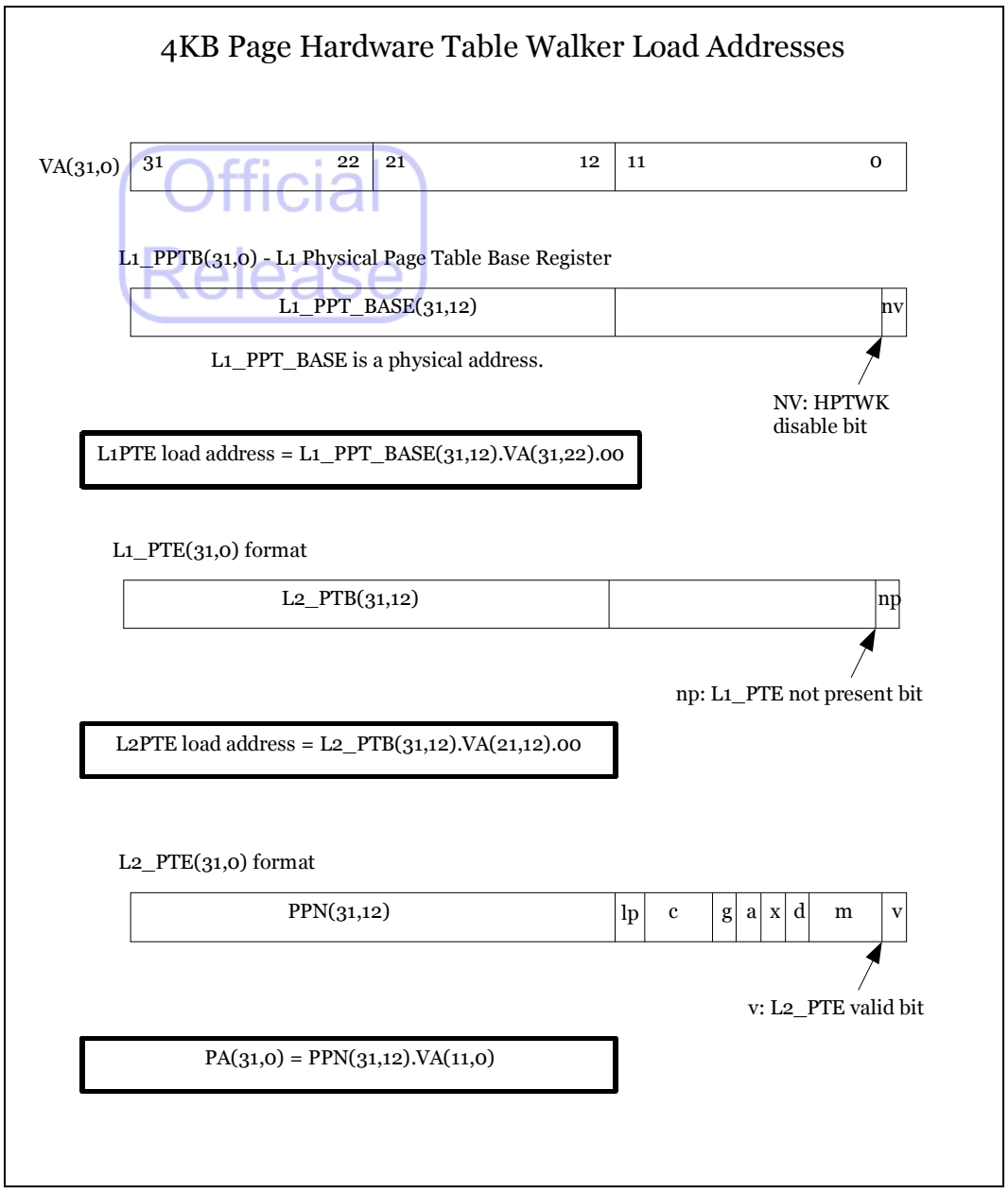
2.9.6. PTE Modification and Insertion Requirement

When inserting a PTE into the TLB, the TLB update instructions (Random Write, Target Write) and the TLB update hardware (page table walker) do not search and invalidate the PTEs in the TLB that cover or overlap the address range of the new PTE. Software is required to remove (i.e. invalidate) all those PTEs that cover/overlap the address range of the new PTE from the TLB before inserting the new PTE.

2.9.7. Page Table Address Formation

To find the correct PTE from the page table in memory, the HPTWK needs to perform two memory load operations to get L1_PTE and then L2_PTE. The two physical load addresses for a 4KB page are formed by the HPTWK based on the following illustration in Figure 3.

Figure 3. 4KB page Hardware Page Table Walker address formation.



The two physical load addresses for an 8KB page are formed by the HPTWK based on the following illustration.

Figure 4. 8KB page Hardware Page Table Walker address formation.

8KB page table format for HPTWK

VA(31,0)

31 24	23 13	12 0
8	11	13

L1PTE load address = L1_PPTB(31,12).00.VA(31,24).00

L2PTE load address = L2_PTB(31,13).VA(23,13).00

PA(31,0) = PPN(31,13).VA(12,0)

2.10. MMU Exception Handling

2.10.1. MMU Exceptions from VA to PA Translation

Nine possible exceptions could be generated during a VA to PA translation process. Their names and generating conditions are listed in the following table.

Table 7. MMU exceptions

Name	Generating condition	Entry vector
TLB fill	(No translation found in Andes TLB structure) and (HPTW is not enabled)	TLB fill
Non-Leaf PTE not present	L1_PTE(“np”) == 1	PTE not present
Leaf PTE not present	L2_PTE(“v”) == 0	PTE not present
Read protection violation	Based on L2_PTE(“m”) and current processor operating mode	TLB misc
Write protection violation	Based on L2_PTE(“m”) and current processor operating mode	TLB misc

Name	Generating condition	Entry vector
Page modified	(L2_PTE(“D”) == 0) and this is a store access	TLB misc
Non-executable page	(L2_PTE(“X”) == 0) and this is an instruction fetch	TLB misc
Access bit	L2_PTE(“A”) == 1	TLB misc
Double TLB miss	TLB miss on LD_VLPT instruction	TLB VLPT miss

TLB fill exception has its own exception entry point to facilitate faster software TLB fill exception handling operations. PTE not present exceptions have their own exception entry point as well. All the other exceptions use the same MMU exception vector. And the exact exception type is remembered in the Processor’s exception reason (ITYPE) register.

On exceptions generated when a PTE is found and valid/present in the TLB, several of them could happen simultaneously. To resolve this situation, their handling priority is listed in the following table.

Table 8. Handling Priority for simultaneous MMU exceptions

Name	Priority
Non-executable page, Read protection violation, Write protection violation	1
Page modified	2
Access bit	3

This priority list is defined as such based on the assumption that an Access bit exception is generated based on a legal memory access and has a less restrictive condition to generate than that of a Page modified exception which is under a legal access as well, while all the other access violation exceptions are all caused by illegal memory accesses. And the priority of reporting illegal accesses should be higher than that of reporting legal accesses.

2.10.2. Multiple match on software-visible part of the TLB

structure

Since the software-visible part of the Andes TLB structure is a set-associative cache memory, it is possible to generate multiple matches when search for a VA translation due to software bugs or hardware errors (e.g. alpha particle hit, etc.). And when multiple matches happen, it is not possible for the hardware to know which entry contains the correct information. Thus, a Machine Error exception will be generated to leave the handling of this situation to software.

The following operations could encounter this multiple match situation and generate a Machine Error exception:

- Instruction fetch – Instruction Machine Error exception
- Load/Store type instructions – Data Machine Error exception
- TLBOP Probe instruction – Imprecise Data Machine Error exception

The following operation could encounter this multiple match situation but will not generate a Machine Error exception since it is very easy to resolve its problem in hardware.

- TLBOP Invalidate VA instruction

2.10.3. All Entries Locked on a Set of the TLB Structure

The Andes MMU TLB structure provides locking support in the software-visible part of the TLB structure. However, since the TLB is not a fully-associate, but a K way set-associate cache, software has to make sure at most only K-1 ways can be locked in each set to allow at least one entry in a set to be used as the replacement entry for a TLB random insertion or HPTWK TLB insertion operation. If this is violated such that all K ways of a set are locked and a TLB insertion operation (from either a TLBOP Random Write instruction or a HPTWK TLB fill action) is performed on the set, then depending on the setting of the TLBALCK field of the MMU Control register, either a “Machine Error” exception will be generated or a hardware random replacement which overwrites a locked entry will be performed. Please see section 9.4.1 MMU Control Register for more detail on the TLBACLK field. And see section 9.3.7 Interruption Type Register for the interruption type encoding of this exception in the Machine Error exception entry point.

2.11. Mixed Endian Support

Andes instruction set and memory architecture provides mechanisms to support a mixed data endian environment to speedup and facilitate the endian exchange process.

2.11.1. Basic Mixed Endian Support

The Processor Status Word (PSW) system register contains a BE bit which determines how load/store instructions move data between memory and registers using little endian or big endian format. Software can change the endian format by executing a SETEND.B or SETEND.L instruction followed by a DSB instruction.

The MMU configuration (MMU_CFG) system register contains a DE (Default Endian) bit which will be copied into PSW.BE bit when an interruption has happened. This makes sure that the interruption handler can have a consistent endian behavior no matter how user programs change the endian format before the interruption event.

2.11.2. Device Register Space Endian Control

To further enhance the flexibility of system construction and OS and device driver pairing, an extra device register space endian control mechanism is provided on top of the basic Andes mixed endian support described in the previous section. This extra mechanism may provide an easy environment to operate an OS in one endian format with all the device components in a different endian format.

The MMU control (MMU_CTL) system register contains a DREE (Device Register Endian Enable) bit that controls if this extra mechanism is turned on or not.

When this extra mechanism is turned on, a device register space access, i.e. memory page attribute C is equal to 0 or NTC in a partition is equal to 0, will use the DRBE bit contains in the Processor Status Word (PSW) system register to determine the access

endian format while an access to all other memory attribute space will still use PSW.BE for the endian format. The PSW.DRBE bit can be updated using a privileged MTSR instruction followed by a DSB instruction.

The MMU configuration (MMU_CFG) system register contains a DRDE (Device Register Default Endian) bit that will be copied into PSW.DRBE bit when an interruption has happened. This also makes sure that the interruption handler can have a consistent device register endian behavior no matter how kernel programs change the endian format before the interruption event.

Chapter 3
Memory Protection Unit

This chapter describes a simple memory management unit (called Memory Protection Unit) and contains the following sections.

3.1. Introduction

This chapter describes a Memory Protection Unit scheme which uses less hardware and memory resources to manage memory compared with the Andes Memory Management Unit. This Memory Protection Unit scheme (called Andes MPU) allows memory remapping, and bound checking on high boundaries. It re-uses most of the software interfaces defined by the Andes Memory Management Unit with some definition changes.

The Andes MPU contains an 8-entry TLB-like structure in hardware. The information inside the TLB structure is completely managed by software using TLBOP target write and target read instruction. The information in the MPU TLB includes the following items:

- Address remapping
- Address bound checking
- Cacheability attribute
- Accessibility attribute

It is similar to Andes MMU that when the PSW.it is 1, the MPU TLB structure will be used for instruction fetch to generate memory remapping, various attributes, and protection information. When the PSW.it is 0, the memory remapping for instruction fetch is off and the NTC fields in the MMU control system register will be used to generate cacheability attribute.

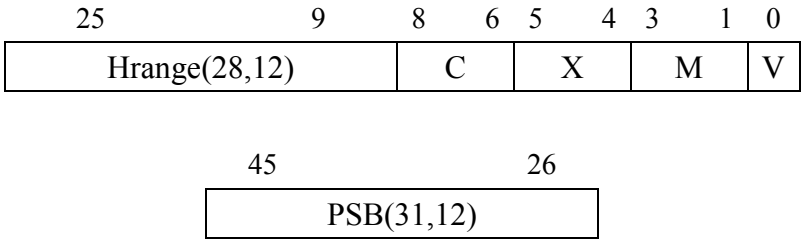
Similarly, when the PSW.dt is 1, the MPU TLB structure will be used for load/store instruction to generate memory remapping, various attributes, and protection information. When the PSW.dt is 0, the memory remapping for load/store instruction is off and the NTC fields in the MMU control system register will be used to generate cacheability attribute.

3.2. MPU TLB Structure

The MPU TLB structure is an eight-entry hardware table indexed by the top 3-bits of the virtual address [31:29]. An implementation may implement a merged MPU structure for both instruction fetching and data accesses or build separate MPU structures for instruction fetching and data accesses respectively. For separate instruction and data MPU structures, the MPU management instructions need to be able to manipulate each structure individually.

3.2.1. MPU Table Entry

Each MPU table entry has the following format:



It contains a 4KB-aligned Physical Section Base which re-maps the starting position of the 512MB-aligned Virtual Section to a real physical memory. It also contains one 4KB-aligned upper virtual section limits for memory access range protection.

The detailed meanings of these fields are defined as follows:

Field	Bits	Description ("-" means Reserved)
PSB	20 (45:26)	Physical Section Base address of the physical memory section. The physical address is generated using the following formula: PA(31,0) = {PSB(31,12) + VA(28,12)}.VA(11,0)
Hrange	17 (25:9)	Upper bound virtual address in the 512MB virtual address section. Read/Write/Execute protection exception will be generated if VA(28,12) >= Hrange(28,12) depending on

<table>
<tr><th>Field</th><th>Bits</th><th>Description (“-“ means Reserved)</th></tr>
<tr><td></td><td></td><td>load, store, or instruction fetch.</td></tr>
<tr><td>C</td><td>3
(8,6)</td><td>Cacheability:
<table>
<tr><th>Encoding</th><th>Meaning</th></tr>
<tr><td>0</td><td>Device space</td></tr>
<tr><td>1</td><td>Device space, write bufferable/coalescable</td></tr>
<tr><td>2</td><td>Non-cacheable memory</td></tr>
<tr><td>3</td><td></td></tr>
<tr><td>4</td><td>Cacheable, write-back, write-allocate memory</td></tr>
<tr><td>5</td><td>Cacheable, write-through, no-write-allocate memory</td></tr>
<tr><td>6</td><td>Cacheable, non-shared, write-back, write-allocate memory</td></tr>
<tr><td>7</td><td>Cacheable, non-shared, write-through, no-write-allocate memory</td></tr>
</table>
Using of Reserved values in this field will generate Reserved Attribute exception at the time of reading out the MPU TLB entry.</td></tr>
<tr><td>X</td><td>2
(5:4)</td><td>Execution permission: exception when fetching a section without proper execution permission.
Bit-5: superuser mode execution is allowed when this bit is set.
Bit-4: user mode execution is allowed when when this bit is set.
<table>
<tr><th>Bit-5 (S)</th><th>Bit-4 (U)</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>None executable page for all modes</td></tr>
<tr><td>0</td><td>1</td><td>User only code (Less useful)</td></tr>
<tr><td>1</td><td>0</td><td>Privileged code</td></tr>
<tr><td>1</td><td>1</td><td>User code</td></tr>
</table>
Non-executable code exception will be generated if an instruction fetch does not get execution permission based on the processor operating mode.</td></tr>
</table>

<table>
<tr><th>Field</th><th>Bits</th><th>Description (“-“ means Reserved)</th></tr>
<tr><td>M</td><td>3
(3:1)</td><td>Read/Write Access mode:
<table>
<tr><th>M[4:2]</th><th>User mode</th><th>Superuser mode</th></tr>
<tr><td>3’b000</td><td>-</td><td>-</td></tr>
<tr><td>3’b001</td><td>Read only</td><td>Read only</td></tr>
<tr><td>3’b010</td><td>Read only</td><td>Read/Write</td></tr>
<tr><td>3’b011</td><td>Read/Write</td><td>Read/Write</td></tr>
<tr><td>3’b100</td><td>-</td><td>-</td></tr>
<tr><td>3’b101</td><td>No Read/Write access</td><td>Read only</td></tr>
<tr><td>3’b110</td><td>-</td><td>-</td></tr>
<tr><td>3’b111</td><td>No Read/Write access</td><td>Read/Write</td></tr>
</table>
Note that this Access mode only governs the access permission of using load/store instructions to access the page. It does not govern the access permission of an instruction fetching and execution. Using of Reserved values in this field will generate Reserved Attribute exception at the time of reading out the MPU TLB entry.</td></tr>
<tr><td>V</td><td>1
(0)</td><td>This MPU Table Entry is valid and present.</td></tr>
</table>

3.3. MPU TLB Management Instructions

Two TLB management instructions are defined to manage the Andes MPU TLB structure. They are summarized in the following table and described in detail in the following sections.

Instruction	Operation
TLBOP Rx, TargetRead (TRD)	Read targeted MPU TLB entry
TLBOP Rx, TargetWrite (TWR)	Write targeted MPU TLB entry

3.3.1. TLB Target Read

The TLB Target Read operation reads a specified entry in the MPU TLB structure. The specified entry is in the Rx register. The read result is placed in the TLB_VPN and the TLB _DATA register. The formats of the TLB_VPN and the TLB_DATA registers are different from the formats of those registers defined for the MMU memory management scheme. Their MPU-defined formats will be described in section 3.4.

The TLB Target Read operation is performed using the following instruction:

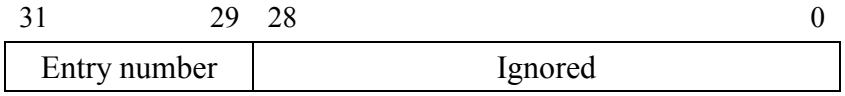
"TLBOP Rx, TargetRead" or "TLBOP Rx, TRD"

The normal instruction sequence of performing the TLB Target Read operation is as follows:

```
movi   Rx, TLB_rd_entry  // prepare read entry number
tlbop  Rx, TRD           // read TLB
dsb                      // data serialize barrier
mfsr Ry, TLB_DATA        // move read out to reg
mfsr Rz, TLB_VPN
```

The "TLB_rd_entry" in the code represents the entry number of a TLB entry you want to

read out. The MPU TLB entry number is defined as follows:



3.3.2. TLB Target Write

The TLB Target Write operation writes a specified entry in the MPU TLB structure. The specified entry is in the Rx register. The TLB entry write input operands are in the TLB_VPN and the TLB _DATA registers. The formats of the TLB_VPN and the TLB_DATA registers are different from the formats of those registers defined for the MMU memory management scheme. Their MPU-defined formats will be described in section 3.4.

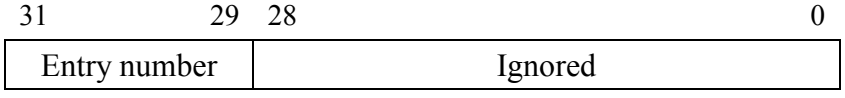
The TLB target write operation is performed using the following instruction:

"TLBOP Rx, TargetWrite" or "TLBOP Rx, TWR"

The normal instruction sequence of performing the TLB Target Write operation is as follows:

```
mtsr    Rb, TLB_DATA   // prepare PSB, etc.
mtsr    Rc, TLB_VPN    // prepare High range.
dsb                          // data serialize barrier
movi    Rx, TLB_wr_entry     // prepare write index
tlbop   Rx, TWR              // idx TLB write
isb                          // inst serialize barrier
```

The TLB entry number is defined as follows:



3.4. MPU-related System Registers

3.4.1. MMU Control Register

Mnemonic Name: mr0 (MMU_CTL)
IM Requirement: required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 0, 0}

10 9	8 7	6 5	4 3	2 1	0
Reserved	NTC3	NTC2	NTC1	NTC0	Reserved

31 20	19	18 17	16 15	14 13	12 11
Reserved	DREE	NTM3	NTM2	NTM1	NTM0

When MMU_CFG..MMPS is 0 or 1, the D, TBALCK, MPZIU fields may become Reserved or not. It is implementation-dependent. For the purpose of this specification, MMU_CFG..MMPS should be 1.

When MMU_CFG..NTME is 0, the NTM0-NTM3 fields should become Reserved fields.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>Reserved</td><td>1
(0)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
<tr><td>NTC0</td><td>2
(2,1)</td><td>Indicates Non-Translated Cacheability memory attribute for partition 0.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Non-cacheable/Non-coalesable</td></tr>
<tr><td>1</td><td>Non-cacheable/Coalesable</td></tr>
<tr><td>2</td><td>Cacheable/Write-Back</td></tr>
<tr><td>3</td><td>Cacheable/Write-Through</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>NTC1</td><td>2</td><td>Indicates Non-Translated Cacheability</td><td>RW</td><td>0</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
	(4,3)	memory attribute for partition 1. See NTC0 description for its value definition.		
NTC2	2 (6,5)	Indicates Non-Translated Cacheability memory attribute for partition 2. See NTC0 description for its value definition.	RW	0
NTC3	2 (8,7)	Indicates Non-Translated Cacheability memory attribute for partition 3. See NTC0 description for its value definition.	RW	0
Reserved	2 (10,9)	RAZWI	-	0
NTM0 (MMU_CFG..NTME == 1)	2 (12,11)	Indicates Non-Translated VA to PA[31,30] mapping for partition 0.	RW	0
NTM1 (MMU_CFG..NTME == 1)	2 (14,13)	Indicates Non-Translated VA to PA[31,30] mapping for partition 1.	RW	1
NTM2 (MMU_CFG..NTME == 1)	2 (16,15)	Indicates Non-Translated VA to PA[31,30] mapping for partition 2.	RW	2
NTM3 (MMU_CFG..NTME == 1)	2 (18,17)	Indicates Non-Translated VA to PA[31,30] mapping for partition 3.	RW	3
DREE	1 (19)	Device register endian control enable. Value: 0 — Device register endian control function is disabled. A data memory access uses PSW.BE as the endian type. Value: 1 — Device register endian control function is enabled. A data memory access with the following attributes, (C==0 or NTC==0), uses PSW.DRBE as the endian	RW	0

Field Name	Bits	Description	Type	Reset
		type while a data memory access with the other attributes uses PSW.BE as the endian type.		
Reserved	12 (31,20)	RAZWI	-	0

NTC0-NTC3 is the Non-translated Cacheability attributes used when the VA to PA translation process is turned off. The value definition of the NTC field is listed in Table 2 in section 2.2.2. When only VA(31) is used to access the NTC fields, only NTC0/NTC1 are used. When VA(31,30) is used to access the NTC fields, all NTC fields will be used. The NTC field assignment for each case is listed in Table 25 and Table 26.

NTM0-NTM3 is the Non-translated VA(31,30) to PA(31,30) Mapping fields when MMU_CFG.NTME is equal to 1. It is most often used to improve memory access performance on a small system without MMU or MPU by mapping different NTC attributes (e.g. cacheable/WB, cacheable/WT) to the same PA partition. Thus, the memory access type can be controlled using VA(31,30). When only VA(31) is used to access the NTM fields, only NTM0/NTM1 are used. When VA(31,30) is used to access the NTM fields, all NTM fields will be used. The NTM field assignment for each case is listed in Table 25 and Table 26.

Table 9. NTC field assignment for indexing using VA(31)

VA(31)	NTC field	NTM field
1`b0	NTC0	NTM0
1`b1	NTC1	NTM1

Table 10. NTC field assignment for indexing using VA(31,30)

VA(31,30)	NTC field	NTM field
2`b00	NTC0	NTM0
2`b01	NTC1	NTM1
2`b10	NTC2	NTM2
2`b11	NTC3	NTM3

3.4.2. TLB Access VPN Register

Mnemonic Name: mr2 (TLB_VPN)
IM Requirement: MPU optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 2, 0}

31 29	28 12	11 0
Reserved	Hrange(28,12)	Reserved

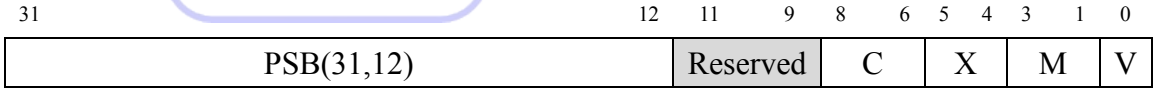
For TLB target write operation, this register contains the high virtual address limit of the MPU TLB entry to be installed. For TLB target read operation, this register holds the same read out.

Field Name	Bits	Description	Type	Reset
Hrange	17 (28,12)	Contains the the Hrange field to and from the MPU TLB structure.	RW	DC

Note that this register does not get updated with any exception VPN information as done in the MMU specification. This is due to the fact that it is Hrange, not exception VPN, which gets written into the MPU TLB structure. So there is no need for hardware to prepare the exception VPN in advance in the TLB_VPN register to reduce software effort.

3.4.3. TLB Access Data Register

Mnemonic Name: mr3 (TLB_DATA)
IM Requirement: MPU optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 3, 0}



For TLB target write operation, this register contains the physical section base, and various memory attribute information of the MPU TLB entry to be installed. For TLB target read operation, this register contains the same read out.

Field Name	Bits	Description	Type	Reset
V	1 (0)	This entry is valid and present.	RW	DC
M	3 (3,1)	Contains the M field to and from the MPU TLB structure.	RW	DC
X	2 (5,4)	Contains the X field to and from the MPU TLB structure.	RW	DC
C	3 (8,6)	Contains the C field to and from the MPU TLB structure.	RW	DC
Reserved	3 (11,9)	RAZWI		0
PSB	18 (31,14)	Contains the PSB field to and from the MPU TLB structure.	RW	DC

3.5. MPU Exceptions

Five possible exceptions could be generated after accessing the MPU TLB entry. Their names and generating conditions are listed in the following table.

Name	Generating condition	Entry vector
MPU TLB Invalid	The MPU TLB entry is invalid, that is, MPU[VA(31,29)].V == 0	TLB fill (the same entry point of the MMU TLB fill exception)
Read protection violation	The load operation has no read permission, that is, a condition based on MPU[VA(31,29)].M and processor operating mode.	TLB misc
Write protection violation	The store operation has no write permission, that is, a condition based on MPU[VA(31,29)].M and processor operating mode.	TLB misc
Non-executable code	The fetched code has no permission to be executed, that is, a condition based on MPU[VA(31,29)].X and processor operating mode.	TLB misc (the same entry point of the MMU non-executable page exception)
Reserved attribute	The value read out from the MPU TLB entry MPU[VA(31,29)] contains any reserved value in C, and M field.	TLB misc (the same entry point of the MMU reserved PTE attribute exception)

Chapter 4
Interruption Architecture

This chapter describes interrupt architecture and contains the following sections

4.1. Introduction

In Andes architecture, an *Interruption* is a control flow change of normal instruction execution generated by an *Interrupt* or an *Exception*. When an interruption happens, the processor stops processing current flow of instructions, saves enough states for later resuming of the interrupted current flow, disable interrupts, enters SuperUser Mode, and starts executing a software interruption handler.

An *interrupt* is a control flow change event generated by an asynchronous internal or external source. It includes Hardware and Software Interrupts. Hardware interrupt is any interrupt event generated from external agents or AndesCore™. Software interrupt is an interrupt event generated by an instruction executing in the AndesCore™.

An *exception* is a control flow change event generated as a by-product of instruction execution. For example, if an instruction with invalid format is encountered by an Andes processor core, a Reserved Instruction exception will be generated. An exception can also be classified as *precise*, *next-precise*, and *imprecise*.

An exception is said to be *precise* if when the exception handler is entered the instruction that causes the exception has not been completed. For example, the Reserved Instruction exception mentioned above is a precise exception.

An exception is said to be *next-precise* if when the exception handler is entered the instruction that causes the exception has been completed but the next instruction has not been completed. For example, a data watchpoint exception can be implemented as a next-precise exception.

An exception is said to be *imprecise* if when the exception handler is entered the instruction that causes the exception has been completed in the processor pipeline but the exact instruction complete/incomplete boundary following that instruction is unpredictable. For example, a Bus Error exception caused by a prefetch instruction is an imprecise exception. Most of the exceptions in the Andes Architecture are precise exceptions.

4.2. Interruption Handling in Hardware

In Andes architecture, interruption is handled in hardware based on the “interruption stack level transition” (ISLT). Currently, four interruption stack levels are defined in the architecture, 0-3. And interruption stack level 0 means no interruption. Thus, hardware behaviors for handling an interruption are defined for three interruption stack level transitions, 0/1, 1/2, and 2/3.

4.2.1. Interruption handling for interruption stack level transition 0/1 and 1/2

For ISLT 0/1 and 1/2, key interruption states will be saved and restore from and to hardware interruption stack registers. These saving and restoring states are needed since when transition into higher levels of interruption (0 → 1, and 1 → 2), the following states will be updated before fetching and executing the first instruction in the interruption handler:

- (PSW) Interruption Stack Level ← Interruption Stack Level ++
- (PSW) Global Interrupt enable ← 0
- (PSW) Privilege Mode ← 1 (Superuser mode)
- (PSW) IT/DT ← 0
- (PSW) BE ← default endian (MMU_CFG.DE)
- (PSW) DRBE ← MMU_CFG.DRDE
- (PSW) IME ← (Instruction Machine Error) ? 1 : IME
- (PSW) DME ← (Data Machine Error) ? 1 : DME
- (PSW) DEX ← (Debug Exception) ? 1 : DEX
- (PSW) HSS ← (PSW.HSS) ? INT_MASK.DSSIM : 0
- Interruption Type
- Exception VA (meaningful for only certain exceptions)
- Program Counter ← Interruption vectored entry point

So before updating these values, the old content of these registers will be saved to the lower level interruption stack registers.

Saving/restoring operations for ISLT 0/1

When entering into interruption stack level 1 from level 0, the saving of registers will be performed as follows:

- IPSW ← PSW
- IPC ← PC

When returning from interruption stack level 1 to level 0, the restoring of registers will be performed as follows:

- IPSW → PSW
- IPC → PC

Saving/restoring operations for ISLT 1/2

When entering into interruption stack level 2 from level 1, the saving of registers will be performed as follows:

- P_IPSW ← IPSW ← PSW
- P_IPC ← IPC ← PC
- P_ITYPE ← ITYPE
- P_EVA ← EVA
- P_P0 ← P0
- P_P1 ← P1

When returning from interruption stack level 2 to level 1, the restoring of registers will be performed as follows:

- P_IPSW → IPSW → PSW
- P_IPC → IPC → PC
- P_ITYPE → ITYPE
- P_EVA → EVA
- P_P0 → P0
- P_P1 → P1

This register stack design allows nested debug/error exception handling in the code area where software has not prepared itself to handle nested interruption event.

4.2.2. Interruption handling for interruption stack level transition 2/3

When an AndesCore™ transition from interruption stack level 2 to interruption stack level 3, since the hardware register stack has been used up for level 2, only very limited interruption states will be updated. The assumption is that interruption stack level 3 will be entered only when severe error condition happens during debugging or handling of a nested kernel exception. In these cases, the following states will be updated to permit minimum handling:

- (PSW) Interruption Stack Level ← 3 (i.e. maximum stack level)
- (PSW) IME ← (Instruction Machine Error) ? 1 : IME
- (PSW) DME ← (Data Machine Error) ? 1 : DME
- (PSW) DEX ← (EDM_CFG.VER < 0x0030 && Debug Exception) ? 1 : DEX
- Overflow_IPC ← Program Counter
- Program Counter ← Interruption vectored entry point

And, the following hardware behaviors still change based on the fact that the interruption stack level is 3 without updating the corresponding control states.

- Disable interrupt
- Turn off instruction/data address translation
- Use “default endian (MMU_CFG.DE)” as the data access endian
- Use “MMU_CFG.DRDE” as the device register access endian if MMU_CTL.DREE is asserted.
- For EDM_CFG.VER ≧ 0x0030, disable Hardware Single Stepping

When an AndesCore™ returns from interruption stack level 3 to interruption stack level 2 executing “return from interruption” instructions, the following states will be updated:

- Program Counter ← Overflow_IPC
- Interruption Stack Level ← 2 (i.e. maximum stack level minus 1)

4.2.3. Maximum Interruption Stack Level Option

For implementations to save hardware cost, Andes architecture provides a configuration option to allow an implementation to choose the maximum interruption stack level to be either 3 or 2. The INTLC field in the MISC_CFG system register records this choice.

The previous two sections (4.2.1 and 4.2.2) describes the operations performed at the interruption stack level transition between 0/1, 1/2, and 2/3 when the INTLC field is set to zero (i.e. maximum interruption stack level is 3).

When the INTLC field is set to 1 (i.e. maximum interruption stack level is 2), all the P_* interruption stack system registers will be removed from an implementation. And the operations performed at the interruption stack level transition between 1/2 will be changed to operations equivalent to those ones performed at the interruption stack level transition between 2/3 when the maximum interruption stack level is 3.

4.2.4. Software Lowering Interruption Stack Level

Since the level depth of the hardware interruption stack is limited, for software to handle nested interruptions, lowering interruption stack level is needed to allow unlimited number of nested interruptions to happen. The operations of lowering interruption stack level include saving lower-level interruption stack registers, reducing the interruption stack level, and later after the nested interruptions complete, restoring lower-level interruption stack registers, increasing the interruption stack level. These procedures can be shown in the following figures.

Figure 5. Operations of lowering interruption stack level from 2 to 1

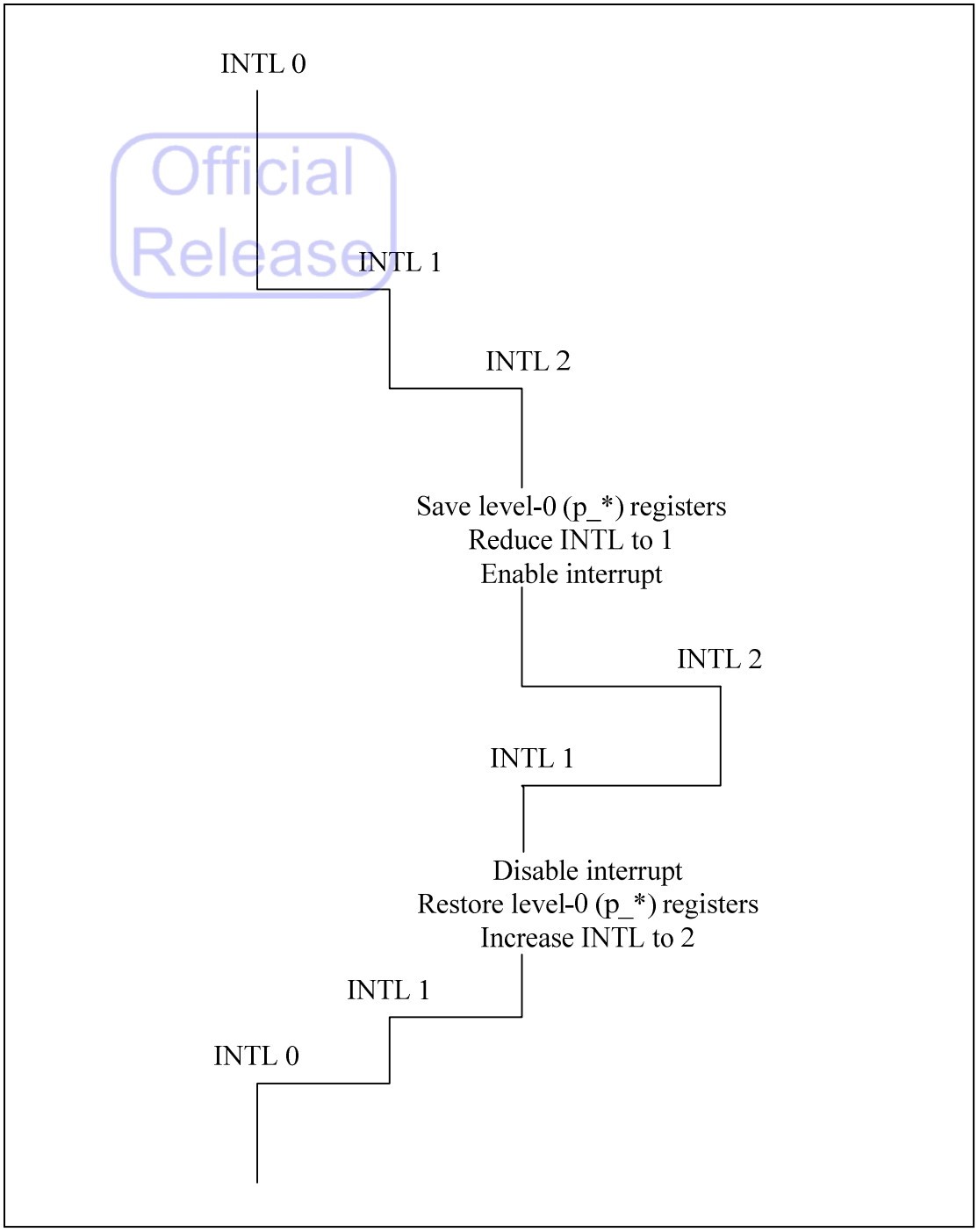
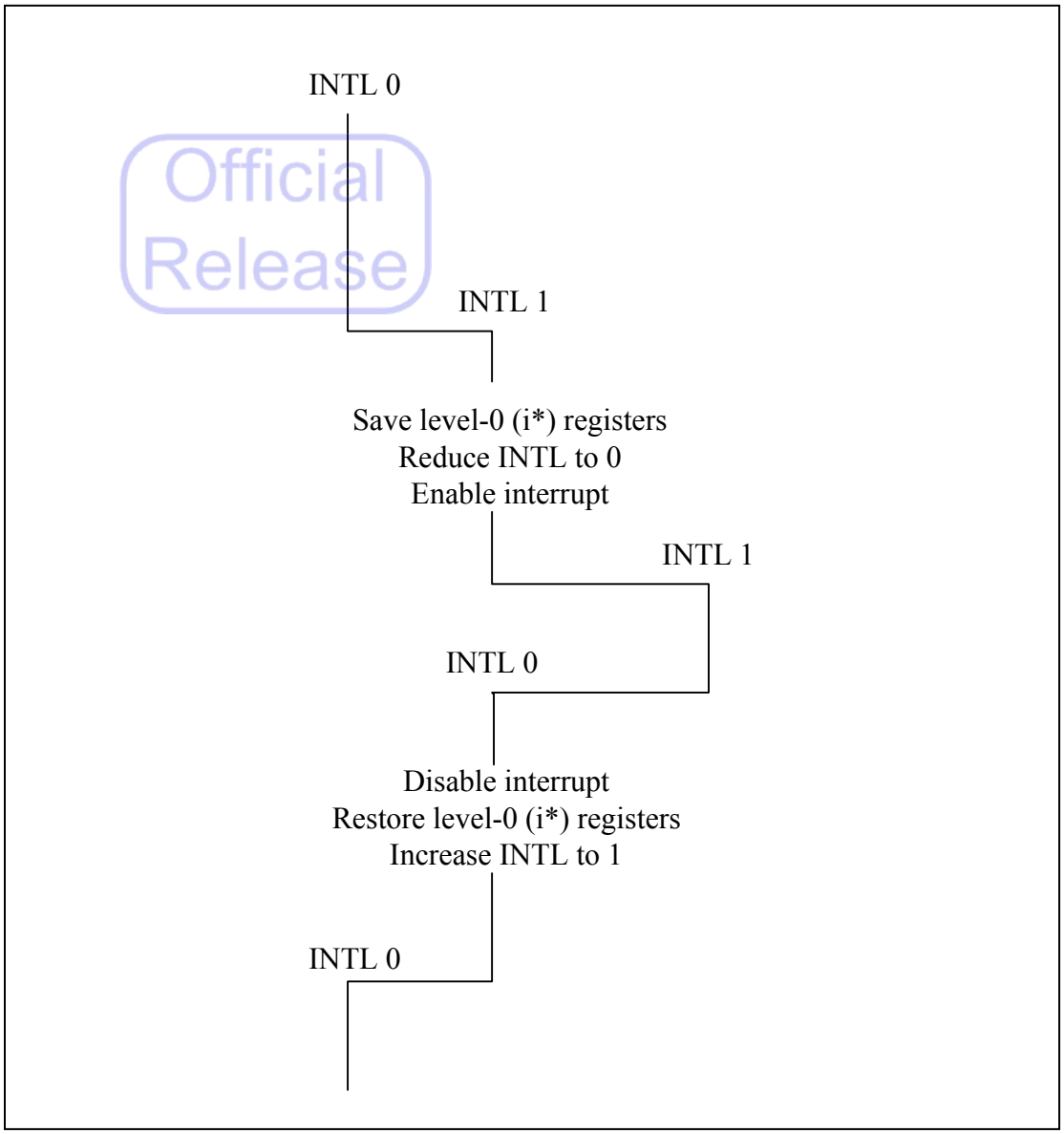


Figure 6. Operations of lowering interruption stack level from 1 to 0



Lowering interruption stack level from 2 to 1

As Figure 5 shown, when in interruption stack level 2, the interruption stack contains level-0 (p_*), level-1 (i*), and level-2 information. Lowering interruption stack level from 2 to 1 means that changing the level-1 (i*) to level-0, changing the level-2 to level-1, and the level-0 (p_*) states are no longer valid/protected. So the level-0 (p_*) information needs to be saved before the lowering happens. The system registers needed to be saved in this case include P_IPSW, P_IPC, P_ITYPE, P_EVA, P_P0, and P_P1.

Lowering interruption stack level from 1 to 0

As Figure 6 shown, when in interruption stack level 1, the interruption stack contains level-0 (i*), and level-1 information. Lowering interruption stack level from 1 to 0 means that changing the level-1 to level-0, and the level-0 (i*) states are no longer valid/protected. So the level-0 (i*) information needs to be saved before the lowering happens. The system registers needed to be saved in this case include IPSW, IPC, ITYPE, EVA, P0, and P1.

Lowering interruption stack level from 3 to 2

Interruption stack level 3 is an overflow interruption level. When in interruption stack level 3, the interruption stack contains level-0 (p_*), level-1 (i*), and level-2 information plus additional level-2 O_IPC state. In this level, the level-2 interruption type and exception VA is not recorded in any register.

If there is a need to lower the interruption stack level from 3 to 2, more steps are required to re-arrange the interruption stack states to the correct stack level hierarchy. Basically, software is required to save the level-0 (p_*) information, and then emulates a stack push operation by (1) moving IPSW, IPC, ITYPE, EVA, P0, and P1 to their corresponding P_* registers, then (2) moving PSW to IPSW, O_IPC to IPC, and (3) updating PSW to the intended operating values along with the stack level change. Once the update of PSW completes, the CPU will be operating under the interruption stack level 2.

4.3. Types of Interruptions

This section lists the types of Interruptions which can be recognized by the combination of individual vectored entry point and type information stored in the ITYPE system register. They are as follows:

- Reset/NMI
  - Cold Reset
  - Warm Reset

- Non-Maskable Interrupt (NMI)

- Interrupt
  - External interrupt
  - Performance counter interrupt
  - Software interrupt
- Debug exception
  - Instruction breakpoint
  - Data address & value break
  - BREAK exception
- TLB fill exception (Instruction/Data)
- PTE not present exception (Instruction/Data)
  - Non-Leaf PTE not present
  - Leaf PTE not present
- TLB miscellaneous exception
  - Read protection violation (Data)
  - Write protection violation (Data)
  - Page modified (Data)
  - Non-executable page (Instruction)
  - Access bit (Instruction/Data)
  - Reserved PTE Attribute (Instruction/Data)
  - TLB VLPT miss (Instruction/Data)
- System Call exception
- General exception
  - Trap exception
  - Arithmetic exception
  - Reserved instruction exception
  - Reserved value exception
  - Privileged instruction exception
  - Alignment check exception (Instruction/Data)
  - Precise bus error (Instruction/Data)
  - Imprecise bus error (Instruction/Data)
  - Nonexistence local memory address (Instruction/Data)
  - MPZIU control (Instruction/Data)
  - Coprocessor N not-usable exception
  - Coprocessor N-related exception
- Machine error exception (Instruction/Data)

- Cache locking error
- TLB locking error
- TLB multiple hit
- Cache parity/ECC error
- unimplemented page size error
- Illegal parallel memory accesses (Audio extension)

4.4. Interruption Vectored Entry Point

When an interruption happens, instruction execution will be directed to the interruption handler which can handle the particular interruption event. To speedup the interruption handling based on the importance of different interruption events, individual hardware entry point in a vectored (offset) table form is provided for separating out types of interruptions to reduce the software effort of dispatching interruption handling actions in the interruption handler. The main design principle here is that the TLB-fill and external interrupt interruption events are more frequent and are much more important performance-wise to handle them quickly.

The Andes Interruption Vectored Entry Points are defined for two different modes. The first mode is Internal Vectored Interrupt Controller (IVIC) mode where interrupts are prioritized inside an AndesCore™ using the internal interrupt controller logic. The second mode is External Vectored Interrupt Controller (EVIC) mode where interrupts are prioritized outside AndesCore™ using an external interrupt controller. These two modes can be configured using the “EVIC” field/bit of the Interruption Vector Base (IVB) system register.

The Andes Interruption Vectored Entry Points for the two modes are listed below. The starting address of the vectored offset table can be programmed using the “Interruption Vector Base” (IVB) system register. And the starting base address is 64KB-aligned. The size of the vectored entry point can be configured using “ESZ” field of the IVB system register from 4 bytes to 16/64/256 bytes.

4.4.1. Entry Points for Internal VIC Mode

- 16 entry points (9 exception + 7 interrupt)
- 4 bits offset

Offset	Entry point
0	Reset/NMI
1	TLB fill
2	PTE not present
3	TLB misc
4	TLB VLPT miss
5	Machine Error
6	Debug related
7	General exception
8	Syscall
9	HW 0
10	HW 1
11	HW 2
12	HW 3
13	HW 4
14	HW 5
15	SW 0

4.4.2. Entry Points for External VIC Mode

- 73 entry points (9 exception + 64 interrupt)
- 7 bits offset

Offset	Entry point
0	Reset/NMI
1	TLB fill
2	PTE not present
3	TLB misc
4	TLB VLPT miss
5	Machine Error

Offset	Entry point
6	Debug related
7	General exception
8	Syscall
9-72	VEP 0-63

For each entry point, multiple interruptions can map to it. Thus the interruption type (ITYPE) system register is needed to record the different types of interruptions. See "Interruption Type" system register description for detailed type information.

4.5. Priority of Interruptions

When multiple interruptions occur simultaneously, the interruption having the highest priority in the following table will be taken by the AndesCore™ as the selected interruption event for an interruption handler to process.

Table 11. Interruption Priority Table

Interruption Name (highest to lowest)	Vectored Entry Point
cold reset/warm reset	Reset/NMI
external debug request (debug interrupt)	Debug
Hardware single-stepping	Debug
NMI	Reset/NMI
Debug Data value watchpoint - imprecise	Debug
Debug Data address watchpoint – next precise	Debug
Debug Data value watchpoint – next precise	Debug
Interrupt	(Based on interrupt priority)
Debug Instruction breakpoint	Debug
Instruction alignment check	General exception
ITLB multiple hit	Machine error
ITLB fill	TLB fill
ITLB VLPT miss	TLB VLPT miss
IPTE not present (Non-leaf)	PTE not present

Interruption Name (highest to lowest)	Vectored Entry Point
IPTE not present (Leaf)	PTE not present
Reserved IPTE attribute	TLB misc
Instruction MPZIU control	General exception
ITLB non-execute page	TLB misc
IAccess bit	TLB misc
Instruction Machine Error	Machine error
Instruction nonexistent local memory address	General exception
Instruction bus error (precise)	General exception
Reserved instruction	General exception
Privileged instruction	General exception
Reserved value	General exception
Unimplemented page size error	Machine error
Break	Debug
Coprocessor	General exception
Trap/syscall	General exception
Arithmetic	General exception
Debug Data address watchpoint	Debug
Data alignment check	General exception
DTLB multiple hit	Machine error
DTLB fill	TLB fill
DTLB VLPT miss	TLB VLPT miss
DPTE not present (Non-leaf)	PTE not present
DPTE not present (Leaf)	PTE not present
Reserved DPTE attribute	TLB misc
Data MPZIU control	General exception
DTLB permission (R/W)	TLB misc
Dpage modified	TLB misc
DAccess bit	TLB misc
Data Machine Error	Machine error
Data nonexistent local memory address	General exception
Debug Data value watchpoint - precise	Debug
Data bus error - precise	General exception
Data bus error - imprecise (e.g. HW prefetch)	General exception
Instruction bus error - imprecise (e.g. HW prefetch)	General exception

4.6. Interruption Related Registers

Interruption related registers can be classified into three different categories. One category (Interruption Control Register) contains registers which control interruption behaviors. The second category (Interruption Info Register) contains registers which are used to save certain information when an interruption happens. The third category (Interruption Shadow Register) contains registers which are used to shadow Interruption Info Registers for handling nested interruptions. A summary of these registers is listed in Table 12. Please see Section 9.3 Interruption System Registers for their detailed definitions.

Table 12. Interruption Related Registers

Name	Type	Updater	Section
Interruption Control Register			
Processor Status Word	R/W	SW/HW	9.3.1
Interruption Vector Base	R/W	SW	9.3.4
Interruption Masking	R/W	SW	9.3.15
Interruption Pending	R/W	SW/HW	9.3.16
Interruption Info Register			
Exception Virtual Address	RO	HW	9.3.5
Interruption Type	RO or RW	SW/HW	9.3.7
Machine Error Log	R/W	HW	9.3.9
Interruption Shadow Register			
Interruption PSW	R/W	SW/HW	9.3.2
Previous IPSW	R/W	SW/HW	9.3.3
Previous EVA	R/W	SW/HW	9.3.6
Previous ITYPE	R/W	SW/HW	0
Interruption Program Counter	R/W	SW/HW	9.3.10
Previous IPC	R/W	SW/HW	9.3.11
Overflow IPC	R/W	SW/HW	9.3.12
Previous P0 / Previous P1	R/W	SW/HW	9.3.13

Basically, the following register stacks are built to handle potentially 2 levels + 1 error level nested interruptions in hardware.

⊗ Updates performed on interruption stack level transition 0/1:

- PC ⇔ IPC
- PSW ⇔ IPSW
- VA ⇒ EVA
- Interruption ⇒ ITYPE

⊗ Updates performed on interruption stack level transition 1/2:

- PC ⇔ IPC ⇔ P_IPC
- PSW ⇔ IPSW ⇔ P_IPSW
- VA ⇒ EVA ⇔ P_EVA
- Interruption ⇒ ITYPE ⇔ P_ITYPE
- P0 ⇔ P_P0
- P1 ⇔ P_P1

⊗ Updates performed on interruption stack level transition 2/3

- PC ⇔ O_IPC

The level 2 interruption is typically used to handle TLB VLPT miss exception, debug exception for privileged code, recoverable error condition, or hypervisor mode (if implemented) interception. The level 3 interruption is typically used to handle error/debug events in the level 2 interruptions. If level 3 interruption is entered, the interruption handler may not be able to return to the interrupted level 2 interruption instruction flow.

4.7. Imprecise Exception

An imprecise exception is generated after the completion of the instruction that causes the exception. And the delay between the reporting of an imprecise exception and the completion of its generating instruction is unpredictable. So which instruction or even which process will be interrupted by the reporting of an imprecise exception is unknown.

However, to provide minimum protection for higher interruption stack levels from interfering imprecise exceptions of lower interruption stack levels, an implementation may only allow an imprecise exception to be taken when the interrupt stack level that generates the imprecise exception is equal or larger than the current interrupt stack level. This way, a group of successive imprecise exceptions from user program will not cause unexpected imprecise exceptions to suerperuser mode exception handlers.

4.8. Details of Individual Interruptions

4.8.1. Cold Reset

A Cold Reset exception happens when the Reset signal is asserted to an AndesCore™ and the AndesCore™ is in an in-active state (power-down, sleep, etc). This exception is not maskable. To handle the Cold Reset exception, an AndesCore™ performs a complete processor state initialization.

4.8.2. Warm Reset

A Warm Reset exception happens when the Reset signal is asserted to the AndesCore™ and the AndesCore™ is in a running state. This exception is not maskable. To handle the Warm Reset exception, an AndesCore™ performs a subset of the complete processor state initialization. The purpose of the warm reset is to keep certain processor states around without initialization to aid debugging.

4.8.3. Non-Maskable Interrupt (NMI)

A Non-Maskable Interrupt exception happens when the NMI signal is asserted to an AndesCore™. This exception is not maskable including using PSW.GIE bit. Note that although the name has the word “interrupt” in it, this is really an exception, not an interrupt.

4.8.4. Machine/Cache Error

Due to the nature of handing machine/cache errors for this exception handler, the instruction and data memory accesses within this exception handler will use *un-cached* space depending on if it is an instruction related or data related error.

The machine error exception includes several different types of errors. Please see Table 21 (page 128) for detailed error types.

4.8.5. Instruction Alignment Check

The Instruction Alignment Check exception is triggered for each instruction fetch which does not have bit 0 of the fetch address be 0. When this exception happens, since the PC has already been updated before the instruction fetch, the IPC will contain the mis-aligned instruction fetch address while the EVA will have the address of the previous instruction which generates the mis-aligned instruction fetch address.

4.8.6. Reserved PTE Attribute

The Reserved PTE Attribute exception is generated when a valid PTE detected by hardware during load/store instructions or instruction fetching contains a reserved value in the “M” or “C” fields of the PTE.

To clear/fix this exception, software is required to invalidate the incorrect, but valid, PTE in the TLB before inserting a new PTE that fixes the problem.

4.9. Internal Vector Interrupt Controller

The Internal Vector Interrupt Controller is a simpler interrupt controller which prioritizes

among 6 hardware and 1 software interrupts. It is used for a simpler SoC platform which does not require a more sophisticated and bigger External Vector Interrupt Controller. The priority of these 7 interrupts is defined as follows:

Priority (0: highest, 6: lowest)	Interrupt
0	HW0
1	HW1
2	HW2
3	HW3
4	HW4
5	HW5
6	SW0

Chapter 5
Performance Monitoring

This chapter describes interrupt architecture and contains the following sections

The AndesCore™ provides performance monitoring mechanism for programmers to obtain program running statistics and performance data. These performance data will help the programmers to understand the behaviors of their programs so that they can easily pin-point the performance bottlenecks in their code and continuously tuning the code to remove those bottlenecks.

The performance monitoring mechanism consists of several 32-bit performance monitoring counters and their associated control registers. The exact number of counters is implementation-dependent. However, at least one cycle/instruction counter and two event counters will be available to provide performance monitoring functionalities.

The performance counters and their controls provide the following performance monitoring features:

- One cycle/instruction counter.
- Two event counters.
- Individual controls:
  - Counter enable.
  - Interrupt enable.
  - Overflow status/clear bit.
  - Counting event selection among maximum of 64 events.
  - Superuser and user mode event counting filters.
- Readable/Writable in privileged mode.
- Continuing counting after overflow.

The performance monitoring related registers are all privileged resources. And they are described in detail in Section 9.6 Performance Monitoring Registers. A summary of these registers is listed in Table 13.

Table 13. Performance Monitoring related registers

Name	Type	Updater	Section
Performance Counters	R/W	SW/HW	9.6.1
Performance Monitoring Control	R/W	SW/HW	9.6.2

5.1. Interrupt Interface

When any of the performance monitoring counters overflows and the interrupt enable control is set, an interrupt will be generated. The interrupt will be exported out of the core through an interface signal, PM_INT_PEND. This signal can then be connected with the internal or external interrupt controller. This signal could be synchronous or asynchronous, depending on the implementation.

Note that if software wants to know which counter generated the interrupt, the software has to examine the individual overflow bits to get this information.

Chapter 6
Local Memory

In addition to caches, Andes architecture provides support for fast-accessed low latency local memories in the AndesCore™. The local memories are divided into instruction local memory (ILM) and data local memory (DLM) for increased performance and design simplicity. This chapter describes the detail of local memory and contains the following sections

6.1. Local Memory Base Address

At present, there are two different rules for the base physical address alignment of the Andes local memory. It is indicated in the BSAV field of the ICM_CFG and DCM_CFG system registers.

When the BSAV field is equal to 0, the base physical address of the Andes local memory is aligned to 1MB boundary for any local memory whose size is smaller than or equal to 1MB regardless of the real size. Any access outside of the local memory address range within the allocated 1MB region will cause a "nonexistent local memory address" exception. For a local memory with *Double-Buffer* mode turned on, the existent non-exception address range is equal to half of the physical local memory size.

When the BSAV field is equal to 1, the base physical address of the Andes local memory is aligned to a power-of-2 byte boundary that has to be the local memory size. At this configuration, there is no "nonexistent local memory address" exception regardless if the *Double-Buffer* mode is disabled or enabled. Any access outside the local memory address range (physical DLM size if *Double-Buffer* mode is disabled or half of physical DLM size if *Double-Buffer* mode is enabled) will access the other parts of the system memory.

If the instruction local memory base address is programmed to be the same as the data local memory base address, UPREDICTABLE behavior will happen to the registers and the local memory content.

6.2. Data Local Memory Access Modes

To provide different data processing models for different applications, the Andes core data local memory has two different access modes for the processor pipeline and the DLM DMA engine. The first mode is the *normal* access mode where the DLM address range, seen by the processor pipeline and the DMA engine, covers the whole DLM size. In this mode at any moment, the processor core and the DMA engine will see the same

DLM address space and it is possible for them to access the same DLM location at the same time using the same address. Please note that the performance of the DLM data access may be degraded if there are frequent simultaneous DLM accesses from the core pipeline and the DMA engine. This performance characteristic of DLM access is implementation-dependent.

The second mode is the *Double-Buffer* access mode where the DLM is divided into two banks and the processor pipeline is directed to access one bank while the DMA engine is directed to access the other bank. In this mode, the address range, seen by the processor pipeline and the DMA engine, is only half of the physical DLM size; and the memory space of this address range is different for the processor pipeline and the DMA engine. If they access DLM using the same address at the same time, they will access different memory locations simultaneously without any contention. This arrangement is good for the processor pipeline and DMA engine doing their own work without interfering with each other. However, to facilitate sharing and coarse-grained task-level pipelining, the accessed bank can be switched between the processor pipeline and the DMA engine. So the data prepared by the DMA engine can be used by the processor pipeline after the bank switch; and vise versa. The bank switch of this mode is the DBB field of the DLMB register. In the *Double-Buffer* mode, any access outside the banked address range (half of physical DLM size) will cause a "Nonexistent local memory address" exception.

Transition between the double buffer mode and the normal mode may cause UNPREDICTABLE data lost in the data local memory content. Software should re-initialize the data local memory after changing the access mode.

6.3. Local Memory Address Range

The local memory address ranges are listed in the following tables. LMB_BPA represents the based address field of the ILM and DLM local memory base address system registers (ILMB_IBPA in Section 9.4.7 and DLMB_DBPA in Section 9.4.8). And "." serves as a bit concatenation operator. 0(m,n) and 1(m,n) represent a replication of bit 0 or bit 1 in the bit range specified.

- Non-Double-buffer mode (for ILM and DLM):

LM Size	Start	End
4KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,12).1(11,0)
8KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,13).1(12,0)
16KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,14).1(13,0)
32KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,15).1(14,0)
64KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,16).1(15,0)
128KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,17).1(16,0)
256KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,18).1(17,0)
512KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,19).1(18,0)
1024KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).1(19,0)

- Double-buffer mode (for DLM only):

LM Size	Start	End
4KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,11).1(10,0)
8KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,12).1(11,0)
16KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,13).1(12,0)
32KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,14).1(13,0)
64KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,15).1(14,0)
128KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,16).1(15,0)
256KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,17).1(16,0)
512KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,18).1(17,0)
1024KB	LMB_BPA(31,20).0(19,0)	LMB_BPA(31,20).0(19,19).1(18,0)

6.4. Local Memory Use Constraints

Local memories are specially built memories for fast access. So to reduce hardware overhead an implementation may impose certain usage constraints on the local memory such that it may not be used as general as a regular memory. Example constraints are listed below. For accurate constraints of an implementation, please consult the implementation documents.

- The local memory needs to be mapped onto an uncacheable region; otherwise, UPREDICTABLE behavior may happen to the local memory content.

- The local memory needs to be mapped onto a page whose size is larger than or equal to the local memory size.
- The data local memory content will be changed UNPERDICTABLLY when the data local memory access mode is transitioned between the double buffer mode and the normal (non-double buffer) mode.

Chapter 7
Local Memory DMA

The local memory DMA is provided to transfer blocks of data between AndesCore™ local memory and external, off-core, memory in parallel with the AndesCore™ execution pipeline. To transfer large amounts of data to or from local memory, it is more efficient to use the DMA engine instead of the AndesCore™ load/store instructions. This chapter describes the DMA detail and contains the following sections

7.1. Features

The AndesCore™ Local Memory DMA engine has the following features:

- Two channels
- One active channel
- Programmed using physical addressing
- Software takes care of address translation and permission check
- Only accessed by privileged mode (e.g. Superuser)
- For both instruction and data local memory
- External address can be incremented with stride
- Aligned internal address
- Aligned external address for basic configuration
- Optional Un-aligned External Address (UNEA) feature which allows un-aligned external address for element transfer.
- Optional 2-D Element Transfer (2DET) feature which provides an easy way to transfer two-dimensional blocks from external memory.

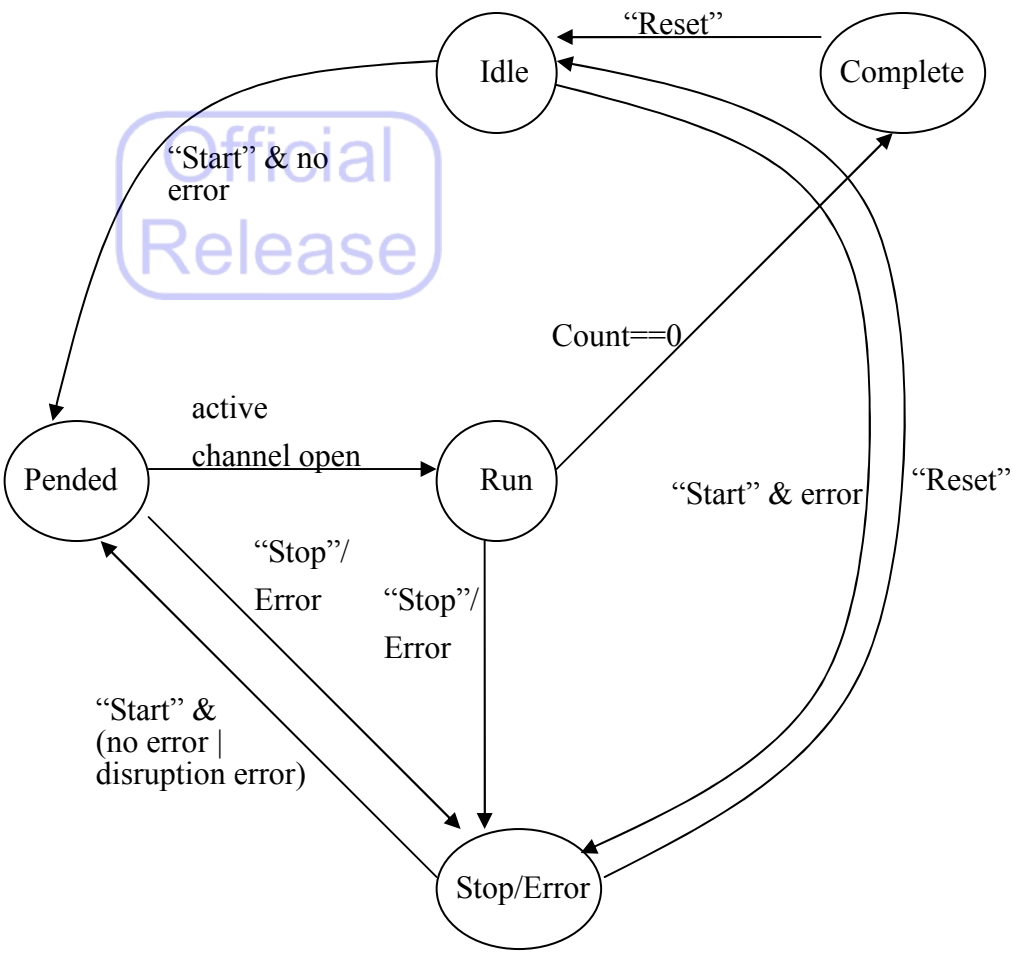
7.2. Basic Rules of Operation

A. Architecturally, a maximum of two DMA channels can be implemented. However, future expansion to more DMA channels is possible. And if more than one DMA channels are implemented, only one DMA channel can be active.

B. Each DMA channel has its own set of control and status registers, and its own operating state.

C. A DMA channel will be transitioned between the following states: Idle, Run, Pended, Stop/Error, and Complete. The following figure illustrates these states and the transitions between them. In the figure, “Start”, “Stop”, and “Reset” are the DMA action command written to the DMA Action register for the channel. And the active channel is occupied if the other channel is in the Run state.

Figure 7. DMA Channel State Transition Diagram



D. The Stop and Error share the same state, but they can be distinguished by checking if there is any error recorded in the DMA Status register.

E. If an error has happened, software needs to perform the following operations to restart the DMA transfer (except DMA transfer disruption error):

- Fixes the errors based on the error conditions in the DMA status register.
- Writes the DMA Action register with a "Reset" command to clear the error conditions/interrupt generated, and put the DMA channel into the "Idle" state.
- Checks the DMA Status register to make sure the DMA channel is in the "Idle" state.
- Writes the DMA Action register with a "Start" command to re-start the DMA transfer.

If a DMA transfer disruption error has happened, since the original DMA setup

values will be maintained without being overwritten, software can perform fewer steps to restart the DMA transfer:

- Checks the DMA Status register to make sure the DMA channel is in the "Idle" state.
- Writes the DMA Action register with a "Start" command to re-start the DMA transfer. The disruption error condition in the DMA status register will be automatically cleared on this re-start.

F. To pick a DMA channel to use, software can check the DMA Global Control and Status Word register to see if any channel is in the "Idle" state.

G. A DMA channel data transfer region is programmed using physical addresses. And software is responsible for virtual address to physical address translation and all access permission checks.

H. The baseline DMA engine supports only transfer element size aligned internal and external memory addresses. However, optional "Unaligned External Address" feature may be implemented to support transferring data between unaligned external memory addresses and aligned internal memory addresses.

I. There is no coherency support between the level-one caches and the DMA accesses. Thus the DMA accesses must be either from un-cached space or if from cached space, the transfer region should have already been properly written back by software instructions.

J. While a DMA channel is running, there is no ordering requirement of memory accesses between those generated by the DMA and those generated by the reads and writes of an Andes core. When a DMA channel has completed running, all its transfers are visible to all other readers in the system. All memory accesses of the DMA transfers are ordered by the channel running sequence and within a running channel, all memory accesses are ordered from the starting address to the end address of the transfer.

K. An interrupt can be generated when a DMA channel transfer is completed, explicitly stopped, or terminated due to an error. There are individual control bits to specify the intended interrupt generation behavior.

L. Once a DMA-related interrupt is generated, the interrupt status can be cleared by issuing either a "Reset" command or when in the "no-error Stop" state a "Start" command.

7.3. Related registers

The local memory DMA is controlled and initiated by accessing the related DMA system registers.

Local memory DMA related registers can be classified into two different categories. One category (DMA Global Register) contains registers which represent or control overall DMA status and behaviors. The second category (DMA per Channel Register) contains registers which are used to reflect the status and control the behaviors of a selected DMA channel. Table 14 lists all the DMA related registers. All these registers require superuser mode privilege to access. Please see Section 9.7 Local Memory DMA Registers for their detailed definitions.

Table 14. Local memory DMA related registers.

Name	Type	Updater	Section
DMA Global Registers			
DMA Configuration	RO	-	9.7.1
DMA Global Control and Status Word	R/W	Inst/Hardware	9.7.2
DMA Channel Select	R/W	Inst	9.7.3
DMA per Channel Registers			
DMA Action	WO	Inst	9.7.4
DMA Setup	R/W	Inst	9.7.5
DMA Internal Starting Address	R/W	Inst/Hardware	9.7.6
DMA External Starting Address	R/W	Inst/Hardware	9.7.7
DMA Transfer Element Count	R/W	Inst/Hardware	9.7.8
DMA Status	RO	Hardware	9.7.9
DMA 2D Setup	R/W	Inst	9.7.10
DMA 2D Startup Control	R/W/WS	Inst/Hardware	9.7.11

7.4. Interrupt Interface

When a DMA channel transfer is completed, explicated stopped, or stopped due to an error and the corresponding interrupt enable control is set, an interrupt will be generated. The interrupt will be exported out of the core through an interface signal, DMA_INT_PEND. This signal can then be connected with the internal or external interrupt controller. This signal could be synchronous or asynchronous, depending on the implementation.

Note that if software wants to know which DMA channel or what state of a DMA channel generated the interrupt, the software has to examine the corresponding status bits to get this information.

Once a DMA-related interrupt is generated, the interrupt status can be cleared by issuing either a “reset” command or when in the “no-error Stop” state a “start” command.

7.5. Example Codes

The “`li reg, imm32`” instruction in the example codes is a software pseudo assembly instruction of “loading a 32-bit immediate value into a register”. It will be translated into the following two hardware instructions by the NDS32 assembler:

- `sethi reg, hi20(imm32)`
- `ori reg, lo12(imm32)`

where hi20(imm32) is the higher 20-bits of the imm32 value and lo12(imm32) is the lower 12-bits of the imm32 value.

7.5.1. DMA Setup

The following code sequence illustrates a typical DMA setup flow.

```
li r1, 0x80000000    ! Enable DMA
mtsr r1, DMA_GCSW    ! ignored if DMA already enabled
li r1, 0x00000000    ! Channel number
mtsr r1, DMA_CHNSEL ! Channel selection
dsb                  ! make sure the selected channel is ready
li r1, SETUP_CONST
mtsr r1, DMA_SETUP   ! DMA transfer setup
li r1, ISADDR_CONST
mtsr r1, DMA_ISADDR  ! Setup internal starting address
li r1, ESADDR_CONST
mtsr r1, DMA_ESADDR  ! Setup external starting address
li r1, TCNT_CONST
mtsr r1, DMA_TCNT    ! Setup total transfer count
li r1, 0x00000003
mtsr r1, DMA_ACT     ! Reset action
li r1, 0x00000001
mtsr r1, DMA_ACT     ! Start action
dsb                 ! make sure channel has started
! example code ends here ...
mfsr r1, DMA_STATUS  ! Polling DMA channel status
.............
```


7.5.2. DMA Channel Selection

If you do not know which DMA channel is idle for use, you can use the following code sequence to find an idle DMA channel.

```
Find_Idle_Channel:
   mfsr r1, DMA_GCSW
   andi r2, r1, 0x07        ! Extract channel 1 status
   beqz r2, Channel1_Free
   andi r2, r1, 0x38        ! Extract channel 2 status
   beqz r2, Channel2_Free
   j No_Free_Channel
Channel1_Free:
   ……
Channel2_Free:
   ……
No_Free_Channel:
   ……
```


7.5.3. Polling DMA Channel Status

The following code sequence polls the "complete" status of a selected DMA channel.

```
......... ! DMA channel setup and action command
... dsb   !  make sure channel has started
! make sure a "dsb" is between DMA channel setup and this code
.........
! example code begins here ...
Check_Status:
   mfsr r1, DMA_STATUS
   andi r0, r1, 0x4
   bnez r0, Transfer_Complete
   j Check_Status
Transfer_Complete:
   ……
```

Chapter 8
High Speed Memory Port

This chapter describes high speed memory port on the next page.

In addition to AMBA AHB-style system interface, to reduce memory access delays and thus increase system performance, an Andes core may provide a high speed memory port interface which has higher bus protocol efficiency and can run at a higher frequency to connect to a memory controller. It is called High Speed Memory Port.

This high speed memory port will use a protocol that has higher speed than the AMBA AHB bus protocol. Please see Andes core implementation documents for detailed protocol definitions and signals.

Version Note: Version 1 of HSMP definition is only implemented in the following implementations:

- N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)

Please see Version 2 definitions for other implementations.

The High Speed Memory Port is defined within a memory region aligned at the 1MB boundaries. The beginning and ending of this region are defined in two system registers.

Table 15. High Speed Memory Port related registers

Name	Type	Updater	Section
High Speed Memory Port Starting Address	R/W	Inst	9.4.10
High Speed Memory Port Ending Address	R/W	Inst	9.4.11

HSMP Version 2

The High Speed Memory Port is defined within a power-of-2 range of aligned memory region from 1MB to 4GB. The beginning and range of this region are defined in one system register.

Table 16. High Speed Memory Port related registers

Name	Type	Updater	Section
High Speed Memory Port Starting Address	R/W	Inst	9.4.10

Configuration Note:

The High Speed Memory Port and the defined registers only have their proper usages and meanings when a main bus port exists. The main bus port can use any possible protocol such as AHB, OCP, AXI, etc. Please do not confuse the “High Speed Memory Port” with this “main bus protocol”. They are related here, but are actually different concepts.

Chapter 9
System Register Definitions

This chapter describes system registers and contains the following sections

9.1. Terminology

- **IM**: Implementation dependent/determined.
- **DC**: The reset value is don't care.
- **WT**: WhitTiger implementation.
- **WT=X**: Expected value is X in AndesCore™ N12 implementation.
- **RO**: Read Only register/field by software. Any software write to RO register/field will be silently ignored by Hardware. So it can also be denoted as ROWI.
- **WO**: Write Only register/field by software. Any software read of WO register/field will get a value of 0. So it an also be denoted as WORAZ.
- **RW**: Read/Write register/field by software.
- **WS**: Write shadow of another register. That is, writing to another register will update this field as well.
- **W1C:** Write one to clear the state of the field.
- **RAZWI**: Read As Zero Write Ignored register/field by software. This behavior is for reserved register fields.
- **SR Encoding**: System Register instruction encoding index. It is 10 bits. It is partitioned into 3 parts, Major, Minor, and Extension. The Major index is 3 bits. The Minor index is 4 bits. The Extension index is 3 bits.
- **Reserved** value in a **RW** field: Writing a reserved value into a RW field will cause a "Reserved Value" exception.

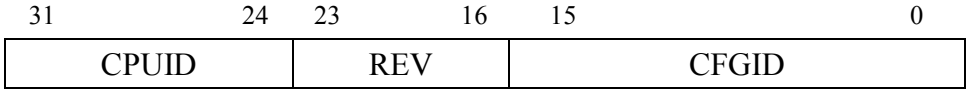
9.2. Configuration System Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
cr0	CPU_VER	0	0	0	90
cr1	ICM_CFG	0	1	0	91
cr2	DCM_CFG	0	2	0	93
cr3	MMU_CFG	0	3	0	96
cr4	MSC_CFG	0	4	0	101
cr5	CORE_ID	0	0	1	106

9.2.1. CPU Version Register

Mnemonic Name: cr0 (CPU_VER)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {0, 0, 0}



Field name	Bits	Description	Type	Reset
Config ID (CFGID)	16 (15,0)	Used to distinguish any other minor configuration differences. The current definition is as follows: Bit (n) – Meaning Bit (0) – Performance extension exists? Bit (1) – 16-bit extension exists? Bit (2) – Performance extension 2 exists? Bit (3) – COP/FPU extension exists? Bit (4) – String extension exists?	RO	IM
Revision (REV)	8 (23,16)	Used to indicate the Revision number of a particular CPU version.	RO	IM
CPU ID (CPUID)	8 (31,24)	Used to distinguish major CPU versions. The current definition is as follows: CPU version – Value N12 – 0x0C N10 – 0x0A N9 – 0x09	RO	IM

9.2.2. Instruction Cache/Memory Configuration Register

Mnemonic Name: cr1 (ICM_CFG)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {0, 1, 0}

31 15	14 13	12 10	9	8 6	5 3	2 0
Reserved	BSAV	ILMB	ILCK	ISZ	IWAY	ISET

Field name	Bits	Description	Type	Reset
Icache sets per way (ISET)	3 (2,0)	Icache sets (# of cache lines) per way: Encoding: Meaning 0: 64 1: 128 2: 256 3: 512 4: 1024 5: 2048 6: 4096 7: Reserved	RO	IM
Icache ways (IWAY)	3 (5,3)	Icache ways: Encoding: Meaning 0: Direct-mapped 1: 2 way 2: 3 way 3: 4 way 4: 5 way 5: 6 way 6: 7 way 7: 8 way	RO	IM
Icache line size	3 (8,6)	Icache block (line) size: Encoding: Meaning	RO	IM

<table>
<tr><th>Field name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>(ISZ)</td><td></td><td><table>
<tr><td>0</td><td>No Icache</td></tr>
<tr><td>1</td><td>8 bytes</td></tr>
<tr><td>2</td><td>16 bytes</td></tr>
<tr><td>3</td><td>32 bytes</td></tr>
<tr><td>4</td><td>64 bytes</td></tr>
<tr><td>5</td><td>128 bytes</td></tr>
<tr><td>6-7</td><td>Reserved</td></tr>
</table></td><td></td><td></td></tr>
<tr><td>Icache locking support (ILCK)</td><td>1
(9)</td><td>Icache locking support:<table>
<tr><td>Encoding</td><td>Meaning</td></tr>
<tr><td>0</td><td>No locking support</td></tr>
<tr><td>1</td><td>With locking support</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
<tr><td>On-chip Instruction Local Memory Base (ILMB)</td><td>3
(12,10)</td><td>Indicate how many On-chip Instruction Local Memory Base register exist.<table>
<tr><td>Encoding</td><td>Meaning</td></tr>
<tr><td>0</td><td>No ILMB exists.</td></tr>
<tr><td>1</td><td>One ILMB exists.</td></tr>
<tr><td>2-7</td><td>Reserved</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
<tr><td>BSAV</td><td>2
(14,13)</td><td>ILM base register alignment version indication:<table>
<tr><td>Encoding</td><td>Meaning</td></tr>
<tr><td>0</td><td>ILM base address is 1MB-aligned.</td></tr>
<tr><td>1</td><td>ILM base address is aligned to local memory size that has to be power of 2.</td></tr>
<tr><td>2-3</td><td>Reserved</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
<tr><td>Reserved</td><td>17
(31,15)</td><td></td><td>RAZWI</td><td>0</td></tr>
</table>

9.2.3. Data Cache/Memory Configuration Register

Mnemonic Name: cr2 (DCM_CFG)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {0, 2, 0}

31 15	14 13	12 10	9	8 6	5 3	2 0
Reserved	BSAV	DLMB	DLCK	DSZ	DWAY	DSET

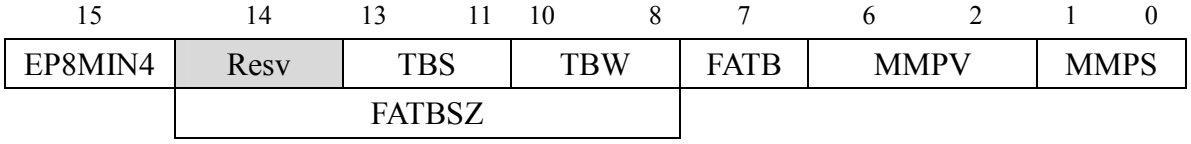
Field name	Bits	Description	Type	Reset
Dcache sets per way (DSET)	3 (2,0)	Dcache sets (# of cache lines) per way: 0: 64 1: 128 2: 256 3: 512 4: 1024 5: 2048 6: 4096 7: Reserved	RO	IM
Dcache ways (DWAY)	3 (5,3)	Dcache ways: 0: Direct-mapped 1: 2 way 2: 3 way 3: 4 way 4: 5 way 5: 6 way 6: 7 way 7: 8 way	RO	IM
Dcache line size	3 (8,6)	Dcache block (line) size: Encoding | Meaning	RO	IM

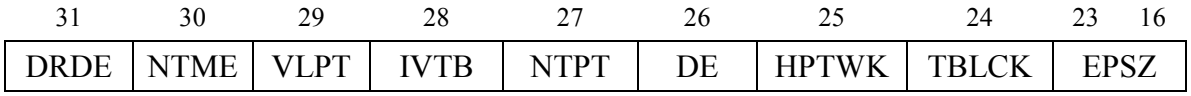
<table>
<tr><th>Field name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>(DSZ)</td><td></td><td><table>
<tr><td>0</td><td>No Dcache</td></tr>
<tr><td>1</td><td>8 bytes</td></tr>
<tr><td>2</td><td>16 bytes</td></tr>
<tr><td>3</td><td>32 bytes</td></tr>
<tr><td>4</td><td>64 bytes</td></tr>
<tr><td>5</td><td>128 bytes</td></tr>
<tr><td>6-7</td><td>Reserved</td></tr>
</table></td><td></td><td></td></tr>
<tr><td>Dcache locking support (DLCK)</td><td>1
(9)</td><td>Dcache locking support:<table>
<tr><td>Encoding</td><td>Meaning</td></tr>
<tr><td>0</td><td>No locking support</td></tr>
<tr><td>1</td><td>With locking support</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
<tr><td>On-chip Data Local Memory Base (DLMB)</td><td>3
(12,10)</td><td>How many On-chip Data Local Memory Base register exist. It also indicates if double buffer mode is supported or not. All external data local memory configuration will have the “One bank” value which means “no double buffer mode support”.<table>
<tr><td>Encoding</td><td>Meaning</td></tr>
<tr><td>0</td><td>No DLMB exists.</td></tr>
<tr><td>1</td><td>One DLMB exists and no double buffer mode support.</td></tr>
<tr><td>2</td><td>One DLMB exists and with double buffer mode support.</td></tr>
<tr><td>3-7</td><td>Reserved</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
<tr><td>BSAV</td><td>2
(14,13)</td><td>DLM base register alignment version indication:<table>
<tr><td>Encoding</td><td>Meaning</td></tr>
<tr><td>0</td><td>DLM base address is 1MB-aligned.</td></tr>
<tr><td>1</td><td>DLM base address is aligned to local</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
</table>

Field name	Bits	Description		Type	Reset
			memory size that has to be power of 2.		
		2-3	Reserved		
Reserved	17 (31,15)			RAZWI	0

9.2.4. MMU Configuration Register

Mnemonic Name: cr3 (MMU_CFG)
IM Requirement: Required
Access Mode: Superuser
SR Value {Major, Minor, Extension}: {0, 3, 0}





Field Name	Bits	Description	Type	Reset
Memory Management Protection Scheme (MMPS)	2 (1,0)	The major category for the Memory Management Protection Scheme: Value: Meaning 0: No memory management 1: Protection Unit 2: TLB MMU 3: Reserved	RO	IM
Memory Management Protection Version Number (MMPV)	5 (6,2)	Indicates the version number of the memory management protection scheme.	RO	IM
Fully-associative TLB (FATB)	1 (7)	Under the TLB MMU memory management scheme, is the TLB a fully-associative structure: Value: Meaning 0: Non-fully-associative 1: Fully-associative	RO	IM
TLB size	7	If the TLB is fully-associative, this	RO	IM

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>(fully-associative)
(FATBSZ)</td><td>(14,8)</td><td>field indicates the number of TLB entries. This 7 bits field will be reused in the following fields if the TLB is non-fully-associative.</td><td></td><td></td></tr>
<tr><td>TLB ways
(non-fully-associative)
(TBW)</td><td>3
(10,8)</td><td>The number of ways in a TLB cache:<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Direct-mapped</td></tr><tr><td>1</td><td>2 ways</td></tr><tr><td>2</td><td>3 ways</td></tr><tr><td>3</td><td>4 ways</td></tr><tr><td>4</td><td>5 ways</td></tr><tr><td>5</td><td>6 ways</td></tr><tr><td>6</td><td>7 ways</td></tr><tr><td>7</td><td>8 ways</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>TLB sets per way
(non-fully-associative)
(TBS)</td><td>3
(13,11)</td><td>The sets per way of the TLB cache:<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>4</td></tr><tr><td>1</td><td>8</td></tr><tr><td>2</td><td>16</td></tr><tr><td>3</td><td>32</td></tr><tr><td>4</td><td>64</td></tr><tr><td>5</td><td>128</td></tr><tr><td>6</td><td>256</td></tr><tr><td>7</td><td>Reserved</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>Reserved
(non-fully-associative)</td><td>1
(14)</td><td>Reserved</td><td>RO</td><td>IM</td></tr>
<tr><td>Is 8KB page supported while minimum page is 4KB?
(EP8MIN4)</td><td>1
(15)</td><td>Indicates if 8KB page is supported under the minimum 4KB page configuration. For AndesCore N12, pages supported is (4KB, 1MB) or (8KB, 1MB), but not (4KB, 8KB, 1MB).<table><tr><th>Value</th><th>Meaning</th></tr></table></td><td>RO</td><td>IM
(WT=0)</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table><tr><td>0</td><td>8KB page is not supported under the 4KB page configuration.</td></tr><tr><td>1</td><td>8KB page is supported under the 4KB page configuration.</td></tr></table></td><td></td><td></td></tr>
<tr><td>Extra page size supported (EPSZ)</td><td>8
(23,16)</td><td>Bit enable to indicate additional supported page sizes.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>‘bxxxxxxx1</td><td>16KB</td></tr><tr><td>‘bxxxxxx1x</td><td>64KB</td></tr><tr><td>‘bxxxxx1xx</td><td>256KB</td></tr><tr><td>‘bxxxx1xxx</td><td>1MB</td></tr><tr><td>‘bxxx1xxxx</td><td>4MB</td></tr><tr><td>‘bxx1xxxxx</td><td>16MB</td></tr><tr><td>‘bx1xxxxxx</td><td>64MB</td></tr><tr><td>‘b1xxxxxxx</td><td>256MB</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>TLB locking support (TBLCK)</td><td>1
(24)</td><td>Indicate if TLB has locking support:<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>No Locking support</td></tr><tr><td>1</td><td>With Locking support</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>Hardware Page Table Walker implemented (HPTWK)</td><td>1
(25)</td><td>Indicate if a Hardware Page Table Walker is implemented or not:<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>No HPTWK</td></tr><tr><td>1</td><td>With HPTWK</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>Default Endian (DE)</td><td>1
(26)</td><td>The default endian bit. This bit will be copied to PSW.BE on reset and on interruption stack level transition 0->1 and 1->2.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Default is Little Endian</td></tr><tr><td>1</td><td>Default is Big Endian</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>Partitions for</td><td>1</td><td>Indicates how many address space</td><td>RO</td><td>IM</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>non-translated attributes (NTPT)</td><td>(27)</td><td>partitions for non-translated attributes.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>2 partitions from VA(31)</td></tr><tr><td>1</td><td>4 partitions from VA(31,30)</td></tr></table></td><td></td><td></td></tr>
<tr><td>Invisible TLB (IVTB)</td><td>1
(28)</td><td>Indicates if a software-invisible TLB that includes additional PTEs from software-visible TLB exists.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No such invisible TLB exists.</td></tr><tr><td>1</td><td>Such invisible TLB exists.</td></tr></table></td><td>RO</td><td>IM
(WT=0)</td></tr>
<tr><td>VLPT</td><td>1
(29)</td><td>Indicates if the feature of VLPT for fast TLB fill handling is implemented or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The VLPT feature is not implemented.</td></tr><tr><td>1</td><td>The VLPT feature is implemented.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>NTME</td><td>1
(30)</td><td>Indicates if the non-translated VA to PA mapping function is implemented or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The non-translated VA to PA mapping function is not implemented.</td></tr><tr><td>1</td><td>The non-translated VA to PA mapping function is implemented.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>DRDE</td><td>1</td><td>Device register default endian. This</td><td>RO</td><td>IM</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
	(31)	bit will be copied to PSW.DRBE on reset and on interruption stack level transition 0->1 and 1->2. Value: 0 — Meaning: Default is Little Endian Value: 1 — Meaning: Default is Big Endian		

9.2.5. Misc Configuration Register

Mnemonic Name: cr4 (MSC_CFG)
IM Requirement: Required
Access Mode: Superuser
SR Value {Major, Minor, Extension}: {0, 4, 0}

9	8	7	6	5	4	3	2	1	0
L2C	AUDIO	MAC	DIV	TRACE	HSMP	PFM	LMDMA	EDM	

31	17	16	15	13	12	11	10
Reserved		NOD	BASEV		INTLC	ADR24	RDREG (GPR16)

Field Name	Bits	Description	Type	Reset
EDM	1 (0)	Indicates if an EDM exists or not. Value: Meaning 0: EDM is not implemented. 1: EDM is implemented.	RO	IM
LMDMA	1 (1)	Indicates if a local memory DMA engine exists or not. Value: Meaning 0: DMA is not implemented. 1: DMA is implemented.	RO	IM
PFM	1 (2)	Indicates if the performance monitoring mechanism (counters) exists or not. Value: Meaning 0: Performance monitoring mechanism is not implemented. 1: Performance monitoring mechanism is implemented.	RO	IM
HSMP	1	Indicates if a High Speed Memory Port	RO	IM

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td>(3)</td><td>exists or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>An High Speed Memory Port is not implemented.</td></tr><tr><td>1</td><td>An High Speed Memory port is implemented.</td></tr></table></td><td></td><td></td></tr>
<tr><td>TRACE</td><td>1
(4)</td><td>Indicates if a Debug Tracer Unit exists or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>A Debug Tracer Unit is not implemented.</td></tr><tr><td>1</td><td>A Debug Tracer Unit is implemented.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>DIV</td><td>1
(5)</td><td>Indicates if DIV/DIVS instructions are supported in the core or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>DIV/DIVS instructions are not implemented.</td></tr><tr><td>1</td><td>DIV/DIVS instructions are implemented.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>MAC</td><td>1
(6)</td><td>Indicates if Multiply/Multiply-Add instructions are supported in the core or not. The instructions affected are listed in the following table.<table><tr><td>MADD32</td></tr><tr><td>MADD64</td></tr><tr><td>MADDS64</td></tr><tr><td>MSUB32</td></tr><tr><td>MSUB64</td></tr><tr><td>MSUBS64</td></tr><tr><td>MUL</td></tr><tr><td>MULT32</td></tr><tr><td>MULT64</td></tr><tr><td>MULTS64</td></tr></table></td><td>RO</td><td>IM</td></tr>
</table>

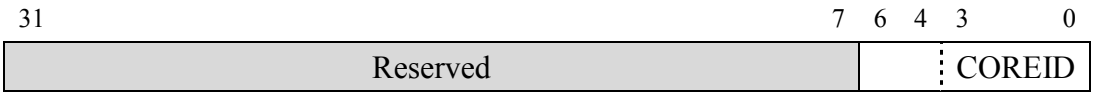
<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Multiply/Multiply-Add related instructions are not implemented.</td></tr><tr><td>1</td><td>Multiply/Multiply-Add related instructions are implemented.</td></tr></table></td><td></td><td></td></tr>
<tr><td>AUDIO</td><td>2
(8,7)</td><td>Indicates if AUDIO ISA extension support exists or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Audio extension is not supported</td></tr><tr><td>1</td><td>32-bit Audio extension is supported.</td></tr><tr><td>2</td><td>24-bit Audio extension is supported.</td></tr><tr><td>3</td><td>Both 24-bit and 32-bit Audio extensions are supported.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>L2C</td><td>1
(9)</td><td>Indicates if a second-level I/D unified cache exists or not. This configuration affects the behaviors of the CCTL 1level and CCTL alevel operations.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>L2 unified cache is not implemented.</td></tr><tr><td>1</td><td>L2 unified cache is implemented.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>RDREG or (GPR16)</td><td>1
(10)</td><td>Indicates if Reduced Register configuration option is selected or not. If it is selected, then the number of general purpose registers in this implementation is reduced to 16.<table><tr><th>Value</th><th>Meaning</th></tr></table></td><td>RO</td><td>IM</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table><tr><td>0</td><td>Reduced Register configuration is not implemented.</td></tr><tr><td>1</td><td>Reduced Register configuration is implemented.</td></tr></table></td><td></td><td></td></tr>
<tr><td>ADR24</td><td>1
(11)</td><td>Indicates if a Reduced Address Space to 24-bit configuration is implemented or not. If it is implemented, the VA and PA address space are all reduced from 32 bits to 24 bits.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Reduced Address Space to 24-bit configuration is not implemented.</td></tr><tr><td>1</td><td>Reduced Address Space to 24-bit configuration is implemented.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>INTLC</td><td>1
(12)</td><td>Indicates interruption level configuration.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Interruption level 1, 2, and 3 are implemented. Level 3 is the overflow level.</td></tr><tr><td>1</td><td>Interruption level 1 and 2 are implemented. Level 2 is the overflow level.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>BASEV</td><td>3
(15,13)</td><td>Indicates the version of Baseline instructions.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Baseline version 1</td></tr><tr><td>1</td><td>Baseline version 2 (include version 1 + new instructions for version 2)</td></tr><tr><td>2-7</td><td>Reserved</td></tr></table></td><td>RO</td><td>IM</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>NOD</td><td>1
(16)</td><td>Indicates if the Dx registers exist or not. If the Dx registers do not exist, the instructions using Dx registers should generate Reserved Instruction exception. It is meaningful after and including baseline version 2.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Dx registers and the instructions involving Dx registers exist.</td></tr>
<tr><td>1</td><td>Dx registers do not exist and the instructions involving Dx registers do not exist. When MISC.AUDIO is not 0 (i.e. Audio ISA extension is supported), this bit can never be set to 1.</td></tr>
</table>
The instructions involved are:
<table>
<tr><td>DIVS/DIV</td></tr>
<tr><td>MULT32/MULTS64/MULT64</td></tr>
<tr><td>MADD32/MADDS64/MADD64</td></tr>
<tr><td>MSUB32/MSUBS64/MSUB64</td></tr>
<tr><td>MFUSR Dx / MTUSR Dx</td></tr>
</table></td><td>RO</td><td>IM</td></tr>
<tr><td>Reserved</td><td>19
(31,13)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

9.2.6. Core Identification Register

Mnemonic Name: cr5 (CORE_ID)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {0, 0, 1}



This register does not exist for the following AndesCore™ implementations:

- N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)

Field name	Bits	Description	Type	Reset
COREID	4 ~ 7 (6~3,0)	Core Identification Number. This is used to distinguish different processor cores in a multi-core system. An implementation should find ways to hardwire this field to different values for each core in a multi-core system. The minimum number of bits for this field is 4 bits while the maximum number of bits for this field can be up to 7 bits.	RO	IM
Reserved	25 (31,7)	RAZWI	-	0

9.2.7. FPU and Coprocessor Existence Configuration Register

Mnemonic Name: cr6 (FUCOP_EXIST)
IM Requirement: COP/FPU extension
Access Mode: Superuser
SR Value {Major, Minor, Extension}: {0, 5, 0}

This system register indicates the existence status of the coprocessors and the floating-point unit. Note that the floating-point unit existence is a combination of "CP0EX" and "CP0ISFPU" (i.e. CR6.CP0EX is 1 and CR6.CP0ISFPU is 1).

31	30 4	3	2	1	0
CP0ISFPU		CP3EX	CP2EX	CP1EX	CP0EX

Field Name	Bits	Description	Type	Reset
CP0EX	1 (0)	Coprocessor #0 existence status bit. Value: 0 — Coprocessor #0 does not exist. Any encountering of Coprocessor #0 instruction will cause "Reserved instruction" exception. Value: 1 — Coprocessor #0 exists.	RO	IM
CP1EX	1 (1)	Coprocessor #1 existence status bit. Value: 0 — Coprocessor #1 does not exist. Any encountering of coprocessor #1 instruction will cause "Reserved instruction" exception. Value: 1 — Coprocessor #1 exists.	RO	IM

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>CP2EX</td><td>1
(2)</td><td>Coprocessor #2 existence status bit.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Coprocessor #2 does not exist. Any encountering of coprocessor #2 instruction will cause “Reserved instruction” exception.</td></tr><tr><td>1</td><td>Coprocessor #2 exists.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>CP3EX</td><td>1
(3)</td><td>Coprocessor #3 existence status bit.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Coprocessor #3 does not exist. Any encountering of coprocessor #3 instruction will cause “Reserved instruction” exception.</td></tr><tr><td>1</td><td>Coprocessor #3 exists.</td></tr></table></td><td>RO</td><td>IM</td></tr>
<tr><td>Reserved</td><td>27
(30,4)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
<tr><td>CP0ISFPU</td><td>1
(31)</td><td>Indicates if Coprocessor #0 is FPU or not when CP0EX is 1.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Coprocessor #0 is not FPU when CP0EX is 1.</td></tr><tr><td>1</td><td>Coprocessor #0 is FPU when CP0EX is 1.</td></tr></table></td><td>RO</td><td>IM</td></tr>
</table>

9.3. Interruption System Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
ir0	PSW	1	0	0	110
ir1	IPSW	1	0	1	116
ir2	P_IPSW	1	0	2	117
ir3	IVB	1	1	1	118
ir4	EVA	1	2	1	120
ir5	P_EVA	1	2	2	123
ir6	ITYPE	1	3	1	124
ir7	P_ITYPE	1	3	2	127
ir8	MERR	1	4	1	131
ir9	IPC	1	5	1	132
ir10	P_IPC	1	5	2	133
ir11	OIPC	1	5	3	134
ir12	P_P0	1	6	2	135
ir13	P_P1	1	7	2	135
ir14	INT_MASK	1	8	0	136
ir15	INT_PEND	1	9	0	139

9.3.1. Processor Status Word Register

Mnemonic Name: ir0 (PSW)
IM Requirement: Required
Access Mode: Superuser (few bits for user)
SR Encoding {Major, Minor, Extension}: {1, 0, 0}

The Processor Status Word register contains several fields which controls the global masking behavior for interrupts, the MMU translation modes, the current processor operation mode, data endianness control, and indicates the current interruption stack level, and machine error status.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN	DRBE	HSS	DEX	DME	IME	DT	IT	BE	POM		INTL		GIE

31	15	14
Reserved		WBNA

Field Name	Bits	Description	Type	Reset
GIE (Global Interrupt Enable)	1 (0)	Controls if Interrupt is enabled or not. Note that the Non-Maskable Interrupt (NMI) cannot be disabled by this bit. Value: 0 — Interrupt is disabled. Value: 1 — Interrupt is enabled.	RW	0
INTL (Interruption Stack Level)	2 (2,1)	Controls interruption behaviors. When interruption happens in interruption stack level 2, all the interruption resources are not updated except that the interruption stack level is being updated to 3. Value: 0 — No interruption	RW	1

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table><tr><td>1</td><td>Interruption stack level 1</td></tr><tr><td>2</td><td>Interruption stack level 2</td></tr><tr><td>3</td><td>Interruption stack level 3</td></tr></table></td><td></td><td></td></tr>
<tr><td>POM</td><td>2
(4,3)</td><td>Processor Operation Mode.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>User</td></tr><tr><td>1</td><td>Superuser</td></tr><tr><td>2</td><td>Reserved</td></tr><tr><td>3</td><td>Reserved</td></tr></table></td><td>RW</td><td>1</td></tr>
<tr><td>BE</td><td>1
(5)</td><td>Endian mode for data memory access. This bit is user modifiable using a SETEND instruction.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Little Endian</td></tr><tr><td>1</td><td>Big Endian</td></tr></table></td><td>RW</td><td>cr3.DE (MMU Config)</td></tr>
<tr><td>IT</td><td>1
(6)</td><td>Controls instruction address translation. On interruption stack level transition from 0 to 1 and from 1 to 2, it gets changed to 0.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Instruction address translation is disabled.</td></tr><tr><td>1</td><td>Instruction address translation is enabled.</td></tr></table>Software Note: When changing the state of this bit using a MTSR instruction, the instruction pages which contain any instruction following the MTSR instruction until the ISB instruction need to be identity-mapped (VPN == PPN) in the page table.</td><td>RW</td><td>0</td></tr>
<tr><td>DT</td><td>1
(7)</td><td>Controls data address translation. On interruption stack level transition from 0 to 1 and from 1 to 2, it gets changed</td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>to 0.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Data address translation is disabled.</td></tr><tr><td>1</td><td>Data address translation is enabled.</td></tr></table>Software Note: When changing the state of this bit using a MTSR instruction, the data pages which are referenced by any load/store instruction following the MTSR instruction until the ISB instruction need to be identity-mapped (VPN == PPN) in the page table.</td><td></td><td></td></tr>
<tr><td>IME</td><td>1
(8)</td><td>Instruction Machine Error flag.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No Instruction Machine Error exception.</td></tr><tr><td>1</td><td>An Instruction Machine Error has happened. When it is set, all instruction memory accesses are to un-cached space. Note that CCTL instruction can still access cache in this case.</td></tr></table>It will be set by the following Instruction Machine Error exceptions:
<table><tr><td>Instruction cache locking error</td></tr><tr><td>Instruction cache parity/ECC error</td></tr><tr><td>Instruction TLB locking error</td></tr><tr><td>Instruction TLB multiple hit</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>DME</td><td>1
(9)</td><td>Data Machine Error flag.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No Data Machine Error</td></tr></table></td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>exception.</td></tr>
<tr><td>1</td><td>A Data Machine Error has happened. When it is set, all data memory accesses are to un-cached space. Note that CCTL instruction can still access cache in this case.</td></tr>
</table>
It will be set by the following Data Machine Error exceptions:
<table>
<tr><td>Data cache locking error</td></tr>
<tr><td>Data cache parity/ECC error</td></tr>
<tr><td>Data TLB locking error</td></tr>
<tr><td>Data TLB multiple hit</td></tr>
</table>
</td><td></td><td></td></tr>
<tr><td>DEX</td><td>1
(10)</td><td>Debug Exception. This bit will be set when the processor enters the debug exception entry point.</td><td>RW</td><td>0</td></tr>
<tr><td>HSS</td><td>1
(11)</td><td>Hardware Single Stepping. If hardware single stepping is enabled, the processor will enter debug exception after an instruction is complete.
If this bit is set, it will be changed to the value of DSSIM of the Interrupt Masking register on interruption entrance.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Hardware single stepping is OFF. On interruption entrance, this bit remains at 0.</td></tr>
<tr><td>1</td><td>Hardware single stepping is ON. On interruption entrance, this bit is set to the value of DSSIM of the Interrupt Masking register.</td></tr>
</table>
</td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>DRBE</td><td>1
(12)</td><td>Device Register Endian mode for device register space access when MMU_CTL.DREE is asserted. This Device Register Endian mode allows uncacheable/un-coalesable device registers to have a different endian type than that of the other attribute space.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Little Endian</td></tr><tr><td>1</td><td>Big Endian</td></tr></table></td><td>RW</td><td>cr3.DRDE</td></tr>
<tr><td>AEN</td><td>1
(13)</td><td>Audio ISA special features enable control. This bit exists only if audio ISA extension is supported. The audio special feature includes: zero-overhead loop mechanism and BLW24. If audio ISA extension is supported, system software needs to turn on this bit in order for user programs to use these features. This bit will be set to 0 automatically on interruption entry to a non-highest level. When interruption entry to the highest level, this bit will not be updated, but will have the same AEN == 0 behavior.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Audio special features are disabled.</td></tr><tr><td>1</td><td>Audio special features are enabled.</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>WBNA</td><td>1
(14)</td><td>Changed write-back, write-allocation memory to write-back, no-write-allocation memory.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Write-back memory with</td></tr></table></td><td>RW</td><td>0</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
		write-allocation policy. 1 Write-back memory with no-write-allocation policy.		
Reserved	17 (31,15)		RAZWI	0

9.3.2. Interruption PSW Register

Mnemonic Name: ir1 (IPSW)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 0, 1}

This register is the shadow stack register of the PSW register. It is updated during interruption stack level transition 0 → 1 and 1 ↔ 2 by hardware. And its format is the same as PSW.

It is a RW type register. Its reset value is defined as DC (Don't Care). And writing a reserved value into the corresponding POM field causes a "Reserved Value" exception.

9.3.3. Previous IPSW Register

Mnemonic Name: ir2 (P_IPSW)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 0, 2}

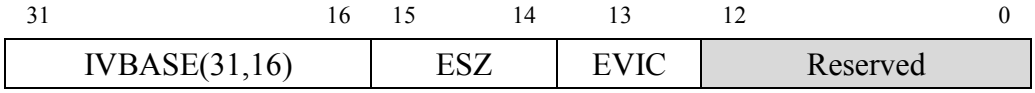
This register is the shadow stack register of the IPSW register. It is updated during interruption stack level transition 1 → 2 by hardware. And its format is the same as PSW.

It is a RW type register. Its reset value is defined as DC (Don't Care). And writing a reserved value into the corresponding POM field causes a "Reserved Value" exception.

9.3.4. Interruption Vector Base Register

Mnemonic Name: ir3 (IVB)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 1, 1}

The IVB register is the base physical address of the interruption vector table which anchors the interruption entry points for the software interruption handlers. Its format is as follows:



Field Name	Bits	Description	Type	Reset
Reserved	13 (12,0)		RAZWI	0
EVIC	1 (13)	Configures the proper Vector Interrupt Controller mode used in the AndesCore™ Value: 0 — Internal Vector Interrupt Controller mode. Value: 1 — External Vector Interrupt Controller mode.	RW	IM
ESZ	2 (15,14)	Defines the size of each vector entry. Value: 0 — 4 Byte Value: 1 — 16 Byte Value: 2 — 64 Byte Value: 3 — 256 Byte	RW	1
IVBASE	16	Defines the base physical address of the	RW	IM

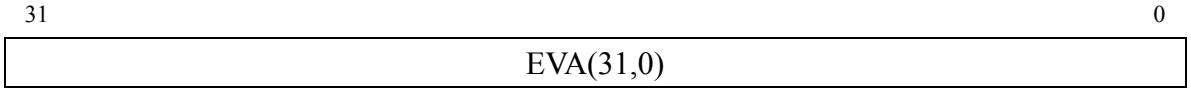
Field Name	Bits	Description	Type	Reset
	(31,16)	interruption vector table. The table has to be 64KB aligned.		

9.3.5. Exception Virtual Address Register

Mnemonic Name: ir4 (EVA)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 2, 1}

The EVA register will be updated with the exception virtual address information needed by the interruption handler when certain exception happens during interruption stack level transition 0 → 1 and level 1 → 2 by hardware. It will also be updated from the P_EVA register when executing a "return from interruption" (IRET) instruction within interruption stack level 2. It is used by the exception handler routine to fix the exception problem, or to debug and log the problem if the exception can not be solved immediately by the handler.

Its register format is as follows:



Field Name	Bits	Description	Type	Reset
EVA	32 (31,0)	The virtual address which causes the exception. For instruction fetch related exceptions, this register records part of the same virtual address captured in the IPC register. For data access related exceptions, this register records the data access virtual address which is different from the virtual address in the IPC register. Detailed exceptions and their EVA values are listed in Table 17. For other exceptions not listed in the table, the EVA update behavior and its update value will be	RO	DC

Field Name	Bits	Description	Type	Reset
		implementation-dependent.		

The EVA register will be updated by hardware for the following exceptions that is related to memory access or memory access hardware structures during interruption stack level transition 0 → 1 and 1 → 2:

Table 17. Exceptions and EVA values

Exceptions updating EVA	EVA value
ITLB fill	4KB-aligned instruction fetch virtual address (i.e. lower 12 bits 0)
ITLB VLPT miss	4KB-aligned instruction fetch virtual address (i.e. lower 12 bits 0)
ITLB misc (all)	4KB-aligned instruction fetch virtual address (i.e. lower 12 bits 0)
I access PTE not present (all)	4KB-aligned instruction fetch virtual address (i.e. lower 12 bits 0)
DTLB fill	full 32-bit data access virtual address
DTLB VLPT miss	full 32-bit data access virtual address
DTLB misc (all)	full 32-bit data access virtual address
D access PTE not present (all)	full 32-bit data access virtual address
alignment check	full 32-bit virtual address of the instruction which generates the misaligned address
precise bus error	full 32-bit data access virtual address
TLB locking error generated by a TLBOP instruction (in machine error entry point)	4KB-aligned virtual page number (i.e. lower 12-bit is 0)
machine error (fetch/load/store related, thus excluding unimplemented page size error)	full 32-bit instruction fetch or data access virtual address
precise/next precise data address watchpoint	full 32-bit watched VA address (regardless of VA or PA address match)
precise/next precise data value watchpoint	full 32-bit watched VA address (regardless of VA or PA address match)
illegal parallel memory accesses	full 32-bit data access virtual address of the first

Exceptions updating EVA	EVA value
	address operand.

9.3.6. Previous EVA Register

Mnemonic Name: ir5 (P_EVA)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 2, 2}

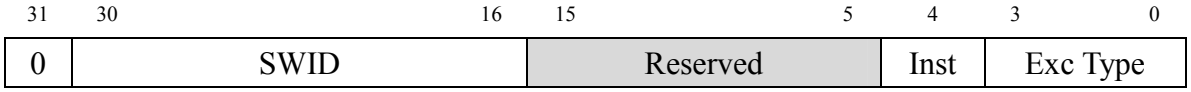
This register is the shadow stack register of the EVA register. It is updated during interruption stack level transition 1 → 2 by hardware. And its format is the same as EVA register.

It is a RW type register. Its reset value is defined as DC (Don't Care).

9.3.7. Interruption Type Register

Mnemonic Name: ir6 (ITYPE)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 3, 1}

The ITYPE register will be updated with the interruption type information needed by the interruption handler when an interruption happens within interruption stack level 0 and level 1. It will also be updated from the P_ITYPE register when executing a "return from interruption" (IRET) instruction within interruption stack level 1 and level 2. Its register format is as follows:



Field Name	Bits	Description	Type	Reset
ETYPE	4 (3,0)	Exception type information. Its definition is based on the exception entry point. Detailed definition will be listed in the following tables. **Entry Point — Definition** Reset/NMI — Table 18 PTE not present — Table 19 TLB misc — Table 20 Machine error — Table 21 Debug — Table 22 General exception — Table 23 Syscall — Table 24	RO	IM
INST	1 (4)	Indicates if an exception is caused by an instruction fetch or a data memory access for the following exceptions:	RO	IM

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>TLB fill</td></tr>
<tr><td>TLB VLPT miss</td></tr>
<tr><td>TLB read protection</td></tr>
<tr><td>TLB write protection</td></tr>
<tr><td>TLB non-executable page</td></tr>
<tr><td>TLB page modified</td></tr>
<tr><td>TLB Access bit</td></tr>
<tr><td>PTE not present (all)</td></tr>
<tr><td>Reserved PTE Attribute</td></tr>
<tr><td>Alignment check</td></tr>
<tr><td>Machine error</td></tr>
<tr><td>Precise bus error</td></tr>
<tr><td>Imprecise bus error</td></tr>
<tr><td>Nonexistent local memory address</td></tr>
<tr><td>MPZIU Control</td></tr>
<tr><td>Cache locking error</td></tr>
<tr><td>TLB locking error</td></tr>
<tr><td>TLB multiple hit</td></tr>
<tr><td>Cache parity/ECC error</td></tr>
</table>
All other exceptions not in the above table should have the INST field of the ITYPE register set to 0.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Exception is caused by a data memory access.</td></tr>
<tr><td>1</td><td>Exception is caused by an instruction fetch access.</td></tr>
</table>
</td><td></td><td></td></tr>
<tr><td>Reserved</td><td>11
(15,5)</td><td></td><td>RAZWI</td><td>0</td></tr>
<tr><td>SWID</td><td>15
(30,16)</td><td>The SWID of BREAK, BREAK16, SYSCALL, and TRAP instruction which caused the corresponding exception.</td><td>RO</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>Zero</td><td>RO</td><td>0</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
	(31)			

For the arithmetic exception in the general exception vector entry point, the following fields are defined as follows:

31	30 … 22	21	20 … 19	16	15 … 5	4	3 … 0
0	Reserved	CPID	Sub_Type		Reserved	Inst	Exc Type

Field Name	Bits	Description	Type	Reset
SUB_TYPE	4 (19,16)	1. Indicates arithmetic exception type when an arithmetic exception has happened. The encoding is as follows: Value / Meaning 0 / Reserved 1 / INT Divide by Zero (for DIV, DIVS) 2 / Integer Overflow (for DIVS) 3-15 / - 2. Indicates coprocessor exception type when a coprocessor exception has happened. The encoding is as follows: Value / Meaning 0 / Reserved 1 / Coprocessor disabled exception 2 / Coprocessor exception. Please check coprocessor exception status for details. 3 / Audio extension disabled exception.	RO	IM

Field Name	Bits	Description	Type	Reset
		4 — Audio extension zero-overhead loop error. 5-15 — -		
CPID	2 (21,20)	Indicates the ID number of the coprocessor which generates the exception.	RO	IM

Detailed Definitions for Exception Type (ETYPE)

For Reset/NMI exception entry point, the ETYPE is defined as follows:

Table 18. Reset/NMI Exception ETYPE definition

Encoding	Exception (Qualified with Inst field)
0	Cold reset
1	Warm reset
2	NMI
3-15	-

For PTE not present exception entry point, the ETYPE is defined as follows:

Table 19. PTE Not Present Exception ETYPE definition

Encoding	Exception (Qualified with Inst field)
0	Non-leaf PTE not present (+I/D bit)
1	Leaf PTE not present (+I/D bit)
2-15	-

For TLB misc exception entry point, the ETYPE is defined as follows:

Table 20. TLB MISC Exception ETYPE definition

Value	Exception (Qualified with Inst field)
0	TLB read protection (data)
1	TLB write protection (data)

Value	Exception (Qualified with Inst field)
2	TLB non-executable page (Inst)
3	TLB page modified (data)
4	TLB Access bit (+I/D bit)
5	Reserved PTE Attribute (+I/D bit)
6-15	-

For Machine Error exception entry point, the ETYPE is defined as follows:

Table 21. Machine Error Exception ETYPE definition

Encoding	Exception (Qualified with Inst field)
0	Cache locking error (+I/D bit)
1	TLB locking error (+I/D bit)
2	TLB multiple hit (+I/D bit)
3	Cache parity/ECC error (+I/D bit)
4	Unimplemented page size error
5	Illegal parallel memory accesses
6-15	-

For Debug exception entry point, the ETYPE is defined as follows:

Table 22. Debug Exception ETYPE definition

Encoding	Exception
0	BREAK
1	BREAK16
2	Instruction breakpoint
3	Data address watchpoint (precise)
4	Data value watchpoint (precise)
5	Data value watchpoint (imprecise)
6	Debug interrupt
7	Hardware single stepping
8	Data address watchpoint (next precise)
9	Data value watchpoint (next precise)

10-15	-

For General Exception entry point, the ETYPE is defined as follows:

Table 23. General Exception ETYPE definition

Value	Exception (Qualified with Inst field)
0	Alignment check (+I/D bit)
1	Reserved instruction
2	Trap
3	Arithmetic
4	Precise bus error (+I/D bit)
5	Imprecise bus error (+I/D bit)
6	Coprocessor
7	Privileged instruction
8	Reserved value
9	Nonexistent local memory address (+I/D bit)
10	MPZIU Control (+I/D bit)
11-15	-

For Syscall and Interrupt entry point, the ETYPE is defined as follows:

Table 24. Syscall Exception, Interrupt ETYPE definition

Value	Exception (Qualified with Inst field)
0-15	Don’t care

9.3.8. Previous ITYPE Register

Mnemonic Name: ir7 (P_ITYPE)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 3, 2}

This register is the shadow stack register of the ITYPE register. It is updated during interruption stack level transition 1 → 2 by hardware. And its format is the same as ITYPE register.

It is a RW type register. Its reset value is defined as DC (Don't Care).

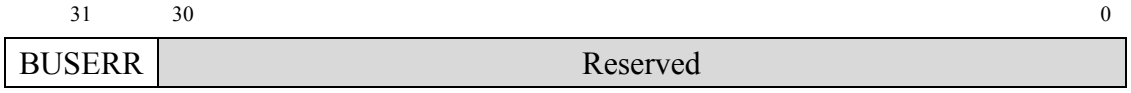
9.3.9. Machine Error Log Register

Mnemonic Name: ir8 (MERR)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 4, 1}

The MERR register will be updated with the Machine error information needed by the Machine error exception handler when a machine error exception happens at any interruption stack level. Its register format is implementation-dependent.

No shadow stack registers is specified for this register since we do not expect nested Machine Error exceptions to happen.

The format and encoding of this register for AndesCore™ N12 is as follows:

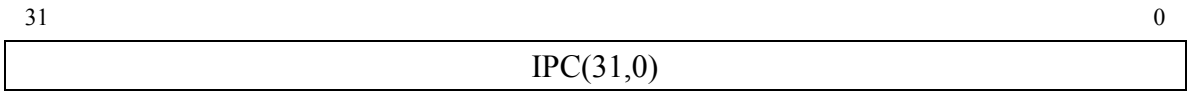


Field Name	Bits	Description	Type	Reset
Reserved	31 (30,0)	RAZWI	-	0
BUSERR	1 (31)	Indicates that a bus error caused by a load instruction has happened.	W1C	0

9.3.10. Interruption Program Counter Register

Mnemonic Name: ir9 (IPC)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 5, 1}

This register is the shadow stack register of the Program Counter. It is updated during interruption stack level transition 0 → 1 and 1 ↔ 2 by hardware. And its format is the same as the Program Counter as follows:

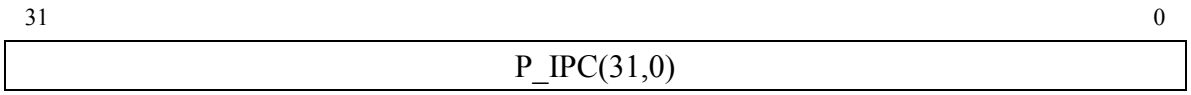


It is a RW type register. Its reset value is defined as DC (Don't Care).

9.3.11. Previous IPC Register

Mnemonic Name: ir10 (P_IPC)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 5, 2}

This register is the shadow stack register of the IPC register. It is updated during interruption stack level transition 1 → 2 by hardware. And its format is the same as IPC as follows:



It is a RW type register. Its reset value is defined as DC (Don't Care).

9.3.12. Overflow Interruption Program Counter Register

Mnemonic Name: ir11 (OIPC)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 5, 3}

This register is the special shadow stack register of the Program Counter. It is updated during interruption stack level transition 2 → 3 by hardware. And its format is the same as the Program Counter.

It is a RW type register. Its reset value is defined as DC (Don't Care).

9.3.13. Previous P0 Register

Mnemonic Name: ir12 (P_P0)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 6, 2}

This register is the shadow stack register of the P0 register. It is updated during interruption stack level transition 1 → 2 by hardware. And its format is the same as a general register.

It is a RW type register. Its reset value is defined as DC (Don't Care).

9.3.14. Previous P1 Register

Mnemonic Name: ir13 (P_P1)
IM Requirement: Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 7, 2}

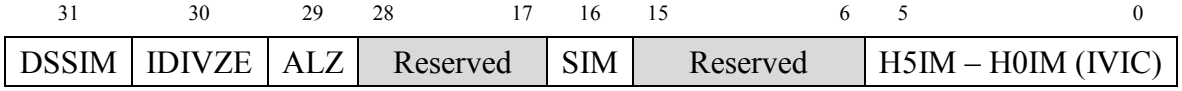
This register is the shadow stack register of the P1 register. It is updated during interruption stack level transition 1 → 2 by hardware. And its format is the same as a general register.

It is a RW type register. Its reset value is defined as DC (Don't Care).

9.3.15. Interruption Masking Register

Mnemonic Name: ir14 (INT_MASK)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 8, 0}

The Interruption Masking Register is used to mask the one software interrupt and the six hardware interrupts in the Internal Vector Interrupt Controller mode. It also controls if an exception is generated on Divide-by-Zero condition for integer divide instructions.



Field Name	Bits	Description	Type	Reset
H5IM – H0IM	6 (5,0)	Hardware Interrupt X Mask bits. These bits are used in the Internal Vector Interrupt Controller mode. The hardware interrupt to bit assignment is as follows: (H5,H4,H3,H2,H1,H0) == (5,4,3,2,1,0). Each bit is defined below. Value: 0 – The corresponding hardware interrupt is disabled Value: 1 – The corresponding hardware interrupt is enabled	RW	0 for all bits
Reserved	10 (15,6)		RAZWI	0
SIM	1 (16)	Software Interrupt Mask bit. This bit is used in the Internal Vector Interrupt Controller mode. Value: 0 – Software interrupt is disabled	RW	0

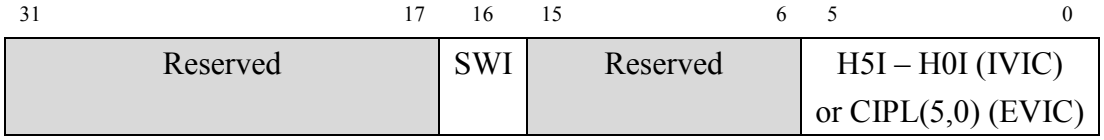
Field Name	Bits	Description	Type	Reset
		1 — Software interrupt is enabled		
Reserved	12 (28,17)		RAZWI	0
ALZ	1 (29)	All zero opcode (LBI R0,[R0+0]) reserved instruction exception masking control. This is a baseline version 2 instruction set special behavior and is mainly used for debugging purpose. 0 — All zero opcode reserved instruction exception is disabled. 1 — All zero opcode reserved instruction exception is enabled.	RW	0
IDIVZE	1 (30)	Controls if Arithmetic exception is enabled or not for "Divide-By-Zero" condition. 0 — Arithmetic exception is disabled. 1 — Arithmetic exception is enabled.	RW	1
DSSIM	1 (31)	Default Single Stepping Interruption Mask. This bit determines the value of PSW.HSS on interruption entrance when PSW.HSS is set (i.e. hardware single-stepping is on). 0 — If (PSW.HSS == 1) then PSW.HSS ← 0 (OFF) on interruption entrance. 1 — If (PSW.HSS == 1) then PSW.HSS ← 1 (ON) on	RW	1

Field Name	Bits	Description	Type	Reset
		interruption entrance.		

9.3.16. Interrupt Pending Register

Mnemonic Name: ir15 (INT_PEND)
IM Requirement: Required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {1, 9, 0}

The Interrupt Pending register is used to generate the software interrupt or record the status of the hardware interrupt. Software generates a software interrupt by writing to the SWI field of this register. The now pended software interrupt will be fed into the processor core's internal interrupt controller logic or exported through a pin to an external interrupt controller.



Field Name	Bits	Description	Type	Reset
H5I – H0I	6 (5,0)	Hardware Interrupt X pending bits. These bits are used in the Internal Vector Interrupt Controller mode. The hardware interrupt to bit assignment is as follows: (H5,H4,H3,H2,H1,H0) == (5,4,3,2,1,0). Each bit is defined below. Value: 0 — The corresponding hardware interrupt is not pending. Value: 1 — The corresponding hardware interrupt is pending.	RO	0
CIPL	6 (5,0)	The CIPL field is used in External Vectored Interrupt Controller (EVIC) mode. It contains the Current Interrupt	RO	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>Priority Level requested by the external interrupt controller.</td><td></td><td></td></tr>
<tr><td>Reserved</td><td>10
(15,6)</td><td></td><td>RAZWI</td><td>0</td></tr>
<tr><td>SWI</td><td>1
(16)</td><td>Indicates a pending software interrupt. This bit is used in Internal Vector Interrupt Controller mode.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>No software interrupt pending.</td></tr>
<tr><td>1</td><td>A software interrupt is pending.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>Reserved</td><td>25
(31,7)</td><td></td><td>RAZWI</td><td>0</td></tr>
</table>

9.4. MMU System Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
mr0	MMU_CTL	2	0	0	142
mr1	L1_PPTB	2	1	0	146
mr2	TLB_VPN	2	2	0	147
mr3	TLB_DATA	2	3	0	149
mr4	TLB_MISC	2	4	0	154
mr5	VLPT_IDX	2	5	0	156
mr6	ILMB	2	6	0	158
mr7	DLMB	2	7	0	160
mr8	CACHE_CTL	2	8	0	160
mr9	HSMP_SADDR	2	9	0	165
mr10	HSMP_EADDR	2	9	1	168

9.4.1. MMU Control Register

Mnemonic Name: mr0 (MMU_CTL)
IM Requirement: required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 0, 0}

10	9	8	7	6	5	4	3	2	1	0
MPZIU	TBALCK	NTC3		NTC2		NTC1		NTC0		D

31	20	19	18	17	16	15	14	13	12	11
Reserved		DREE	NTM3		NTM2		NTM1		NTM0	

When MMU_CFG..MMPS is 0 or 1, the D, TBALCK, MPZIU fields may become Reserved or not. It is implementation-dependent.

When MMU_CFG..NTME is 0, the NTM0-NTM3 fields should become Reserved fields.

Field Name	Bits	Description	Type	Reset
Default minimum page size (D)	1 (0)	Indicates the default minimum page size: Value: 0 = 4KB; 1 = 8KB	RW	0
NTC0	2 (2,1)	Indicates Non-Translated Cachebility memory attribute for partition 0. Value: 0 = Non-cacheable/Non-coalesable; 1 = Non-cacheable/Coalesable; 2 = Cacheable/Write-Back; 3 = Cacheable/Write-Through	RW	0
NTC1	2 (4,3)	Indicates Non-Translated Cachebility memory attribute for partition 1. See NTC0 description for its value definition.	RW	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>NTC2</td><td>2
(6,5)</td><td>Indicates Non-Translated Cachebility memory attribute for partition 2. See NTC0 description for its value definition.</td><td>RW</td><td>0</td></tr>
<tr><td>NTC3</td><td>2
(8,7)</td><td>Indicates Non-Translated Cachebility memory attribute for partition 3. See NTC0 description for its value definition.</td><td>RW</td><td>0</td></tr>
<tr><td>TBALCK</td><td>1
(9)</td><td>Controls TLB all-lock resolution scheme:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Generating “Machine Error” exception.</td></tr>
<tr><td>1</td><td>Hardware random/LRU replacement.</td></tr>
</table>
Operation exception behavior affected by this bit is listed as follows:
<table>
<tr><td>HPTWK TLB fill operation</td></tr>
<tr><td>TLB Random Write</td></tr>
<tr><td>TLB Random Write and Lock</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>MPZIU</td><td>1
(10)</td><td>Multiple Page Size In Use bit. This bit is for OS to indicate to hardware that it is currently using multiple page sizes at the same time. If this bit is not set properly while OS tries to use multiple page sizes, the TLB search for the PTE of large page size may always miss. If this bit is not set while the hardware page table walker is instructed by a PTE to insert the PTE as a large page size, then a MPZIU Control exception will be generated and the PTE will not be inserted into TLB.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>No multiple page size in use. Hardware TLB search for the PTE of large page size may always fail.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table><tr><td>1</td><td>Multiple page size in use. Hardware TLB will perform TLB search for the PTE of large page size. But this may take more cycles.</td></tr></table></td><td></td><td></td></tr>
<tr><td>NTM0 (MMU_CFG..NTME == 1)</td><td>2 (12,11)</td><td>Indicates Non-Translated VA to PA[31,30] mapping for partition 0.</td><td>RW</td><td>0</td></tr>
<tr><td>NTM1 (MMU_CFG..NTME == 1)</td><td>2 (14,13)</td><td>Indicates Non-Translated VA to PA[31,30] mapping for partition 1.</td><td>RW</td><td>1</td></tr>
<tr><td>NTM2 (MMU_CFG..NTME == 1)</td><td>2 (16,15)</td><td>Indicates Non-Translated VA to PA[31,30] mapping for partition 2.</td><td>RW</td><td>2</td></tr>
<tr><td>NTM3 (MMU_CFG..NTME == 1)</td><td>2 (18,17)</td><td>Indicates Non-Translated VA to PA[31,30] mapping for partition 3.</td><td>RW</td><td>3</td></tr>
<tr><td>DREE</td><td>1 (19)</td><td>Device register endian control enable.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Device register endian control function is disabled. A data memory access uses PSW.BE as the endian type.</td></tr><tr><td>1</td><td>Device register endian control function is enabled. A data memory access with the following attributes, (C==0 or NTC==0), uses PSW.DRBE as the endian type while a data memory access with the other attributes uses PSW.BE as the endian type.</td></tr></table></td><td>RW</td><td>IM</td></tr>
<tr><td>Reserved</td><td>12</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
	(31,20)			

NTC0-NTC3 is the Non-translated Cacheability attributes used when the VA to PA translation process is turned off. The value definition of the NTC field is listed in Table 2 in section 2.2.2. When only VA(31) is used to access the NTC fields, only NTC0/NTC1 are used. When VA(31,30) is used to access the NTC fields, all NTC fields will be used. The NTC field assignment for each case is listed in Table 25 and Table 26.

NTM0-NTM3 is the Non-translated VA(31,30) to PA(31,30) Mapping fields when MMU_CFG.NTME is equal to 1. It is most often used to improve memory access performance on a small system without MMU or MPU by mapping different NTC attributes (e.g. cacheable/WB, cacheable/WT) to the same PA partition. Thus, the memory access type can be controlled using VA(31,30). When only VA(31) is used to access the NTM fields, only NTM0/NTM1 are used. When VA(31,30) is used to access the NTM fields, all NTM fields will be used. The NTM field assignment for each case is listed in Table 25 and Table 26.

Table 25. NTC field assignment for indexing using VA(31)

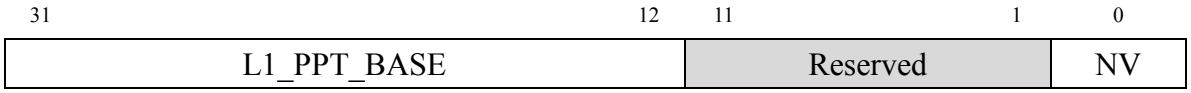
VA(31)	NTC field	NTM field
1`b0	NTC0	NTM0
1`b1	NTC1	NTM1

Table 26. NTC field assignment for indexing using VA(31,30)

VA(31,30)	NTC field	NTM field
2`b00	NTC0	NTM0
2`b01	NTC1	NTM1
2`b10	NTC2	NTM2
2`b11	NTC3	NTM3

9.4.2. L1 Physical Page Table Base Register

Mnemonic Name: mr1 (L1_PPTB)
IM Requirement: MMU hardware page table walker optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 1, 0}

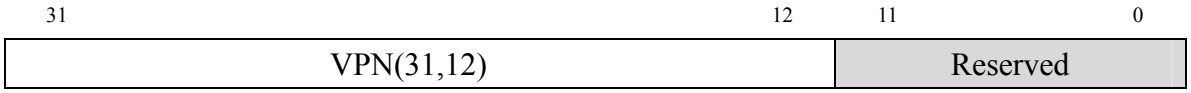


This is a pointer to the current level 1 physical page table. The value of the pointer should be in physical addresses space. This pointer is used by the HPTWK to construct the load address to locate the level 2 physical page table base.

Field Name	Bits	Description	Type	Reset
NV	1 (0)	Used to enable/disable Hardware Page Table Walker (HPTWK). Value: 0 — HPTWK is enabled Value: 1 — HPTWK is disabled	RW	1
Reserved	11 (11,1)	Read As Zero, Write ignored (RAZWI)	RAZWI	0
L1 PPT Base Address (L1_PPT_BASE)	20 (31,12)	Indicates the first level physical page table base address. It is 4KB-aligned.	RW	0

9.4.3. TLB Access VPN Register

Mnemonic Name: mr2 (TLB_VPN)
IM Requirement: MMU optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 2, 0}



For TLB writes (target or random), this is the virtual page number (VPN) of the page translation entry to be installed.
For TLB target reads, this register holds the VPN of the read out.
When the following TLB exception happens, this register will be updated to the exception VPN.

TLB fill
PTE not present
Page modified
TLB permission

So, for example, after a TLB fill exception, this register will contain the correct VPN content for the TLB fill operation needed to be performed in the exception handler.

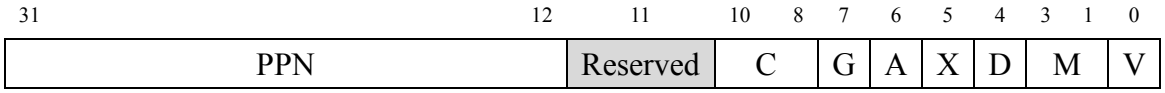
No shadow stack register is specified for the TLB_VPN.VPN field since we assume that nested exceptions specified above cannot happen.

Field Name	Bits	Description	Type	Reset
Reserved	12 (11,0)	Read As Zero, Write Ignored.	RAZWI	0
Virtual Page Number (VPN)	20 (31,12)	Virtual Page Number. It is updated by hardware on the following TLB exceptions: TLB Fill, PTE not present,	RW	DC

Field Name	Bits	Description	Type	Reset
		Page Modified, and TLB permission, when interruption stack level transition 0 → 1 or 1 → 2 happens. For all minimum page configurations, it is updated with VA(31,12).		

9.4.4. TLB Access Data Register

Mnemonic Name: mr3 (TLB_DATA)
IM Requirement: MMU optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 3, 0}



For TLB target write, this is the page translation entry (PTE) to be written into TLB. For TLB reads, this is the page translation entry read out. The page translation entry translates virtual addresses into physical addresses.

Note that for TLB random write, the PTE data is coming from a general register instead of this register.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>V</td><td>1
(0)</td><td>This PTE is valid and present</td><td>RW</td><td>DC</td></tr>
<tr><td>M
(For MMU version 1)</td><td>3
(3,1)</td><td>This field indicates the page read/write access privilege. If a load instruction accesses a page which has no read privilege, a Read Protection Violation exception is generated. If a store instruction accesses a page which has no write privilege, a Write Protection Violation exception is generated.
<table>
<tr><th>Value</th><th>User mode</th><th>Superuser mode</th></tr>
<tr><td>0</td><td>-</td><td>-</td></tr>
<tr><td>1</td><td>Read only</td><td>Read only</td></tr>
<tr><td>2</td><td>Read only</td><td>Read/Write</td></tr>
</table></td><td>RW</td><td>DC</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>3</td><td>Read/write</td><td>Read/write</td></tr>
<tr><td>4</td><td>-</td><td>-</td></tr>
<tr><td>5</td><td>No access</td><td>Read only</td></tr>
<tr><td>6</td><td>-</td><td>-</td></tr>
<tr><td>7</td><td>No access</td><td>Read/Write</td></tr>
</table>
Note that this Access mode only governs the access permission of using load/store instructions to access the page. It does not govern the access permission of an instruction fetching and execution.
Using of Reserved values (0, 4, 6) will generate UNPREDICTABLE result for the following implementations:
● N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
● N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)
But will generate Reserved PTE Attribute exception (Data) for all other implementations.
</td><td></td><td></td></tr>
<tr><td>M
(For MMU version 2)</td><td>3
(3,1)</td><td>Read/Write Access mode + user mode fetching restriction on superuser mode page.
<table>
<tr><th>M[3:1]</th><th>User mode</th><th>Superuser mode</th></tr>
<tr><td>3’b000
X==0</td><td>-</td><td>-</td></tr>
<tr><td>3’b000
X==1</td><td>No Read/Write</td><td>Read/Write</td></tr>
<tr><td>3’b001</td><td>Read only</td><td>Read only</td></tr>
<tr><td>3’b010</td><td>Read only</td><td>Read/Write</td></tr>
<tr><td>3’b011</td><td>Read/Write</td><td>Read/Write</td></tr>
<tr><td>3’b100</td><td>-</td><td>-</td></tr>
<tr><td>3’b101</td><td>No Read/Write/Fetch</td><td>Read only</td></tr>
</table>
</td><td>RW</td><td>DC</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>access</td><td></td></tr>
<tr><td>3’b110</td><td>-</td><td>-</td></tr>
<tr><td>3’b111</td><td>No Read/Write/Fetch access</td><td>Read/Write</td></tr>
</table>
Note that this Access mode governs the access permission of using load/store instructions to access the page. And it governs the user mode instruction fetching/execution permission when M[3:1] == 0b101 or M[3:1] == 0b111.
Using of Reserved values 0 (when X is 0), 4, and 6 will generate Reserved PTE Attribute exception for load/store instructions and instruction fetching.
</td><td></td><td></td></tr>
<tr><td>D</td><td>1
(4)</td><td>This is the Dirty bit.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>If this bit is not set, a store to this page generates a Page Modified Exception.</td></tr>
<tr><td>1</td><td>No Page Modified Exception will be generated.</td></tr>
</table>
</td><td>RW</td><td>DC</td></tr>
<tr><td>X
(For MMU version 1)</td><td>1
(5)</td><td>Indicates if this page is executable or not.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>This page is not executable. An instruction fetch to this page generates a Non-Executable Page exception.</td></tr>
<tr><td>1</td><td>This page is executable.</td></tr>
</table>
</td><td>RW</td><td>DC</td></tr>
<tr><td>X
(For MMU version 2)</td><td>1
(5)</td><td>Executable bit.
● In superuser mode, a non-executable page exception will be generated when fetching a page without this bit set.
● In user mode and when M[3:1] != 0b101</td><td>RW</td><td>DC</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>and M[3:1] != 0b111, a non-executable page exception will be generated when fetching a page without this bit set.
● In user mode and when M[3:1] == 0b101 or M[3:1] == 0b111, a non-executable page exception will always be generated. The X bit is don’t care in this case.</td><td></td><td></td></tr>
<tr><td>A</td><td>1
(6)</td><td>This bit controls if an Access Bit exception will be generated or not.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>No Access Bit exception will be generated</td></tr>
<tr><td>1</td><td>Any access to this page generates Access Bit exception.</td></tr>
</table></td><td>RW</td><td>DC</td></tr>
<tr><td>G</td><td>1
(7)</td><td>This bit indicates if this page is shared across contexts.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>This page is not shared with other context.</td></tr>
<tr><td>1</td><td>This page is shared across contexts.</td></tr>
</table></td><td>RW</td><td>DC</td></tr>
<tr><td>C</td><td>3
(10,8)</td><td>Indicates Cachebility attributes.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>device space</td></tr>
<tr><td>1</td><td>device space, write bufferable / coalescable</td></tr>
<tr><td>2</td><td>non-cacheable memory</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>4</td><td>cacheable, write-back, write-allocate memory</td></tr>
<tr><td>5</td><td>cacheable, write-through, no-write-allocate memory</td></tr>
<tr><td>6</td><td>cacheable, non-shared,</td></tr>
</table></td><td>RW</td><td>DC</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
		<table><tr><td></td><td>write-back, write allocate memory</td></tr><tr><td>7</td><td>cacheable, non-shared, write-through, no-write-allocate memory</td></tr></table> Using of the reserved value (3) will generate UNPREDICTABLE result for the following implementations: ● N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003) ● N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003) But will generate Reserved PTE Attribute exception (Instruction/Data) for all other implementations.		
Reserved	1 (11)	RAZWI		0
PPN	20 (32,12)	This is the Physical Page Number of a page.	RW	DC

9.4.5. TLB Access Misc Register

Mnemonic Name: mr4 (TLB_MISC)
IM Requirement: MMU optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 4, 0}

31 ... 13	12 ... 4	3 ... 0
Reserved	CID	ACC_PSZ

For TLB reads, this register holds the context ID and page size field of the read out. For TLB writes (target or random), this is the context ID and page size of the page translation entry to be installed.

The CONTEXT_ID field of this register is used as the current context ID to tag virtual addresses to avoid the TLB flushing penalty between context switches. The operating system is responsible for assigning a new context ID for each context switch. Since changing this field affects how PTE is searched in the TLB, care must be taken when changing this field. The code that changes this field must be either executed with TLB translation off or contained in a global page PTE where changing CID will not cause any TLB translation exceptions.

The ACC_PAGE_SIZE field of this register is the page size of the page translation entry installed in the software-visible part of the TLB. Note that not all page sizes are implemented in an implementation for the value range between 0 and 9. When this field is written with an unimplemented page size in the value range between 0 and 9, a "Machine Error" exception will be generated.

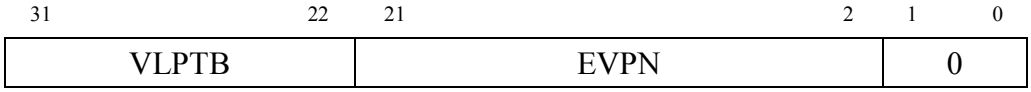
Field Name	Bits	Description	Type	Reset
Access page size (ACC_PSZ)	4 (3,0)	Indicates the page size of a PTE entry. Not all page sizes are implemented in an implementation. The MMU_CFG configuration register lists the page size supported in an implementation. Writing an	RW	DC

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>unimplemented page size into this field will generate a “Machine Error” exception.
<table>
<tr><th>Value</th><th>Page Size</th></tr>
<tr><td>0</td><td>4KB</td></tr>
<tr><td>1</td><td>8KB</td></tr>
<tr><td>2</td><td>16KB</td></tr>
<tr><td>3</td><td>64KB</td></tr>
<tr><td>4</td><td>256KB</td></tr>
<tr><td>5</td><td>1MB</td></tr>
<tr><td>6</td><td>4MB</td></tr>
<tr><td>7</td><td>16MB</td></tr>
<tr><td>8</td><td>64MB</td></tr>
<tr><td>9</td><td>256MB</td></tr>
<tr><td>10-15</td><td>Reserved</td></tr>
</table></td><td></td><td></td></tr>
<tr><td>CID</td><td>9
(12,4)</td><td>Current Context ID. It will be installed into TLB along with a PTE entry.</td><td>RW</td><td>DC</td></tr>
<tr><td>Reserved</td><td>19
(31,13)</td><td>RAZWI</td><td></td><td>0</td></tr>
</table>

9.4.6. Virtual Linear Page Table Index Register

Mnemonic Name: mr5 (VLPT_IDX)
IM Requirement: MMU optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 5, 0}

AndesCore™ Info: N12 does not have this feature implemented.



Optionally, to speed up software page table walk operations in the TLB fill exception handler, operating system can provide a Virtual Linear Page Table which maps the physical page tables into a linear table in a process's virtual address space. If this VLPT mapping exists in the TLB structure, then a two-level physical page table walk using two load instructions will be simplified to a one-level virtual memory load access in the TLB fill exception handler. This is beneficial if there are spare TLB entries which can be used to map VLPT pages. And a 4KB VLPT page can map 4MB of user address space. The VLPT is aligned on a 4MB boundary in the virtual address space. If the VLPT page mapping does not exist in the TLB structure, a nested TLB miss (called double TLB miss exception) will be raised. The handling of the double TLB miss exception is TBD.

The "EVPN" field of this register will be updated to the exception VPN when the following TLB-related exception occurs.

TLB fill
PTE not present
Page modified

So when a TLB fill exception happens, this register will already contains the correct VLPT index where the PTE translation is located for the exception VA in the exception handler.

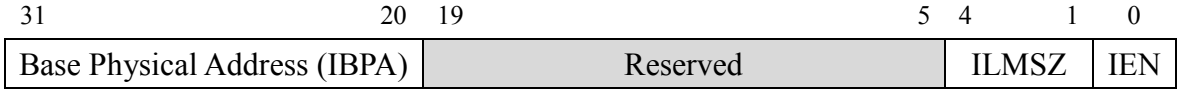
No shadow stack register is specified for the VLPT_IDX.EVPN field since we assume that nested exceptions specified above cannot happen.

Field Name	Bits	Description	Type	Reset
Zero	2 (1,0)	Always 0.	RO	0
EVPN	20 (21,2)	Exception Virtual Page Number. It is updated by hardware on the following TLB exceptions: TLB Fill, PTE not present, and Page Modified, when interruption stack level transition 0 → 1 or 1 → 2 happens. For a 4KB minimum page configuration, it is updated with VA(31,12). For a 8KB minimum page configuration, it is updated with 0.VA(31,13).	RO	0
VLPTB	10 (31,22)	The base virtual address of a process's Virtual Linear Page Table. It is 4MB-aligned.	RW	0

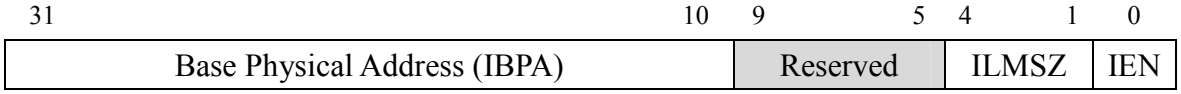
9.4.7. ILM (Instruction Local Memory) Base Register

Mnemonic Name: mr6 (ILMB)
IM Requirement: ILM optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 6, 0}

ICM_CFG.BSAV == 0



ICM_CFG.BSAV == 1



Field Name	Bits	Description	Type	Reset
Enable (IEN)	1 (0)	Enable bit for the Instruction Local Memory. This controls Andes core pipeline access only, including instruction fetch, and instruction execution. It does not affect DMA engine operations. Value: Meaning 0: ILM is disabled 1: ILM is enabled	RW	0
ILM_Size (ILMSZ)	4 (4,1)	Indicates the size of ILM: Value: Meaning 0: 4KB 1: 8KB 2: 16KB 3: 32KB 4: 64KB 5: 128KB	RO	IM

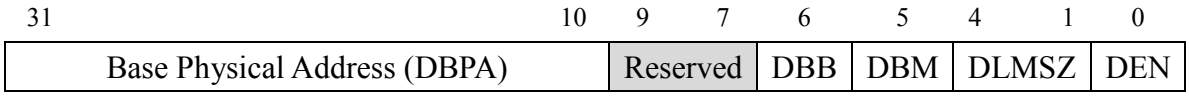
Field Name	Bits	Description	Type	Reset
		6: 256KB 7: 512KB 8: 1024KB 9: 1KB 10: 2KB 11-14: Reserved 15: 0 Byte (used for unconnected external ILM)		
Reserved (ICM_CFG.BSAV == 0)	15 (19,5)	RAZWI	RAZWI	0
Reserved (ICM_CFG.BSAV == 1)	5 (9,5)	RAZWI	RAZWI	0
ILM Base Physical Address (IBPA) (ICM_CFG.BSAV == 0)	12 (31,20)	The base physical address of ILM. It has to be aligned to 1MB boundary for any ILM whose size is smaller than or equal to 1MB. Any access outside of the ILM within the allocated 1MB region will cause a “Nonexistent local memory address” exception.	RW	DC
ILM Base Physical Address (IBPA) (ICM_CFG.BSAV == 1)	22 (31,10)	The base physical address of ILM. It has to be aligned to multiple of ILM size that has to be power of 2.	RW	DC

9.4.8. DLM (Data Local Memory) Base Register

Mnemonic Name: mr7 (DLMB)
IM Requirement: DLM optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 7, 0}

DCM_CFG.BSV == 0

DCM_CFG.BSV == 1



Field Name	Bits	Description	Type	Reset
Enable (DEN)	1 (0)	Enable bit for the Data Local Memory. This controls Andes core pipeline access only. It does not affect DMA engine operations. Value: Meaning 0: DLM is disabled 1: DLM is enabled	RW	0
DLM_Size (DLMSZ)	4 (4,1)	Indicates the size of DLM: Value: Meaning 0: 4KB 1: 8KB 2: 16KB 3: 32KB 4: 64KB 5: 128KB 6: 256KB 7: 512KB	RO	IM

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>8</td><td>1024KB</td></tr>
<tr><td>9</td><td>1KB</td></tr>
<tr><td>10</td><td>2KB</td></tr>
<tr><td>11-14</td><td>Reserved</td></tr>
<tr><td>15</td><td>0 Byte (used for unconnected external ILM)</td></tr>
</table></td><td></td><td></td></tr>
<tr><td>DBM</td><td>1
(5)</td><td>Enable Double-Buffer Mode for data local memory. When in Double-Buffer mode, the data local memory is divided into two address overlapping regions where one can only be accessed by the processor and the other can only be accessed by the DMA engine. The next field, DBB, determines which half of the local memory can be accessed by the processor.<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>0</td><td>Double-buffer mode is disabled.</td></tr>
<tr><td>1</td><td>Double-buffer mode is enabled.</td></tr>
</table>Toggling/Changing this bit will cause UNPREDICTABLE change in the data local memory content. Software should re-initialize the data local memory content after changing this bit.</td><td>RW</td><td>0</td></tr>
<tr><td>DBB</td><td>1
(6)</td><td>Double-buffer bank which can be accessed by the processor. This field is used only when the previous field, DBM, is set to 1.<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>0</td><td>Double-buffer bank 0 is accessed by the processor.</td></tr>
<tr><td>1</td><td>Double-buffer bank 1 is</td></tr>
</table></td><td>RW</td><td>0</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
		accessed by the processor.		
Reserved (DCM_CFG.BSV == 0)	13 (19,7)	RAZWI	RAZWI	0
Reserved (DCM_CFG.BSV == 1)	3 (9,7)	RAZWI	RAZWI	0
DLM Base Physical Address (DBPA) (DCM_CFG.BSV == 0)	12 (31,20)	The base physical address of DLM. It has to be aligned to 1MB boundary for any DLM whose size is smaller than or equal to 1MB. Any access outside of the DLM within the allocated 1MB region will cause a “Nonexistent local memory address” exception.	RW	DC
DLM Base Physical Address (DBPA) (DCM_CFG.BSV == 1)	22 (31,10)	The base physical address of DLM. It has to be aligned to multiple of DLM size that has to be power of 2.	RW	DC

9.4.9. Cache Control Register

Mnemonic Name: mr8 (CACHE_CTL)
IM Requirement: required
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 8, 0}

31 6	5	4	3	2	1	0
Reserved	DCPMW	DCCWF	DCALCK	ICALCK	DC_EN	IC_EN

Field Name	Bits	Description	Type	Reset
Icache Enable (IC_EN)	1 (0)	Controls if first-level instruction cache is enabled or not. Value: 0 — Instruction cache is disabled. Value: 1 — Instruction cache is enabled.	RW	0
Dcache Enable (DC_EN)	1 (1)	Controls if first-level data cache is enabled or not. Value: 0 — Data cache is disabled. Value: 1 — Data cache is enabled.	RW	0
ICALCK	1 (2)	Controls I-cache all-lock resolution scheme. Value: 0 — Generates “Machine Error” exception Value: 1 — Changes a cacheable request to the all-locked cache set into an un-cacheable request.	RW	0
DCALCK	1 (3)	Controls D-cache all-lock resolution scheme. Value: 0 — Generates “Machine Error” exception	RW	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td><table><tr><td>1</td><td>Changes a cacheable request to the all-locked cache set into an un-cacheable request.</td></tr></table></td><td></td><td></td></tr>
<tr><td>DCCWF</td><td>1
(4)</td><td>Controls if D-cache Critical Word Forwarding functionality is enabled or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>D-cache Critical Word Forwarding is disabled.</td></tr><tr><td>1</td><td>D-cache Critical Word Forwarding is enabled.</td></tr></table></td><td>RW</td><td>1</td></tr>
<tr><td>DCPMW</td><td>1
(5)</td><td>Controls if D-cache concurrent (parallel) miss and write-back processing is enabled or not.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>D-cache concurrent miss and write-back processing is disabled.</td></tr><tr><td>1</td><td>D-cache concurrent miss and write-back processing is enabled.</td></tr></table></td><td>RW</td><td>1</td></tr>
<tr><td>Reserved</td><td>26
(31,6)</td><td>RAZWI</td><td></td><td>0</td></tr>
</table>

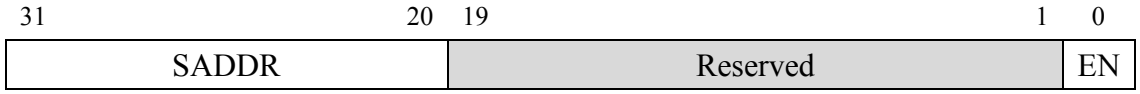
9.4.10. High Speed Memory Port Starting Address

Mnemonic Name: mr9 (HSMP_SADDR)
IM Requirement: High Speed Memory Port optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 9, 0}

Version Note: Version 1 of HSMP definition is only implemented in the following implementations:

- N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)

Please see Version 2 definitions for other implementations.



This register defines the starting physical address of the High Speed Memory Port region.

Field Name	Bits	Description	Type	Reset
EN	1 (0)	Enable control bit for the High Speed Memory port. Value: 0 — The High Speed Memory Port is disabled. Value: 1 — The High Speed Memory Port is enabled.	RW	0
Reserved	19 (19,1)	RAZWI		0
SADDR	12 (31,20)	Starting physical address of the High Speed Memory Port region. It is 1MB-aligned.	RW	DC

HSMP Version 2

31 20	19 13	12 1	0
SADDR	Reserved	RANGE	EN

This register defines the starting base physical address of the High Speed Memory Port region and the power-of-2 range of the region. The region has to be power-of-2 aligned. Any value of SADDR(31,20) outside of the power of 2 range defined in RANGE(12,1) will be ignored (i.e. treated as zero). This register will be removed, if the HSMP interface does not exist.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>EN</td><td>1
(0)</td><td>Enable control bit for the High Speed Memory port.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>The High Speed Memory Port is disabled.</td></tr>
<tr><td>1</td><td>The High Speed Memory Port is enabled.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>RANGE</td><td>12
(12,1)</td><td>The address range is denoted using the following pattern from 1MB to 4GB. If these patterns are not followed, UNPREDICTABLE memory access behavior will be generated.
<table>
<tr><th>Range</th><th>Pattern</th></tr>
<tr><td>1MB</td><td>0b111111111111</td></tr>
<tr><td>2MB</td><td>0b111111111110</td></tr>
<tr><td>4MB</td><td>0b111111111100</td></tr>
<tr><td>8MB</td><td>0b111111111000</td></tr>
<tr><td>16MB</td><td>0b111111110000</td></tr>
<tr><td>32MB</td><td>0b111111100000</td></tr>
<tr><td>64MB</td><td>0b111111000000</td></tr>
<tr><td>128MB</td><td>0b111110000000</td></tr>
<tr><td>256MB</td><td>0b111100000000</td></tr>
<tr><td>512MB</td><td>0b111000000000</td></tr>
</table></td><td>RW</td><td>DC</td></tr>
</table>

Field Name	Bits	Description	Type	Reset
		1GB: 0b110000000000 2GB: 0b100000000000 4GB: 0b000000000000		
Reserved	7 (19,13)	RAZWI		0
SADDR	12 (31,20)	Starting base physical address of the High Speed Memory Port region. It is a power-of-2 range of memory blocks. The value outside of the power-of-2 range defined in RANGE(12,1) will be ignored.	RW	DC

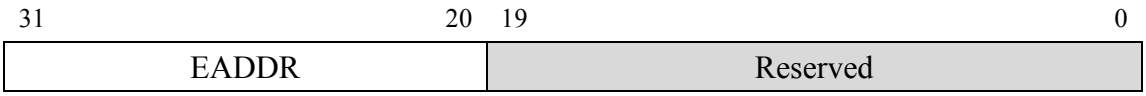
9.4.11. High Speed Memory Port Ending Address

Mnemonic Name: mr10 (HSMP_EADDR)
IM Requirement: High Speed Memory Port optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 9, 1}

Version Note: Version 1 of HSMP definition is only implemented in the following implementations:

- N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)

Please see Version 2 definitions for other implementations.



This register defines the ending inclusive 1MB-aligned physical address of the High Speed Memory Port region. The address starting from "EADDR(31,20) + 1" 1MB-aligned region will not belong to the High Speed Memory Port region.

Field Name	Bits	Description	Type	Reset
Reserved	20 (19,0)	RAZWI		0
EADDR	12 (31,20)	Ending inclusive physical address of the High Speed Memory Port region. It is 1MB-aligned.	RW	DC

HSMP Version 2

In HSMP version 2, this register will be removed from the spec.

9.5. EDM System Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension
dr0+(n*5)	BPCn (n=0..7)	3	0	n
dr1+(n*5)	BPAn (n=0..7)	3	1	n
dr2+(n*5)	BPAMn (n=0..7)	3	2	n
dr3+(n*5)	BPVn (n=0..7)	3	3	n
dr4+(n*5)	BPCIDn (n=0..7)	3	4	n
dr40	EDM_CFG	3	5	0
dr41	EDMSW	3	6	0
dr42	EDM_CTL	3	7	0
dr43	EDM_DTR	3	8	0
dr44	BPMTC	3	9	0
dr45	DIMBR	3	10	0
dr46	TECR0	3	14	0
dr47	TECR1	3	14	1

9.6. Performance Monitoring Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page
pfr0-pfr2	PFMCn (n=0..2)	4	0	n	171
pfr3	PFM_CTL	4	1	0	172

9.6.1. Performance Counter Register 0-2

Mnemonic Name: (pfr0-pfr2) PFMC0-PFMC2
IM Requirement: Performance Monitoring Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {4, 0, 0-2}

31	0
PFMC0-PFMC2	

These are the Performance Counters used to count the events selected in the Performance Counter Control (PMC_CTL) register. They are all readable and writeable by software in Superuser mode. A counter overflows when the counter value wraps around from 0xFFFFFFFF to 0x0. And the corresponding Overflow bit in the PMC_CTL register will be set. If the corresponding Interrupt Enable bit is set as well, then a pending hardware interrupt will be generated to signal the interrupt controller. The software can clear the pending interrupt by writing a one to the Overflow bit in the PMC_CTL register. A counter will continue to count after the overflow event. The event counting can be filtered out individually for Superuser and User mode on each counter. So to stop a counter from counting completely, software can filter out counting from both Superuser and User mode.

When a counter is disabled, an implementation may put the counter into a low power state to preserve energy. When a counter is disabled and then re-enabled again, the counter value is whatever value existed before the re-enabled point.

Field Name	Bits	Description	Type	Reset
PFMC0-PFMC2	32 (31,0)	Performance event counter	RW	DC

9.6.2. Performance Counter Control Register

Mnemonic Name: (pfr3) PFM_CTL
IM Requirement: Performance Monitoring Optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {4, 1, 0}

31 28	27 22	21 16	15	14 12	11 9	8 6	5 3	2 0
Reserved	SEL2	SEL1	SEL0	KU2-0	KS2-0	OVF2-0	IE2-0	EN2-0

Field name	Bits	Description	Type	Reset
EN2-0	3 (2,0)	The enable bit for each performance counter register. Value: 0 — The corresponding performance counter is turned off. Value: 1 — The corresponding performance counter is turned on and starts to count.	RW	0
IE2-0	3 (5,3)	The interrupt enable bit for each performance counter. Value: 0 — The corresponding performance counter interrupt is being masked off. Value: 1 — The corresponding performance counter interrupt is enabled for reporting.	RW	0

<table>
<tr><th>Field name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>OVF2-0</td><td>3
(8,6)</td><td>The overflow bit for each performance counter. An interrupt request will be generated if this bit is set and the interrupt enable bit is set as well.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The corresponding performance counter has not overflowed.</td></tr><tr><td>1</td><td>The corresponding performance counter has overflowed</td></tr></table></td><td>W1C</td><td>0</td></tr>
<tr><td>KS2-0</td><td>3
(11,9)</td><td>The Superuser mode event counting control for each performance counter.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The event counting is on in Superuser mode for the corresponding performance counter.</td></tr><tr><td>1</td><td>The event counting is off in Superuser mode for the corresponding performance counter.</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>KU2-0</td><td>3
(14,12)</td><td>The User mode event counting control for each performance counter.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The event counting is on in User mode for the corresponding performance counter.</td></tr><tr><td>1</td><td>The event counting is off in User mode for the corresponding performance counter.</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>SEL0</td><td>1</td><td>The event selection for performance</td><td>RW</td><td>0</td></tr>
</table>

Field name	Bits	Description	Type	Reset
	(15)	counter 0. Value: 0 = Cycles; 1 = Completed instructions		
SEL1	6 (21,16)	The event selection for performance counter 1. The events included are implementation-dependent. Please see Table 27 for the basic event list.	RW	0
SEL2	6 (27,22)	The event selection for performance counter 2. The events included are implementation-dependent. Please see Table 27 for the basic event list.	RW	0
Reserved	4 (31,28)	RAZWI	-	0

Table 27. Basic Counting Events for Counter 1 and Counter 2

SEL	C1	C2
0	total cycles	
1	completed instructions	
2	conditional branch	conditional branch mispredict
3	taken conditional branches	taken conditional branch mispredict
4	prefetch insts	prefetch with $ hit
5	RET inst	RET mispredict
6	JR (non-RET) inst	immeidate J insts (exclude JAL)
7	JAL/JRAL insts	multiply insts
8	NOP instruction	16-bit instructions
9	SCW instruction	failed SCW
10	ISB/DSB instruction	ld-after-st conflict replays
11	CCTL instruction	-
12	taken interrupt	exceptions taken
13	loads completed (LMW count as 1)	stores completed (SMW count as 1)
14	uITLB access	uITLB miss
15	uDTLB access	uDTLB miss
16	MTLB access	MTLB miss

SEL	C1	C2
17	I$ access	I$ miss
18	data dependency stall cycles (when it is the sole cause of stall)	no inst from IQ stall cycles
19	D$ miss stall cycles	D$ writebacks
20	D$ access (LMW/SMW count as many, not including CCTL)	
21	D$ miss	
22	LD D$ access (LMW count as many, not including CCTL)	LD D$ miss (LMW count as many, not including CCTL)
23	ST D$ access (SMW count as many, not including CCTL)	ST D$ miss (LMW count as many, not including CCTL)
24	ILM access	DLM access
25	LSU BIU cycles (D$ fill, noncacheable, write-back, write-through)	LSU BIU request (D$ fill, noncachable, write-back, write-through)
26	HPTWK BIU cycles	HPTWK BIU request
27	DMA BIU cycles	DMA BIU request
28	I$ fill BIU cycles	I$ fill BIU request
29	legal unaligned D$ access	external event (implementation-dep)
30-63	-	-

* "-" in the above table means "un-assigned value". Writing un-assigned value will result in UNPREDICTABLE event collection which is implementation-dependent.

Note: The following event counts in the above table should not include counts during stall cycles, and should exclude, with reasonable efforts, extra counts due to replay cycles.

uITLB access	uITLB miss
uDTLB access	uDTLB miss
MTLB access	MTLB miss
I$ access	I$ miss
D$ access	D$ miss
ILM access	DLM access

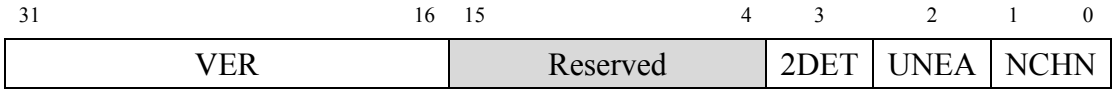
9.7. Local Memory DMA Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
dmar0	DMA_CFG	5	0	0	177
dmar1	DMA_GCSW	5	1	0	179
dmar2	DMA_CHNSEL	5	2	0	181
dmar3	DMA_ACT	5	3	0	182
dmar4	DMA_SETUP	5	4	0	184
dmar5	DMA_ISADDR	5	5	0	188
dmar6	DMA_ESADDR	5	6	0	190
dmar7	DMA_TCNT	5	7	0	191
dmar8	DMA_STATUS	5	8	0	192
dmar9	DMA_2DSET	5	9	0	195
dmar10	DMA_2DSCTL	5	9	1	199

9.7.1. DMA Configuration Register

Mnemonic Name: dmar0 (DMA_CFG)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 0, 0}
Per DMA Channel: No



The DMA Configuration register specifies the implemented DMA features in an implementation.

Field Name	Bits	Description	Type	Reset
NCHN	2 (1,0)	The number of DMA channels implemented. Value: Meaning 0: 1 DMA channel 1: 2 DMA channels 2: Reserved 3: Reserved	RO	IM
UNEA	1 (3)	Indicates if the Un-aligned External Address transfer feature is implemented. Value: Meaning 0: The feature is not implemented 1: The feature is implemented.	RO	IM
2DET	1 (4)	Indicates if the 2-D Element Transfer feature is implemented. Value: Meaning 0: The feature is not implemented 1: The feature is implemented.	RO	IM
Reserved	12	RAZWI	-	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td>(15,4)</td><td></td><td></td><td></td></tr>
<tr><td>VER</td><td>16
(31,16)</td><td>Indicates the DMA architecture and implementation version. It is suggested to partition this 16-bit number into two parts: a 12-bit part and a 4-bit part. The 12-bit part is for major architecture changes and the 4-bit part is for minor implementation updates. The current assignments are as follows:
<table>
<tr><th>Implemenatation</th><th>Number</th></tr>
<tr><td>N1213 Hardcore
N1213_43U1HA0</td><td>0x0000</td></tr>
<tr><td>N1213 Hardcore
N1213_43U1HB0</td><td>0x0001</td></tr>
<tr><td>N1213-S</td><td>0x0011</td></tr>
</table>
</td><td>RO</td><td>IM</td></tr>
</table>

9.7.2. DMA Global Control and Status Word Register

Mnemonic Name: dmar1 (DMA_GCSW)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 1, 0}
Per DMA Channel: No

31	30 ... 14	13	12	11 ... 6	5 ... 3	2 ... 0
EN	Reserved	C1INT	C0INT	Reserved	C1STAT	C0STAT

The DMA Global Control and Status Word register enables DMA engine logic. The enable state will be exported out of the core as information for power management. When the DMA engine is enabled, it also collects status information from the two DMA channels. Software can use this register to determine if there is any "Idle" DMA channel to use.

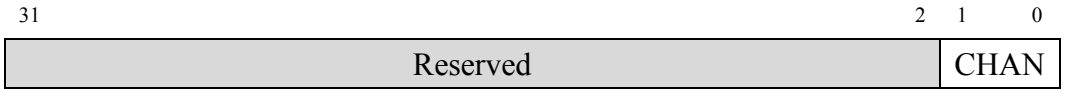
Disabling the DMA global enable bit will immediately put the two DMA channels in the "Idle" state and stop all ongoing DMA transactions.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>C0STAT</td><td>3
(2,0)</td><td>The current state the DMA channel 0 is in.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Idle ("Reset" action value)</td></tr>
<tr><td>1</td><td>Run</td></tr>
<tr><td>2</td><td>Pended</td></tr>
<tr><td>3</td><td>Stop/Error</td></tr>
<tr><td>4</td><td>Complete</td></tr>
<tr><td>5-7</td><td>-</td></tr>
</table></td><td>RO</td><td>0</td></tr>
<tr><td>C1STAT</td><td>3
(5,3)</td><td>The current state the DMA channel 1 is in. Please see the above field (C0STAT) for value definitions.</td><td>RO</td><td>0</td></tr>
<tr><td>Reserved</td><td>6</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td>(11,6)</td><td></td><td></td><td></td></tr>
<tr><td>C0INT</td><td>1
(12)</td><td>Indicates if an interrupt has been generated from DMA channel 0.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No interrupt</td></tr><tr><td>1</td><td>An interrupt has been generated.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>C1INT</td><td>1
(13)</td><td>Indicates if an interrupt has been generated from DMA channel 1. Please see the above field (C0INT) for value definitions.</td><td>RO</td><td>0</td></tr>
<tr><td>Reserved</td><td>17
(30,14)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
<tr><td>EN</td><td>1
(31)</td><td>DMA engine global enable bit. This state will be exported out of AndesCore™ as one of the information for external power and frequency control.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>DMA engine is disabled. No DMA transfer is possible. Any write to DMA Action register will be ignored. Disabling this bit will immediately put the two DMA channels in the “Idle” state and stop all ongoing DMA transactions.</td></tr><tr><td>1</td><td>DMA engine is enabled. DMA Action register can be programmed to perform DMA operations.</td></tr></table></td><td>RW</td><td>0</td></tr>
</table>

9.7.3. DMA Channel Selection Register

Mnemonic Name: dmar2 (DMA_CHNSEL)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 2, 0}
Per DMA Channel: No

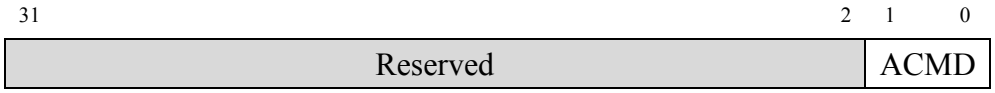


The DMA channel selection register is used to control the accessibility of the DMA per channel registers. Once the CHAN field is set to a particular DMA channel, all later per channel register accesses will occur on registers belong to that particular channel. Note that to read a register of the selected channel after a write to this register, a "data serialization barrier" (DSB) instruction must be present in between the write operation and the read operation. However, to write a register of the selected channel after a write to this register, the DSB instruction does not need to be present in between these two write operations. All implementations will guarantee that after a channel selection operation the newly selected channel can be written immediately.

Field Name	Bits	Description	Type	Reset
CHAN	2 (1,0)	Selected channel number. Value / Meaning 0 – Channel 0 is selected. 1 – Channel 1 is selected. 2 – Reserved 3 – Reserved	RW	0
Reserved	30 (31,2)	RAZWI		0

9.7.4. DMA Action Register

Mnemonic Name: dmar3 (DMA_ACT)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 3, 0}
Per DMA Channel: Yes



The DMA Action register activates/de-activates the transfer operation of the DMA engine based on the value written into the ACMD field.

Field Name	Bits	Description	Type	Reset
ACMD	2 (1,0)	DMA Action Command. Value: Meaning 0: No DMA action 1: Start the DMA transfer 2: Stop the DMA transfer 3: Reset DMA status/error	WO	0
Reserved	30 (31,2)	RAZWI		0

Details of DMA Action Command:

- Start

  The "Start" command activates the DMA transfer operation from the "Idle" or "Stop/Error" states. If the other channel is not in the "Run" state, this DMA channel will be transitioned into the "Run" state. If the other channel is in the "Run" state, this DMA channel will be transitioned into the "Pended" state. The above state transitions are only true if there is no error detected. If there is any error detected, the DMA will be put into the "Stop/Error" state.

- Stop

  The "Stop" command deactivates the DMA transfer operation from the "Run" or "Pended" states. When the DMA transfer operation finally stops, the DMA channel

will be put into the “STOP” state. Software needs to check the status of the DMA channel to make sure it really stops after it executes this command. When the DMA channel stops, the Internal/External Address and Transfer Element Count registers contain the values that can restart the remaining DMA transfer transactions. There is no effect that will happen if DMA channel is in other states when receiving this command.

- Reset

  The “Reset” command clears out the channel’s status, interrupt, and error information and put the channel into the “Idle” state when the DMA channel is in the “Complete”, or “Stop/Error” states.

9.7.5. DMA Setup Register

Mnemonic Name: dmar4 (DMA_SETUP)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 4, 0}
Per DMA Channel: Yes

20	19	18	17	16	15 ... 4	3 2	1	0
2DE (opt)	UE (opt)	EIE	SIE	CIE	ESTR	TES	TDIR	LM

31 ... 24	23 ... 21
Reserved	CA

The DMA Setup Register specifies all the essential DMA transfer parameters for the DMA engine to perform its transfer work. When the DMA channel is in the RUN or the PENDED state, writing to this register will cause a DMA Transfer Disruption error.

Field Name	Bits	Description	Type	Reset
LM	1 (0)	Local Memory Selection. Value: 0 — ILM is selected for DMA transfer. Value: 1 — DLM is selected for DMA transfer.	RW	0
TDIR	1 (1)	Transfer Direction. Value: 0 — From external memory to LM. Value: 1 — From LM to external memory.	RW	0
TES	2 (3,2)	Transfer Element Size. Value: 0 — Byte Value: 1 — 2 Bytes (Half-word)	RW	0

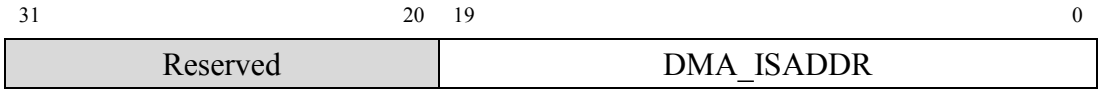
Field Name	Bits	Description	Type	Reset
		2 — 4 Bytes (Word) 3 — 8 Bytes (Double-word) optional When the 8 bytes transfer element size is not supported, writing this value to this field will generate a reserved value exception, or unpredictable results for some implementations.		
ESTR	12 (15,4)	External memory transfer Stride. This is the byte value increments on the external memory address between successive DMA transfers. A value of 0 will keep the external memory address to be the same for all transfers. This stride value has to be aligned to the transfer element size if the optional Unaligned External Address feature is not implemented; otherwise an Un-aligned error will be reported in the DMA Channel Status register when the DMA transfer starts. If the optional Unaligned External Address feature is implemented, the error reporting condition will depend on if the feature is enabled, not reporting error, or not, reporting error. In the 2-D Element transfer mode, this stride is the “width stride” of a transfer. And an additional “height stride” is needed and is defined in the DMA 2D Setup register. Please see section 9.7.10 for more detail.	RW	0
CIE	1 (16)	Interrupt Enable on Completion. Value — Meaning 0 — No interrupt on DMA transfer completion. 1 — Generate interrupt on DMA transfer completion.	RW	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>SIE</td><td>1
(17)</td><td>Interrupt Enable on explicit Stop. This may be necessary since after an explicit Stop command is being issued to the DMA engine, the DMA engine may not stop immediately until completing its current outstanding bus transactions.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>No interrupt when the DMA transfer is explicitly stopped by a STOP command.</td></tr>
<tr><td>1</td><td>Generate interrupt when the DMA transfer is explicitly stopped by a STOP command.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>EIE</td><td>1
(18)</td><td>Interrupt Enable on Error.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>No interrupt on DMA errors.</td></tr>
<tr><td>1</td><td>Generate interrupt on DMA errors.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>UE
(optional)</td><td>1
(19)</td><td>This is used when the Un-aligned External Address (UEA) Feature is implemented. This bit is used to enable the Un-aligned External Address feature.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>The UEA feature is disabled.</td></tr>
<tr><td>1</td><td>The UEA feature is enabled.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>2DE
(optional)</td><td>1
(20)</td><td>This is used when the 2-D External Transfer (2DET) feature is implemented. This bit is used to enable the 2-D External Transfer feature.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>The 2DET feature is disabled.</td></tr>
<tr><td>1</td><td>The 2DET feature is enabled.</td></tr>
</table></td><td>RW</td><td>0</td></tr>
<tr><td>CA</td><td>3
(23,21)</td><td>Cachebility attribute of external memory access. This is used for external cache</td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>memory management of the DMA transferred data. The cachebility attribute is defined as follows:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Device space</td></tr>
<tr><td>1</td><td>Device space, write bufferable/coalescable</td></tr>
<tr><td>2</td><td>Non-cacheable memory</td></tr>
<tr><td>3</td><td>-</td></tr>
<tr><td>4</td><td>cacheable, write-back, write-allocate memory (shared)</td></tr>
<tr><td>5</td><td>cacheable, write-through, no-write-allocate memory (shared)</td></tr>
<tr><td>6</td><td>cacheable, non-shared, write-back, write allocate memory</td></tr>
<tr><td>7</td><td>cacheable, non-shared, write-through, no-write-allocate</td></tr>
</table></td><td></td><td></td></tr>
<tr><td>Reserved</td><td>8
(31,24)</td><td>RAZWI</td><td></td><td>0</td></tr>
</table>

9.7.6. DMA Internal Start Address Register

Mnemonic Name: dmar5 (DMA_ISADDR)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 5, 0}
Per DMA Channel: Yes



The DMA Internal Start Address register specifies the starting physical address offset from the local memory base address the data transfer will happen in local memory. Based on the real local memory size and the double buffer mode control, some of the upper bits of this address offset from Bit-19 to Bit-11 will be ignored by an AndesCore™. The effective bit range is listed in the following table.

LM Size	Non-double-buffer range	double buffer range
4KB	(11,0)	(10,0)
8KB	(12,0)	(11,0)
16KB	(13,0)	(12,0)
32KB	(14,0)	(13,0)
64KB	(15,0)	(14,0)
128KB	(16,0)	(15,0)
256KB	(17,0)	(16,0)
512KB	(18,0)	(17,0)
1024KB	(19,0)	(18,0)

The physical address offset specified must be aligned to the transfer element size set in the DMA Setup register; otherwise, an Un-aligned error will be reported in the DMA Channel Status register.

During a DMA transfer, if the address offset increment overflows, an Out-Of-Range error

will be generated. And for moving data into local memory, the data of local memory around the zero offset may be corrupted due to this error. Software should be careful to avoid this problem.

When the DMA channel is in the RUN state, the content of this register does not change. However, when the DMA channel is stopped, by a STOP command or an error, this register will contain the address that is required to restart the transaction. When the DMA channel completes its transfer, this register will contain the next transfer address offset of the last transfer of the transaction.

When the DMA channel is in the RUN or the PENDED state, writing to this register will cause a DMA Transfer Disruption error.

This is a RW type register. The reset value of this register is DC.

9.7.7. DMA External Start Address Register

Mnemonic Name: dmar6 (DMA_ESADDR)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 6, 0}
Per DMA Channel: Yes

The DMA External Start Address register specifies the starting physical address in the external memory outside the core the data transfer will happen. And the address specified must be aligned to the transfer element size set in the DMA Setup register if the optional Unaligned External Address feature is not implemented; otherwise, an Un-aligned error will be reported in the DMA Channel Status register. If the optional Unaligned External Address feature is implemented, the error reporting condition will depend on if the feature is enabled, not reporting error, or not, reporting error.

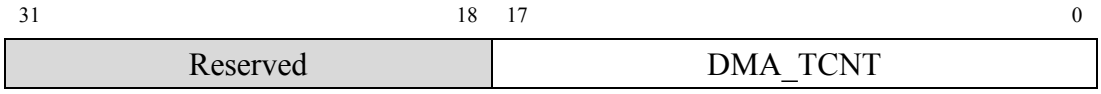
When the DMA channel is in the RUN state, the content of this register does not change. However, when the DMA channel is stopped, by a STOP command or an error, this register will contain the address that is required to restart the transaction. When the DMA channel completes its transfer, this register will contain the next transfer address of the last transfer of the transaction. To be more specific, for 1-D transfer, the next transfer address is the last transfer address plus DMA_SETUP.ESTR. For 2-D transfer, the next transfer address is the last transfer address plus DMA_2DSET.HTSTR or, if the last transfer address is not the last element of a row, the last transfer address plus DMA_SETUP.ESTR..

When the DMA channel is in the RUN or the PENDED state, writing to this register will cause a DMA Transfer Disruption error.

This is a RW type register. The reset value of this register is DC.

9.7.8. DMA Transfer Element Count Register

Mnemonic Name: dmar7 (DMA_TCNT)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 7, 0}
Per DMA Channel: Yes



The DMA Transfer Element Count register specifies the total number of element transfers the DMA engine will perform. During a DMA transfer transaction, at any time if the internal transfer address offset overflows, an Out-Of-Range error will be generated.

When the DMA channel is in the RUN state, the content of this register does not change. However, when the DMA channel is stopped, by a STOP command or an error, this register will contain the remaining transfer count that is required to complete the transaction. When the DMA channel completes its transfer, this register will contain a value of 0.
When the DMA channel is in the RUN or the PENDED state, writing to this register will cause a DMA Transfer Disruption error.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>DMA_TCNT</td><td>18
(17,0)</td><td>The DMA transfer element count.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The DMA completes immediately and no transfer will happen.</td></tr><tr><td>> 0</td><td>The number of element transfers the DMA will perform.</td></tr></table></td><td>RW</td><td>DC</td></tr>
<tr><td>Reserved</td><td>14
(31,18)</td><td>RAZWI</td><td></td><td>0</td></tr>
</table>

9.7.9. DMA Status Register

Mnemonic Name: dmar8 (DMA_STATUS)
IM Requirement: DMA optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 8, 0}
Per DMA Channel: Yes

31 10	9	8	7	6	5	4	3	2 0
Reserved	ESUP	EBUS	IOOR	IUNA	EUNA	DERR	STUNA	STAT

The DMA Status register is a read-only register. It shows the current status, state and error, in the selected DMA channel. A "Reset" action command written to the DMA Action register will change the DMA state to "Idle" and clears all error indications. Note that for the read of this DMA status register to get the correct status after a DMA action command or a DMA channel selection change, a "data serialization barrier" (DSB) instruction must be present in between the DMA action command and the read instruction (MFSR) of this register.

The various error flags/indications in this register are only valid when the DMA channel state is in the Stop/Error state.

Field Name	Bits	Description	Type	Reset
STAT	3 (2,0)	The current state the DMA channel is in. Value: Meaning 0: Idle ("Reset" action value) 1: Run 2: Pended 3: Stop/Error 4: Complete 5-7: -	RO	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>STUNA</td><td>1
(3)</td><td>Indicates if an Un-aligned error has happened on the External Stride value.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>DERR</td><td>1
(4)</td><td>DMA Transfer Disruption Error.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>EUNA</td><td>1
(5)</td><td>Indicates if an Un-aligned error has happened on the External address.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>IUNA</td><td>1
(6)</td><td>Indicates if an Un-aligned error has happened on the Internal address.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>IOOR</td><td>1
(7)</td><td>Indicates if an Our-Of-Range error has happened on the Internal address.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>EBUS</td><td>1
(8)</td><td>Indicates if a Bus Error has happened on an External DMA transfer.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
</table>

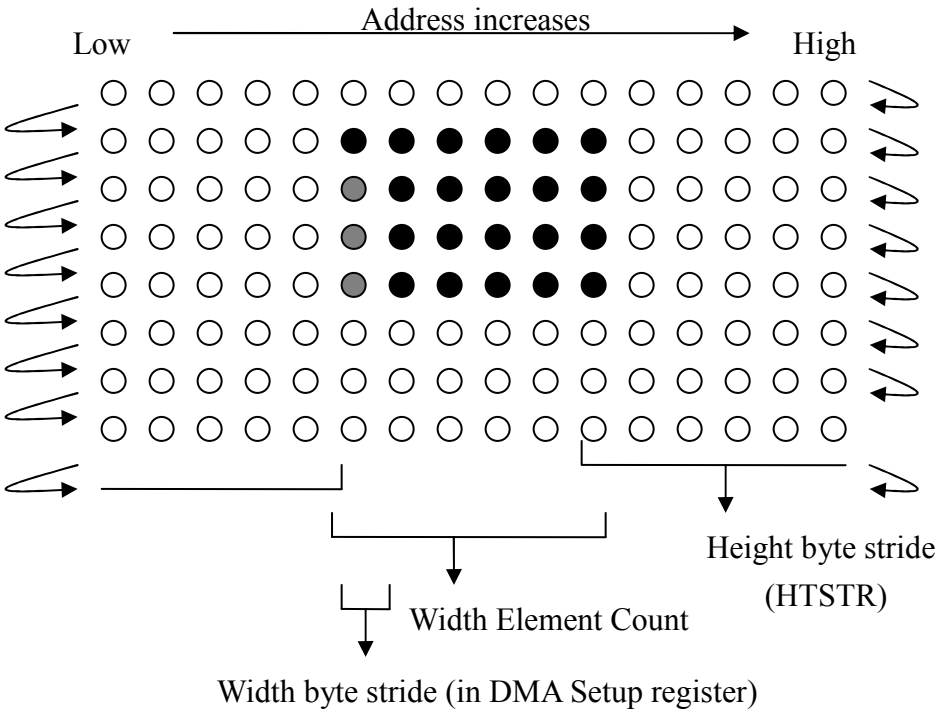
<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>ESUP</td><td>1
(9)</td><td>Indicates if a DMA setup error has happened. A value of 0 in DMA_2DSET.WECNT or DMA_2DSCTL.STWECNT when a DMA transfer starts will generate this error.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error (“Reset” action value)</td></tr><tr><td>1</td><td>The error has happened.</td></tr></table></td><td>RO</td><td>0</td></tr>
<tr><td>Reserved</td><td>23
(31,9)</td><td>RAZWI</td><td></td><td>0</td></tr>
</table>

9.7.10. DMA 2D Setup Register

Mnemonic Name: dmar9 (DMA_2DSET)
IM Requirement: DMA 2-D Element Transfer feature optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 9, 0}
Per DMA Channel: Yes

31 16	15 0
HTSTR	WECNT

The DMA 2D Setup register specifies two values which, along with the stride value defined in the DMA Setup register, define a 2-dimensional region in the external memory that needs to be transferred into a linear array in the local memory. The 2-dimensional region in the external memory can be visualized using the following figure.



The Height Byte Stride (HTSTR) shown in the above picture represents the byte value between the starting address of the rightmost transfer element of the current line of the 2-D region to the starting address of the leftmost transfer element of the next line of the 2-D region regardless of the element transfer size.

If the 2-D Element Transfer function is enabled, the DMA engine will

(1) Use the External Starting address to transfer the first data element, (at this point, only one element has transferred and the transfer address still points to the External starting address)
(2) Use the "Width stride" defined in the DMA Setup register as address increment to transfer "Startup Width element count" (defined in DMA 2D Startup Control register) minus one numbers of data elements from external memory (at this point, all the elements of the first line of the 2-D region has been transferred, and the transfer address points at the starting address of the rightmost element of the first line), and then
(3) Use the "Height stride" defined in this register as address increment to transfer the leftmost element of the next line (shown as a gray dot in the above 2-D transfer picture), when this is done the transfer address points at the starting address of the leftmost element, and then
(4) Use the "Width stride" defined in the DMA Setup register as address increment to transfer "Width element count" (defined in this register) minus one numbers of data elements from external memory.
(5) Repeat (3) and (4) until the "Transfer element count" numbers of data element have been transferred.

The gray dots shown in the figure are the data elements whose transfer addresses are calculated by adding the starting address of the rightmost transfer element of the previous line and the "Height stride" (HTSTR) in step (3).

When the DMA channel is in the RUN or the PENDED state, writing to this register will cause a DMA Transfer Disruption error.

Field Name	Bits	Description	Type	Reset
WECNT	16	The Width Element Count for a 2-D transfer	RW	DC

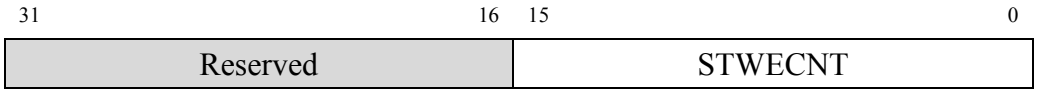
	(15,0)	region. A value of 1 degenerates the 2D transfer feature into a linear stride transfer and is not recommended for its use. A value of 0 is illegal and will generate a DMA setup error when a DMA transfer starts.		
HTSTR	16 (31,16)	The Height Stride for a 2-D transfer region. This is the byte value increment on the external memory address between successive "Width Element Count" numbers of DMA transfers. This stride value has to be aligned to the transfer element size if the optional Unaligned External Address feature is not implemented; otherwise an Un-aligned error will be reported in the DMA Channel Status register when the DMA transfer starts. If the optional Unaligned External Address feature is implemented, the error reporting condition will depend on if the feature is enabled, not reporting error, or not, reporting error. ***Implementation Constraints:*** Due to implementation complexity, an implementation may add the following constraints to the legal value of this field. ● For byte transfer, HTSTR cannot be 0. ● For halfwod transfer, HTSTR cannot be 0 or 1. ● For word transfer, HTSTR cannot be 0, 1, 2, or 3.	RW	DC

Implementation Note: The following implementations have a slightly different definition for WECNT. In these implementations, the WECNT is defined to be "width minus 1" instead of "width".

- N1213 hardcore N1213_43U1HA0 (CPU_VER == 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER == 0x0C020003)

9.7.11. DMA 2D Startup Control Register

Mnemonic Name: dmar10 (DMA_2DSCTL)
IM Requirement: DMA 2-D Element Transfer feature optional
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {5, 9, 1}
Per DMA Channel: Yes



The 2D Startup Control register specifies the first width element count used in a DMA 2D block transfer operation. This register is a read/write register, and a write-shadow of the WECNT field of the 2D Setup register.

When the DMA channel is in the RUN state, the content of this register does not change. However, when the DMA channel is stopped, by a STOP command or an error, this register will contain the “starting width element count” that is required to restart the transaction. When the DMA channel completes its transfer, the STWECNT field of this register will contain a value that is implementation-dependent.

When the DMA channel is in the RUN or the PENDED state, writing to this register will cause a DMA Transfer Disruption error.

Field Name	Bits	Description	Type	Reset
STWECNT	16 (15,0)	The Startup (first) Width Element Count for a 2-D transfer region. This is a write shadow of the WECNT field of the 2D Setup register. So a write to the WECNT field of the 2D Setup register will also update this field. But a write to this field will not change the value in the WECNT field of the 2D Setup register. A value of 0 is illegal and will generate a DMA setup	RW/ WS	DC

Field Name	Bits	Description	Type	Reset
		error when a DMA transfer starts.		
Reserved	16 (31,16)	RAZWI	-	0

Implementation Note: The following implementations have a slightly different definition for WECNT. In these implementations, the WECNT is defined to be "width minus 1" instead of "width".

- N1213 hardcore N1213_43U1HA0 (CPU_VER == 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER == 0x0C020003)

9.8. Resource Access Control Registers

Brief Summary

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page
	PRUSR_ACC_CTL	4	4	0	202
fucpr	FUCOP_CTL	4	5	0	204

PRUSR_ACC_CTL register exists when either MSC_CFG.PFM is 1 or MSC_CFG.LMDMA is 1 except for the following implementations.

- N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)

9.8.1. Privileged Resource User Access Control Registers

Mnemonic Name: PRUSR_ACC_CTL
IM Requirement: Required when either MSC_CFG.PFM is 1 or MSC_CFG.LMDMA is 1
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {4, 4, 0}

31 2	1	0
Reserved	PFM_USR_EN	DMA_USR_EN

This register controls whether a user mode program is allowed to directly access a group of related system registers (or user special registers) and the corresponding functionality for a particular privileged feature. The individual control bit is also aliased to the USR register 0 of a special feature USR group.

This register exists when either MSC_CFG.PFM is 1 or MSC_CFG.LMDMA is 1 except this register does not exist for the following AndesCore™ implementations:

- N1213 hardcore N1213_43U1HA0 (CPU_VER = 0x0C010003)
- N1213 hardcore N1213_43U1HB0 (CPU_VER = 0x0C020003)

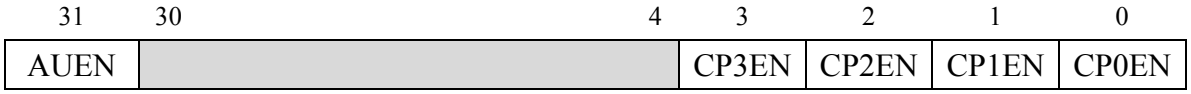
Field Name	Bits	Description	Type	Reset
DMA_USR_EN	1 (0)	The DMA registers user mode access enable control. Value: 0 — User mode access of DMA registers and feature is not allowed. Value: 1 — User mode access of DMA registers and feature is allowed.	RW	0
PFM_USR_EN	1 (1)	The Performance Monitoring registers user mode access enable control. Value: 0 — User mode access of Performance Monitoring registers	RW	0

Field Name	Bits	Description		Type	Reset
			and feature is not allowed.		
		1	User mode access of Performance Monitoring registers and feature is allowed.		
Reserved	30 (31,2)	RAZWI		-	0

9.8.2. FPU and Coprocessor Enable Control Register

Mnemonic Name: fucpr (FUCOP_CTL)
IM Requirement: COP/FPU or Audio extension
Access Mode: Superuser
SR Value {Major, Minor, Extension}: {4, 5, 0}

This system register controls enabling/disabling of audio extension, coprocessors, and the floating-point unit. Note that the floating-point unit enable is "CP0EN" when the floating-point unit is supported (i.e. CR6.CP0EX is 1 and CR6.CP0ISFPU is 1).



Field Name	Bits	Description	Type	Reset
CP0EN	1 (0)	Coprocessor #0 enable bit. 0: Coprocessor #0 is disabled. Any encountering of Coprocessor #0 instruction when Coprocessor #0 exists will cause "Coprocessor disabled" exception. 1: Coprocessor #0 is enabled.	RW	0
CP1EN	1 (1)	Coprocessor #1 enable bit. 0: Coprocessor #1 is disabled. Any encountering of coprocessor #1 instruction when coprocessor #1 exists will cause "Coprocessor disabled" exception. 1: Coprocessor #1 is enabled.	RW	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>CP2EN</td><td>1
(2)</td><td>Coprocessor #2 enable bit.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Coprocessor #2 is disabled. Any encountering of coprocessor #2 instruction when coprocessor #2 exists will cause “Coprocessor disabled” exception.</td></tr><tr><td>1</td><td>Coprocessor #2 is enabled.</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>CP3EN</td><td>1
(3)</td><td>Coprocessor #3 enable bit.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Coprocessor #3 is disabled. Any encountering of coprocessor #3 instruction when coprocessor #3 exists will cause “Coprocessor disabled” exception.</td></tr><tr><td>1</td><td>Coprocessor #3 is enabled.</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>Reserved</td><td>27
(30,4)</td><td>RAZWI</td><td>-</td><td>-</td></tr>
<tr><td>AUEN</td><td>1
(31)</td><td>Audio extension enable bit.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Audio extension is disabled. Any encountering of Audio extension instruction when Audio extension exists will cause “Audio extension disabled” exception.</td></tr><tr><td>1</td><td>Audio extension is enabled.</td></tr></table></td><td>RW</td><td>0</td></tr>
</table>

9.9. Implementation-Dependent Registers

The following system register index range will be reserved for use by an implementation. Software should expect register definition changes in these registers from an implementation generation to the next implementation generation.

Major	Minor	Extension
2	15	0-7

Brief Summary

N12:

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
idr0	SDZ_CTL	2	15	0	207
idr1	N12MISC_CTL	2	15	1	210

N10:

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
idr0	SDZ_CTL	2	15	0	212
idr1	MISC_CTL	2	15	1	215

N9:

Simple Mnemonics	Symbolic Mnemonics	Major	Minor	Extension	Page*
idr0	SDZ_CTL	2	15	0	217
idr1	MISC_CTL	2	15	1	219

9.9.1. N12 Implementation

9.9.1.1. Structure Downsizing Control Register

Implementation: AndesCore™ N12
Mnemonic Name: SDZ_CTL
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 15, 0}

31 12	11 9	8 6	5 3	2 0
Reserved	BTBDZ	MTBDZ	DCDZ	ICDZ

This control register allows customers to emulate smaller cache structures on Andes hardcore. This control feature is intended to help customers to figure out the best cache configurations for their final products. Writing an un-assigned value into one of the defined fields of this register will result in UNPREDICTABLE structure size which is implementation-dependent.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>ICDZ</td><td>3
(2,0)</td><td>Instruction Cache downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 4-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / 2-way)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 4-way)</td></tr>
<tr><td>3</td><td>maximum size/4 (half set / 2-way)</td></tr>
<tr><td>4-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the following configurations when both Icache and Dcache are configured to 4-way:
<table>
<tr><th>Maximum size</th><th>Supported non-default values</th></tr>
<tr><td>16KB</td><td>1</td></tr>
</table></td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>32KB</td><td>1, 2, 3</td></tr>
<tr><td>64KB</td><td>1, 2, 3</td></tr>
</table>
The limited 16KB downsizing support is due to the minimum size-per-way constraint of 4KB.
</td><td></td><td></td></tr>
<tr><td>DCDZ</td><td>3
(5,3)</td><td>
Data Cache downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 4-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / 2-way)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 4-way)</td></tr>
<tr><td>3</td><td>maximum size/4 (half set / 2-way)</td></tr>
<tr><td>4-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the following configurations when both Icache and Dcache are configured to 4-way:
<table>
<tr><th>Maximum size</th><th>Supported non-default values</th></tr>
<tr><td>16KB</td><td>1</td></tr>
<tr><td>32KB</td><td>1, 2, 3</td></tr>
<tr><td>64KB</td><td>1, 2, 3</td></tr>
</table>
The limited 16KB downsizing support is due to the minimum size-per-way constraint of 4KB.
</td><td>RW</td><td>0</td></tr>
<tr><td>MTBDZ</td><td>3
(8,6)</td><td>
MTLB downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 4-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / 2-way)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 4-way)</td></tr>
<tr><td>3</td><td>maximum size/4 (half set / 2-way)</td></tr>
<tr><td>4-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the maximum size of 128, 64, and 32 entries.
When changing the value of this field, it is required that (1) the change can only be done when PSW.IT == 0 and PSW.DT == 0 and (2)
</td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>after the change all MTLB entries need to be invalidated before turning-on instruction/data address translation. If these rules are not followed, UNPREDICTABLE memory address may be generated for instruction fetch and data access.</td><td></td><td></td></tr>
<tr><td>BTBDZ</td><td>3
(11,9)</td><td>Branch Target Table downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 2-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / 1-way)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 2-way)</td></tr>
<tr><td>3</td><td>maximum size/4 (half set / 1-way)</td></tr>
<tr><td>4-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the maximum size of 256, 128, 64, and 32 entries.</td><td>RW</td><td>0</td></tr>
<tr><td>Reserved</td><td>20
(31,12)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

9.9.1.2. N12 Miscellaneous Control Register

Implementation: AndesCore™ N12
Mnemonic Name: N12MISC_CTL
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 15, 1}

31 ... 3	2	1	0
Reserved	PTEPF	RTP	BTB

This control register disables various structures on the Andes N12 core.

Field Name	Bits	Description	Type	Reset
BTB	1 (0)	Disable Branch Target Buffer. Value: 0 — Branch Target Buffer functionality is enabled. Value: 1 — Branch Target Buffer functionality is disabled. All branches will be predicted not taken in the early N12 pipeline stage.	RW	1
RTP	1 (1)	Disable Return Target Predictor. Value: 0 — Return Target Predictor functionality is enabled. Value: 1 — Return Target Predictor functionality is disabled.	RW	1
PTEPF	1 (2)	Disable Hardware Page Table Walker L2 PTE prefetch function. Value: 0 — HPTWK L2 PTE prefetch functionality is enabled.	RW	0

Field Name	Bits	Description		Type	Reset
		1	HPTWK L2 PTE prefetch functionality is disabled.		
Reserved	30 (31,3)	RAZWI		-	0

9.9.2. N10 Implementation

A particular field of these registers is defined only when the respective feature is configured.

9.9.2.1. Structure Downsizing Control Register

Implementation: AndesCore™ N10
Mnemonic Name: idr0 or SDZ_CTL
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 15, 0}

31 ... 12	11 ... 9	8 ... 6	5 ... 3	2 ... 0
Reserved	BTBDZ	MTBDZ	DCDZ	ICDZ

This control register allows customers to emulate smaller cache structures on Andes hardcore. This control feature is intended to help customers to figure out the best cache configurations for their final products. Writing an un-assigned value into one of the defined fields of this register will result in UNPREDICTABLE structure size which is implementation-dependent. A particular field is defined only when the respective feature is configured.

Note: N1003 does not have BTBDZ and MTBDZ fields defined.

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td>ICDZ</td><td>3
(2,0)</td><td>Instruction Cache downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 2-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / direct-mapped)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 2-way)</td></tr>
<tr><td>3-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the following</td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>configurations when both Icache and Dcache are configured to 2-way:
<table>
<tr><th>Maximum size</th><th>Supported non-default values</th></tr>
<tr><td>8KB</td><td>1</td></tr>
<tr><td>16KB</td><td>1, 2</td></tr>
<tr><td>32KB</td><td>1, 2</td></tr>
</table>
The limited 8KB downsizing support is due to the minimum size-per-way constraint of 4KB.</td><td></td><td></td></tr>
<tr><td>DCDZ</td><td>3
(5,3)</td><td>Data Cache downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 2-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / direct-mapped)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 2-way)</td></tr>
<tr><td>3-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the following configurations when both Icache and Dcache are configured to 2-way:
<table>
<tr><th>Maximum size</th><th>Supported non-default values</th></tr>
<tr><td>8KB</td><td>1</td></tr>
<tr><td>16KB</td><td>1, 2</td></tr>
<tr><td>32KB</td><td>1, 2</td></tr>
</table>
The limited 8KB downsizing support is due to the minimum size-per-way constraint of 4KB.</td><td>RW</td><td>0</td></tr>
<tr><td>MTBDZ</td><td>3
(8,6)</td><td>Note: N1003 does not have this field defined.
MTLB downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 4-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / 2-way)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 4-way)</td></tr>
<tr><td>3</td><td>maximum size/4 (half set / 2-way)</td></tr>
</table></td><td>RW</td><td>0</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>4-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the maximum size of 128, 64, and 32 entries.
When changing the value of this field, it is required that (1) the change can only be done when PSW.IT == 0 and PSW.DT == 0 and (2) after the change all MTLB entries need to be invalidated before turning-on instruction/data address translation. If these rules are not followed, UNPREDICTABLE memory address may be generated for instruction fetch and data access.
</td><td></td><td></td></tr>
<tr><td>BTBDZ</td><td>3
(11,9)</td><td>Note: N1003 does not have this field defined.
Branch Target Table downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 2-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / 1-way)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 2-way)</td></tr>
<tr><td>3</td><td>maximum size/4 (half set / 1-way)</td></tr>
<tr><td>4-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the maximum size of 256, 128, 64, and 32 entries.
</td><td>RW</td><td>0</td></tr>
<tr><td>Reserved</td><td>20
(31,12)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

9.9.2.2. Miscellaneous Control Register

Implementation: AndesCore™ N10
Mnemonic Name: idr1 or MISC_CTL
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 15, 1}

31 ... 4	3	2	1	0
Reserved	TCM	Reserved	RTP	BTB

This control register disables various structures on the Andes N10 core. A particular field is defined only when the respective feature is configured.

Note: N1003 does not have PTEPF and BTB fields defined.

Field Name	Bits	Description	Type	Reset
BTB	1 (0)	Note: N1003 does not have this field defined. Disable Branch Target Buffer. Value: 0 — Branch Target Buffer functionality is enabled. Value: 1 — Branch Target Buffer functionality is disabled. All branches will be predicted not taken in the early N10 pipeline stage.	RW	1
RTP	1 (1)	Disable Return Target Predictor. Value: 0 — Return Target Predictor functionality is enabled. Value: 1 — Return Target Predictor functionality is disabled.	RW	1
Reserved	1	RAZWI		0

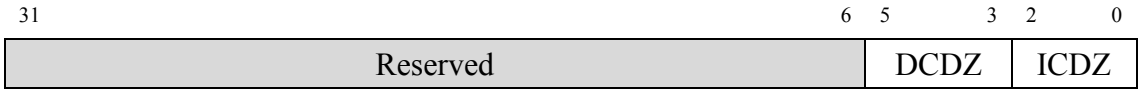
<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td>(2)</td><td></td><td></td><td></td></tr>
<tr><td>TCM</td><td>1
(3)</td><td>Note: N1003 does not have this field defined.
Disable Two Cycle Multiplication.<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Two cycle multiplication functionality is enabled.</td></tr><tr><td>1</td><td>Two cycle multiplication functionality is disabled.</td></tr></table></td><td>RW</td><td>0</td></tr>
<tr><td>Reserved</td><td>28
(31,4)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

9.9.3. N9 Implementation

A particular field of these registers is defined only when the respective feature is configured.

9.9.3.1. Structure Downsizing Control Register

Implementation: AndesCore™ N9
Mnemonic Name: idr0 or SDZ_CTL
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 15, 0}



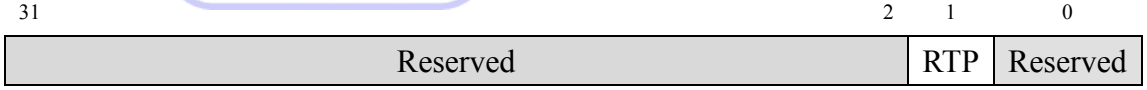
This control register allows customers to emulate smaller cache structures on Andes hardcore. This control feature is intended to help customers to figure out the best cache configurations for their final products. Writing an un-assigned value into one of the defined fields of this register will result in UNPREDICTABLE structure size which is implementation-dependent. A particular field is defined only when the respective feature is configured.

Field Name	Bits	Description	Type	Reset
ICDZ	3 (2,0)	Instruction Cache downsizing control. Value / Meaning 0: maximum size (full set / 2-way) 1: maximum size/2 (full set / direct-mapped) 2: maximum size/2 (half set / 2-way) 3-7: Unassigned/UNPREDICTABLE This control field works on the following configurations when both Icache and Dcache are configured to 2-way:	RW	0

<table>
<tr><th>Field Name</th><th>Bits</th><th>Description</th><th>Type</th><th>Reset</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Maximum size</th><th>Supported non-default values</th></tr>
<tr><td>8KB</td><td>1</td></tr>
<tr><td>16KB</td><td>1, 2</td></tr>
<tr><td>32KB</td><td>1, 2</td></tr>
</table>
The limited 8KB downsizing support is due to the minimum size-per-way constraint of 4KB.
</td><td></td><td></td></tr>
<tr><td>DCDZ</td><td>3
(5,3)</td><td>
Data Cache downsizing control.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>maximum size (full set / 2-way)</td></tr>
<tr><td>1</td><td>maximum size/2 (full set / direct-mapped)</td></tr>
<tr><td>2</td><td>maximum size/2 (half set / 2-way)</td></tr>
<tr><td>3-7</td><td>Unassigned/UNPREDICTABLE</td></tr>
</table>
This control field works on the following configurations when both Icache and Dcache are configured to 2-way:
<table>
<tr><th>Maximum size</th><th>Supported non-default values</th></tr>
<tr><td>8KB</td><td>1</td></tr>
<tr><td>16KB</td><td>1, 2</td></tr>
<tr><td>32KB</td><td>1, 2</td></tr>
</table>
The limited 8KB downsizing support is due to the minimum size-per-way constraint of 4KB.
</td><td>RW</td><td>0</td></tr>
<tr><td>Reserved</td><td>26
(31,6)</td><td>RAZWI</td><td>-</td><td>0</td></tr>
</table>

9.9.3.2. Miscellaneous Control Register

Implementation: AndesCore™ N9
Mnemonic Name: idr1 or MISC_CTL
Access Mode: Superuser
SR Encoding {Major, Minor, Extension}: {2, 15, 1}



This control register disables various structures on the Andes N9 core. A particular field is defined only when the respective feature is configured.

Field Name	Bits	Description	Type	Reset
Reserved	1 (0)	RAZWI	-	0
RTP	1 (1)	Disable Return Target Predictor. Value: 0 — Return Target Predictor functionality is enabled. Value: 1 — Return Target Predictor functionality is disabled.	RW	1
Reserved	30 (31,2)	RAZWI	-	0

9.10. Summary of System Register Existence

The following table summarizes the configuration condition(s) that determines the existence of each system register. For any implementation-dependent condition, please see the implementation data sheets for more details.

Simple Mnemonics	Symbolic Mnemonics	Existence configuration condition(s)
Configuration		
cr0	CPU_VER	Always exist
cr1	ICM_CFG	
cr2	DCM_CFG	
cr3	MMU_CFG	
cr4	MSC_CFG	
cr5	CORE_ID	Always exist except on implementations with (CPU_VER.CPUID == 0xC) && (CPU_VER[23:18] == 0x0)
cr6	FUCOP_EXIST	CPU_VER[3] == 1
Interruption		
ir0	PSW	Always exist
ir1	IPSW	
ir3	IVB	
ir4	EVA	
ir6	ITYPE	
ir9	IPC	
ir11	OIPC	
ir14	INT_MASK	
ir15	INT_PEND	
ir2	P_IPSW	MSC_CFG.INTLC != 1
ir5	P_EVA	
ir7	P_ITYPE	
ir10	P_IPC	

ir12	P_P0	
ir13	P_P1	
ir8	MERR	IM-dependent (Exists for both N12/N10 cores)
Memory Management		
mr0	MMU_CTL	always exist
mr1	L1_PPTB	MMU_CFG.MMPS == 2
mr4	TLB_MISC	
mr2	TLB_VPN	MMU_CFG.MMPS > 0
mr3	TLB_DATA	
mr5	VLPT_IDX	(MMU_CFG.MMPS == 2) & IM-dependent (Does not exist for both N12/N10 cores)
mr6	ILMB	ICM_CFG.ILMB != 0
mr7	DLMB	DCM_CFG.DLMB != 0
mr8	CACHE_CTL	(ICM_CFG.ISZ != 0) & (DCM_CFG.DSZ != 0)
mr9	HSMP_SADDR	MSC_CFG.HSMP == 1
mr10	HSMP_EADDR	Exists only on implementations with (CPU_VER.CPUID == 0xC) && (CPU_VER[23:18] == 0x0)
Performance Monitoring		
pfr0-pfr2	PFMCn	MSC_CFG.PFM == 1
pfr3	PFM_CTL	
Local Memory DMA		
dmar0	DMA_CFG	MSC_CFG.LMDMA == 1
dmar1	DMA_GCSW	
dmar2	DMA_CHNSEL	
dmar3	DMA_ACT	
dmar4	DMA_SETUP	
dmar5	DMA_ISADDR	
dmar6	DMA_ESADDR	
dmar7	DMA_TCNT	
dmar8	DMA_STATUS	
dmar9	DMA_2DSET	
dmar10	DMA_2DSCTL	

Resource Access Control		
	PRUSR_ACC_CTL	(MSC_CFG.LMDMA == 1) | (MSC_CFG.PFM == 1) except on implementations with (CPU_VER.CPUID == 0xC) && (CPU_VER[23:18] == 0x0)
fucpr	FUCOP_CTL	(CPU_VER[3] == 1) | (MSC_CFG[8:7] != 0)
Embedded Debug Module		
dr0+(n*5)	BPCn	MSC_CFG.EDM == 1 (The range of n is defined in EDM_CFG.BC)
dr1+(n*5)	BPAn	
dr2+(n*5)	BPAMn	
dr3+(n*5)	BPVn	
dr4+(n*5)	BPCIDn	
dr40	EDM_CFG	
dr41	EDMSW	
dr42	EDM_CTL	
dr43	EDM_DTR	
dr44	BMPTC	
dr45	DIMBR	
dr46	TECR0	MSC_CFG.TRACE == 1
dr47	TECR1	
Implementation-dependent		
idr0	SDZ_CTL	IM-dependent (Exists in N12 cores)
idr1	MISC_CTL	IM-dependent (Exists in N12 cores)

Chapter 10
AndesCore™ Optional Feature List

Some of the features in the Andes System Privilege Architecture are optional. This section lists the supporting of these optional features for the following implementations.

10.1. AndesCore™ N1213-S

Optional Features	AndesCore™ N12	Section
Virtual Linear Page Table Index register	no	9.4.6
LW_VLPT instruction / Double TLB miss exception	no	2.8.9
Machine Error Log register	yes	9.3.9
Instruction Local Memory	4KB-1MB	Chapter 6
Data Local Memory	4KB-1MB	Chapter 6
Local memory DMA	yes	Chapter 7
Local memory DMA Un-aligned External Address Transfer (UNEA)	yes	Chapter 7
Local memory DMA 2-Dimentional Element Transfer (2DET)	yes	Chapter 7
Performance Monitoring	yes	Chapter 5
EDM	yes	9.5
High Performance Memory Port	yes	Chapter 8
Hardware Page Table Walker	yes	2.1
Icache Locking support	yes	9.2.2
Dcache Locking support	no	9.2.3

I$/D$ Size	Way	Line size (Byte)	Lines/way	Downsizing support
8KB	2	16	256	No
	2	32	128	No
16KB	2	16	512	No
	2	32	256	No
	4	16	256	Yes (limited)
	4	32	128	Yes (limited)
32KB	2	16	1024	No

I$/D$ Size	Way	Line size (Byte)	Lines/way	Downsizing support
	2	32	512	No
	4	16	512	Yes
	4	32	256	Yes
64KB	4	16	1024	Yes
	4	32	512	Yes

Other detail information regarding N1213-S, please refer to N1213-S data sheet.

10.2. AndesCore™ N1033-S

Optional Features	Support	Section
Virtual Linear Page Table Index register	no	9.4.6
LW_VLPT instruction / Double TLB miss exception	no	2.8.9
Machine Error Log register	yes	9.3.9
Instruction Local Memory	1KB-1MB	Chapter 6
Data Local Memory	1KB-1MB	Chapter 6
Local memory DMA	yes	Chapter 7
Local memory DMA Un-aligned External Address Transfer (UNEA)	yes	Chapter 7
Local memory DMA 2-Dimentional Element Transfer (2DET)	yes	Chapter 7
Performance Monitoring	yes	Chapter 5
EDM	yes	9.5
High Performance Memory Port	no	Chapter 8
Hardware Page Table Walker	yes	2.1
Icache Locking support	yes	9.2.2
Dcache Locking support	no	9.2.3

I$/D$ Size	Way	Line size (Byte)	Lines/way	Downsizing support
4KB	1	16	256	No
	1	32	128	No
8KB	2	16	256	Yes (limited)
	2	32	128	Yes (limited)
16KB	2	16	512	Yes

I$/D$ Size	Way	Line size (Byte)	Lines/way	Downsizing support
	2	32	256	Yes
32KB	2	16	1024	Yes
	2	32	512	Yes

Other detail information regarding N1033-S, please refer to N1033-S data sheet.

10.3. AndesCore™ N903-S

Optional Features	Support	Section
Virtual Linear Page Table Index register	no	9.4.6
LW_VLPT instruction / Double TLB miss exception	no	2.8.9
Machine Error Log register	yes	9.3.9
Instruction Local Memory	1KB-1MB	Chapter 6
Data Local Memory	1KB-1MB	Chapter 6
Local memory DMA	no	Chapter 7
Local memory DMA Un-aligned External Address Transfer (UNEA)	no	Chapter 7
Local memory DMA 2-Dimentional Element Transfer (2DET)	no	Chapter 7
Performance Monitoring	no	Chapter 5
EDM	yes	9.5
High Performance Memory Port	no	Chapter 8
Hardware Page Table Walker	no	2.1
Icache Locking support	yes	9.2.2
Dcache Locking support	no	9.2.3

I$/D$ Size	Way	Line size (Byte)	Lines/way	Downsizing support
4KB	1	16	256	No
	1	32	128	No
8KB	2	16	256	Yes (limited)
	2	32	128	Yes (limited)
16KB	2	16	512	Yes

I$/D$ Size	Way	Line size (Byte)	Lines/way	Downsizing support
	2	32	256	Yes
32KB	2	16	1024	Yes
	2	32	512	Yes

Other detail information regarding N903-S, please refer to N903-S data sheet.

Proprietary Notice

Confidentiality status

This document is Confidential. This document has restriction on distribution.

Feedback on this Manual

If you have any problems with this Manual, Please contact Andes Technology Corporation by email at support@andestech.com
or on the Internet at www.andestech.com for support giving:

- 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 also welcome.