GOWIN
PROGRAMMING FOR THE FUTURE

Gowin RiscV AE250

User Guide

Revision History

Date	Version	Description
01/23/2019	1.0E	Initial version published.
01/21/2020	1.1E	● The number of UART, SPI, IIC, PIT increased; ● Enable AHB and APB interfaces; ● C BIN file prestorage added; ● ACE and AXI removed; ● The configuration options in chapter 5 updated; ● Figures and tables in chapter 4 to chapter 23 updated; ● IP name updated.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

The Gowin RiscV AE250 IP user guide helps users learn the features and usages of the Gowin RiscV AE250 IP product.

This guide targets application developers. It provides complete information on how to use the Gowin RiscV AE250 microcontroller memory and peripherals.

The AE250 is a family of microcontrollers with different memory sizes and peripherals. For ordering information, mechanical and electrical device characteristics please refer to the datasheets. For information on the RISC-V ISA, please refer to the RISC-V Instruction Set Manual.

1.2 Supported Products

The information in this guide applies to the following products:

1. GW2A series of FPGA products

2. GW2AR series of FPGA products

1.3 Related Documents

The latest user guides are available on GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:

1. DS102, GW2A series of FPGA Products Data Sheet

2. DS226, GW2AR series of FPGA Products Data Sheet

3. SUG100, Gowin YunYuan Software User Guide

The related documents are available on RISCV organization website:riscv.org：

1. The RISC-V Instruction Set Manual Volume I: User-Level ISA

2. The RISC-V Instruction Set Manual Volume II: Privileged Architecture

1.4 Abbreviations and Terminology

The abbreviations and terminology used in this guide are shown in Table 1-1 below.

Table 1-1 Abbreviations and Terminology

Abbreviations and Terminology	Meaning
IPIPE	Integer Pipeline
RF	Register File
CSR	Control and Status Register
ALU	Arithmetic Logic Unit
MDU	Multiplication and Division Unit
FASTMUL	Fast Multiplier
IFU	Instruction Fetch Unit
ICU	Instruction Cache Unit
LSU	Load Store Unit
DCU	Data Cache Unit
ILM	Instruction Local Memory Controller
DLM	Data Local Memory Controller
BIU	Bus Interface Unit
MMU	Memory Manager Unit
TRIGM	Trigger Module
FPU	Floating Point Unit
ACE	Andes Custom Extension

1.5 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly using the information provided below.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Overview

The Gowin RiscV AE250 IP is a 32-bit CPU IP core based on AndeStar™ V5 architecture which incorporated RISC-V technology. It is capable of delivering high per-MHz performance and operating at high frequencies, at the same time it is small in gate count. This Gowin IP also supports single and double precision floating point instructions and half precision load/store. The Gowin RiscV AE250 IP comes with options, including branch prediction for efficient branch execution, Instruction and Data caches, Local Memories for low-latency accesses, and ECC for L1 memory soft error protection.

The Gowin RiscV AE250 IP's 5-stage pipeline is optimized for high operating frequency and small gate count. Features also includes PLIC and vectored interrupts for serving various types of system events, AHB 32-bit bus, Power Brake and WFI mode for low power and power management, and JTAG debug interface for development support.

Table 2-1 Gowin RiscV AE250

Gowin RiscV AE250	
IP Core Application	
Device Support	GW2A、GW2AR Series
Logic Resource	See Table 3-1
Delivery	
Design Kit	Verilog (encrypted)
Reference Design	Verilog
TestBench	---
Verification	
Systhesis	Synplify_Pro
IDE	GoWinYunYuan

3 Features and Performance

3.1 Features

- AndeStar™ V5 Instruction Set Architecture (ISA), compliant to RISC-V technology
- Floating point extensions (N25F)
- Andes extensions, architected for performance and functionality enhancements
- Separately licensable Andes Custom Extension™ (ACE) for customized acceleration[1]
- 32-bit, 5-stage pipeline CPU architecture 16/32-bit mixable instruction format for compacting code density
- Branch predication to speed up control code
- Return Address Stack (RAS) to speed up procedure returns
- Physical Memory Protection (PMP)
- Flexibly configurable Platform-Level Interrupt Controller (PLIC) for supporting wide range of system event scenarios
- Enhancement of vectored interrupt handling for real-time performance
- Advanced CoDense™ technology to reduce program code size

Note !

The current version does not support ACE.

3.2 Resources Utilization

RTL code is realized by Verilog language. Performance and resource utilization may vary depending on the density, speed, and grade of devices used.

Take Gowin GW2A-18 series of FPGA as an example. The resource utilization is shown in Table 3-1. For application verification on other Gowin FPGA, please pay attention to the information released later.

Table 3-1 Gowin RiscV AE250 Resources Utilization

Device	Speed Grade	Resources	Usage	Remarks
GW2A-18	-5	LUT	10K	-
		ALU	4K	
		REG	6K	
		BSRAM	0-max	

3.3 The Clock Frequency

There are no special requirements in this IP. GW2A-18 chips in the default conditions can run at 30MHz.

4 CPU Core Overview

4.1 N25(F) Processor Features

The features of N25(F) processor are as follows:

CPU Core

- 5-stage in-order execution pipeline
- Hardware multiplier
  radix-2/radix-4/radix-16/radix-256/fast
- Hardware divider
- Optional branch prediction
  Static branch prediction, or * 4-entry return address stack (RAS)
- Dynamic branch prediction
  - * 32/64/128/256-entry branch target buffer (BTB)
  - * 256-entry branch history table
  - * 8-bit global branch history
  - * 4-entry return address stack (RAS)
- Machine mode and supervisor mode and User mode
- Optional performance monitors
- Misaligned memory access
- RISC-V physical memory protection

AndeStar V5 ISA

- RISC-V RV32I base integer instruction set
- RISC-V RVC standard extension for compressed instructions
- RISC-V RVM standard extension for integer multiplication and division
- Optional RISC-V RVA standard extension for atomic instructions
- Optional RISC-V F and D single/double precision floating point
- Andes Performance extension
- Andes CoDense™ extension

Memory Subsystem

- I & D caches
  - Cache size: 8KiB/16KiB/32KiB/64KiB
  - Cache line size: 32B
  - Set associativity: Direct-mapped/2-way/4-way
  - Custom cache control operation through CSR read/write
- I & D local memories
  - Size: 4KiB to 16MiB
  - Optional local memory slave port
  - Interface: RAM or AHB-Lite
- Memory subsystem soft-error protection
  - Protection scheme: parity-checking or error-checking-and-correction (ECC)
  - Automatic hardware error correction
  - Protected memories:

    a) * I & D cache tag RAM and data RAM

    b) * I & D local memories

Bus

- Interface Protocol
  Synchronous AHB (32-bit data width)

Debug

- RISC-V External Debug Support Version 0.13
- Configurable number of breakpoints: 2/4/8
- External debug module with AHB interfaces
- External JTAG debug transport module
  JTAG: IEEE Std 1149.1 style 4-wire JTAG interface

Power Management

Wait-for-interrupt (WFI) mode

AndeStar Extension

- StackSafe hardware stack protection extension
- PowerBrake simple power/performance scaling extension
- Custom performance counter events

Platform-Level Interrupt Controller (PLIC)

- AHB interface
- Configurable number of interrupts: 1–1023

- Configurable number of interrupt priorities: 3/7/15/31/63/127/255
- Configurable number of targets: 1–16
- Andes Vectored Interrupt extension

Trace

Optional instruction trace

4.2 Block Diagram

Figure 4-1 N25(F) Block Diagram
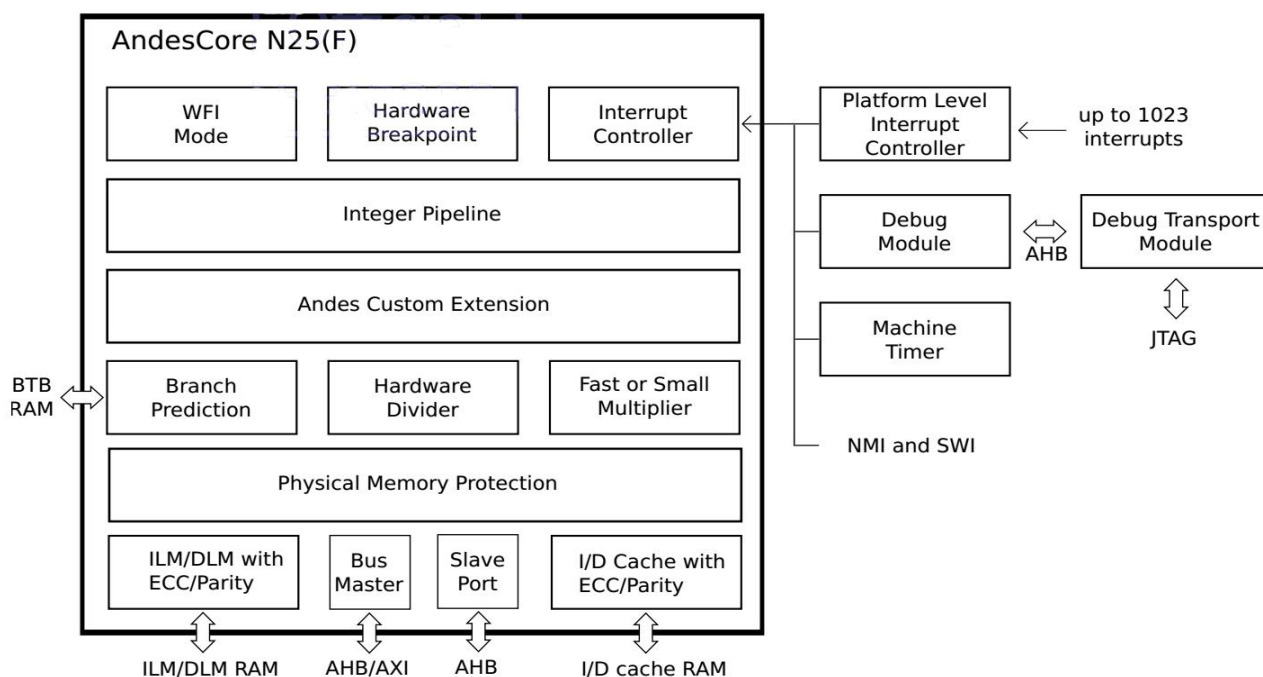
4.3 Major Components

The following table describes the major components of the N25(F) processor.

Table 4-1 Major Components

Abbreviations and Terminology	Meaning
IPIPE	Integer Pipeline
RF	Register File
CSR	Control and Status Register
ALU	Arithmetic Logic Unit
MDU	Multiplication and Division Unit
FASTMUL	Fast Multiplier
IFU	Instruction Fetch Unit
ICU	Instruction Cache Unit
LSU	Load Store Unit
DCU	Data Cache Unit

Abbreviations and Terminology	Meaning
ILM	Instruction Local Memory Controller
DLM	Data Local Memory Controller
BIU	Bus Interface Unit
MMU	Memory Manager Unit
TRIGM	Trigger Module
FPU	Floating Point Unit

4.4 Pipeline Stages and Activities

N25(F) implements a five-stage pipeline architecture. The following figure shows the pipeline stages of the processor.

Figure 4-2 Pipeline Stages and Activities
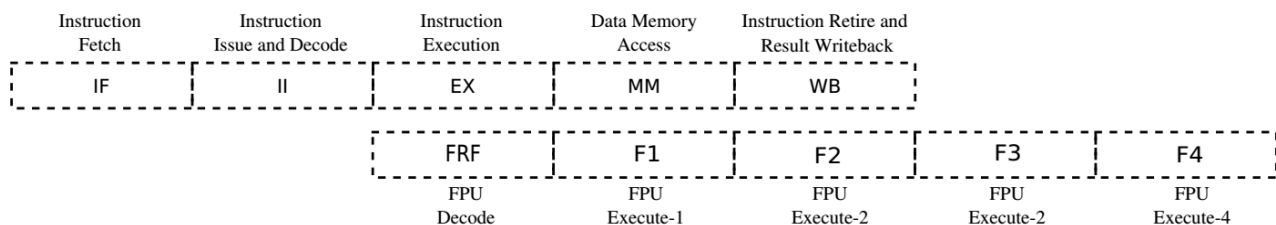
The pipeline activities of the corresponding stages are:

IF—Instruction Fetch

● Fetching an instruction word from ILM/Instruction Cache/Bus
● Dynamic branch prediction

II—Instruction Decode and Issue

● 16/32-bit instruction alignment
● Instruction decoding
● Register file read
● Resolving data dependency
● Static branch prediction

EX—Instruction Execution

● ALU instruction execution
● Load/Store address generation

MM—Memory Access

● DLM/Data Cache access
● Division instruction execution
● Multiplication instruction execution
● Branch resolution

WB — Instruction Retire and Result Write-Back

- Interrupt resolution
- Instruction retire
- Register file write back
- ILM access
- Bus access

FRF — FPU Instruction Decode

- Instruction decoding
- Register file read

F1~F4 — FPU Instruction Execution

- Float-point arithmetic execution
- Data Exchange between Integer/FPU pipelines

4.5 Design Hierarchy

4.5.1 AHB Platform

The design hierarchy of AE250 is illustrated in Figure 4-3. The top of N25(F) processor design is vc_core. The vc_core design does not include any SRAM cells. All of the required SRAM cells are expected to be instantiated outside of vc_core to simplify the integration effort.

The ae250_chip module is a reference AHB platform that instantiates the peripheral IPs. The ae250_uncore module is a reference subsystem that instantiates N25(F) processor and all the required memory modules. These two modules are open for modification to meet system requirements.

To support all bus interface configures of the processor, ae250_uncore contains several optional adapter modules. The ae250_cpu_to_ahb32 module converts the 64-bit AHB interface to the 32-bit AHB interface. It is an optional module and could be removed when the type of the AE250 bus interface is 32-bit AHB. The ae250_slv_to_lm module contains glue logics connected to the slave port of vc_core.

The ae250_debug_subsystem, nceplmt100, and nceplic100 modules are platform IPs as specified in the RISC-V architecture for proper operation of a RISC-V system. The ae250_debug_subsystem module implements the debug functionality. The nceplmt100 implements the RISC-V machine timer, and nceplic100 implements the RISC-V platform interrupt controller (PLIC).

The PLIC module is instantiated twice, one (u_plic) for arbitrating

interrupts from peripheral devices, and the other (u_plic_sw) for supporting software interrupts. The u_plic_sw instantiation only needs to use the programmability of the PLIC registers to generate (software programmable) interrupts, so all its interrupt sources are tied to zero.

The atcbmc200, atcspi200, ae250_smu, atcrtc100, atcapbbrg100, atcwdt200, atciic100, atcpit100, atcuart100, and atcgpio100 modules are pre-integrated Andes platform/peripheral IPs that may be shipped together with the AE250 package, depending on licensing agreements.

Figure 4-3 Design Hierarchy
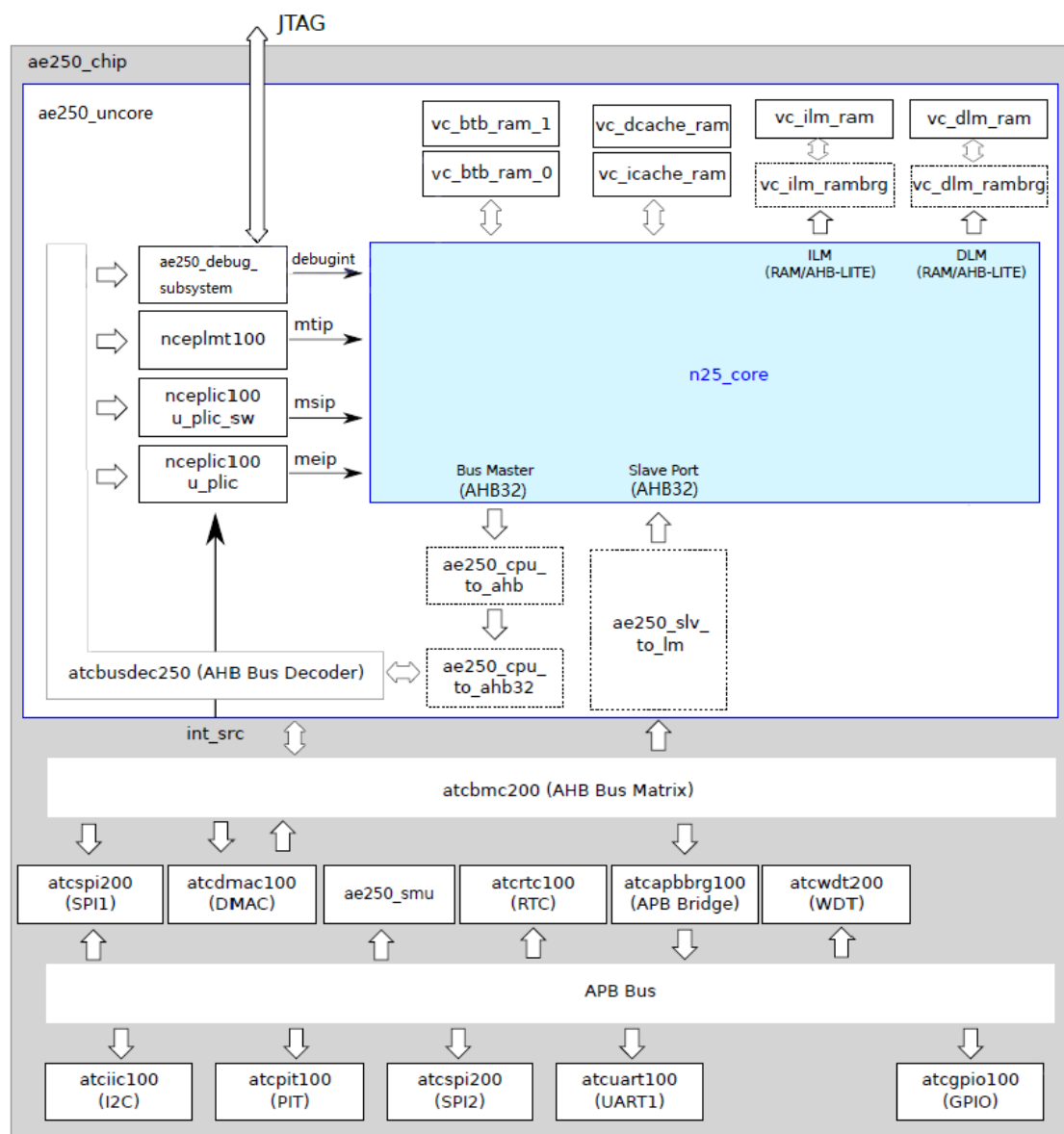
5 Processor Configuration Options

The configuration of the AE250 processor is through a config tool and Verilog parameters.

5.1 CPU Core Configuration

Figure 5-1 CPU Core Configuration GUI

Note！

The screenshot only serves to show how the tool looks like to facilitate the description above. It is not a goal for this figure to show the most updated version of the tool. The actual content of the tool may differ slightly as the tool gets updated with new features added to AE250.

5.1.1 ISA

RISC-V User-Level Interrupt Extension

Specifying this option to "yes" enables the RISC-V User-Level Interrupt Extension (RVN). RVN supports user-level trap handling.

RISC-V Atomic Instruction Extension

Specifying this option to "yes" enables the RISC-V "A" Standard Extension for Atomic Instructions (RVA). RVA includes instructions that atomically read-modify-write memories for synchronization between multiple hardware threads (harts)

RISC-V P-extension (draft) DSP/SIMD ISA

Specifying this option to "yes" enables the RISC-V DSP (Digital Signal Processing) instruction. The DSP instruction includes all the instructions of AndeStar DSP ISA Extension.Boost the performance of voice, audio, image and signal processing.

RISC-V Floating-Point Instruction Extension

This option determines the floating-point extension type.

- none: no floating-point extension
- single precision: "F" standard extension for single precision floating-point instruction
- double+single precision: "F" and "D" standard extension for single and double precision floatingpoint instruction

Note！

The floating-point extension is available depending on AE250 licensing agreements.

5.1.2 Privilege Architecture

Privilege Modes

The option determines the number of supported privileges levels. The AE250 processor supports most 3 privilege levels

Table 5-1 Supported Combination of Privilege Modes

Number of Levels	Supported Modes	Intended Usage
1	Machine	Simple embedded systems
2	Machine,User	Secure embedded systems
3	Machine,User,Supervisor	Systerm running Unix-like operating systems

The RISC-V privilege architecture specifies four privilege levels as shown in Table 5-2. Machine mode (M-mode) has the highest privileges and is the mandatory privilege level for a RISC-V hardware platform. User mode (U-mode) restricts privileges to protect against incorrect or malicious application codes. And Supervisor mode (S-mode) is provided for Unix-like operating systems with address translating and protection requirements.

Table 5-2 RISC-V Privilege Levels

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	Reserved	
3	11	Machine	M

Performance Monitors

Specifying this option to "yes" enables hardware performance monitors, including mcycle and minstret Control and Status Registers (CSR). The mcycle register counts the elapsed clock cycles while the minstret register counts the number of retired instructions.

Number of Shared TLB Entries

● This option selects the number of supported Shared TLB entries
   Number of iTLB Entries
● This option selects the number of supported iTLB entries
   Number of dTLB Entries
● This option selects the number of supported dTLB entries
   Physical Memory Protection
● This option selects the number of supported PMP entries.
   Performance Monitors

Specifying this option to "yes" enables hardware performance monitors, including mcycle and minstret Control and Status Registers (CSR). The mcycle register counts the elapsed clock cycles while the minstret register counts the number of retired instructions.

Misaligned Memory Access

AE250 implements the misaligned memory access to support accessing misaligned addresses without triggering any misaligned exceptions.

Andes Vectored PLIC Extension

Specifying this option to "yes" enables the Andes Vectored PLIC extension for reducing interrupt latency.

Andes StackSafe Extension

Specifying this option to "yes" enables the StackSafe hardware stack protection extension.

The extension is a hardware mechanism for tracking and guarding against the stack pointer overflows/underflows.

Andes PowerBrake Extension

Specifying this option to "yes" enables the PowerBrake power/performance scaling extension.

The PowerBrake extension throttles performance by reducing instruction executing rate instead of slowing down clock frequency. The performance and hence power consumption can be switched to a different level in a couple of instructions. This is an ultra-low latency mechanism for performance & power scaling, as compared to the latencies of frequency scaling through PLL programming.

Andes CoDense Support

Specifying this option to "yes" enables CoDense Support.

5.1.3 Bus Interface

AHB Low Latency Support

The Low Access-Latency AHB Bus option reduces the system bus access latency at the expense of maximum frequency (around 70% of configurations without this option). This option is expected to be used under configurations where there are no caches.

5.1.4 Micro Architecture

Multiplier Implementation

This option selects the implementation of the hardware multiplier in AE250. Valid values and the respective performance are:

- radix2: 1-bit/cycle
- radix4: 2-bit/cycle
- radix16: 4-bit/cycle
- radix256: 8-bit/cycle
- fast: two-stage pipelined.

The radix multiplier options realize the multiplier through serial additions, while the fast multiplier option implements a two-stage pipelined parallel multiplier.

Fast Multiplier Algorithm

This option selects the type of fast multiplier algorithm.

Multiplier Accelerator Type

This option selects the type of multiplier Accelerator.

Branch Prediction

This option selects the branch prediction scheme. The AE250 processor supports both static and dynamic branch prediction, and the number of BTB entries is configurable.

The static branch prediction algorithm predicts that all backward branches will be taken and all forward branches will not.

The dynamic branch prediction algorithm uses a 2-way branch target buffer (BTB) and a branch history table to predict the target address and direction (taken/not-taken) of each branch.

5.1.5 Trace Interface

Specifying this option to "instruction" enables instruction trace interface.

5.1.6 Device Regions

Device regions are memory locations where accesses to the location may cause side effects. Accesses to device regions are non-cacheable and strong-ordered. Memory operations to these regions will be conservative without the need of excessive FENCE instructions. At most eight device regions could be specified in AE250. A device region n (n = 0–7) is specified by two parameters:

- DEVICE_REGIONn_BASE: the base address of the region,
- DEVICE_REGIONn_MASK: the address mask of the region.

An address is inside a region when:

(address & DEVICE_REGIONn_MASK) == DEVICE_REGIONn_BASE.

A device region can be disabled by setting DEVICE_REGIONn_BASE to all ones and DEVICE_REGIONn_MASK to all zeros.

Device regions and ILM/DLM/WRITETHROUGH should not overlap with each other. The behavior is undefined when device regions are overlapped with ILM/DLM/WRITETHROUGH.

5.1.7 Write-through Regions

Write-through regions are memory locations where memory writes will update both D-Cache and the system memory with write-no-allocate policy.

At most eight write-through regions could be specified. A write-through region n (n = 0–7) is specified by two parameters:

- WRITETHROUGH_REGIONn_BASE: the base address of the region,
- WRITETHROUGH_REGIONn_MASK: the address mask of the region.

An address is inside a region when:

(address & WRITETHROUGH_REGIONn_MASK) == WRITETHROUGH_REGIONn_BASE.

A write-through region can be disabled by setting WRITETHROUGH_REGIONn_BASE to all ones and WRITETHROUGH_REGIONn_MASK to all zeros.

Write-through regions and ILM/DLM/DEVICE regions should not overlap with each other. The behavior is undefined when write-through regions are overlapped with ILM/DLM/DEVICE regions.

5.2 Local Memory Configuration

5.2.1 Instruction Local Memory

ILM Size

The ILM Size option selects the size of the instruction local memory in KiB. Only power-of-2 sizes are supported. Specifying 0 to the option unconfigures the instruction local memory.

ILM Base

ILM_BASE is a Verilog parameter for specifying the base address of the instruction local memory, and it must be aligned to the size of the instruction local memory.

ILM Soft Error Protection

ILM Soft Error Protection option selects the soft-error protection scheme for the instruction local memory:

- none: no protection
- parity: single-error detection
- ecc: single-error correction, and double-error detection

ILM Initialize File

To make the executable files loadable , user must provide binary data, and then converted to the ASCII format readable by the $readmemh() SystemVerilog system task.

5.2.2 Data Local Memory (DLM)

DLM Size

The DLM Size option selects the size of the data local memory in KiB. Only power-of-2 sizes are supported. Specifying 0 to DLM_SIZE_KB unconfigures the data local memory.

DLM Base

DLM_BASE is a Verilog parameter for specifying the base address of the data local memory, and it must be aligned to the size of the data local memory.

DLM Soft Error Protection

DLM Soft Error Protection option selects the soft-error protection scheme for the data local memory:

- none: no protection
- parity: single-error detection
- ecc: single-error correction, and double-error detection

5.2.3 Slave Port

The interface protocol is AHB, and the data width is identical to bus data width. The value of parameter SLAVE_PORT_DATA_WIDTH is identical to the value of parameter BIU_DATA_WIDTH.

Slave Port Support

Specifying this option to "yes" enables the local memory slave port. The local memory slave port enables bus masters to access the local memories of AE250.

Slave Port Base Address

Slave Port Base Address is a Verilog parameter for specifying the base address of the local memory slave port, and it must be aligned to the size of the local memory.

Slave Port Space Size

The Slave Port Space Size option selects the size of the data local memory in MiB.

5.3 Cache Configuration

5.3.1 Cache

Cache Line Size

The option cache line size only support 32 bytes.

NON_CRITICAL_WORD_FIRST

Specifying this option to "yes" support the non-critical-word-first.

LSU_PREFETCH

Specifying this option to "yes" support the LSU prefetch.

5.3.2 Instruction Cache

I-Cache Size

The I-Cache Size option selects the size of the instruction cache in KiB. Only power-of-2 sizes are supported. Specifying 0 to the option unconfigures the instruction cache.

I-Cache Replacement Policy

The I-Cache Replacement Policy option selects the replacement policy of instruction cache on cache misses.

I-Cache Associativity

The I-Cache Associativity option selects the associativity of instruction cache.

I-Cache Soft Error Protection

The I-Cache Soft Error Protection option selects the soft-error protection scheme for the instruction cache:

● none: no protection
● parity: single-error correction
● ecc: single-error and double-error correction

5.3.3 Data Cache

D-Cache Size

The D-Cache Size option selects the size of the data cache in KiB. Only power-of-2 sizes are supported. Specifying 0 to the option unconfigures the data cache.

D-Cache Replacement Policy

The D-Cache Replacement Policy option selects the replacement policy of data cache when cache miss.

D-Cache Associativity

The D-Cache Associativity option selects the associativity of data cache.

D-Cache Soft Error Protection

The D-Cache Soft Error Protection option selects the soft-error protection scheme for the data cache:

- none: no protection
- parity: single-error detection
- ecc: single-error correction, and double-error detection

5.4 Debug Configuration

5.4.1 Debug Support

Specifying this option to "yes" enables debug support.

5.4.2 DEBUG_VEC

DEBUG_VEC defines the entry point address of the exception handler for servicing debug exceptions in the debug mode. It should be set to the address of the Debug ROM of the NCEPLDM200 debug module, which is also the base address of the module. This address space should be a device region for proper operation of the external debug support. See **Debug** Subsystem Configuration Options

Please see Section 5.4 for the description of Debug Subsystem Configuration Options.

NCEPLDM200 for more information.

The AE250 memory map assigns the base address of NCEPLDM200 to 0xe68000000.

5.4.3 Number of Trigger

The Number of Trigger option selects the number of hardware breakpoints.

5.4.4 Program Buffer Size

The Program Buffer Size option selects the size of program buffer. NCEPLDM200 supports program buffer 0–7 only. Program buffer 8–15 are hardwired to 32'h00100073 (the EBREAK instruction).

5.4.5 System Bus Access Support

The option determines the the system interface port of debug systerm.

6 Signals Description

The interface ports of the AE250 core are described in this section. All signals are Active-High unless otherwise indicated.

6.1 General Signals

Table 6-1 General Signals

Signal Name	Direction	Description
hart_id	input	This signal indicates the CPU hart ID. Hart IDs might not necessarily be numbered contiguously in a multiprocessor system, but at least one hart must have a hart ID of zero.
core_reset_n	input	CPU reset (Active-Low)
core_clk	input	CPU clock input
reset_vector[31:0]	input	Default program counter value upon reset. It should normally be a 4-byte aligned value but 2-byte aligned value is also allowed; bit 0 of this input signal should be zero.
bus_clk_en	input	This is a core_clk clock domain signal indicating that the data from the bus clock domain can be sampled at the coming rising edge of core_clk.
core_wfi_mode	input	This signal indicates that the processor is in the wait-for-interrupt mode.

6.2 Interrupt Signals

Table 6-2 Interrupt Signals

Signal Name	Direction	Description
meip	output	External interrupt acknowledgment, used in the vector interrupt mode
meiid[9:0]	input	Timer interrupt pending
meiack	input	Software interrupt pending

Signal Name	Direction	Description
mtip	input	Non-maskable interrupt
msip	input	Supervisor-mode external interrupt pending
nmi	input	Non-maskable interrupt
seip	input	Supervisor-mode external interrupt pending

6.3 Debug Signals

Table 6-3 Debug Signals

Signal Name	Direction	Description
debugint	output	Debug interrupt
hart_unavail	input	This signal indicates that the processor is not available for accesses by the external debugger. The processor could be in the reset or some kind of power-gating state.
hart_halted	input	This signal indicates that the processor is halted.
hart_under_reset	input	This signal indicates that the processor is under reset.
stoptime	input	This signal indicates that the processor is in Debug Mode and timers should stop counting. This signal is controllable by dcsr.STOPTIME.

6.4 Trace Signals

Table 6-4 Trace Signals

Signal Name	Direction	Description
trace_ivalid	output	This signal indicates that an instruction has retired or trapped (exception).
trace_iexception	output	This signal indicates that an instruction has trapped (exception).
trace_cause	output	This signal indicates the cause of an exception
trace_tval	output	This signal indicates the exception data
trace_interrupt	output	This signal indicates that the processor is under reset.
trace_iaddr	output	This signal indicates the cause of an exception
trace_instr	output	This signal indicates the instruction
trace_priv	output	This signal indicates the privilege mode during execution

6.5 AHB Interface Signals

AHB interface signals provide connectivity to the AHB system bus. These signals are used only when BIU_BUS is ahb. Otherwise, they are left unconnected. They are sampled/driven in the core_clk domain on

bus_clk_en pulses.

Table 6-5 AHB Interface Signals

Signal Name	Direction	Description
hgrant	input	Bus grant
hrdata[BIU_DATA_WIDTH-1:0]	input	Read data bus
hready	input	Transfer done
hresp[1:0]	input	Transfer response
haddr[31:0]	output	Address bus
hburst[2:0]	output	Burst type
hbusreq	output	Bus request
hlock	output	Locked transfer
hprot[3:0]	output	Protection control
hsize[2:0]	output	Transfer size
htrans[1:0]	output	Transfer type
hwdata[BIU_DATA_WIDTH-1:0]	output	Write data bus
hwrite	output	Transfer direction

6.6 Instruction Local Memory Interface Signals

ILM interface signals provide connectivity to the Instruction Local Memory. Two interface types, ram and ahb-lite, could be configured through the Local Memory Interface. If the value of option is ram, SRAM style interface is configured. If the value is ahb-lite, the AHB-Lite interface will be used.

The SRAM style interface signals provide connectivity to the Instruction Local Memory RAM of the processor. These signals are present on the processor interface when ILM is configured and the ram type of Local Memory Interface is selected.

Table 6-6 Instruction Local Memory Interface Signals

Signal Name	Direction	Description
ilm_cs	output	Chip select
ilm_byte_we[3:0]	output	Write enable
ilm_addr[ILM_RAM_AW-1:0]	output	Address
ilm_wdata[ILM_RAM_DW-1:0]	output	Write data
ilm_rdata[ILM_RAM_DW-1:0]	input	Read data

Table 6-7 Instruction Local Memory Address Bit-Width

Size (KiB)	ILM_RAM_AW
4	10
8	11
16	12
32	13
64	14
128	15
256	16
512	17
1024	18
2048	19
4096	20
8192	21
16384	22

Table 6-8 Instruction Local Memory Data Bit-Width

Protection Scheme	ILM_RAM_DW
none	32
parity	36
ecc	39

Table 6-9 ILM Byte Write Enable Mapping

BWE Bit	Protection Scheme: none	Protection Scheme: parity	Protection Scheme: ecc
0	wdata[7:0]	{wdata[32], wdata[7:0]}	wdata[38:0]
1	wdata[15:8]	{wdata[33], wdata[15:8]}	wdata[38:0]
2	wdata[23:16]	{wdata[34], wdata[23:16]}	wdata[38:0]
3	wdata[31:24]	{wdata[35], wdata[31:24]}	wdata[38:0]

6.7 Data Local Memory Interface Signals

DLM interface signals provide connectivity to the DATA Local Memory. Two interface types, ram and ahb-lite, could be configured through the Local Memory Interface. If the value of option is ram, SRAM style interface is configured. If the value is ahb-lite, the AHB-Lite interface will be used.

The SRAM style interface signals provide connectivity to the Instruction Local Memory RAM of the processor. These signals are present on the processor interface when DLM is configured and the ram type of Local Memory Interface is selected.

Table 6-10 Data Local Memory Interface Signals

Signal Name	Direction	Description
dlm_cs	output	Chip select
dlm_byte_we[3:0]	output	Write enable
dlm_addr[DLM_RAM_AW-1:0]	output	Address
dlm_wdata[DLM_RAM_DW-1:0]	output	Write data
dlm_rdata[DLM_RAM_DW-1:0]	input	Read data

Table 6-11 Data Local Memory Address Bit-Width

Size (KiB)	DLM_RAM_AW
4	10
8	11
16	12
32	13
64	14
128	15
256	16
512	17
1024	18
2048	19
4096	20
8192	21
16384	22

Table 6-12 Data Local Memory Data Bit-Width

Protection Scheme	DLM_RAM_DW
none	32
parity	36
ecc	39

Table 6-13 DLM Byte Write Enable Mapping

BWE Bit	Protection Scheme: none	Protection Scheme: parity	Protection Scheme: ecc
0	wdata[7:0]	{wdata[32], wdata[7:0]}	wdata[38:0]
1	wdata[15:8]	{wdata[33], wdata[15:8]}	wdata[38:0]
2	wdata[23:16]	{wdata[34], wdata[23:16]}	wdata[38:0]
3	wdata[31:24]	{wdata[35], wdata[31:24]}	wdata[38:0]

6.8 Instruction Cache Interface Signals

I-Cache interface signals provide connectivity to the I-Cache SRAMs of the processor. These signals are always present on the processor interface but they are used only when I-Cache is configured. Otherwise, they are left unconnected.

Table 6-14 Instruction Cache Interface Signals

Signal Name	Direction	Description
icache_disable_init	input	Disable the initialization of I-Cache RAMs when the processor is out of reset. Assertion of this signal is to speed up the power-gating wake up process when the content of I-Cache SRAM is preserved during power-down.
icache_tag$N$_cs	output	Chip select, N=0, 1, 2, 3
icache_tag$N$_we	output	Write enable
icache_tag$N$_addr[ICACHE_TAG_RAM_AW-1:0]	output	Address
icache_tag$N$_wdata[ICACHE_TAG_RAM_DW-1:0]	output	Write data
icache_tag$N$_rdata[ICACHE_TAG_RAM_DW-1:0]	input	Read data
icache_data$N$_cs	output	Chip select, N=0, 1, 2, 3
icache_data$N$_we	output	Write enable
icache_data$N$_addr[ICACHE_DATA_RAM_AW-1:0]	output	Address
icache_data$N$_wdata[ICACHE_DATA_RAM_DW-1:0]	output	Write data
icache_data$N$_rdata[ICACHE_DATA_RAM_DW-1:0]	input	Read data

Table 6-15 I-Cache Tag Address Bit-Width

Associativity	Size(KiB)	ICACHE_TAG_AW
1	8	8
	16	9
	32	10
	64	11
2	8	7
	16	8
	32	9
	64	10
4	8	6
	16	7
	32	8
	64	9

Table 6-16 I-Cache Tag Data Bit-Width

Protection Scheme	Associativity	Size (KiB)	ICACHE_TAG_RAM_DW
none	1	8	23
		16	23
		32	23
		64	23
	2	8	23
		16	23
		32	23
		64	23
	4	8	24
		16	23
		32	23
		64	23
parity	1	8	27
		16	27
		32	27
		64	27
	2	8	27
		16	27
		32	27
		64	27
	4	8	28
		16	27
		32	27
		64	27

Protection Scheme	Associativity	Size (KiB)	ICACHE_TAG_RAM_DW
ecc	1	8	30
		16	30
		32	30
		64	30
	2	8	30
		16	30
		32	30
		64	30
	4	8	31
		16	30
		32	30
		64	30

Table 6-17 I-Cache Data Address Bit-Width

Associativity	Size(KiB)	ICACHE_DATA_AW
1	8	11
	16	12
	32	13
	64	14
2	8	10
	16	11
	32	12
	64	13
4	8	9
	16	10
	32	11
	64	12

6.9 Data Cache Interface Signals

D-Cache interface signals provide connectivity to D-Cache SRAMs of the processor. These signals are always present on the processor interface but they are used only when D-Cache is configured. Otherwise, they are left unconnected.

Table 6-18 Data Cache Interface Signals

Signal Name	Direction	Description
dcache_disable_init	input	Disable the initialization of D-Cache RAMs when the processor is out of reset. Assertion of this signal is to speed up the power-gating wake up process when the content of D-Cache SRAM is preserved during power-down.
dcache_tagN_cs	output	Chip select, N=0, 1, 2, 3
dcache_tagN_we	output	Write enable
dcache_tagN_addr[DCACHE_TAG_RAM_AW-1:0]	output	Address
dcache_tagN_wdata[DCACHE_TAG_RAM_DW-1:0]	output	Write data
dcache_tagN_rdata[DCACHE_TAG_RAM_DW-1:0]	input	Read data
dcache_dataN_cs	output	Chip select, N=0, 1, 2, 3
dcache_dataN_we[DCACHE_DATA_RAM_BWEW-1:0]	output	Write enable
dcache_dataN_addr[DCACHE_DATA_RAM_AW-1:0]	output	Address
dcache_dataN_wdata[DCACHE_DATA_RAM_DW-1:0]	output	Write data
dcache_dataN_rdata[DCACHE_DATA_RAM_DW-1:0]	input	Read data

Table 6-19 D-Cache Tag Address Bit-Width

Associativity	Size(KiB)	DCACHE_TAG_AW
1	8	8
	16	9
	32	10
	64	11
2	8	7
	16	8
	32	9
	64	10
4	8	6
	16	7
	32	8
	64	9

Table 6-20 D-Cache Tag Data Bit-Width

Protection Scheme	Associativity	Size (KiB)	DCACHE_TAG_RAM_DW
none	1	8	22
		16	21
		32	20
		64	19
	2	8	23
		16	22
		32	21
		64	20
	4	8	24
		16	23
		32	22
		64	21
parity	1	8	26
		16	25
		32	24
		64	23
	2	8	27
		16	26
		32	25
		64	24
	4	8	28
		16	27
		32	26
		64	25

ecc	1	8	29
		16	28
		32	27
		64	26
	2	8	30
		16	29
		32	28
		64	27
	4	8	31
		16	30
		32	29
		64	28

Table 6-21D-Cache Data Address Bit-Width

Associativity	Size (KiB)	DCACHE_DATA_AW
1	8	11
	16	12
	32	13
	64	14
2	8	10
	16	11
	32	12
	64	13
4	8	9
	16	10
	32	11
	64	12

Table 6-22 D-Cache Data Bit-Width

Protection Scheme	DCACHE_DATA_RAM_DW	DCACHE_DATA_RAM_BWEW
none	32	4
parity	36	4
ecc	39	4

Table 6-23 D-Cache Byte Write Enable Mapping

BWE Bit	Protection Scheme: none	Protection Scheme: parity	Protection Scheme: ecc
0	wdata[7:0]	{wdata[32], wdata[7:0]}	wdata[38:0]
1	wdata[15:8]	{wdata[33], wdata[15:8]}	wdata[38:0]
2	wdata[23:16]	{wdata[34], wdata[23:16]}	wdata[38:0]
3	wdata[31:24]	{wdata[35], wdata[31:24]}	wdata[38:0]

6.10 AHB Slave Port Signals

The AHB Slave Port signals allows external agents to access the local memories of the processor through the AHB interface protocol. These signals are always present on the processor interface but they are used only when SLAVE_PORT_SUPPORT is yes. Otherwise, they are left unconnected. They are sampled/driven in the core_clk domain on bus_clk_en pulses.

Table 6-24 AHB Slave Port Signals

Signal Name	Direction	Description
slv_hsel	input	Bus grant
slv_huser	input	Select ILM/DLM (0:ILM, 1:DLM)
slv_hready	input	Ready in
slv_hreadyout	output	Ready out
slv_haddr[31:0]	input	Address bus
slv_hburst[2:0]	input	Burst type
slv_hprot[3:0]	input	Protection control
slv_hsize[2:0]	input	Transfer size
slv_htrans[1:0]	input	Transfer type
slv_hwrite	input	Transfer direction
slv_hwdata[SLAVE_PORT_DATA_WIDTH-1:0]	input	Write data bus
slv_hrdata[SLAVE_PORT_DATA_WIDTH-1:0]	output	Read data bus
slv_hresp[1:0]	output	Transfer response

6.11 BTB Interface Signals

BTB interface signals provide connectivity to the BTB SRAMs of the processor. These signals are always present on the process interface but they are used only when BTB is configured. Otherwise, they are left unconnected. BTB in AE250 is 2-way associative.

Table 6-25 BTB Memory Interface Signals

Signal Name	Direction	Description
btb0_cs	output	Chip select for BTB memory 0 0: disable 1: enable
btb0_we	output	Write enable for BTB memory 0 0: disable 1: enable
btb0_addr[BTB_RAM_ADDR_WIDTH-1:0]	output	Address for BTB memory 0
btb0_wdata[37:0]	output	Write data for BTB memory 0
btb0_rdata[37:0]	input	Read data for BTB memory 0
btb1_cs	output	Chip select for BTB memory 1 0: disable 1: enable
btb1_we	output	Write enable for BTB memory 1 0: disable 1: enable
btb1_addr[BTB_RAM_ADDR_WIDTH-1:0]	output	Address for BTB memory 1
btb1_wdata[37:0]	output	Write data for BTB memory 1
btb1_rdata[37:0]	input	Read data for BTB memory 1

Table 6-26 BTB RAM Address Bit-Width

BTB Size	BTB_RAM_ADDR_WIDTH	RAM Dimension
32	4	16×38
64	5	32×38
128	6	64×38
256	7	128×38

7 Reset and Clocking Scheme

7.1 Reset

AE250 only has one reset pin (core_reset_n) for resetting its only clock domain (core_clk). The reset signal should be synchronized to the core_clk clock domain before connecting to AE250.

In addition, to maintain proper reset ordering, the reset signal should only be released after the release of the reset signal to the bus clock domain, even though AE250 does not take the reset signal to the bus clock domain as its input. Figure 7-1 illustrates a reference design for reset synchronization.

Figure 7-1 Suggested Design for Reset Synchronization
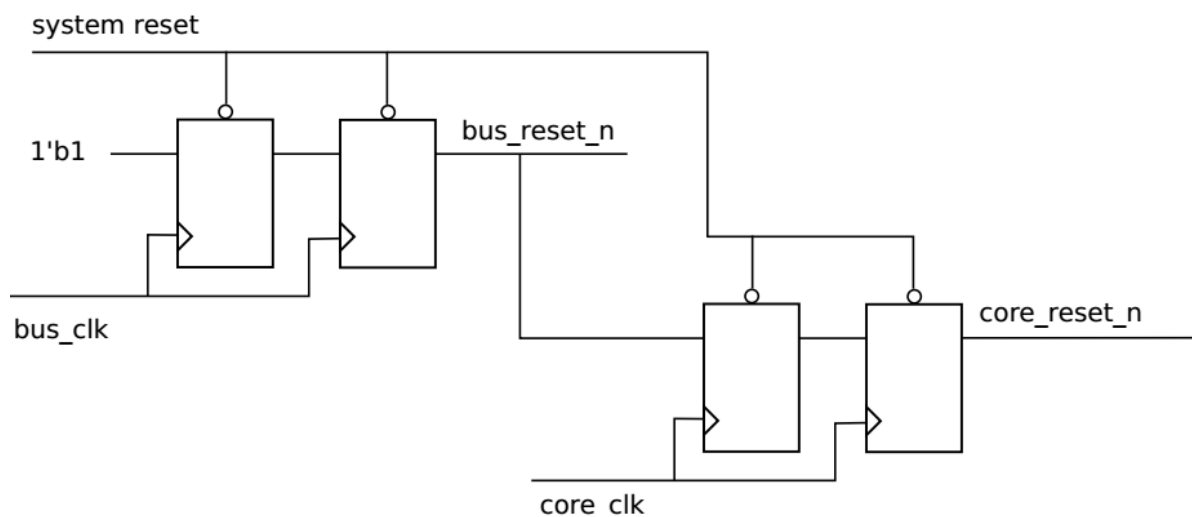
7.2 Clock Domains

AE250 has only one clock domain: core_clk. However, its bus interface signals may be operating in a separate synchronous bus clock domain. The synchronous bus clock is a virtual clock to the AE250 design. The clock enable input signal bus_clk_en serves as the clock enable signal

for generating the virtual clock. The core_clk domain signal should be asserted for one core_clk cycle before the rising edge of the bus clock, as shown in Figure 7-2. AE250 uses bus_clk_en to determine valid cycles to sample/drive the bus interface signals.

Figure 7-2 BUS_CLK_EN Waveform for N:1 (3:1) Clock Ratio
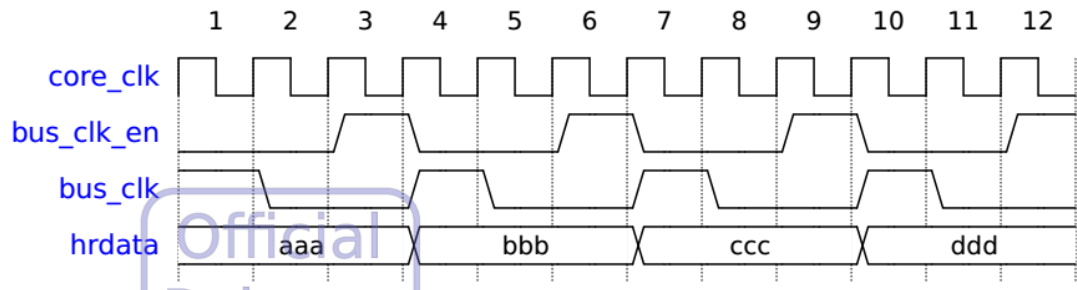
Detailed clock domain constraints can be found in the synthesis script timing_con.tcl as described latter.

8 Instruction Set Overview

8.1 Introduction

AE250 implements The RISC-V Instruction Set Manual, Volume I: User-Level ISA (TD001) V2.2. The following instruction sets are implemented:

- RV32I base integer instruction set
- RISC-V "C" standard extension
- RISC-V "M" standard extension
- RISC-V "A" standard extension
- AndeStar V5 instruction extension

For detailed information, please see The RISC-V Instruction Set Manual, Volume I: User-Level ISA (TD001) V2.2 and AndeStar V5 Instruction Extension Specification (UM165).

8.2 Integer Registers

Table 8-1 Integer Registerslists all general purpose integer registers.

Table 8-1 Integer Registers

Register	Signal Name	Description
x0	zero	Hard-wired zero
x1	ra	Return address
x2	sp	Stack pointer
x3	gp	Global pointer
x4	tp	Thread pointer
x5	t0	Temporary/alternate link register
x6–x7	t1–t2	Temporaries
x8	s0/fp	Saved register/frame pointer
x9	s1	Saved register
x10–x11	a0–a1	Function arguments/return values
x12–x17	a2–a7	Function arguments
x18–x27	s2–s11	Saved registers
x28–x31	t3–t6	Temporaries

8.3 Atomic Instructions

The RVA extension includes load-reserved/store-conditional and atomic memory operation (AMO) instructions.

8.3.1 Load-Reserved/Store-Conditional Instruction

AE250 tracks at most one physical address location for LR-SC instructions at a time. The reservation made by LR instruction is canceled after any memory operation or exception happens. The address of SC instructions must match the reserved address for SC to succeed.

8.3.2 Atomic Memory Operation Instruction

An atomic memory operation is expanded to LR-modify-SC sequences in AE250. The memory content is first loaded with the LR instruction, then the required operation is performed on the retrieved data, and the final result is written back to the memory by the SC instruction. The sequence will be retried again until it succeeds if the SC instruction fails.

8.4 Misaligned Memory Access

AE250 implements misaligned memory access to support accessing misaligned address without triggering any misaligned exceptions.

By controlling the mmisc_ctl CSR, the scheme can be enabled or disabled. Please see Machine Miscellaneous Control Register for details.

8.4.1 Limitation

In the following cases, AE250 will still trigger misaligned exceptions even though the misaligned memory access scheme is enabled.

- Accesses to device regions
- Accesses across two different memory regions
- Atomic Accesses

8.5 Floating Point ISA Extension

AE250 supports the "F" and "D" Standard Extensions for accelerating the performance of floating point heavy applications. The supported configuration is indicated in the Machine ISA (misa) configuration register.

8.5.1 FPU Features

AE250 supports the following features:

- Fully pipelined MAC instructions
- Hardware subnormal handling
- All rounding modes
- Support of custom half-precision load/store. (See FLHW and FSHW of AndeStar V5 Instruction Extension Specification (UM165) for more information.

9 Physical Memory Attributes

9.1 Introduction

The memory space of the AE250 processor is divided into memory regions. Each region has a memory type, which determines its memory attributes. Two types of memories are provided: Device memory and Normal memory.

Normal memory regions are considered as cacheable regions. The content of these memory regions can be cached in the processor. This attribute is ignored if no caches are configured in a processor implementation.Normal memory regions can be further divided into write-back regions and write-through regions. Table 9-1 Normal Memory Attributeable 9-1 lists the behavior of these two regions.

Table 9-1 Normal Memory Attribute

Regions	Write hit	Write-miss
Write-Back	Write to D-Cache	Write-allocate
Write-Through	Write-through	Write-no-allocate

Device memory regions (a.k.a. "device regions") are memory locations where an access to these locations may cause side effects. Device regions are non-cacheable and strongly-ordered. Accesses to device regions are guaranteed to be non-speculative and issued in program order. An access to a device region is not issued until all preceding accesses to device regions are finished.

On the other hand, accesses to the normal memory regions could be speculative and the order of accessing normal memory regions is not guaranteed. A load access to a normal memory region might bypass an earlier store access if there is no data dependency. In such a scenario, an explicit fence instruction is required to guarantee order.

Table 9-2 Memory Access Ordering shows ordering of two instructions

A and B, where A < B in program order.

Table 9-2 Memory Access Ordering

A<B in program order	B	
A	Normal Memory	Device
Normal Memory	-	-
Device	-	<

9.2 Device Regions

Device regions are memory locations where accesses to the location may cause side effects. Accesses to device regions are non-cacheable and strongly-ordered.

Device regions and ILM/DLM/WRITETHROUGH should not overlap with each other. The behavior is undefined when device regions are overlapped with ILM/DLM/WRITETHROUGH.

9.3 Write-through Regions

Write-through regions are memory locations where memory writes will update both D-Cache and the system memory with write-no-allocate policy.

Write-through regions and ILM/DLM/DEVICE regions should not overlap with each other. The behavior is undefined when write-through regions are overlapped with ILM/DLM/DEVICE regions.

10 Local Memory

10.1 Introduction

Local memories store data or instructions that might either be accessed frequently or require deterministic access latency, such as interrupt service routines, system calls, video data, real-time systems, etc. AE250 supports both instruction local memory (ILM) and data local memory (DLM). They are dedicated address spaces that are independent of the memory subsystem. Accesses to them bypass the memory subsystem to achieve minimal latency. The Local Memory Base Address is specified by processor parameters described in Processor Configuration Options.

Local memories are memories and accesses to them are treated the same as to the cacheable memory space. It is not suitable to map device registers in the local memories.

The details of local memory usages are described in the subsequent sections.

10.2 Local Memory Spaces

The AE250 processor supports three address spaces: the instruction local memory, the data local memory and the system bus (AHB) address spaces. The ILM address space is defined by ILM_ BASE and ILM_SIZE_KB configuration parameters, and the DLM address space is defined by DLM_ BASE and DLM_SIZE_KB configuration parameters. The base addresses of the Andes local memory should be aligned to its size (a power-of-2 size). See Processor Configuration Options for more information regarding the configuration parameters. Any addresses outside the local memory address spaces belong to the system bus address space.

Instruction fetches may go to the instruction local memory or the system bus while load/store data accesses may access all three regions of

spaces. The address spaces for ILM and DLM should not overlap each other for maximum compatibility across Andes processor products. The exact address space access priorities for the AE250 processor are defined in Table 10-1 Priorities for Instruction Fetches for instruction fetches and Table 10-2 Priorities for Data Accesses for load/store data accesses.

It is not recommended to set the instruction local memory base address to be the same as the data local memory base address. Otherwise, UNPREDICTABLE behavior might happen.

Table 10-1 Priorities for Instruction Fetches

Address hit the ILM space	Address hit the DLM space	Actual space accessed
No	No	AHB/AXI address space
No	Yes	AHB/AXI address space (not recommended; the ILM and DLM spaces should not overlap)
Yes	No	ILM
Yes	Yes	ILM (not recommended; the ILM and DLM spaces should not overlap)

Table 10-2 Priorities for Data Accesses

Address hit the ILM space	Address hit the DLM space	Actual space accessed
No	No	AHB/AXI address space
No	Yes	DLM
Yes	No	ILM
Yes	Yes	DLM (not recommended; the ILM and DLM spaces should not overlap)

10.3 Local Memory Address Range

The local memory address ranges are listed in Table 10-3 Local Memory Address Range (for ILM and DLM). LM_BASE represents the base address field of the ILM and DLM local memory base address system registers (milmb.IBPA and mdlmb.DBPA).

Table 10-3 Local Memory Address Range (for ILM and DLM)

LM Size	Start	End
4KiB	LM_BASE[63:12]	LM_BASE[63:12] + 0x000000FFF
8KiB	LM_BASE[63:13]	LM_BASE[63:13] + 0x000001FFF
16KiB	LM_BASE[63:14]	LM_BASE[63:14] + 0x000003FFF
32KiB	LM_BASE[63:15]	LM_BASE[63:15] + 0x000007FFF
64KiB	LM_BASE[63:16]	LM_BASE[63:16] + 0x00000FFFF
128KiB	LM_BASE[63:17]	LM_BASE[63:17] + 0x00001FFFF
256KiB	LM_BASE[63:18]	LM_BASE[63:18] + 0x00003FFFF
512KiB	LM_BASE[63:19]	LM_BASE[63:19] + 0x00007FFFF
1024KiB	LM_BASE[63:20]	LM_BASE[63:20] + 0x0000FFFFF
2048KiB	LM_BASE[63:21]	LM_BASE[63:21] + 0x0001FFFFF
4096KiB	LM_BASE[63:22]	LM_BASE[63:22] + 0x0003FFFFF
8192KiB	LM_BASE[63:23]	LM_BASE[63:23] + 0x0007FFFFF
16384KiB	LM_BASE[63:24]	LM_BASE[63:24] + 0x000FFFFFF

10.4 Local Memory Usage Constraints

Local memories are optimized for access latency. As a result, the design imposes the following usage restrictions:

- The virtual address mapping for local memories should maintain consistent address offset mappings for the offset part of the address that indexes into the local memory.
- Accesses to the local memory are speculative. Devices with side effects on reads should not be mapped to this region.

10.5 Local Memory Interface

The local memory interface could be SRAM or AHB-Lite. This is configured through the Local Memory Interface option. If the value of option is ram, SRAM style of interface is configured. If the value is ahb-lite, the AHB-Lite interface will be used.

Table 10-4 Possible AHB-Lite Transactions Used by Local Memory Interfacesshows the possible transactions of AHB-Lite interface used by the Local Memory Interfaces.

Table 10-5 Instruction Local Memory Protection Control Signaland Table 10-6 Data Local Memory Protection Control Signalsummarize the possible HPROT combinations on AHB-lite interfaces for instruction/data local memories.

Table 10-4 Possible AHB-Lite Transactions Used by Local Memory Interfaces

Request types	Transaction Types
Write transfers	SINGLE WORD
	SINGLE HALFWORD
	SINGLE BYTE
Read transfers	SINGLE WORD

Table 10-5 Instruction Local Memory Protection Control Signal

ILM_HPROT[3] Cacheable	ILM_HPROT[2] Bufferable	ILM_HPROT[1] Privileged	ILM_HPROT[0] Data/Instruction	Description
1	0	0	0	User Instruction Fetch
		0	1	User Data Access
		1	0	Privileged Instruction Fetch
		1	1	Privileged Data Access

Table 10-6 Data Local Memory Protection Control Signal

DLM_HPROT[3] Cacheable	DLM_HPROT[2] Bufferable	DLM_HPROT[1] Privileged	DLM_HPROT[0] Data/Instruction	Description
1	0	0	1	User Data Access
		1	1	Privileged Data Access

11 Local Memory Slave Port

11.1 Introduction

The local memory slave port enables external bus masters to access the local memories of AE250. When an address exceeds ILM/DLM size, the higher address bits are ignored by the slave port. The slv_huser signal of the local memory slave port interface selects which local memory to access:

Table 11-1 Local Memory Slave Port Selection

slv_huser[0]	Selection
0	ILM
1	DLM

Local memory slave port contains a four-entry buffer for burst accesses or write accesses. Therefore, write transfers might temporarily be stored in the buffer. Local memory slave port performs prefetch on burst read transfers to shorten the total wait state of burst transfers.

Slave port accesses have lower priority than load/store operations and instruction fetches. When a slave port access is not granted in 4 cycles, the access is granted the highest priority to avoid starvation.

Note that AE250 does not include logics to guarantee atomicity of atomic instructions accessing the local memory address space when external masters access the same location through the local memory slave port. And AE250 also does not include the protection feature on LM slave port.

11.2 Latency of Transfer

The table below summarizes the minimum latency of transfers accessing LM slave port. For read transfers, the latency will be larger than the listed number if the request of LM slave port is not granted by AE250

instantly. For write transfers, the latency will be larger than the listed number when the internal 4-entry buffer is full.

Table 11-2 Local Memory Slave Port Transfer Latency

Access Type	Minimal Latency
Single Read	5
Single Write	1
Burst Read(N Beats)	N+4
Burst Write(N Beats)	N

11.3 Basic Transfer

The waveform below illustrates the best case of single-read accesses with 4 cycle wait states and single-write accesses with no wait state.

Note that BUS_CLK is a pseudo-clock, please see Clock Domains for more information.

Figure 11-1 Single Access on the Local Memory Slave Port
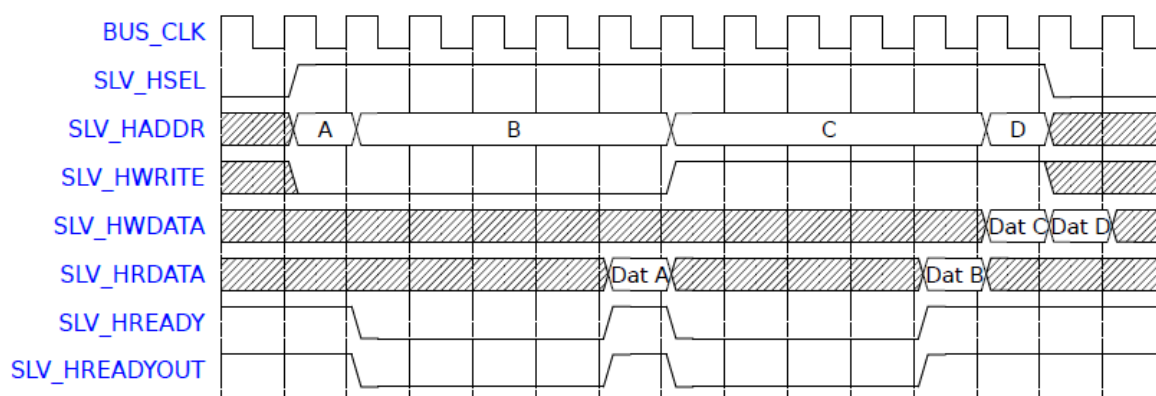
11.4 Burst Transfer

Figure 11-2 Burst Read Access on the Local Memory Slave Port and Figure 11-3 Burst Write Access on the Local Memory Slave Port illustrate a 16-beat incremental burst read access and 16-beat wrapped burst write access. Note that extra wait states may be inserted if the data width of local memory slave port is not equal to the data width of local memory.

Figure 11-2 Burst Read Access on the Local Memory Slave Port
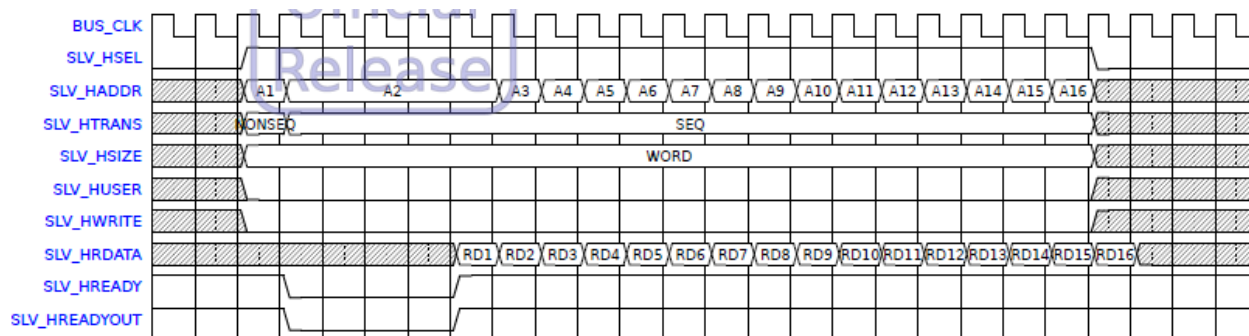
Figure 11-3 Burst Write Access on the Local Memory Slave Port
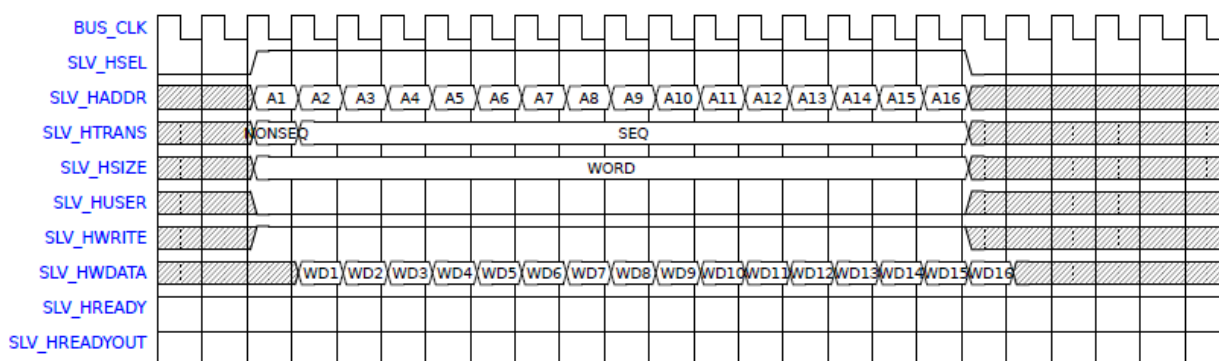
11.5 Support for Soft Error Protection

The local memory slave port would return bus errors to AHB masters as well as trigger local interrupts to AE250 when a 2-bit ECC error or parity error is detected.

The behavior of ECC logic for local memories is controlled by milmb.ECCEN/mdlmb.ECCEN. The encoding for errors encountered through accesses from the local memory slave port is summarized in the table below.

Correctable ECC errors only trigger local interrupts when ECCEN is equal to 3. Uncorrectable ECC errors would trigger local interrupts when ECCEN is equal to 2 or 3. The triggering of local interrupts is controlled by mie.IMECCI and the interrupt status is reported in mip.IMECCI. See Machine Interrupt Enable and Machine Interrupt Pending for detail.

Data returned through the local memory slave port is the ECC corrected version. For uncorrectable ECC errors, bus errors are reported when ECCEN is equal to 2 or 3.

Value	Local Interrupts	Data Returned
0	Disable parity/ECC.	Uncorrected data
1	Reserved.	Reserved
2	Generate local interrupts only on uncorrectable parity/ECC errors.	Corrected data or bus errors
3	Generate local interrupts on any type of parity/ECC errors.	Corrected data or bus errors

Extra wait state is needed while the size of a write transfer differs to the size of local memory. Figure 11-4 Various Size Write Transfers with ECC demonstrates an example of various size write transfers with ECC.

Figure 11-4 Various Size Write Transfers with ECC
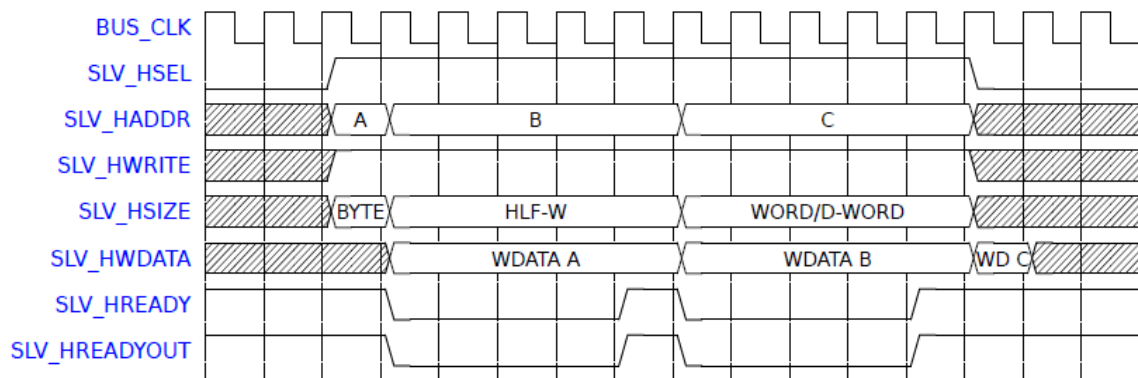
11.6 Local Memory Slave Port Operation under WFI Mode

The local memory slave port is still accessible in WFI mode if the following conditions are met:

- The clock to AE250 should resume running while there is a transfer issued from the local memory slave port if core clk is deactivated in WFI mode.
- The clock to AE250 should only be turned off in the WFI mode when there is no more outstanding transfer on the local memory slave port.

12 Caches

12.1 Introduction

AE250 has two caches, the instruction cache and the data cache. Both can be configured to 8KiB, 16KiB, 32KiB and 64KiB in size.

The cache organization information can be collected from the cache/memory configuration registers--micm_cfg for the instruction cache and mdcm_cfg for the data cache. The configuration choices are listed below and the format of the configuration registers could be found in Instruction Cache/Memory Configuration Register and Data Cache/Memory Configuration Register.

Table 12-1 Configuration Choices for the Instruction Cache

Items	Field Name (micm_cfg)	Choices
Cache lines per way	ISET	64, 128, 256, 512, 1024, 2048, 4096
Ways	IWAY	Direct-mapped, 2-way, 4-way
Line size (bytes)	ISZ	32

Table 12-2 Configuration Choices for the Data Cache

Items	Field Name (mdcm_cfg)	Choices
Cache lines per way	DSET	64, 128, 256, 512, 1024, 2048, 4096
Ways	DWAY	Direct-mapped, 2-way, 4-way
Line size (bytes)	DSZ	32

Total cache size = Cache lines per x way x Ways Line size

The replacement policy for 2-way and 4-way caches can be random or pseudo-LRU. The replacement policy for direct-mapped caches is irrelevant.

12.2 Cache Access Latency

The access latency of the instruction cache and the data cache are listed below.

Table 12-3 Access Latency of the Instruction Cache

Instruction	Throughput (cycles/instruction)	Latency (cycles)
Fetch from I-Cache (hit)	1	2
Fetch from I-Cache (miss)	6	7

Note!

The latency should take system delay into consideration

Table 12-4 Access Latency of the Data Cache

Instruction	Throughput (cycles/instruction)	Latency (cycles)
Load word from D-Cache (hit)	1	2
Load word from D-Cache (miss)	5	7
Load byte/halfword from D-Cache (hit)	1	3
Load byte/halfword from D-Cache (miss)	5	7

Note!

The latency should take system delay into consideration

12.3 I-Cache Fill Operation

The instruction cache fill operation starts when a cacheable line is not in the I-Cache. A burst read request for the missed cache line is always sent first to the system bus to minimize the miss latency.

The fill operation may be aborted by system bus errors. A precise instruction access fault is triggered for the instruction fetch causing the cache miss operation if the error is on the critical word. If the error occurs on non-critical words, the fill operation will be canceled and the missed line will not be installed into I-Cache. Instruction fetches before the non-critical error word will not be affected since they have received the required data.

In Debug Mode, instruction fetches will not affect I-Cache contents. All I-Cache misses will not cause cache replacements in Debug Mode.

12.4 D-Cache Fill Operations

The D-Cache fill operation starts when a cacheable line is not in the D-Cache. A burst read request for the missed cache line is always sent first to the system bus to minimize the miss latency. The read request will be followed by a burst write request if cache eviction is required.

The fill operation may be aborted by system bus errors. A precise load/store access fault is triggered for the load/store instruction causing the cache miss operation if the error is for the critical word. If the error occurs on non-critical words, the fill operation will be canceled and the missed line will not be installed into D-Cache. Load instructions will not be affected by these errors since they have received the required data (the critical words). Store instructions will send a single bus request to write the data directly to the bus.

In Debug Mode, load/store instructions will minimally affect D-Cache contents. All cache misses will not cause cache replacements, and only dirty bits may be affected by accesses to cache lines that are already in D-Cache.

12.5 D-Cache Eviction Operations

A burst write request will be sent to the system bus if a dirty line is evicted out of D-Cache. An imprecise bus-write error exception is triggered if the burst write request encounters system bus errors.

12.6 FENCE/FENCE.I Operations

FENCE/FENCE.I instructions can be used to write back or invalidate the cache content. The behavior of FENCE/FENCE.I is summarized in Table 12-5 Effects of FENCE/FENCE.I Instructions.

Table 12-5 Effects of FENCE/FENCE.I Instructions

Cache	FENCE	FENCE.I
I-Cache	None	Invalidate all cache lines
D-Cache	Write back and invalidate all cache lines	Write back all cache lines

FENCE do not flush (write back and invalidate) D-cache on U-mode.

13 Bus Interface Unit

13.1 Introduction

The bus interface unit (BIU) is responsible for off-CPU accesses which include system memory accesses and memory-mapped register accesses in devices. AE250 supports AHB bus protocols.

13.2 BIU Block Diagram

The following figure shows the block diagram of the bus interface unit. Requests for the external bus can come from fetch, load, and store accesses to memory. BIU includes an arbiter, a command FIFO, a write-data FIFO, a response FIFO, and a bus master.

Figure 13-1 BIU Block Diagram
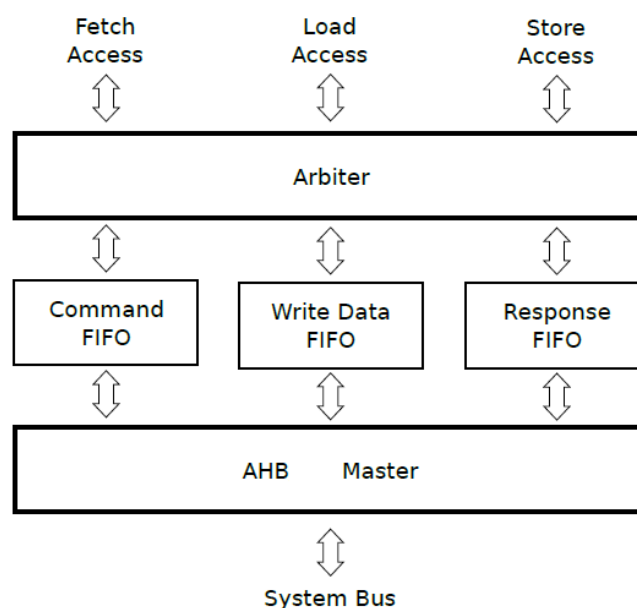
13.3 Supported Transaction Types

Table 13-1 AHB Transactions Used by   summarizes the transactions that BIU uses to access the AHB bus.

Table 13-1 AHB Transactions Used by BIU

Request types	Transaction Types for 64-bit Data Bus	Transaction Types for 32-bit Data Bus
Basic transfers	SINGLE WORD	SINGLE WORD
	SINGLE HALF WORD	SINGLE HALFWORD
	SINGLE BYTE	SINGLE BYTE
Additional transfers when caches are configured	WRAP4 DOUBLEWORD	WRAP8 WORD

13.4 Atomic Operations

Atomic operations are implemented by locking the bus using the HLOCK signal when AHB is supported. The HLOCK signal is asserted to lock the AHB bus once the LR instruction is executed. BIU will hold the lock until it executes the SC instruction or the internal lock is cleared.

Please see the usage descriptions for LR and SC instructions in The RISC-V Instruction Set Manual, Volume I: User-Level ISA (TD001) V2.2.

13.5 Low Latency AHB Access Mode

The low latency access option is designed to improve the bus access latency under low clock frequency conditions.

This mode comes with a couple restrictions:

1. Only AHB bus is supported.

2. Bus must be synchronous. Although the bus clock could run slower (N:1 clock ratio) to reduce power consumption, the timing constraint must be 1:1 with respect to core_clk during synthesis.

3. Cycle time is sacrificed in order to shorten the access latency. Bus signals are used directly without flops so internal critical paths will appear in the SoC timing report — 1/2 of the bus cycle time should be reserved for internal paths of BIU.

14 Trap

14.1 Introduction

According to RISC-V Privileged Architecture, a trap is a control flow change of normal instruction execution caused by an interrupt or an exception. An interrupt is a control flow change event initiated by an external source. An exception is a control flow change event generated as a by-product of instruction execution. When a trap happens, the processor stops processing the current flow of instructions, disables interrupts, saves enough states for later resumption, and starts executing a trap handler.

14.2 Interrupt

AE250 provides three interrupt inputs: timer interrupt, software interrupt and external interrupt. Timer interrupts and software interrupts are local interrupts in a RISC-V platform, which means each processor in the platform receives its own timer/software interrupts. External interrupts are global interrupts in a RISC-V platform, which means they are shared by all processors in a RISC-V platform. External interrupts are arbitrated and distributed by a platform level interrupt controller (PLIC) to processors in the RISC-V platform. Each external interrupt source can be assigned its own priority, and each interrupt target (e.g., RISC-V processors) could select which external interrupt sources it would handle. PLIC routes the highest priority interrupt source to the target processor. PLIC is described in Platform-Level Interrupt Controller (PLIC).

The mip CSR contains pending bits of these three interrupts, and mie CSR contains enable bits of these interrupts. The processor can selectively enable interrupts by manipulating the mie CSR, or globally disable interrupts by clearing the mstatus.MIE bit.

In addition to external interrupts, AE250 may generate internal interrupts for the following events:

- Local memory slave port parity/ECC error (See Floating Point ISA Extension and mie.IMECCI and mip.IMECCI)
- Bus-write transaction error (See mie.BWEI and mip.BWEI)
- Performance monitor overflow (See mie.PMOVI and mip.PMOVI)

14.3 Exception

AE250 implements the following exceptions:

- Instruction address misaligned exceptions
  Jump to misaligned addresses
- Instruction access faults
  - Bus errors caused by instruction fetches
  - Uncorrectable ECC errors when fetching trap handlers under the vector PLIC mode
- Illegal instructions
  - Unsupported instructions
  - Privileged instructions
  - Accessing non-existent CSRs
  - Accessing privileged CSRs
  - Writing to read-only CSRs
  - Executing Andes-specific instructions in the RISC-V compatibility mode (mmisc_ctl.RVCOMPM== 1)
- Breakpoint exceptions
- Load address misaligned exceptions
- Load access faults
  - Bus errors caused by load instructions
  - ECC errors caused by load instructions
- Store/AMO address misaligned exceptions
- Store/AMO access faults
  ECC errors caused by store instructions
- Environment calls
- Stack overflow/underflow exception with StackSafe supported

14.4 Trap Handling

All traps are handled in M-mode. When a trap occurs, the following operations are applied:

- mepc is set to the current program counter.
- mstatus is updated.
  - The MPP field is set to the current privilege mode

- – The MPIE field is set to mie
- – The MIE field is set to 0
- ● mcause is updated.
- ● mtval is updated when any of address-misaligned, access-fault, or page-fault exceptions occur.
- ● The privilege mode is changed to M-mode.
- ● When mmisc_ctl.VEC_PLIC is 0, the program counter is set to the address specified by mtvec.
- ● When mmisc_ctl.VEC_PLIC is 1, the mtvec register will be the base address register of a vector table with 4-byte entries storing addresses pointing to interrupt service routines.
- – mtvec[0] is for exceptions and non-external local interrupts. For these traps, the mcause register records the trap type based on RISC-V definitions.
- – mtvec[N] is for external PLIC interrupt source N. For the external PLIC interrupts, the mcause register records the interrupt source ID. The RISC-V architecture defines a two-level stack of interrupt-enable bits and privilege modes. To support nested traps, the trap handler should backup trap handling CSRs and enable the interrupt-enable bit.

After handling a trap, the MRET instruction can be executed for returning to the privilege mode before the trap. When a MRET instruction is executed, the following operations are applied:

- ● The program counter is set to mepc
- ● The privilege mode is set to mstatus.MPP
- ● mstatus is updated
- – The MPP field is set to U-mode (or M-mode if U-mode is not supported)
- – The MIE field is set to mpie
- – The MPIE field is set to 1

15 Reset and Non-Maskable Interrupts

15.1 Reset

When the core_reset_n input signal is deasserted, the following operations are applied:

- CSRs are set to their reset values
- All integer registers (listed in Table 8-1 Integer Registers) are set to zero
- BTB is initialized
- Program execution starts with the address specified by the reset_vector input signal

15.2 Non-Maskable Interrupts

Non-maskable interrupts (NMIs) are intended for handling hardware error conditions and are assumed to be non-resumable. They are triggered through the nmi input signal. The rising edge of the signal causes an immediate jump to an address stored in the mnvec register and transition of the privilege level to M-mode, regardless of the state of a hart's interrupt enable bit.

The following operations are applied when an NMI is taken:

- The mepc register is written with the address of the next instruction when the NMI was taken.
- The mcause register is set to 1, indicating that NMI is caused by the nmi signal.
- The mstatus.MPP field records the privilege mode before NMI was taken.
- The mstatus.MPIE field is set to the value of mstatus.xIE before NMI was taken. The "x" is active privilege mode before the NMI was taken.
- The mstatus.MIE field is set to 0.

16 Power Management

16.1 Wait-For-Interrupt Mode

The RISC-V WFI instruction enables the processor to enter the wait-for-interrupt (WFI) mode for reducing power consumption and power-gating of the processor should only happen when the processor is in the WFI mode.

Upon execution of a WFI instruction, the processor stops all activities and asserts the core_wfi_mode output signal to indicate that the processor is in WFI mode.

Once in WFI mode, memory transactions that are started before the execution of WFI are guaranteed to have been completed, all transient states of memory handling are flushed and no new memory accesses will take place. The core_clk and bus_clk_en input signals can be safely changed in this period for frequency scaling. This is also the safe period to power-gate the processor while making the I/D-Cache SRAMs enter the state retention mode.

Slave port accesses are not disabled in WFI mode, but the processor requires core_clk to access local memories. When slave port accesses are still needed in WFI mode, core_clk should not be gated off.

When the processor is in WFI mode and is awoken by the nmi or debugint signals, it will resume and start to execute from the first instruction of NMI or debug interrupt service routine.

The processor can be awoken by the interrupts regardless the value of the global interrupt enable bit (mstatus.MIE). When the core is awoken by a taken interrupt, it will resume and start to execute from the corresponding interrupt service routine. When the core is awoken by a pending interrupt and mstatus.MIE is disabled, it will resume and start to execute from the instruction after the WFI instruction. Note that interrupts disabled by the mip CSR will not be able to awake the processor in WFI mode.

17 Memory Subsystem Error Protection

17.1 Introduction

AE250 includes support for soft-error protection for the memory subsystems

17.1.1 Memory Subsystem Error Protection Scheme

Two memory subsystem error protection schemes are supported:

- Parity check
- – Memory error detection through even parity check
- – Detecting one bit error per byte
- – Each 8-bit data requires one extra bit to store the parity bit
- ECC
- – Single-Error-Correction, Double-Error-Detection (SEC-DED) ECC
- – Single-bit errors can be detected and recovered
- – Double-bit errors can be detected but may not be recovered
- – Each 32-bit data requires seven extra bits to store the ECC code

17.1.2 Error Protected Memory Subsystem

The AE250 memory subsystems protected by the Parity/ECC scheme include:

- Cache memories
- – Instruction caches (Tag RAM and Data RAM)
- – Data caches (Tag RAM and Data RAM)
- Local memories
- – ILM
- – DLM

17.1.3 Read-Modify-Write Operations

For data RAMs, the unit of ECC protection is one 32-bit word (four bytes). The ECC code is computed and written to the data RAMs along with the data word.

To write narrower data (e.g., a byte or a half word) into these RAMs, the processor must read data from the RAM, merge the read data with the write data, and then generate the ECC code for the merged data before writing back the merged data and the ECC code. This process is referred to as read-modify-write operations and these operations are done by hardware automatically in AE250.

For parity protected RAMs, the unit of parity protection is one byte. The parity bits could be generated directly for all kinds of partial word (bytes and half-words) writes without the need for the read-modify- write operations.

17.2 Parity/ECC Control Mode and Access types

For each protected memory, a Parity/ECC enable control flag is defined with three modes:

- Parity/ECC checking disabled
- Generating exceptions on uncorrectable Parity/ECC errors
- Generating exceptions on Parity/ECC errors

17.2.1 Parity/ECC Checking Disabled

The behavior of the Parity/ECC logic when Parity/ECC checking is disabled is:

- Reads from RAMs will not trigger the ECC circuit at all. The raw data is directly returned. But all writes to RAMs still update/regenerate Parity/ECC codes.
- No exceptions would be generated.
- No Parity/ECC related registers will be updated.
- The processor uses read-modify-write operations to store a partial word.

17.2.2 Generating Exceptions on Uncorrectable Parity/ECC Errors

The behavior of the Parity/ECC logic under this mode is:

- Parity/ECC checking is enabled and all writes to the RAMs update/regenerate Parity/ECC codes. Generating Exceptions on Parity/ECC Errors;

- The processor gets the error-corrected data from RAMs;
- The processor uses read-modify-write operations to store a partial word;
- For accesses by the main pipeline n the CPU core:
- No exception would be generated for correctable errors.
- Exceptions would be generated for uncorrectable errors.
- The related error reporting registers would be updated on all Parity/ECC errors.
- For accesses by the local memory slave port:
- No exception would be generated for any detected Parity/ECC errors.
- The standard bus error reporting mechanism is used to report uncorrectable errors for the slave port accesses.
- The slave port triggers local interrupts to signal that uncorrectable errors are detected.

17.2.3 Generating Exceptions on Parity/ECC Errors

The behavior of the Parity/ECC logic under this mode is:

- Parity/ECC checking is enabled and all writes to the RAMs update/regenerate Parity/ECC codes.
- The processor gets the gets the error-corrected data from RAMs.
- The processor uses read-modify-write operations to store a partial word.
- For accesses by the main pipeline in CPU core:
- All detected Parity/ECC errors would generate exceptions.
- The related error reporting registers would be updated on all Parity/ECC errors.
- For accesses by the local memory slave port:
- No exception would be generated for any detected Parity/ECC errors.
- The standard bus error reporting mechanism is used to report uncorrectable errors for the slave port accesses.
- The slave port triggers local interrupts to signal that uncorrectable errors are detected.

17.2.4 Error Handling in Caches

The behavior of the cache ECC logic is controlled by the following CSR. See Cache Control Register for more information.

mcache_ctl.IC_ECCEN

mcache_ctl.DC_ ECCEN.

The definitions of correctable and uncorrectable errors and their handling are summarized in Table 17-1 Handling Correctable Errors in Caches and Table 17-2 Handling Uncorrectable Errors in Caches.

Table 17-1 Handling Correctable Errors in Caches

Error Type	Error Handling Action
One-bit parity errors or one-bit/two-bit ECC errors in clean cache lines	• The clean lines are invalidated and the correct copies from the next-level memory are brought back into the cache. • All lines in I-Cache are considered clean.
One-bit ECC errors in dirty cache lines	• All data in dirty lines are written back to the next-level memory after ECC correction. • The dirty lines are then invalidated and the correct copies from the next-level memory are brought back into the cache.

Table 17-2 Handling Uncorrectable Errors in Caches

Error Type	Error Handling Action

One-bit parity errors or two-bit ECC errors in dirty cache lines	• All data in the dirty lines are written back to the next-level memory.
	  – Data without ECC errors and with one-bit ECC errors are written back after correction.
	  – Data with one-bit parity error or two-bit ECC errors cannot be recovered and they are written back without correction.
	• The dirty lines are then invalidated.

17.2.5 Error Handling in ILM and DLM

The actions when a Parity/ECC error is detected in ILM/DLM are listed in Table 17-3 Local Memory Parity/ECC Error Handling.

Table 17-3 Local Memory Parity/ECC Error Handling

Error Type	Error Handling Action
Correctable errors	The data is corrected and written back to ILM/DLM.
Uncorrectable errors	The data is not corrected and a *Machine Error* exception is triggered.

17.2.6 Behavior of Parity/ECC-Supported Local Memory Accesses

The behavior of Local Memory ECC logic is controlled by milmb.ECCEN/mdlmb.ECCEN. See Introduction and Data Local Memory Base Register for more information.

Table 17-4 Parity/ECC Behavior for Local Memory Operations

Operation	Parity/ECC Error Checking
Instruction fetches and load/store instructions	Controlled by `milmb.ECCEN`/`mdlmb.ECCEN`.
Slave port accesses	• Controlled by `milmb.ECCEN`/`mdlmb.ECCEN`. • No exception would be generated for detected Parity/ECC errors. • Uncorrectable errors would be reported through error response. • Uncorrectable errors trigger local interrupts.

Table 17-5 Types of Parity/ECC Error Exception

Access Type	Target RAM	Precise/Imprecise
Instruction Fetches	Local memories	Precise
Load-type instructions	Local memories	Precise
Store-type instructions	Local memories	Precise

18 Control and Status Registers

18.1 Introduction

The sections below describe the registers in detail.

18.1.1 System Register Type

Term	Description
IM	Implementation dependent/determined
RO	Read-Only register/field. Any software write to RO register/field will be silently ignored by hardware.
RW	Read/Write register/field
W1S	Write 1 to Set
WLRL	Write/Read Only Legal Values

18.1.2 Reset Value

Term	Description
DC	The reset value is Don't-Care.

18.1.3 CSR Listing

Table 18-1 Machine Information Registers

Mnemonic Name	CSR Address
mvendorid	0xf11
marchid	0xf12
mimpid	0xf13
mhartid	0xf14

Table 18-2 Machine Trap Related Registers

Mnemonic Name	CSR Address
mstatus	0x300
misa	0x301
medeleg	0x302
mideleg	0x303
mie	0x304
mtvec	0x305
mscratch	0x340
mepc	0x341
mcause	0x342
mtval	0x343
mip	0x344
mxstatus	0x7c4
mdcause	0x7c9

Table 18-3 Counter Related Registers

Mnemonic Name	CSR Address
mcycle	0xb00
mcycleh	0xb80
minstret	0xb02
minstreth	0xb82
mhpmcounter3	0xb03
mhpmcounter4	0xb04
mhpmcounter5	0xb05
mhpmcounter6	0xb06
mhpmcounter3h	0xb83
mhpmcounter4h	0xb84
mhpmcounter5h	0xb85
mhpmcounter6h	0xb86
mcounteren	0x306
mhpmevent3	0x323
mhpmevent4	0x324
mhpmevent5	0x325
mhpmevent6	0x326
mcounterwen	0x7ce
mcounterinten	0x7cf
mcountermask_m	0x7d1
mcountermask_s	0x7d2
mcountermask_u	0x7d3
mcounterovf	0x7d4

Table 18-4 Configuration Control & Status Registers

Mnemonic Name	CSR Address
micm_cfg	0xfc0
mdcm_cfg	0xfc1
mmsc_cfg	0xfc2

Table 18-5 Trigger Registers

Mnemonic Name	CSR Address
tselect	0x7a0

Mnemonic Name	CSR Address
tdata1	0x7a1
tdata2	0x7a2
tdata3	0x7a3
tinfo	0x7a4
mcontrol	0x7a1
icount	0x7a1
itrigger	0x7a1
etrigger	0x7a1

Table 18-6 Debug Registers

Mnemonic Name	CSR Address
dcsr	0x7b0
dpc	0x7b1
dscratch0	0x7b2
dscratch1	0x7b3
dexc2dbg	0x7e0
ddcause	0x7e1

Table 18-7 Memory and Miscellaneous Registers

Mnemonic Name	CSR Address
milmb	0x7c0
mdlmb	0x7c1
mecc_code	0x7c2
mnvec	0x7c3
mpft_ctl	0x7c5
mcache_ctl	0x7ca
mcctlbeginaddr	0x7cb
mcctlcommand	0x7cc
mcctldata	0x7cc
ucctlbeginaddr	0x80b
ucctlcommand	0x80c
mmisc_ctl	0x7d0

Table 18-8 Hardware Stack Protection and Recording Registers

Mnemonic Name	CSR Address
mhsp_ctl	0x7c6
msp_bound	0x7c7
msp_base	0x7c8

Table 18-9 CoDense Registers

Mnemonic Name	CSR Address
uitb	0x800

Table 18-10 PMP Registers

Mnemonic Name	CSR Address
pmpcfg0	0x3a0
pmpcfg1	0x3a1
pmpcfg2	0x3a2
pmpcfg3	0x3a3
pmpaddr0	0x3b0
pmpaddr1	0x3b1
pmpaddr2	0x3b2
pmpaddr3	0x3b3
pmpaddr4	0x3b4
pmpaddr5	0x3b5
pmpaddr6	0x3b6
pmpaddr7	0x3b7
pmpaddr8	0x3b8
pmpaddr9	0x3b9
pmpaddr10	0x3b10
pmpaddr11	0x3b11
pmpaddr12	0x3b12
pmpaddr13	0x3b13
pmpaddr14	0x3b14
pmpaddr15	0x3b15

18.2 Machine Information Registers

18.2.1 Machine Vendor ID Register

Mnemonic Name: mvendorid

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xf11 (standard read only)

31	0
MVENDORID	
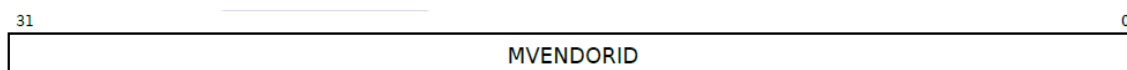
This read-only register provides the Andes JEDEC manufacturer ID: 0x0000031e.

Field Name	Bits	Description	Type	Reset
MVENDORID	[31:0]	The manufacturer ID of Andes.	RO	0x0000031e

18.2.2 Machine Architecture ID Register

Mnemonic Name: marchid

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xf12 (standard read only)

31	30		0
1		CPU_ID	
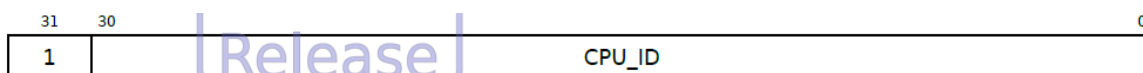
This register provides the micro-architecture id of AndesCore processor implementations. For AE250, marchid.CPU_ID will be 0x0025. Note that the MSB of this register is 1 for commercial implementations of RISC-V processors.

Field Name	Bits	Description	Type	Reset
CPU_ID	[30:0]	Andes CPU ID.	RO	0x0025

18.2.3 Machine Implementation ID Register

Mnemonic Name: mimpid

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xf13 (standard read only)

31	8	7	4	3	0
MAJOR		MINOR		EXTENSION	
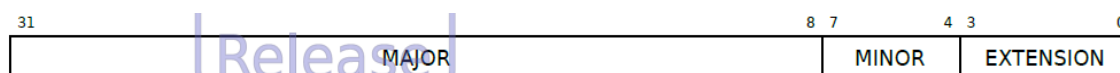
This register provides the revision number of the AE250 processor. Please see AndesCore N25(F) Release Note (RN168) for exact values. It is documented in the release note as MAJOR.MINOR. EXTENSION.

Field Name	Bits	Description	Type	Reset
EXTENSION	[3:0]	Revision extension.	RO	IM
MINOR	[7:4]	Revision minor.	RO	IM
MAJOR	[31:8]	Revision major.	RO	IM

18.2.4 Hart ID Register

Mnemonic Name: mhartid

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xf14 (standard read only)

31		0
	MHARTID	
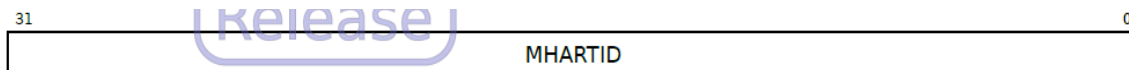
This register provides the ID of the hardware thread. It is required that one of the hart IDs must be zero on a RISC-V platform. The value of this register is determined by the HART0_ID parameter.

Field Name	Bits	Description	Type	Reset
MHARTID	[31:0]	Hart ID.	RO	HART0_ID

18.3 Machine Trap Related CSRs

18.3.1 Machine Status

Mnemonic Name: mstatus

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x300 (standard read/write)

31	30		23	22		20
SD		0			0	

19	18	17	16	15 14	13 12	11 10	8	7	6	5	4	3 2	1	0
MXR	0	MPRV	XS	FS	MPP	0		MPIE	0		UPIE	MIE	0	UIE
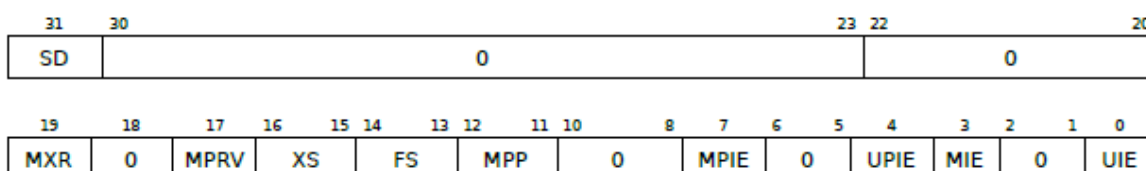
Field Name	Bits	Description	Type	Reset
UIE	[0]	U-mode interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
MIE	[3]	M-mode interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
UPIE	[4]	UPIE holds the value of the UIE bit prior to a trap.	RW	0
MPIE	[7]	MPIE holds the value of the MIE bit prior to a trap.	RW	0
MPP	[12:11]	MPP holds the privilege mode prior to a trap. Encoding is in Table 4. When U-mode is not available, this field is hardwired to 3.	WLRL	3
FS	[14:13]	FS holds the status of the architectural states of the floating-point unit, including the fcsr CSR and f0 – f31 floating-point data registers. The value of this field is zero and read-only if the processor does not have FPU. This field is primarily managed by software. The processor hardware assists the state managements in two regards:	WLRL	0

- Illegal instruction exceptions are triggered when FS is Off.
- FS is updated to the Dirty state with the execution of floating-point instructions when FS is not Off.

Changing the setting of this field has no effect on the contents of the floating-point register states. In particular, setting FS to Off does not destroy the states, nor does setting FS to Initial clear the contents.

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

Field Name	Bits	Description	Type	Reset
XS	[16:15]	XS holds the status of the architectural states (ACE registers) of ACE instructions. The value of this field is zero if ACE extension is not configured. This field is primarily managed by software. The processor hardware assists the state managements in two regards: • Illegal instruction exceptions are triggered when XS is Off. • XS is updated to the Dirty state with the execution of ACE instructions when XS is not Off. Changing the setting of this field has no effect on the contents of ACE states. In particular, setting XS to Off does not destroy the states, nor does setting XS to Initial clear the contents. 	Value	Meaning	 0	Off	 1	Initial	 2	Clean	 3	Dirty		RO	0
MPRV	[17]	When the MPRV bit is set, the memory access privilege for load and store are specified by the MPP field.	RW	0
MXR	[19]	MXR controls whether execute-only pages are readable. It has no effect when page-based virtual memory is not in effect. 	Value	Meaning	 0	Execute-only pages are not readable	 1	Execute-only pages are readable		RW	0
SD	[31]	SD summarizes whether either the FS field or XS field is dirty.	RO	0

When supervisor mode or N extension is not supported, the corresponding bits in mstatus are hardwired to zero.

18.3.2 Machine ISA Register

Mnemonic Name: misa

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x301 (standard read/write)

31 30 29	26 25		0
Base	0	[Release]	Extensions

Field Name	Bits	Description	Type	Reset
Extensions	[25:0]	See Table 71.	RO	IM
Base	[31:30]	The general purpose register width of the native base integer ISA.	RO	1

Value	Meaning
0	Reserved
1	32
2	64
3	128

Table 18-11 RISC-V Definition of the Extensions Field

Bit	Extension	Description
0	A	Atomic extension
1	B	*Tentatively reserved for Bit operations extension*
2	C	Compressed extension
3	D	Double-precision floating-point extension
4	E	RV32E base ISA
5	F	Single-precision floating-point extension
6	G	Additional standard extensions present
7	H	Reserved
8	I	RV32I/64I/128I base ISA
9	J	*Tentatively reserved for Dynamically Translated Languages extension*
10	K	Reserved
11	L	*Tentatively reserved for Decimal Floating-Point extension*
12	M	Integer Multiply/Divide extension

Bit	Extension	Description
13	N	User-level interrupts supported
14	O	Reserved
15	P	*Tentatively reserved for Packed-SIMD extension*
16	Q	Quad-precision floating-point extension
17	R	Reserved
18	S	Supervisor mode implemented
19	T	*Tentatively reserved for Transactional Memory extension*
20	U	User mode implemented
21	V	*Tentatively reserved for Vector extension*
22	W	Reserved
23	X	Non-standard extensions present
24	Y	Reserved
25	Z	Reserved

18.3.3 Machine Interrupt Enable

Mnemonic Name: mie

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x304 (standard read/write)

31 19	18	17	16	15 12	11	10 9	8	7	6 5	4	3	2 1	0
0	PMOVI	BWEI	IMECCI	0	MEIE	0	UEIE	MTIE	0	UTIE	MSIE	0	USIE

Field Name	Bits	Description	Type	Reset
USIE	[0]	U-mode software interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

MSIE	[3]	M-mode software interrupt enable bit.		RW	0
			Value	Meaning	
			0	Disabled	
			1	Enabled	

UTIE	[4]	U-mode timer interrupt enable bit.		RW	0
			Value	Meaning	
			0	Disabled	
			1	Enabled	

MTIE	[7]	M-mode timer interrupt enable bit.		RW	0
			Value	Meaning	
			0	Disabled	
			1	Enabled	

UEIE	[8]	U-mode external interrupt enable bit.		RW	0
			Value	Meaning	
			0	Disabled	
			1	Enabled	

MEIE	[11]	M-mode external interrupt enable bit.		RW	0
			Value	Meaning	
			0	Disabled	
			1	Enabled	

IMECCI	[16]	Imprecise ECC error local interrupt enable bit. The processor may receive imprecise ECC error on slave port accesses or cache writebacks.		RW	0
			Value	Meaning	
			0	Disabled	
			1	Enabled	

Field Name	Bits	Description	Type	Reset
BWEI	[17]	Bus-write transaction error local interrupt enable bit. The processor may receive bus-write errors on store instructions or cache writebacks.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
PMOVI	[18]	Performance monitor overflow local interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

When supervisor mode or N extension is not supported, the corresponding bits in mie are hardwired to zero.

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields above show the default bit location.

18.3.4 Machine Interrupt Pending

Mnemonic Name: mip

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x344 (standard read/write)

31 19	18	17	16	15 12	11	10 9	8	7	6 5	4	3	2 1	0
0	PMOVI	BWEI	IMECCI	0	MEIP	0	UEIP	MTIP	0	UTIP	MSIP	0	USIP

Field Name	Bits	Description	Type	Reset
USIP	[0]	U-mode software interrupt pending bit.	RW	0

Value	Meaning
0	Not pending
1	Pending

Field Name	Bits	Description	Type	Reset
MSIP	[3]	M-mode software interrupt pending bit. <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not pending</td></tr><tr><td>1</td><td>Pending</td></tr></table>	RO	0
UTIP	[4]	U-mode timer interrupt pending bit. <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not pending</td></tr><tr><td>1</td><td>Pending</td></tr></table>	RW	0
MTIP	[7]	M-mode timer interrupt pending bit. <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not pending</td></tr><tr><td>1</td><td>Pending</td></tr></table>	RO	0
UEIP	[8]	U-mode external interrupt pending bit. <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not pending</td></tr><tr><td>1</td><td>Pending</td></tr></table>	RW	0
MEIP	[11]	M-mode external interrupt pending bit. <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not pending</td></tr><tr><td>1</td><td>Pending</td></tr></table>	RO	0
IMECCI	[16]	Imprecise ECC error local interrupt enable bit. The processor may receive imprecise ECC error on slave port accesses or cache writebacks. <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not pending</td></tr><tr><td>1</td><td>Pending</td></tr></table>	RW	0

Field Name	Bits	Description	Type	Reset
BWEI	[17]	Bus-write transaction error local interrupt pending bit. The processor may receive bus-write errors on store instructions or cache writebacks.	RO	0

Value	Meaning
0	Not pending
1	Pending

Field Name	Bits	Description	Type	Reset
PMOVI	[18]	Performance monitor overflow local interrupt pending bit.	RO	0

Value	Meaning
0	Not pending
1	Pending

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields above show the default bit location.

18.3.5 Machine Trap Vector Base Address

Mnemonic Name: mtvec

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x305 (standard read/write)

31		2 1 0
	BASE[31:2]	0

This register determines the base address of the trap vector. The least significant 2 bits are hard-wired to zeros. When the configured address width is less than 32, the upper bits are hard-wired to zeros. When mmisc_ctl.VEC_PLIC is 0 (PLIC is not in the vector mode), this register indicates the entry points for the trap handler and it may point to any 4-byte aligned location in the memory space.

On the other hand, when mmisc_ctl.VEC_PLIC is 1 (PLIC is in the vector mode), this register will be the base address of a vector table with 4-byte entries storing addresses pointing to interrupt service routines. And this register should be aligned to 4096 byte (4 KiB) boundary.

*mtvec[0] is for exceptions and non-external local interrupts.

* mtvec[N] is for external PLIC interrupt source N coming from mip.MEIP.

* mtvec[1024+N] is for external PLIC interrupt source N coming from

** mip.UEIP when mideleg.UEI == 0 for M/U system.

Field Name	Bits	Description	Type	Reset
BASE[31:2]	[31:2]	Base address for interrupt and exception handlers. See description above for alignment requirements when PLIC is in the vector mode.	RW	0

18.3.6 Machine Exception Program Counter

Mnemonic Name: mepc

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x341 (standard read/write)

This register is written with the virtual address of the instruction that encountered traps and/or NMIs when these events occurred.

Field Name	Bits	Description	Type	Reset
EPC	[31:1]	Exception program counter.	RW	0

18.3.7 Machine Cause Register

Mnemonic Name: mcause

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x342 (standard read/write)

This register indicates the cause of trap, reset, NMI or the interrupt source ID of vector interrupt.

This register is updated when trap, reset, NMI or vector interrupt occurs.

Field Name	Bits	Description	Type	Reset
EXCEPTION_CODE	[9:0]	Exception Code.	RW	0
INTERRUPT	[31]	Interrupt.	RW	0

The following tables show the possible values of mcause:

Table 18-12 Possible Values of mcause After Trap

Interrupt	Exception Code	Description
1	3	Machine software interrupt
1	7	Machine timer interrupt
1	11	Machine external interrupt
1	16	Slave port ECC error interrupt (M-mode)
1	17	Bus write transaction error interrupt (M-mode)
1	18	Performance monitor overflow interrupt(M-mode)
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	Reserved
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	Reserved
0	15	Store/AMO page fault
0	32	Stack overflow exception
0	33	Stack underflow exception
0	40-47	Andes Custom Extension exception (see *AndeStar Custom Extension Specification CUM009* for more details)

Table 18-13 Possible Values of mcause after Reset

Interrupt	Exception Code	Description
0	0	Reset by core_reset_n

Table 18-14 Possible Values of mcause after NMI

Interrupt	Exception Code	Description
0	1	NMI triggered

Table 18-15 Possible Values of mcause after Vector Interrupt

mcause	Description
Interrupt source ID	Interrupt source ID when vector interrupt occurs

18.3.8 Machine Trap Value

Mnemonic Name: mtval

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x343 (standard read/write)
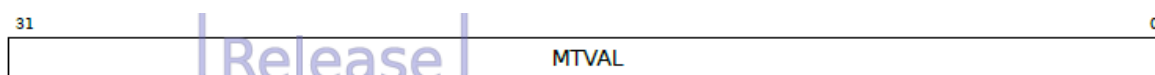
This register is updated when a trap is taken to M-mode. The updated value is dependent on the cause of traps:

● For hardware breakpoint exceptions, address-misaligned exceptions, access-fault exceptions, or page-fault exceptions, it is the effective faulting addresses.

● For illegal instruction exceptions, the updated value is the faulting instruction. If the length of the instruction is less than XLEN bits long, the upper bits of mtval are cleared to zero.

● For other exceptions, mtval is set to zero.

For instruction-fetch access faults, this register will be updated with the address pointing to the portion of the instruction that caused the fault, while the mepc register will be updated with the address pointing to the beginning of the instruction.

When the configured address width is less than 32, the upper bits are hard-wired to zeros.

Field Name	Bits	Description	Type	Reset
MTVAL	[31:0]	Exception-specific information for software trap handling.	RW	0

18.3.9 Machine Scratch Register

Mnemonic Name: mscratch

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x340 (standard read/write)
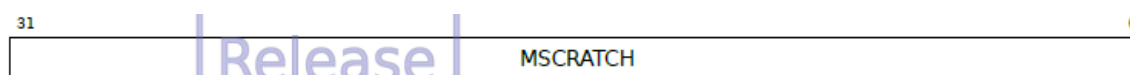
This is a scratch register for temporary data storage, which is typically

used by the M-mode trap handler.

Field Name	Bits	Description	Type	Reset
MSCRATCH	[31:0]	Scratch register storage.	RW	0

18.3.10 Machine Extended Status

Mnemonic Name: mxstatus

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x7c4 (non-standard read/write)

31		10	9	6	5	4	3	2	1	0
0			0		PDME	DME	0		PPFT_EN	PFT_EN

Field Name	Bits	Description	Type	Reset
PFT_EN	[0]	Enable performance throttling. When throttling is enabled, the processor executes instructions at the performance level specified in `mpft_ctl.T_LEVEL`. On entering a trap: • PPFT_EN ⇐ PFT_EN; • PFT_EN ⇐ mpft_ctl.FAST_INT ? 0 : PFT_EN; On executing an MRET instruction: • PFT_EN ⇐ PPFT_EN; This field is hardwired to 0 if the PowerBrake feature is not supported.	RW	0
PPFT_EN	[1]	For saving previous PFT_EN state on entering a trap. This field is hardwired to 0 if the PowerBrake feature is not supported.	RW	0
DME	[4]	Data Machine Error flag. It indicates an exception occurred at the data cache or data local memory (DLM). It will be set by Data parity/ECC error exceptions.	RW	0
PDME	[5]	For saving previous DME state on entering a trap. This field is hardwired to 0 if data cache and data local memory are not supported.	RW	0

18.3.11 Machine Detailed Trap Cause

Mnemonic Name: mdcause

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x7c9 (non-standard read/write)

31		2 1	0
	0		MDCAUSE

Field Name	Bits	Description	Type	Reset
MDCAUSE	[1:0]	This register further disambiguates causes of traps recorded in the mcause register:	RW	0

- When mcause == 1 (Instruction access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP instruction access violation
3	Bus error

- When mcause == 2 (Illegal instruction)

Value	Meaning
0	Please parse mtval CSR
1	FP disabled exception
2	ACE disabled exception

- When mcause == 5 (Load access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP load access violation
3	Bus error

- When mcause == 7 (Store access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP store access violation
3	Bus error

- For other exceptions and interrupts, this register will not be updated.

18.4 Counter Related CSRs

18.4.1 Machine Cycle Counter

Mnemonic Name: mcycle and mcycleh

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xb00 and 0xb80 (standard read/write)

The mcycle CSR counts the number of cycles that the hart has executed since some arbitrary time in the past. The mcycle register has 64-bit precision. For RV32 processors like N25(F), reads of the mcycle CSR return the lower 32 bits, while reads of the mcycleh CSR return bits 63–32 of the mcycle CSR.

18.4.2 Machine Instruction-Retired Counter

Mnemonic Name: minstret and minstreth

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xb02 and 0xb82 (standard read/write)

The minstret CSR counts the number of instructions that the hart has retired since some arbitrary time in the past. The minstret register has 64-bit precision. For RV32 processors like N25(F), reads of the minstret CSR return the low 32 bits, while reads of the minstreth CSR return bits 63–32 of the minstret CSR.

18.4.3 Machine Performance Monitoring Counter

Mnemonic Name: mhpmcounter3–mhpmcounter6 and mhpmcounter3h–mhpmcounter6h

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xb03 to 0xb06 and 0xb83 to 0xb86 (standard read/write)

The mhpmcounter3-mhpmcounter6 and mhpmcounter3h-mhpmcounter6h CSRs count the number of selected events, which are selected by mhpmevent3-mhpmevent6.

18.4.4 Machine Performance Monitoring Event Selector

Mnemonic Name: mhpmevent3–mhpmevent6

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x323 to 0x326 (standard read/write)

The event selectors are defined in Table 18-16 Event Selectors. Micro-architectural events are mostly speculative in nature. The counted

events could include events caused by speculative actions, unless they are defined to be non-speculative in the comment section. In particular, retired instruction counts are non-speculative counts.

Table 18-16 Event Selectors

TYPE	SEL	Event Name	Comment
0	1	Cycle count	Number of elapsed processor clock cycles.
0	2	Retired instruction count	Number of retired instructions.
0	3	Integer load instruction count	Number of retired load instructions (including LR).
0	4	Integer store instruction count	Number of retired store instructions (including SC).
0	5	Atomic instruction count	Number of retired atomic instructions (LR and SC not included).
0	6	System instruction count	Number of retired SYSTEM instructions (instructions with major opcode equal to 0b1110011).
0	7	Integer computational instruction count	Number of retired integer computational instructions.
0	8	Conditional branch instruction count	Number of retired conditional branch instructions.
0	9	Taken conditional branch instruction count	Number of retired conditional branch instructions that are taken.
0	10	JAL instruction count	Number of retired JAL instructions.
0	11	JALR instruction count	Number of retired JALR instructions. This event selector also counts the events monitored by the *return instruction count* event selector defined in the next row.
0	12	Return instruction count	Number of retired return instructions. Return instructions are `JALR` instructions with zero immediate offset and the following operands: • (rd != x1/x5) and (rs1 == x1/x5) • rd == x1 and rs1 == x5 • rd == x5 and rs1 == x1
0	13	Control transfer instruction count	Number of retired unconditional jumps (`JAL` and `JALR`) and conditional branch instructions.
0	14	`EXEC.IT` instruction count	Number of retired `EXEC.IT` instructions.
0	15	Integer multiplication instruction count	Number of retired integer multiplication instructions.
0	16	Integer division instruction count	Number of retired integer division/remainder instructions.
0	17	Floating-point load instruction count	Number of retired floating-point load instructions.
0	18	Floating-point store instruction count	Number of retired floating-point store instructions.

TYPE	SEL	Event Name	Comment
0	19	Floating-point addition instruction count	Number of retired floating-point addition/subtraction instructions.
0	20	Floating-point multiplication instruction count	Number of retired floating-point multiplication instructions.
0	21	Floating-point fused multiply-add instruction count	Number of retired floating-point fused multiply-add/subtraction instructions (`FMADD`, `FMSUB`, `FNMSUB`, `FNMADD`).
0	22	Floating-point division or square-root instruction count	Number of retired floating-point division/square-root instructions.
0	23	Other floating-point instruction count	Number of retired floating-point instructions not counted by the previous floating-point instruction event selectors.
1	0	ILM access	Number of ILM transfers, including speculative instruction fetch, load/store accesses, ECC repair and slave port accesses.
1	1	DLM access	Number of DLM transfers, including speculative load/store accesses, ECC repair and slave port accesses.
1	2	I-Cache access	Number of completed ICACHE fetch access.
1	3	I-Cache miss	Number of ICACHE fetch miss.
1	4	D-Cache access	Number of completed DCACHE load-and-store access. Misaligned load/store accesses might increase this counter by either one or two, depending on access sizes and alignments. Only misaligned accesses crossing two cache lines are guaranteed to result in increment of two.
1	5	D-Cache miss	The event counts the number of DCACHE load-and-store miss. Misaligned load/store accesses might increase this counter by either zero, one or two, depending on access sizes, alignments and whether the accessed lines are in D-Cache.
1	6	D-Cache load access	Number of completed DCACHE load access. See the D-Cache access count event selector for the handling of misaligned load accesses.

TYPE	SEL	Event Name	Comment
1	7	D-Cache load miss	Number of DCACHE load miss. See the D-Cache miss count event selector for the handling of misaligned load accesses.
1	8	D-Cache store access	Number of completed DCACHE store access. See the D-Cache access count event selector for the handling of misaligned load accesses.
1	9	D-Cache store miss	Number of DCACHE store miss. See the D-Cache miss count event selector for the handling of misaligned load accesses.
1	10	D-Cache writeback	Number of D-Cache writeback.
1	11	Cycles waiting for I-Cache fill data	Number of cycles waiting for the return of the critical word of I-Cache misses from the system bus. This event selector does not monitor accesses to I/O regions or accesses to cacheable regions when I-Cache is turned off.
1	12	Cycles waiting for D-Cache fill data	Number of cycles waiting for the return of the critical word of D-Cache misses from the system bus. This event selector does not monitor accesses to I/O regions or accesses to cacheable regions when D-Cache is turned off.
1	13	Uncached fetch data access from bus	Number of cycles waiting for the instruction data to return from the system bus. This event selector monitors accesses to I/O regions or accesses to cacheable regions when I-Cache is not configured or off.
1	14	Uncached load data access from bus	Number of cycles waiting for the load data to return from the system bus. This event selector monitors accesses to I/O regions or accesses to cacheable regions when D-Cache is not configured or off.
1	15	Cycles waiting for uncached fetch data from bus	Number of accesses of uncached instruction data returning from the system bus. This event selector monitors accesses to I/O regions or accesses to cacheable regions when I-Cache is not configured or off.

TYPE	SEL	Event Name	Comment
1	16	Cycles waiting for uncached load data from bus	Number of accesses of uncached load data returning from the system bus. This event selector monitors accesses to I/O regions or accesses to cacheable regions when D-Cache is not configured or off.
2	0	Misprediction of conditional branches (direction)	Number of misprediction of committed conditional branches.
2	1	Misprediction of taken conditional branches (direction)	Number of misprediction of committed taken conditional branches.
2	2	Misprediction of targets of Return instructions	Number of misprediction of committed Return instruction.
2	3	Replay for load-after-store or store-after-store cases	A load-after-store replay happens when a load hits a prior store in the pipeline with overlapping addresses. A store-after-store replay happens when a store hits a prior store in the pipeline with overlapping addresses.

18.4.5 Machine Counter Enable

Mnemonic Name: mcounteren

IM Requirement: Required if User mode is implemented

Access Mode: Machine

CSR Address: 0x306 (standard read/write)

31					7	6	5	4	3	2	1	0
0						HPM6	HPM5	HPM4	HPM3	IR	TM	CY

The machine counter-enable register controls the availability of the hardware performance monitoring counters to the next-lowest privileged mode. The default value of this register is 0.

When CY, TM, IR, HPM3, HPM4, HPM5, or HPM6 in the mcounteren register is clear, attempts to read the cycle, time, instret, hpmcounter3, hpmcounter4, hpmcounter5, or hpmcounter6 registers while executing in U-mode (M/U configuration) or S-mode (M/S/U configuration) will cause an illegal instruction exception. When one of these bits is set, accessing to the corresponding register is permitted in the next implemented privilege mode.

18.4.6 Machine Counter Write Enable

Mnemonic Name: mcounterwen

IM Requirement: mmsc_cfg.PMNDS == 1 and misa[20] == 1

Access Mode: Machine

CSR Address: 0x7CE (non-standard read/write)

31		7	6	5	4	3	2	1	0
	0		HPM6	HPM5	HPM4	HPM3	IR	0	CY

The machine counter write enable register controls the permission of writing the hardware performance monitoring counters in the next-lowest privileged mode and M-mode itself. The default value of this register is 0.

When CY, IR, HPM3, HPM4, HPM5, or HPM6 in the mcounterwen register is clear, attempts to write the cycle, time, instret, hpmcounter3, hpmcounter4, hpmcounter5, or hpmcounter6 registers while executing in U-mode (M/U configuration) or S-mode (M/S/U configuration) will cause an illegal instruction exception. When one of these bits is set, writing to the corresponding register is permitted in the next implemented privilege mode.

18.4.7 Machine Counter Interrupt Enable

Mnemonic Name: mcounterinten

IM Requirement: mmsc_cfg.PMNDS == 1

Access Mode: Machine

CSR Address: 0x7CF (non-standard read/write)

31		7	6	5	4	3	2	1	0
	0		HPM6	HPM5	HPM4	HPM3	IR	0	CY

The machine counter interrupt enable register controls whether a counter overflow interrupt is generated or not. The default value of this register is 0.

When CY, IR, HPM3, HPM4, HPM5, or HPM6 in the mcounterinten register is clear, no overflow interrupt is generated for the corresponding counter. When one of these bits is set, an interrupt will be generated when the corresponding counter overflows (the counter value wraps around back to 0).

18.4.8 Machine Counter Mask for Machine Mode

Mnemonic Name: mcountermask_m

IM Requirement: mmsc_cfg.PMNDS == 1 and misa[20] == 1

Access Mode: Machine

CSR Address: 0x7D1 (non-standard read/write)

31		7	6	5	4	3	2	1	0
	0		HPM6	HPM5	HPM4	HPM3	IR	0	CY
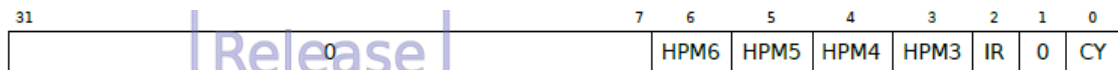
The machine counter mask for M-mode register controls the performance counter behavior in Mmode. The default value of this register is 0.

When CY, IR, HPM3, HPM4, HPM5, or HPM6 in the mcountermask_m register is set, the specific counter will not be incremented in M-mode.

The setting in this register also controls the privileged mode of the overflow local interrupt when the corresponding counter overflows for the M/S/U configuration: For any bit in this register, if the corresponding counter overflows, an M-mode interrupt is triggered if the bit is zero and an S-mode interrupt is triggered if the bit is one.

On the other hand, a counter overflow will always generate an M-mode interrupt for the M/U configuration, regardless of the settings in this register.

18.4.9 Machine Counter Mask for Supervisor Mode

Mnemonic Name: mcountermask_s

IM Requirement: mmsc_cfg.PMNDS == 1 and misa[18] == 1

Access Mode: Machine

CSR Address: 0x7D2 (non-standard read/write)

31		7	6	5	4	3	2	1	0
	0		HPM6	HPM5	HPM4	HPM3	IR	0	CY
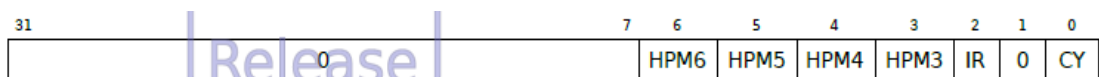
The machine counter mask for S-mode register controls the performance counter behavior in S-mode. The default value of this register is 0.

18.4.10 Machine Counter Mask for User Mode

Mnemonic Name: mcountermask_u

IM Requirement: mmsc_cfg.PMNDS == 1 and misa[20] == 1

Access Mode: Machine

CSR Address: 0x7D3 (non-standard read/write)

31		7	6	5	4	3	2	1	0
	0		HPM6	HPM5	HPM4	HPM3	IR	0	CY

The machine counter mask for U-mode register controls the performance counter behavior in U-mode. The default value of this register is 0.

18.4.11 Machine Counter Overflow Status

Mnemonic Name: mcounterovf

IM Requirement: mmsc_cfg.PMNDS == 1

Access Mode: Machine

CSR Address: 0x7D4 (non-standard read/write)

31		7	6	5	4	3	2	1	0
	0		HPM6	HPM5	HPM4	HPM3	IR	0	CY

The machine counter overflow status register records the overflow status of performance counters. When a bit is set, it indicates that an overflow has happened to the corresponding counter. Write 1 to each bit will clear the overflow state for the corresponding counter.

18.5 Configuration Control & Status Registers

18.5.1 Instruction Cache/Memory Configuration Register

Mnemonic Name: micm_cfg

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xfc0 (non-standard read only)

31 25	24	23	22 21	20	19 15	14 12	11 10	9	8 6	5 3	2 0
0	SETH	ILM_XONLY	ILM_ECC	0	ILMSZ	ILMB	IC_ECC	ILCK	ISZ	IWAY	ISET

This register provides information about configurations of instruction cache and instruction memory.

Field Name	Bits	Description		Type	Reset
ISET	[2:0]	I-Cache sets (# of cache lines per way):		RO	IM
		Value	Meaning		
		0	64		
		1	128		
		2	256		
		3	512		
		4	1024		
		5	2048		
		6	4096		
		7	Reserved		
IWAY	[5:3]	Associativity of I-Cache:		RO	IM
		Value	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		

Field Name	Bits	Description	Type	Reset
ISZ	[8:6]	I-Cache block (line) size:	RO	IM

Value	Meaning
0	No I-Cache
1	8 bytes
2	16 bytes
3	32 bytes
4	64 bytes
5	128 bytes
6,7	Reserved

Field Name	Bits	Description	Type	Reset
ILCK	[9]	I-Cache locking support:	RO	IM

Value	Meaning
0	No locking support
1	With locking support

Field Name	Bits	Description	Type	Reset
IC_ECC	[11:10]	I-Cache soft-error protection scheme:	RO	IM

Value	Meaning
0	No parity/ECC
1	Parity
2	ECC
3	Reserved

Field Name	Bits	Description	Type	Reset
ILMB	[14:12]	Number of ILM base registers present:	RO	IM

Value	Meaning
0	No ILM base register present
1	One ILM base register present
2-7	Reserved

Field Name	Bits	Description		Type	Reset
ILMSZ	[19:15]	ILM Size:		RO	IM

Value	Meaning
0	0 Byte
1	1 KiB
2	2 KiB
3	4 KiB
4	8 KiB
5	16 KiB
6	32 KiB
7	64 KiB
8	128 KiB
9	256 KiB
10	512 KiB
11	1 MiB
12	2 MiB
13	4 MiB
14	8 MiB
15	16 MiB
16-31	Reserved

Field Name	Bits	Description		Type	Reset
ILM_ECC	[22:21]	ILM soft-error protection scheme:		RO	IM

Value	Meaning
0	No parity/ECC
1	Parity
2	ECC
3	Reserved

Field Name	Bits	Description		Type	Reset
ILM_XONLY	[23]	ILM soft-error protection scheme		RO	IM

Value	Meaning
0	ILM is not execute-only
1	ILM is execute only

Field Name	Bits	Description	Type	Reset
SETH	[24]	This bit extends the `ISET` field.	RO	IM

18.5.2 Data Cache/Memory Configuration Register

Mnemonic Name: mdcm_cfg

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xfc1 (non-standard read only)

31	25	24	23	22	21	20	19	15	14	12	11	10	9	8	6	5	3	2	0
0		SETH		0	DLM_ECC		0	DLMSZ		DLMB		DC_ECC		DLCK		DSZ	DWAY		DSET

This register provides information about the configurations of data cache and data memory.

Field Name	Bits	Description	Type	Reset
DSET	[2:0]	D-Cache sets (# of cache lines per way):	RO	IM

When `mdcm_cfg.SETH==0`:

Value	Meaning
0	64
1	128
2	256
3	512
4	1024
5	2048
6	4096
7	Reserved

When `mdcm_cfg.SETH==1`:

Value	Meaning
0	32
1	16
2	8
3~7	Reserved

Field Name	Bits	Description	Type	Reset
DWAY	[5:3]	Associativity of D-Cache	RO	IM

Value	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

Field Name	Bits	Description	Type	Reset
DSZ	[8:6]	D-Cache block (line) size:	RO	IM

Value	Meaning
0	No D-Cache
1	8 bytes
2	16 bytes
3	32 bytes
4	64 bytes
5	128 bytes
6,7	Reserved

Field Name	Bits	Description	Type	Reset
DLCK	[9]	D-Cache locking support:	RO	IM

Value	Meaning
0	No locking support
1	With locking support

Field Name	Bits	Description	Type	Reset
DC_ECC	[11:10]	D-Cache soft-error protection scheme:	RO	IM

Value	Meaning
0	No parity/ECC support
1	Has parity support
2	Has ECC support
3	Reserved

Field Name	Bits	Description	Type	Reset
DLMB	[14:12]	Number of DLM base registers present:	RO	IM

Value	Meaning
0	No DLM base register present
1	One DLM base register present
2-7	Reserved

Field Name	Bits	Description		Type	Reset
DLMSZ	[19:15]	DLM Size:		RO	IM

Value	Meaning
0	0 Byte
1	1 KiB
2	2 KiB
3	4 KiB
4	8 KiB
5	16 KiB
6	32 KiB
7	64 KiB
8	128 KiB
9	256 KiB
10	512 KiB
11	1 MiB
12	2 MiB
13	4 MiB
14	8 MiB
15	16 MiB
16-31	Reserved

Field Name	Bits	Description		Type	Reset
DLM_ECC	[22:21]	DLM soft-error protection scheme:		RO	IM

Value	Meaning
0	No parity/ECC
1	Parity
2	ECC
3	Reserved

Field Name	Bits	Description	Type	Reset
SETH	[24]	This bit extends the DSET field.	RO	IM

18.5.3 Misc. Configuration Register

Mnemonic Name: mmsc_cfg

IM Requirement: Required

Access Mode: Machine

CSR Address: 0xfc2 (non-standard read only)

31 18	17	16	15	14	13	12	11 7	6	5	4	3 2	1 0
Reserved	EFHW	CCTLCSR	PMNDS	LMSLVP	EV5MPE	VPLIC	0	ACE	HSP	PFT	ECD	TLB_ECC	ECC

This register provides information regarding miscellaneous processor configurations.

Field Name	Bits	Description	Type	Reset
ECC	[0]	Indicates whether the parity/ECC soft-error protection is implemented or not.	RO	IM
		<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not implemented.</td></tr><tr><td>1</td><td>Implemented.</td></tr></table> The specific parity/ECC scheme used for each protected RAM is specified by the control bits in the following list. • micm_cfg.IC_ECC • micm_cfg.ILM_ECC • mdcm_cfg.DC_ECC • mdcm_cfg.DLM_ECC • mmsc_cfg.TLB_ECC		
TLB_ECC	[2:1]	TLB parity/ECC support configuration.	RO	IM
		<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No parity/ECC support.</td></tr><tr><td>1</td><td>Has parity support.</td></tr><tr><td>2</td><td>Has ECC support.</td></tr><tr><td>3</td><td>Reserved.</td></tr></table>		
ECD	[3]	Indicates whether the Andes CoDense Extension is implemented or not.	RO	IM
		<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Not implemented.</td></tr><tr><td>1</td><td>Implemented.</td></tr></table>		

Field Name	Bits	Description	Type	Reset
PFT	[4]	Indicates whether the Andes PowerBrake (Performance Throttling) power/performance scaling extension is implemented or not. <table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Not implemented.</td></tr><tr><td>1</td><td>Implemented.</td></tr></table>	RO	IM
HSP	[5]	Indicates whether the Andes StackSafe hardware stack protection extension is implemented or not. <table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Not implemented.</td></tr><tr><td>1</td><td>Implemented.</td></tr></table>	RO	IM
ACE	[6]	Indicates whether the Andes Custom Extension is implemented or not. <table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Not implemented.</td></tr><tr><td>1</td><td>Implemented.</td></tr></table>	RO	IM
VPLIC	[12]	Indicates whether the Andes Vectored PLIC Extension is implemented or not. <table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Not implemented.</td></tr><tr><td>1</td><td>Implemented.</td></tr></table>	RO	IM
EV5PE	[13]	Indicates whether AndeStar V5 Performance Extension is implemented or not. N25(F) always implements AndeStar V5 Performance Extension.	RO	1

Field Name	Bits	Description	Type	Reset
LMSLVP	[14]	Indicates if local memory slave port is present or not.	RO	IM

Value	Meaning
0	Local memory slave port is not present.
1	Local memory slave port is implemented.

Note that atomicity of atomic instructions accessing local memory address space is not guaranteed if external masters modify the same data through the local memory slave port.

Field Name	Bits	Description	Type	Reset
PMNDS	[15]	Indicate if Andes-enhanced performance monitoring feature is present or not.	RO	IM

Value	Meaning
0	Andes-enhanced performance monitoring feature is not supported.
1	Andes-enhanced performance monitoring feature is supported.

Field Name	Bits	Description	Type	Reset
CCTLCSR	[16]	Indicate the presence of CSRs for CCTL operations.	RO	IM

Value	Meaning
0	Feature of CSRs for CCTL operations is not supported.
1	Feature of CSRs for CCTL operations is supported.

Field Name	Bits	Description	Type	Reset
EFHW	[17]	Indicate the support of FLHW and FSHW instructions.	RO	IM

Value	Meaning
0	FLHW and FSHW instructions are not supported.
1	FLHW and FSHW instructions are supported.

18.6 Trigger Registers

18.6.1 Trigger Select

Mnemonic Name: tselect

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a0 (standard read/write)

31	0
TRIGGER_INDEX	
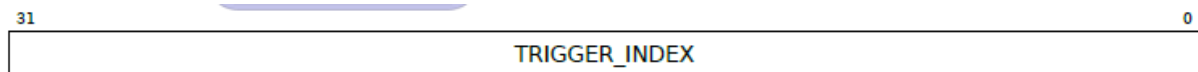
This register determines which trigger is accessible through other trigger registers. The set of accessible triggers must start at 0, and be contiguous. Writes of values greater than or equal to the number of supported triggers might result in a different value in this register than what was written. Debuggers should read back the value to confirm that what they wrote was a valid index.

Since triggers can be used both by Halt mode and Machine mode, the debugger must restore this register after the modification.

18.6.2 Trigger Data 1

Mnemonic Name: tdata1

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a1 (standard read/write)

31	28	27	26	0
TYPE		DMODE	DATA	
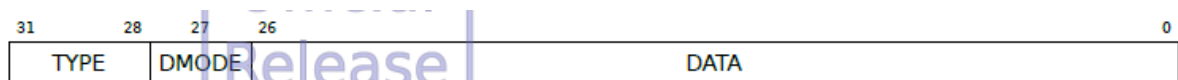
This register provides access to the tdata1 register of the currently selected trigger registers selected by the tselect register.

Field Name	Bits	Description		Type	Reset
DATA	[26:0]	Trigger-specific data		RW	0
DMODE	[27]	Setting this field to indicate the trigger is used by Debug Mode.		RW	0
		Value	Meaning		
		0	Both Debug-mode and M-mode can write the currently selected trigger registers.		
		1	Only Debug Mode can write the currently selected trigger registers. Writes from other modes are ignored.		
TYPE	[31:28]	Indicating the trigger type.		RW	2
		Value	Meaning		
		0	The selected trigger is invalid.		
		2	The selected trigger is an address/data match trigger.		
		3	The selected trigger is an instruction count trigger.		
		4	The selected trigger is an interrupt trigger.		
		5	The selected trigger is an exception trigger.		

18.6.3 Trigger Data 2

Mnemonic Name: tdata2

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a2 (standard read/write)

This register provides access to the tdata2 register of the currently selected trigger registers selected by the tselect register, and it holds trigger-specific data.

18.6.4 Trigger Data 3

Mnemonic Name: tdata3

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a3 (standard read/write)

This register provides access to the tdata3 register of the currently selected trigger registers selected by the tselect register, and it holds trigger-specific data.

18.6.5 Trigger Info

Mnemonic Name: tinfo

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a4 (standard read/write)

31	16	15	0
0		INFO	
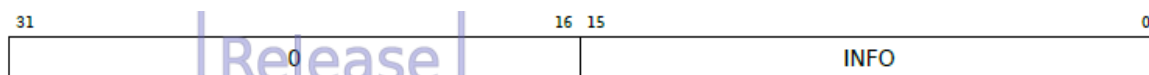
This register provides access to the tinfo register of the currently selected trigger registers selected by the tselect register, and it indicates the supported trigger types of the currently selected trigger.

Field Name	Bits	Description	Type	Reset
INFO	[15:0]	One bit for each possible TYPE in tdata1. Bit N corresponds to type N. If the bit is set, then that type is supported by the currently selected trigger. If the currently selected trigger does not exist, this field contains 1.	RW	0

INFO[2] = 1.
INFO[3] = 1, on trigger 0 or 1 (tselect = 0 or 1).
INFO[4] = 1, on trigger 0 (tselect = 0) or
on trigger 4 (tselect = 4) when Number of Triggers is 8.
INFO[5] = 1, on trigger 1 (tselect = 1) when Number of Triggers is 2 or
on trigger 3 (tselect = 3) when Number of Triggers is 4 or
on trigger 3 or 7 (tselect = 3 or 7) when Number of Triggers is 8.

18.6.6 Trigger Contorl

Mnemonic Name: tcontrol
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a5 (standard read/write)

31	8	7	6	4	3	2	0
0		MPTE	0		MTE	0	
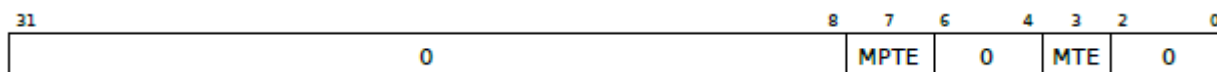
This register provides accesses to the tcontrol register, and it indicates the current native M-Mode debugging settings.

Field Name	Bits	Description	Type	Reset
MTE	[3]	M-mode trigger enable field. When a trap into M-mode is taken, MTE is set to 0. When the MRET instruction is executed, MTE is set to the value of MPTE.	RW	0
MPTE	[7]	M-mode previous trigger enable field. When a trap into M-mode is taken, MPTE is set to the value of MTE.	RW	0

Value	Meaning
0	Triggers do not match/fire while the hart is in M-mode.
1	Triggers do match/fire while the hart is in M-mode.

18.6.7 Machine Context

Mnemonic Name: mcontext
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a8 (standard read/write)

31	6 5	0
0	MCONTEXT	

This register provides access to the mcontext register.

Field Name	Bits	Description	Type	Reset
MCONTEXT	[5:0]	Machine mode software can write a context number to this register, which can be used to set triggers that only fire in that specific context.	RW	0

18.6.8 Match Control

Mnemonic Name: mcontrol

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a1 (standard read/write)

This register is accessible as tdata1 when type is 0 or 2.

31 28	27	26 21	20 18	17 12	11	10 7	6	5	4	3	2	1	0
TYPE	DMODE	MASKMAX	0	ACTION	CHAIN	MATCH	M	0	S	U	EXECUTE	STORE	LOAD

Field Name	Bits	Description	Type	Reset
LOAD	[0]	Setting this field to enable this trigger to compare virtual address of a load	RW	0
STORE	[1]	Setting this field to enable this trigger to compare virtual address of a store	RW	0
EXECUTE	[2]	Setting this field to enable this trigger to compare virtual address of an instruction	RW	0
U	[3]	Setting this field to enable this trigger in U-mode.	RW	0
S	[4]	Setting this field to enable this trigger in S-mode.	RW	0
M	[6]	Setting this field to enable this trigger in M-mode.	RW	0
MATCH	[10:7]	Setting this field to select matching scheme.	RW	0

Value	Meaning
0	Matches when the value equals tdata2
1	Matches when the top M bits of the value match the top M bits of tdata2. M is 31 minus the index of the least-significant bit containing 0 in tdata2.
2	Matches when the value is greater than (unsigned) or equal to tdata2.
3	Matches when the value is less than (unsigned) tdata2.

Field Name	Bits	Description	Type	Reset
CHAIN	[11]	Setting this field to enable trigger chain.	RW	0

Value	Meaning
0	When this trigger matches, the configured action is taken.
1	While this trigger does not match, it prevents the trigger with the next index from matching.

If Number of Triggers is 2, this field is hardwired to 0 on trigger 1 (`tselect` = 1).

If Number of Triggers is 4, this field is hardwired to 0 on trigger 3 (`tselect` = 3).

If Number of Triggers is 8, this field is hardwired to 0 on trigger 3 and trigger 7 (`tselect` = 3 or 7).

ACTION	[17:12]	Setting this field to select what happens when this trigger matches.	RW	0

Value	Meaning
0	Raise a breakpoint exception.
1	Enter Debug Mode. (Only supported when DMODE is 1.)

MASKMAX	[26:21]	Indicating the largest naturally aligned range supported by the hardware is 2^12 bytes.	RO	12
DMODE	[27]	Setting this field to indicate the trigger is used by Debug Mode.	RW	0

Value	Meaning
0	Both Debug-mode and M-mode can write the currently selected trigger registers.
1	Only Debug Mode can write the currently selected trigger registers. Writes from other modes are ignored.

TYPE	[31:28]	Indicating the trigger type.	RW	2

Value	Meaning
0	The selected trigger is invalid.
2	The selected trigger is an address/data match trigger.

Note

The LOAD/STORE fields take no effect and are cleared if the EXECUTE field is set at the same time.

18.6.9 Instruction Count

Mnemonic Name: icount

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a1 (standard read/write)

This register is accessible as tdata1 when type is 3.

This register exists just for single-stepping support so COUNT is hard-wired to 1. After this trigger fires, the mode bits (M, S, U bits) will be cleared instead of decrementing the COUNT bits.

31	28	27	26		11	10	9	8	7	6	5		0
TYPE		DMODE		0		COUNT	M	0	S	U		ACTION	

Field Name	Bits	Description	Type	Reset
ACTION	[5:0]	Setting this field to select what happens when this trigger matches.	RW	0

Value	Meaning
0	Raise a breakpoint exception.
1	Enter Debug Mode. (Only supported when DMODE is 1.)

Field Name	Bits	Description	Type	Reset
U	[6]	Setting this field to enable this trigger in U-mode.	RW	0
S	[7]	Setting this field to enable this trigger in S-mode.	RW	0
M	[9]	Setting this field to enable this trigger in M-mode.	RW	0
COUNT	[10]	This field is hard-wired to 1 for single-stepping support	RO	1
DMODE	[27]	Setting this field to indicate the trigger is used by Debug Mode.	RW	0

Value	Meaning
0	Both Debug-mode and M-mode can write the currently selected trigger registers.
1	Only Debug Mode can write the currently selected trigger registers. Writes from other modes are ignored.

Field Name	Bits	Description	Type	Reset
TYPE	[31:28]	The selected trigger is an instruction count trigger.	RW	3

18.6.10   Interrupt Trigger

Mnemonic Name: itrigger

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a1 (standard read/write)

This register is accessible as tdata1 when TYPE is 4.

This trigger may fire on any of the interrupts configurable in mie. The interrupts to fire on are configured by setting the same bit in tdata2 as would be set in mie to enable the interrupt.

31 28	27 26		10	9	8	7	6	5 0
TYPE	DMODE	0	M	0	S	U	ACTION

Field Name	Bits	Description		Type	Reset
ACTION	[5:0]	Setting this field to select what happens when this trigger matches.		RW	0
		Value	**Meaning**		
		0	Raise a breakpoint exception.		
		1	Enter Debug Mode. (Only supported when DMODE is 1.)		
U	[6]	Setting this field to enable this trigger in U-mode.		RW	0
S	[7]	Setting this field to enable this trigger in S-mode.		RW	0
M	[9]	Setting this field to enable this trigger in M-mode.		RW	0
DMODE	[27]	Setting this field to indicate the trigger is used by Debug Mode.		RW	0
		Value	**Meaning**		
		0	Both Debug-mode and M-mode can write the currently selected trigger registers.		
		1	Only Debug Mode can write the currently selected trigger registers. Writes from other modes are ignored.		
TYPE	[31:28]	The selected trigger is an interrupt trigger.		RW	4

18.6.11 Exception Trigger

Mnemonic Name: etrigger

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug and Machine

CSR Address: 0x7a1 (standard read/write)

This register is accessible as tdata1 when TYPE is 5.

This trigger may fire on up to XLEN of the Exception Codes defined in mcause (with Interrupt=0). Those causes are configured by writing the corresponding bit in tdata2.

31 28	27	26 10	9	8	7	6	5 0
TYPE	DMODE	0	M	0	S	U	ACTION

Field Name	Bits	Description		Type	Reset
ACTION	[5:0]	Setting this field to select what happens when this trigger matches.		RW	0
		Value	**Meaning**		
		0	Raise a breakpoint exception.		
		1	Enter Debug Mode. (Only supported when DMODE is 1.)		
U	[6]	Setting this field to enable this trigger in U-mode.		RW	0
S	[7]	Setting this field to enable this trigger in S-mode.		RW	0
M	[9]	Setting this field to enable this trigger in M-mode.		RW	0
DMODE	[27]	Setting this field to indicate the trigger is used by Debug Mode.		RW	0
		Value	**Meaning**		
		0	Both Debug-mode and M-mode can write the currently selected trigger registers.		
		1	Only Debug Mode can write the currently selected trigger registers. Writes from other modes are ignored.		
TYPE	[31:28]	The selected trigger is an exception trigger.		RW	5

18.7 Debug and Trigger Registers

18.7.1 Debug Control and Status Register

Mnemonic Name: dcsr

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug

CSR Address: 0x7b0 (debug-mode-only)

15	14	13	12	11	10	9	8	6 5	3	2	1	0
EBREAKM	0		EBREAKU	0	STOPCOUNT	STOPTIME	CAUSE		0	STEP		PRV

31	28	27		16
XDEBUGVER			0	

Field Name	Bits	Description	Type	Reset
PRV	[1:0]	The privilege level that the hart was operating in when Debug Mode was entered. The external debugger can modify this value to change the hart's privilege level when exiting Debug Mode.	RW	0
STEP	[2]	This bit controls whether non-Debug Mode instruction execution is in the single step mode. When set, the hart returns to Debug Mode after a single instruction execution. If the instruction does not complete due to an exception, the hart will immediately enter Debug Mode before executing the trap handler, with appropriate exception registers set.	RW	0

PRV:

Value	Meaning
0	User/Application
1	Reserved
2	Reserved
3	Machine

STEP:

Value	Meaning
0	Single Step Mode is off
1	Single Step Mode is on

Field Name	Bits	Description	Type	Reset
CAUSE	[8:6]	Reason why Debug Mode was entered. When there are multiple reasons to enter Debug Mode, the priority to determine the CAUSE value will be: single step > halt request > ebreak > trigger module. Halt requests are requests issued by the external debugger.	RO	0

Value	Meaning
0	Reserved
1	ebreak
2	Trigger module
3	Halt request
4	Single step
5–7	Reserved

Field Name	Bits	Description	Type	Reset
STOPTIME	[9]	This bit controls whether timers are stopped in Debug Mode. The processor only drives its stoptime output pin to 1 if it is in Debug Mode and this bit is set. Integration effort is required to make timers in the platform observe this pin to really stop them.	RW	1

Value	Meaning
0	Do not stop timers in Debug Mode
1	Stop timers in Debug Mode

Field Name	Bits	Description	Type	Reset
STOPCOUNT	[10]	This bit controls whether performance counters are stopped in Debug Mode.	RW	1

Value	Meaning
0	Do not stop counters in Debug Mode
1	Stop counters in Debug Mode

Field Name	Bits	Description	Type	Reset
EBREAKU	[12]	This bit controls the behavior of `ebreak` instructions in User/Application Mode.	RW	0
EBREAKM	[15]	This bit controls the behavior of `ebreak` instructions in Machine Mode.	RW	0
XDEBUGVER	[31:28]	Version of the external debugger. 0 indicates that no external debugger exists and 4 indicates that the external debugger confirms to the *RISC-V External Debug Support (TD003) V0.13*.	RO	4

EBREAKU values:

Value	Meaning
0	Generate a regular breakpoint exception
1	Enter Debug Mode

EBREAKM values:

Value	Meaning
0	Generate a regular breakpoint exception
1	Enter Debug Mode

18.7.2 Debug Program Counter

Mnemonic Name: dpc

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug

CSR Address: 0x7b1 (debug-mode-only)

31		0
	DPC	
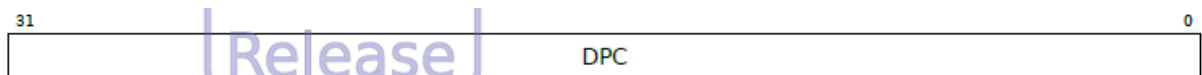
When entering Debug Mode, the dpc CSR is updated with the virtual address of the next instruction to be executed. The behavior is described in more detail in Table 18-17 Virtual Address in DPC upon Debug Mode Entry. When leaving Debug Mode, the hart's PC is updated to the value stored in this register. The external debugger may write this register to change where the hart resumes.

Field Name	Bits	Description	Type	Reset
DPC	[31:0]	Debug Program Counter. Bit 0 is hard-wired to 0.	RW	0

Table 18-17 Virtual Address in DPC upon Debug Mode Entry

Cause	Virtual Address in DPC
ebreak	Address of the `ebreak` instruction
single step	Address of the instruction that would be executed next if no debugging was going on.
trigger module	Address of the instruction which caused the trigger module to fire.
halt request	Address of the next instruction to be executed at the time that Debug Mode was entered

18.7.3 Debug Scratch Register 0

Mnemonic Name: dscratch0

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug

CSR Address: 0x7b2 (debug-mode-only)

31		0
	DSCRATCH0	
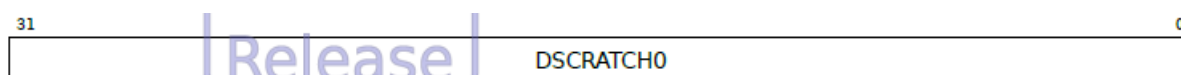
A scratch register that is reserved for use by Debug Module.

18.7.4 Debug Scratch Register 1

Mnemonic Name: dscratch1

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug

CSR Address: 0x7b3 (debug-mode-only)

31		0
	DSCRATCH1	
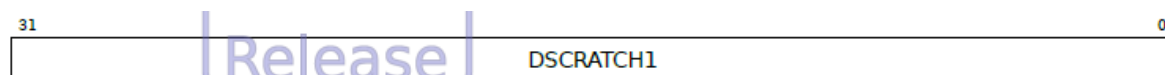
A scratch register that is reserved for use by Debug Module.

18.7.5 Exception Redirection Register

Mnemonic Name: dexc2dbg

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug

CSR Address: 0x7e0 (non-standard read/write)

31 20	19	18 16	15	14	13	12	11	10 9	8	7	6	5	4	3	2	1	0
0	PMOV	0	BWE	IMECC	ACE	HSP	MEC	0	UEC	SAF	SAM	LAF	LAM	NMI	II	IAF	IAM
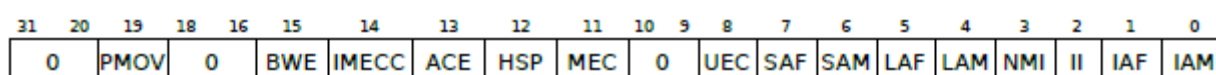
This register redirects selected exceptions to cause the hart to enter Debug Mode instead of performing the standard trap handling.

When an exception is redirected to enter Debug Mode, the dpc CSR will be updated with the virtual address of the instruction causing the exception. The dcsr.CAUSE field will be updated with a value of 1 (ebreak). The actual cause of the exception is saved to the ddcause CSR. The required updates to mepc, mcause, mtval, mstatus, and mxstatus CSRs for exceptions will not be affected by the redirection and these CSRs continue to provide information associated with the corresponding exceptions.

Field Name	Bits	Description	Type	Reset
IAM	[0]	Indicates whether Instruction Access Misaligned exceptions are redirected to enter Debug Mode.	RW	0
IAF	[1]	Indicates whether Instruction Access Fault exceptions are redirected to enter Debug Mode.	RW	0
II	[2]	Indicates whether Illegal Instruction exceptions are redirected to enter Debug Mode.	RW	0
NMI	[3]	Indicates whether Non-Maskable Interrupt exceptions are redirected to enter Debug Mode.	RW	0

IAM [0]:

Value	Meaning
0	Do not redirect
1	Redirect

IAF [1]:

Value	Meaning
0	Do not redirect
1	Redirect

II [2]:

Value	Meaning
0	Do not redirect
1	Redirect

NMI [3]:

Value	Meaning
0	Do not redirect
1	Redirect

Field Name	Bits	Description	Type	Reset
LAM	[4]	Indicates whether Load Access Misaligned exceptions are redirected to enter Debug Mode.	RW	0
LAF	[5]	Indicates whether Load Access Fault exceptions are redirected to enter Debug Mode.	RW	0
SAM	[6]	Indicates whether Store Access Misaligned exceptions are redirected to enter Debug Mode.	RW	0
SAF	[7]	Indicates whether Store Access Fault exceptions are redirected to enter Debug Mode.	RW	0
UEC	[8]	Indicates whether U-mode Environment Call exceptions are redirected to enter Debug Mode.	RW	0
MEC	[11]	Indicates whether M-mode Environment Call exceptions are redirected to enter Debug Mode.	RW	0

LAM:

Value	Meaning
0	Do not redirect
1	Redirect

LAF:

Value	Meaning
0	Do not redirect
1	Redirect

SAM:

Value	Meaning
0	Do not redirect
1	Redirect

SAF:

Value	Meaning
0	Do not redirect
1	Redirect

UEC:

Value	Meaning
0	Do not redirect
1	Redirect

MEC:

Value	Meaning
0	Do not redirect
1	Redirect

Field Name	Bits	Description	Type	Reset
HSP	[12]	Indicates whether Stack Protection exceptions are redirected to enter Debug Mode. This bit is present only when `mmsc_cfg.HSP` is set.	RW	0
ACE	[13]	Indicates whether ACE-related exceptions are redirected to enter Debug Mode. This bit is present only when `mmsc_cfg.ACE` is set.	RW	0
IMECC	[14]	Indicates whether Imprecise ECC Error local interrupts are redirected to enter Debug Mode.	RW	0
BWE	[15]	Indicates whether Bus-write Transaction Error local interrupts are redirected to enter Debug Mode.	RW	0
PMOV	[19]	Indicates whether performance counter overflow interrupts are redirected to enter Debug Mode.	RW	0

HSP:

Value	Meaning
0	Do not redirect
1	Redirect

ACE:

Value	Meaning
0	Do not redirect
1	Redirect

IMECC:

Value	Meaning
0	Do not redirect
1	Redirect

BWE:

Value	Meaning
0	Do not redirect
1	Redirect

PMOV:

Value	Meaning
0	Do not redirect
1	Redirect

18.7.6 Debug Detailed Cause

Mnemonic Name: ddcause

IM Requirement: DEBUG_SUPPORT

Access Mode: Debug

CSR Address: 0x7e1 (non-standard read/write)

31	16 15	8 7	0
0	SUBTYPE	MAINTYPE	
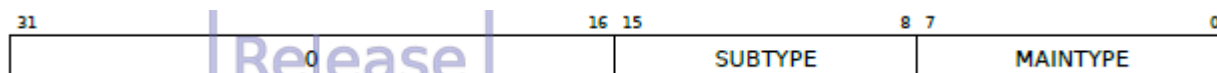
Field Name	Bits	Description	Type	Reset
MAINTYPE	[7:0]	Cause for redirection to Debug Mode.	RO	0

Value	Meaning
0	Software Breakpoint (EBREAK)
1	Instruction Access Misaligned (IAM)
2	Instruction Access Fault (IAF)
3	Illegal Instruction (II)
4	Non-Maskable Interrupt (NMI)
5	Load Access Misaligned (LAM)
6	Load Access Fault (LAF)
7	Store Access Misaligned (SAM)
8	Store Access Fault (SAF)
9	U-mode Environment Call (UEC)
10–11	Reserved
12	M-mode Environment Call (MEC)
13	Load page fault
14	Reserved
15	Store/AMO page fault
16	Imprecise ECC error
17	Bus write transaction error
18	Performance Counter overflow
19–31	Reserved
32	Stack overflow exception
33	Stack underflow exception
34	ACE disabled exception
35–39	Reserved
40–47	ACE exception
$\geq$48	Reserved

SUBTYPE	[15:8]	Subtypes for main type. The table below lists the subtypes for DCSR.CAUSE==1 and DDCAUSE.MAINTYPE==3.	RO	0

Value	Meaning
0	Illegal instruction
1	Privileged instruction
2	Non-existent CSR
3	Read-only CSR update
4	Privilege CSR access

18.8 Memory and Miscellaneous Registers

18.8.1 Instruction Local Memory Base Register

Mnemonic Name: milmb

IM Requirement: ILM_SIZE_KB > 0

Access Mode: Machine

CSR Address: 0x7c0 (non-standard read/write)

31 IBPA 10	9 0 4	3 RWECC	2 ECCEN	1 0 IEN

This register controls instruction local memory.

Field Name	Bits	Description	Type	Reset
IEN	[0]	ILM enable control:	RO	1

Value	Meaning
0	ILM is disabled
1	ILM is enabled

Field Name	Bits	Description	Type	Reset
ECCEN	[2:1]	Parity/ECC enable control:	RW	0

Value	Meaning
0	Disable parity/ECC
1	Reserved
2	Generate exceptions only on uncorrectable parity/ECC errors
3	Generate exceptions on any type of parity/ECC errors

RWECC	[3]	Controls diagnostic accesses of ECC codes of the ILM rams through the `mecc_code` register. This bit can be turned on for injecting ECC errors to test the ECC handler.	RW	0

Value	Meaning
0	Disable diagnostic accesses of ECC codes
1	Enable diagnostic accesses of ECC codes

IBPA	[31:10]	The base physical address of ILM. It has to be an integer multiple of the ILM size.	RO	ILM_BASE[31:10]

18.8.2 Data Local Memory Base Register

Mnemonic Name: mdlmb

IM Requirement: DLM_SIZE_KB > 0

Access Mode: Machine

CSR Address: 0x7c1 (non-standard read/write)

31 .... DBPA 10	9 4	3	2	1 0
DBPA	0	RWECC	ECCEN	DEN

This register controls data local memory.

Field Name	Bits	Description	Type	Reset
DEN	[0]	DLM enable control:	RO	1

Value	Meaning
0	DLM is disabled
1	DLM is enabled

Field Name	Bits	Description	Type	Reset
ECCEN	[2:1]	Parity/ECC enable control:	RW	0

Value	Meaning
0	Disable parity/ECC
1	Reserved
2	Generate exceptions only on uncorrectable parity/ECC errors
3	Generate exceptions on any type of parity/ECC errors

Field Name	Bits	Description	Type	Reset
RWECC	[3]	Controls diagnostic accesses of ECC codes of the DLM rams through the `mecc_code` register. This bit can be turned on for injecting ECC errors to test the ECC handler.	RW	0

Value	Meaning
0	Disable diagnostic accesses of ECC codes
1	Enable diagnostic accesses of ECC codes

Field Name	Bits	Description	Type	Reset
DBPA	[31:10]	The base physical address of DLM. It has to be an integer multiple of the DLM size.	RO	DLM_BASE[31:10]

18.8.3 ECC Code Register

Mnemonic Name: mecc_code

IM Requirement: mmsc_cfg.ECC == 1

Access Mode: Machine

CSR Address: 0x7c2 (non-standard read/write)

31	23	22	21	18	17	16	15	7	6	0
0		INSN	RAMID		P	C	0		CODE	

This register is used for accessing ECC array of ILM and DLM.

Field Name	Bits	Description	Type	Reset
CODE	[6:0]	This field records the ECC value on ECC error exceptions. This field is also used to read/write the ECC codes when diagnostic access of ECC codes are enabled (`milmb.RWECC` or `mdlmb.RWECC` is 1).	RW	1
C	[16]	Correctable error. This bit is updated on parity/ECC error exceptions.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Uncorrectable error</td></tr><tr><td>1</td><td>Correctable error</td></tr></table>	RO	0
P	[17]	Precise error. This bit is updated on parity/ECC error exceptions.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Imprecise error</td></tr><tr><td>1</td><td>Precise error</td></tr></table>	RO	0
RAMID	[21:18]	The ID of RAM that caused parity/ECC errors. This bit is updated on parity/ECC error exceptions.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0–1</td><td>Reserved</td></tr><tr><td>2</td><td>Tag RAM of I-Cache</td></tr><tr><td>3</td><td>Data RAM of I-Cache</td></tr><tr><td>4</td><td>Tag RAM of D-Cache</td></tr><tr><td>5</td><td>Data RAM of D-Cache</td></tr><tr><td>6</td><td>Tag RAM of TLB</td></tr><tr><td>7</td><td>Data RAM of TLB</td></tr><tr><td>8</td><td>ILM</td></tr><tr><td>9</td><td>DLM</td></tr><tr><td>10–15</td><td>Reserved</td></tr></table>	RO	0
INSN	[22]	Indicates if the parity/ECC error is caused by instruction fetch or data access.<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Data access</td></tr><tr><td>1</td><td>Instruction fetch</td></tr></table>	RO	0

18.8.4 NMI Vector Base Address Register

Mnemonic Name: mnvec

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x7c3 (non-standard read/write)

31	0
NMVEC	
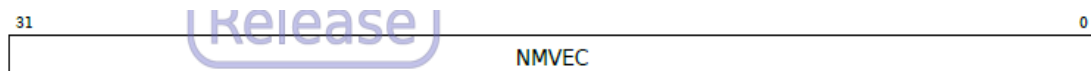
This register indicates the entry point when an NMI occurs.

Field Name	Bits	Description	Type	Reset
NMVEC	[31:0]	Handler base address for NMI. Its value is the zero extended value of the `reset_vector[31:0]` input signal to N25(F).	RO	Pin Configured

18.8.5 Performance Throttling Control Register

Mnemonic Name: mpft_ctl

IM Requirement: POWERBRAKE_SUPPORT = "yes"

Access Mode: Machine

CSR Address: 0x7c5 (non-standard read/write)

31	9	8	7	4	3	0
0		FAST_INT	T_LEVEL		0	

Field Name	Bits	Description	Type	Reset
T_LEVEL	[7:4]	Throttling Level. The processor has the highest performance at throttling level 0 and the lowest performance at throttling level 15.	RW	0

Value	Meaning
0	Level 0 (the highest performance)
1-14	Level 1-14
15	Level 15 (the lowest performance)

Field Name	Bits	Description	Type	Reset
FAST_INT	[8]	Fast interrupt response. If this field is set, MXSTATUS.PFT_EN will be automatically cleared when the processor enters an interrupt handler.	RW	0

18.8.6 Cache Control Register

Mnemonic Name: mcache_ctl

IM Requirement: Cache optional (micm_cfg.ISZ != 0 or mdcm_cfg.DSZ != 0)

Access Mode: Machine

CSR Address: 0x7ca (non-standard read/write)

31	9	8	7	6	5	4	3	2	1	0
0		CCTL_SUEN	DC_RWECC	IC_RWECC	DC_ECCEN		IC_ECCEN		DC_EN	IC_EN

Field Name	Bits	Description	Type	Reset
IC_EN	[0]	Controls if the instruction cache is enabled or not. | Value	Meaning | |---|---| | 0 | I-Cache is disabled | | 1 | I-Cache is enabled |	RW	0
DC_EN	[1]	Controls if the data cache is enabled or not. | Value	Meaning | |---|---| | 0 | D-Cache is disabled | | 1 | D-Cache is enabled |	RW	0
IC_ECCEN	[3:2]	Parity/ECC error checking enable control for the instruction cache. | Value	Meaning | |---|---| | 0 | Disable parity/ECC | | 1 | Reserved | | 2 | Generate exceptions only on uncorrectable parity/ECC errors | | 3 | Generate exceptions on any type of parity/ECC errors |	RW	0

Field Name	Bits	Description		Type	Reset
DC_ECCEN	[5:4]	Parity/ECC error checking enable control for the data cache.		RW	0
		Value	Meaning		
		0	Disable parity/ECC		
		1	Reserved		
		2	Generate exceptions only on uncorrectable parity/ECC errors		
		3	Generate exceptions on any type of parity/ECC errors		
IC_RWECC	[6]	Controls diagnostic accesses of ECC codes of the instruction cache rams through the `mecc_code` register. This bit can be turned on for injecting ECC errors to test the ECC handler.		RW	0
		Value	Meaning		
		0	Disable diagnostic accesses of ECC codes		
		1	Enable diagnostic accesses of ECC codes		
DC_RWECC	[7]	Controls diagnostic accesses of ECC codes of the data cache rams through the `mecc_code` register. This bit can be turned on for injecting ECC errors to test the ECC handler.		RW	0
		Value	Meaning		
		0	Disable diagnostic accesses of ECC codes		
		1	Enable diagnostic accesses of ECC codes		

Field Name	Bits	Description	Type	Reset
CCTL_SUEN	[8]	Enable bit for supervisor and user modes to access ucctlbeginaddr, ucctlcommand, and scctldata CSRs.	RW	0

Value	Meaning
0	Disable ucctlbeginaddr, ucctlcommand, and scctldata accesses in S/U mode
1	Enable ucctlbeginaddr, ucctlcommand, and scctldata accesses in S/U mode

18.8.7 Machine Miscellaneous Control Register

Mnemonic Name: mmisc_ctl

IM Requirement: Required

Access Mode: Machine

CSR Address: 0x7d0 (non-standard read/write)

31					7	6	5	4	3	2	1	0
		0				MSA	ACES		BRPE	RVCOMPM	VEC_PLIC	0

Field Name	Bits	Description	Type	Reset
VEC_PLIC	[1]	Selects the operation mode of PLIC: 	Value	Meaning	 	0	Regular mode	 	1	Vector mode	 Please note that both this bit and the vector mode enable bit (VECTORED) of the Feature Enable Register in NCEPLIC100 should be turned on for the vectored interrupt support to work correctly. See Section 19.5.2 for the definition of the VECTORED bit. This bit is hard-wired to 0 if the vectored PLIC feature is not supported.	RW	0
RVCOMPM	[2]	RISC-V compatibility mode enable bit. If the compatibility mode is turned on, all Andes-specific instructions become reserved instructions. 	Value	Meaning	 	0	Disabled	 	1	Enabled		RW	0
BRPE	[3]	Branch prediction enable bit. This bit controls all branch prediction structures. 	Value	Meaning	 	0	Disabled	 	1	Enabled	 This bit is hard-wired to 0 if branch prediction structure is not supported.	RW	1

.

Field Name	Bits	Description	Type	Reset
ACES	[5:4]	Andes Custom Extension (ACE) extension context status field:	RW	0

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

- This field should not be Off (0) for ACE instructions to execute normally. This is done by software program through CSRW instructions.
- A normal flow to turn on ACE unit will be as follows:
  - ACES is in the Off state.
  - An ACE instruction executed in the Off state triggers an illegal instruction with sdcause/mdcause == 2 (ACE disabled exception).
  - The exception handler initializes all ACE register states, changes this field to the Initial state, and then returns from exception.
  - The ACE instruction is executed again. Since ACES is not in the Off state this time, it shall execute correctly. If any ACE register states are modified, this field (ACES) will be updated to the Dirty state automatically by hardware.

This field is hard-wired to 0 if ACE extension is not configured.

Field Name	Bits	Description	Type	Reset
MSA	[6]	This field controls whether the load/store instructions can access misaligned memory locations without generating an Address Misaligned exception. Supported instructions: LW/LH/LHU/SW/SH	RW/RO	IM

Value	Meaning
0	Misaligned access generates an Address Misaligned exception.
1	Misaligned access is allowed.

18.8.8 Machine CCTL Begin Address

Mnemonic Name: mcctlbeginaddr

IM Requirement: Cache optional (micm_cfg.ISZ != 0 or mdcm_cfg.DSZ != 0) & (mmsc_cfg.CCTLCSR==1)

Access Mode: Machine

CSR Address: 0x7cb (non-standard read/write)

This register holds the address information required by CCTL operations:

For "VA" type of CCTL operations: The mcctlbeginaddr register contains the starting virtual address for CCTL operations triggered by writes to the mcctlcommand register. For CCTL lock operations, the mcctldata register will be updated with a successful (1) or failed (0) status value when the operations complete.

After an update to the mcctlcommand CSR with a "VA" type command, the value of this register will be incremented with the byte size of the corresponding cache line.

For "Index" type of CCTL operations: The mcctlbeginaddr register contains the cache index for

CCTL operations triggered by writes to the mcctlcommand register.

For all Index type commands other than "IX_RDATA" and "IX_WDATA": the "way" field in this register will be incremented. If the incremented "way" wraps to 0 (i.e., the first way of a set), then the "set" index field in this register will be incremented.

For "IX_RDATA" and "IX_WDATA" commands: the "offset" field in this register will be incremented first to the next offset value. If the incremented "offset" field wraps across 0 (i.e., the first word or double word of a cache line), then the "way" field in this register will be incremented. If the incremented "way" wraps to 0 (i.e., the first way of a set), then the "set" index field in this register will be incremented.

18.8.9 Machine CCTL Command

Mnemonic Name: mcctlcommand

IM Requirement: Cache optional (micm_cfg.ISZ != 0 or mdcm_cfg.DSZ != 0) & (mmsc_cfg.CCTLCSR==1)+

Access Mode: Machine

CSR Address: 0x7cc (non-standard read/write)

Writing to this register will trigger a CCTL operation, with the type of the operation specified by the value written. Valid CCTL operations are defined in Table 18-18 CCTL Command Definition.

Table 18-18 CCTL Command Definition

Value		Command	Type
0	0b00_000	L1D_VA_INVAL	VA
1	0b00_001	L1D_VA_WB	VA
2	0b00_010	L1D_VA_WBINVAL	VA
3	0b00_011	L1D_VA_LOCK	VA
4	0b00_100	L1D_VA_UNLOCK	VA
6	0b00_110	L1D_WBINVAL_ALL	-
7	0b00_111	L1D_WB_ALL	-
8	0b01_000	L1I_VA_INVAL	VA
11	0b01_011	L1I_VA_LOCK	VA
12	0b01_100	L1I_VA_UNLOCK	VA
16	0b10_000	L1D_IX_INVAL	Index
17	0b10_001	L1D_IX_WB	Index
18	0b10_010	L1D_IX_WBINVAL	Index
19	0b10_011	L1D_IX_RTAG	Index
20	0b10_100	L1D_IX_RDATA	Index
21	0b10_101	L1D_IX_WTAG	Index
22	0b10_110	L1D_IX_WDATA	Index
23	0b10_111	L1D_INVAL_ALL	-
24	0b11_000	L1I_IX_INVAL	Index
27	0b11_011	L1I_IX_RTAG	Index
28	0b11_100	L1I_IX_RDATA	Index
29	0b11_101	L1I_IX_WTAG	Index
30	0b11_110	L1I_IX_WDATA	Index

18.8.10 Machine CCTL Data

Mnemonic Name: mcctldata

IM Requirement: Cache optional (micm_cfg.ISZ != 0 or mdcm_cfg.DSZ != 0) & (mmsc_cfg.CCTLCSR==1)

Access Mode: Machine

CSR Address: 0x7cd (non-standard read/write)

Table 18-19 CCTL Commands Using the mcctldata Register lists all CCTL operations that access this register.

Table 18-19 CCTL Commands Using the mcctldata Register

Value of mcctlcommand		Command	Type
3	0b00_011	L1D_VA_LOCK	VA
4	0b00_100	L1D_VA_UNLOCK	VA
11	0b01_011	L1I_VA_LOCK	VA
12	0b01_100	L1I_VA_UNLOCK	VA
19	0b10_011	L1D_IX_RTAG	Index
20	0b10_100	L1D_IX_RDATA	Index
21	0b10_101	L1D_IX_WTAG	Index
22	0b10_110	L1D_IX_WDATA	Index
27	0b11_011	L1I_IX_RTAG	Index
28	0b11_100	L1I_IX_RDATA	Index
29	0b11_101	L1I_IX_WTAG	Index
30	0b11_110	L1I_IX_WDATA	Index

    For CCTL lock operations: The mcctldata register will be updated with a successful (1) or failed (0) status value when the operations complete.

● For CCTL index read/write-data operations: The mcctldata[31:0] holds the cache data for the operations.

● For CCTL index read/write-tag operations: The mcctldata register holds the cache tag for the operations. The bit position annotation is marked with the assumption that the width of physical address is PALEN. For caches which are larger than 2KiB per way, mcctldata[1:0] will be zero. For caches which are 2KiB per way, mcctldata[0] will be zero.

● The register format for tag operations is:

XLEN-1	XLEN-2	XLEN-3	XLEN-4	PALEN-10	PALEN-11	0
valid	lock	dirty/lock_dup		0		TAG=PA[(PALEN-1):10]

Note!

Bit XLEN-3 is the dirty bit for D-Cache and it keeps a duplicated copy of the lock bit to tolerate soft-errors for I-Cache.

18.8.11 User CCTL Begin Address

Mnemonic Name: ucctlbeginaddr

IM Requirement: Cache optional

Access Mode: User and above

CSR Address: 0x80b (non-standard read/write)

    The ucctlbeginaddr register contains the starting virtual address for CCTL operations triggered by writes to the ucctlcommand register. For CCTL lock operations, the mcctldata and scctldata register will be updated

with a successful (1) or failed (0) status value when the operations complete.

After an update to the mcctlcommand CSR with a "VA" type command, the value of this register will be incremented with the byte size of the corresponding cache line.

This register is only present when (micm_cfg.ISZ!=0 or mdcm_cfg.DSZ!=0) and mmsc_cfg.CCTLCSR==1 and misa[20]==1.

18.8.12 User CCTL Command

Mnemonic Name: ucctlcommand

IM Requirement: Cache optional

Access Mode: User and above

CSR Address: 0x80c (non-standard read/write)

Writing to this register will trigger a CCTL operation, with the type of the operation specified by the value written. Valid User CCTL operations are defined in Table 18-20 User CCTL Command Definition. When mcache_ctl.CCTL_SUEN is 0, accessing ucctlcommand in U-mode and S-mode will generate an "illegal instruction" exception. When mcache_ctl.CCTL_SUEN is 1, U-mode programs can access ucctlcommand. However, if ucctlcommand is write a command with 0 in U-Mode allowed field in Table 18-20 User CCTL Command Definition, an "illegal instruction" exception will be generated.

This register is only present when (micm_cfg.ISZ!=0 or mdcm_cfg.DSZ!=0) and mmsc_cfg.CCTLCSR==1 and misa[20]==1.

Table 18-20 User CCTL Command Definition

Value		Command	Type	U-Mode allowed
0	0b00_000	L1D_VA_INVAL	VA	1
1	0b00_001	L1D_VA_WB	VA	1
2	0b00_010	L1D_VA_WBINVAL	VA	1
3	0b00_011	L1D_VA_LOCK	VA	0
4	0b00_100	L1D_VA_UNLOCK	VA	0
6	0b00_110	L1D_WBINVAL_ALL	-	0
7	0b00_111	L1D_WB_ALL	-	0
8	0b01_000	L1I_VA_INVAL	VA	1
11	0b01_011	L1I_VA_LOCK	VA	0
12	0b01_100	L1I_VA_UNLOCK	VA	0
16	0b10_000	L1D_IX_INVAL	Index	0
17	0b10_001	L1D_IX_WB	Index	0
18	0b10_010	L1D_IX_WBINVAL	Index	0
19	0b10_011	L1D_IX_RTAG	Index	0
20	0b10_100	L1D_IX_RDATA	Index	0
21	0b10_101	L1D_IX_WTAG	Index	0
22	0b10_110	L1D_IX_WDATA	Index	0
23	0b10_111	L1D_INVAL_ALL	-	0
24	0b11_000	L1I_IX_INVAL	Index	0
27	0b11_011	L1I_IX_RTAG	Index	0
28	0b11_100	L1I_IX_RDATA	Index	0
29	0b11_101	L1I_IX_WTAG	Index	0
30	0b11_110	L1I_IX_WDATA	Index	0

18.9 Hardware Stack Protection and Recording Registers

18.9.1 Machine Hardware Stack Protection Control

Mnemonic Name: mhsp_ctl

IM Requirement: STACKSAFE_SUPPORT = "yes" (mmsc_cfg.HSP == 1)

Access Mode: Machine

CSR Address: 0x7C6 (non-standard read/write)

31	6	5	4	3	2	1	0
0	M	0	U	SCHM	UDF_EN	OVF_EN

Field Name	Bits	Description	Type	Reset
OVF_EN	[0]	Enable bit for the stack overflow protection and recording mechanism. This bit will be cleared to 0 automatically by hardware when a stack protection (overflow or underflow) exception is taken.	RW	0
UDF_EN	[1]	Enable bit for the stack underflow protection mechanism. This bit will be cleared to 0 automatically by hardware when a stack protection (overflow or underflow) exception is taken.	RW	0
SCHM	[2]	Selects the operating scheme of the stack protection and recording mechanism.	RW	0
U	[3]	Enables the SP protection and recording mechanism in User mode.	RW	0
M	[5]	Enables the SP protection and recording mechanism in Machine mode.	RW	0

OVF_EN:

Value	Meaning
0	The stack overflow protection and recording mechanism are disabled.
1	The stack overflow protection and recording mechanism are enabled.

UDF_EN:

Value	Meaning
0	The stack underflow protection is disabled.
1	The stack underflow protection is enabled.

SCHM:

Value	Meaning
0	Stack overflow/underflow detection
1	Top-of-stack recording

U:

Value	Meaning
0	The mechanism is disabled in User mode.
1	The mechanism is enabled in User mode.

M:

Value	Meaning
0	The mechanism is disabled in Machine mode.
1	The mechanism is enabled in Machine mode.

18.9.2 Machine SP Bound Register

Mnemonic Name: msp_bound

IM Requirement: STACKSAFE_SUPPORT = "yes" (mmsc_cfg.HSP == 1)

Access Mode: Machine

CSR Address: 0x7c7 (non-standard read/write)

31	0
MSP_BOUND	
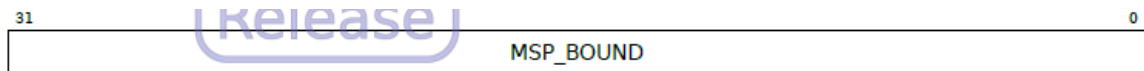
When the SP overflow detection mechanism is properly selected and enabled, any updated value to the SP register (via any instruction) is compared with the msp_bound register. If the updated value to the SP register is smaller than the msp_bound register, a stack overflow exception is generated. The stack overflow exception has an exception code of 32 in the mcause register.

When the top of stack recording mechanism is properly selected and enabled, any updated value to the SP register on any instruction is compared with the msp_bound register. If the updated value to the SP register is smaller than the msp_bound register, the msp_bound register is updated with this updated value. It is a RW type register with the all-one reset value (0xFFFFFFFF for RV32 and 0xFFFFFFFFFFFFFFFF for RV64).

Programming Note:

The "csrrw sp, msp_bound, rs" instruction updates both sp and msp_bound registers at the same time. When the stack overflow detection mechanism is enabled, using this instruction may generate unpredictable exception behavior.

18.9.3 Machine SP Base Register

Mnemonic Name: msp_base

IM Requirement: STACKSAFE_SUPPORT = "yes" (mmsc_cfg.HSP == 1)

Access Mode: Machine

CSR Address: 0x7c8 (non-standard read/write)

31	0
SP_BASE	
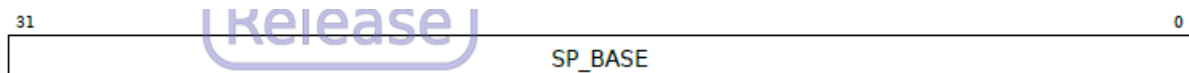
When the SP underflow detection mechanism is properly selected and enabled, any updated value to the SP register (via any instruction) is compared with the msp_base register. If the updated value to the SP register is greater than the msp_base register, a stack underflow exception is generated. The stack underflow exception has an exception code of 33 in the mcause register.

It is a RW type register with the all-one reset value (0xFFFFFFFF for RV32 and 0xFFFFFFFFFFFFFFFF for RV64).

Programming Note:

The "csrrw sp, msp_base, rs" instruction updates both sp and msp_base registers at the same time. When the stack underflow detection mechanism is enabled, using this instruction may generate unpredictable exception behavior.

18.10 CoDense Registers

18.10.1 Instruction Table Base Address Register

Mnemonic Name: uitb

IM Requirement: CODENSE_SUPPORT = "yes"

Access Mode: User

CSR Address: 0x800 (non-standard read/write)

This register defines the base address of the CoDense instruction table. The CoDense instruction table is a table of 32-bit instructions that instructions defined by the Andes CoDense Extension should expand to. The table is typically generated by the compiler for replacing 32-bit instructions with the shorter 16-bit Andes CoDense instructions, hence reducing the code size.

31	2	1	0
ADDR		Reserved	HW

Field Name	Bits	Description	Type	Reset
HW	[0]	This bit specifies if the CoDense instruction table is hard-wired.	RO	IM
Reserved	[1]	Reserved		0
ADDR	[31:2]	The base address of the CoDense instruction table. This field is reserved if UITB.HW == 1.	RW	0

Value	Meaning
0	The instruction table is located in memory. UITB.ADDR should be initialized to point to the table before using the CoDense instructions.
1	The instruction table is hard-wired. Initialization of UITB.ADDR is not needed before using the CoDense instructions.

18.11 Physical Memory Protection Unit Configuration & Address Registers

18.11.1 PMP Configuration Registers

Mnemonic Name: pmpcfg0–pmpcfg3

IM Requirement: PMP_SUPPORT

Access Mode: Machine

CSR Address: 0x3A0 to 0x3A3 (standard read/write)

0x3A0

31 24	23 16	15 8	7 0
pmp3cfg	pmp2cfg	pmp1cfg	pmp0cfg

0x3A1

31 24	23 16	15 8	7 0
pmp7cfg	pmp6cfg	pmp5cfg	pmp4cfg

0x3A2

31 24	23 16	15 8	7 0
pmp11cfg	pmp10cfg	pmp9cfg	pmp8cfg

0x3A3

31 24	23 16	15 8	7 0
pmp15cfg	pmp14cfg	pmp13cfg	pmp12cfg

PMP Configuration Format

7	6 5	4 3	2	1	0
L	WIRI	A	X	W	R

Field Name	Bits	Description			Type	Reset
R	[0]	Read access control.			RW	0
			Value	Meaning		
			0	Read accesses are not allowed.		
			1	Read accesses are allowed.		
W	[1]	Write access control.			RW	0
			Value	Meaning		
			0	Write accesses are not allowed.		
			1	Write accesses are allowed.		

Field Name	Bits	Description			Type	Reset
X	[2]	Instruction execution control.			RW	0
			Value	Meaning		
			0	Instruction execution is not allowed.		
			1	Instruction execution is allowed.		
A	[4:3]	Address matching mode.			RW	0
			Value	Meaning		
			0	OFF: PMP entry is disable.		
			1	TOR: Top of range. For PMP entry 0, it matches any address A < pmpaddr0. For PMP entry i, it matches any address A such that pmpaddr$_i$ > A >= pmpaddr$_{i-1}$. But the 4-byte range is not supported.		
			2	Reserved.		
			3	NAPOT: Naturally aligned power-of-2 region, >= 8 bytes. This mode makes use of the low-order bits of the associated address register to encode the size of the range. See Table 81 for range encoding from the value of a PMP address register.		
WIRI	[6:5]	Reserved WIRI field.			WIRI	0
L	[7]	Write lock and permission enforcement bit for Machine mode.			W1S*	0
			Value	Meaning		
			0	Machine mode writes to PMP entry registers are allowed. R/W/X permissions apply to S and U modes.		
			1	For PMP entry i, writes to pmp$_i$cfg and pmpaddr$_i$ are ignored. Additionally, if pmp$_i$cfg.A is set to TOR, writes to pmpaddr$_{i-1}$ is ignored as well. As for permission enforcement, R/W/X permissions apply to all modes. This bit can only be cleared to 0 with a system reset.		

Note!

The register type of the "L" bit field is W1S because only a system reset can clear this bit.

18.11.2 PMP Address Register

Mnemonic Name: pmpaddr0–pmpaddr15

IM Requirement: PMP_SUPPORT

Access Mode: Machine

CSR Address: 0x3b0 to 0x3bf (standard read/write)

31	0
address[33:2](WARL)	
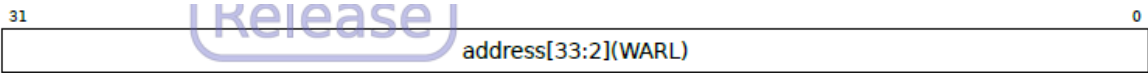
Each PMP address register encodes bits 33–2 of a 34-bit physical address, as shown in the register format. The encoding is described in Table 18-21 AE250 NAPOT range encoding in PMP address and configuration registers. The "a" in the table represents one bit address, with arbitrary values.

Table 18-21 AE250 NAPOT range encoding in PMP address and configuration registers

Register Content	Match Size(Byte)
aaaa...aaa0	8
aaaa...aa01	16
aaaa...a011	32
...	...
aa01...1111	2^{XLEN}
a011...1111	2^{XLEN+1}
0111...1111	2^{XLEN+2}
1111...1111	2^{XLEN+2}

19 Instruction Latency

This chapter lists instruction throughput and latency. The instruction throughput is the number of cycles before executing the next independent instruction with the same type. The instruction latency is the number of cycles before executing the next instruction with read-after-write dependency.

19.1 ALU Instructions

The latency and the throughput of ALU instructions are both 1 cycle. ALU instructions include:

- Add/Sub: ADD, SUB, ADDI, ADDW, SUBW, ADDIW
- Shift: SLL, SRL, SRA, SLLI, SRLI, SRAI, SLLW, SRLW, SRAW, SLLIW, SRLIW, SRAIW
- Logical: AND, OR, XOR, ANDI, ORI, XORI
- Compare: SLT, SLTU, SLTI, SLTIU
- LUI and AUIPC
- Load effective address instructions
- ADDIGP
- String processing: FFB, FFZMISM, FFMISM, FLMISM
- Bit field operation: BFOS, BFOZ

19.2 Load Instructions

Table 19-1 Load Instruction Latency

Instruction	Throughput (cycles/instruction)	Latency (cycles)
load word from DLM/D-Cache	1	2
load word from ILM	2	4
load word from from AXI/AHB	3	4
load word from low access-latency AHB	1	2
load byte/halfword from DLM/D-Cache	1	3
load byte/halfword from ILM	2	4
load byte/halfword from AXI/AHB	3	5
load byte/halfword from low access-latency AHB	1	3

19.3 Multiply Instructions

The latency and throughput of multiply instructions depend on the multiplier implementation.

Table 19-2 Multiply Instruction Latency: Radix Multiplier

Instruction	Throughput (cycles/instruction)	Latency (cycles)
MULHU	4 + 32 / LOG2(MUL_RADIX)	6 + 32 / LOG2(MUL_RADIX)
MUL, MULH, MULHSU	5 + 32 / LOG2(MUL_RADIX)	7 + 32 / LOG2(MUL_RADIX)

Table 19-3 Multiply Instruction Latency: Fast Multiplier

Instruction	Throughput (cycles/instruction)	Latency (cycles)
MUL, MULH, MULHU, MULHSU	1	3

19.4 Divide and Remainder Instructions

The divide and remainder instructions are implemented using the non-restoring division algorithmwith early termination detection.

Instruction	Throughput (cycles/instruction)	Latency (cycles)
DIVU, REMU	5–36	7–38
DIV, REM	6–37	8–39

19.5 Branch and Jump Instruction

The branch and jump instruction throughput is 1 cycle/instruction. Branch mis-prediction penalty is3 cycles.

19.6 Trap Return Instruction

The trap return instruction flushes the entire pipeline, and the penalty is 5 cycles.

20 AE250 AHB Platform

The AE250 AHB platform is a pre-integrated platform implementing the AE250 memory map that contains an N25(F) processor. The block diagram is depicted in Figure 4-3 Design Hierarchy. The peripheral platform IPs may be available as either unencrypted RTL or encrypted RTL depending on the N25(F) licensing agreements.

20.1 I/O Signal

The top-level module of this platform is ae250_chip_wrap. I/O signals of ae250_chip_wrap are described in Table 20-1. Signal types are listed below:

Term	Description
I	Input signals
O	Output signals
I/O	Bi-direction signal

Table 20-1 I/O Signal

Interface	Signal Name	Type	Description
General	X_om	I	Operation mode
	X_aopd_por_b	I	Power-on reset in the always-on power domain
	X_por_b	I	Power-on reset in the main power domain
	X_hw_rstn	I	Hardware reset
	X_oschin	I	High frequency oscillator input
	X_osclin	I	Low frequency oscillator input
	X_mpd_pwr_off	O	Main power domain power off indication
	X_rtc_wakeup	O	Alarm wake-up event
	X_wakeup_in	I	External wake-up event
JTAG	X_tdi	I	Test data input
	X_tdo	O	Test data output
	X_tms	I	Test mode select
	X_tck	I	Test clock

Interface	Signal Name	Type	Description
	X_trst	I/O	Test reset
SPIn (n=1~4)	X_spin_clk	I/O	SPI clock
	X_spin_csn	I/O	SPI chip select (Active-Low)
	X_spin_mosi	I/O	SPI bus master output / slave input
	X_spin_miso	I/O	SPI bus master input / slave output
	X_spin_holdn	I/O	SPI hold (Active-Low)
	X_spin_wpn	I/O	SPI WP (Active-Low)
I^{2}Cn (n=1~4)	X_i2cn_scl	I/O	I^{2}C clock
	X_i2cn_sda	I/O	I^{2}C data
UARTn (n=1~4)	X_uartn_rxd	I	UART serial data input
	X_uartn_txd	O	UART serial data output
	X_uartn_ctsn	I	UART modem clear to send (Active-Low)
	X_uartn_rtsn	O	UART modem request to send (Active-Low)
	X_uartn_dcdn	I	UART modem data carrier detect (Active-Low)
	X_uartn_dsrn	I	UART modem data set ready (Active-Low)
	X_uartn_dtrn	O	UART modem data terminal ready (Active-Low)
	X_uartn_out1n	O	UART user-defined output 1 (Active-Low)
	X_uartn_out2n	O	UART user-defined output 2 (Active-Low)
	X_uartn_rin	I	UART modem ring indicator (Active-Low)
PITn (n=1~2)	X_pitn_pwm_ch0	O	PWM channel 0
	X_pitn_pwm_ch1	O	PWM channel 1
	X_pitn_pwm_ch2	O	PWM channel 2
	X_pitn_pwm_ch3	O	PWM channel 3
GPIO	X_gpio	I/O	General purpose I/O
	gpio_pullup	O	GPIO output pull_up enable signal
	gpio_pulldown	O	GPIO output pull_down enable signal
AHB External Master/Slave (n=1~15)	hclk_o	O	AHB clock
	hresetn_o	O	AHB reset signal (Active-Low)
	exthmn_haddr	I	Address bus of AHB-Lite master *n*
	exthmn_hburst	I	Burst type signal of AHB-Lite master *n*
	exthmn_hport	I	Protect signal of AHB-Lite master *n*
	exthmn_hsize	I	Data size signal of AHB-Lite master *n*
	exthmn_htrans	I	Transfer type signal of AHB-Lite master *n*
	exthmn_hwrite	I	Read/write control signal of AHB-Lite master n
	exthmn_hwdata	I	Write data bus of AHB-Lite master *n*
	exthmn_hrdata	O	Read data bus of AHB-Lite master *n*
	exthmn_hready	O	Hready signal of AHB-Lite master *n* Indicates that a transfer has finished on the bus
	exthmn_hresp	O	Response signal of AHB-Lite master *n*

Interface	Signal Name	Type	Description
	exthsn_hrdata	I	Read data bus of AHB slave *n*. The read data bus transfers data from bus slaves to bus masters for read operations.
	exthsn_hready	I	AHB hready from slave *n*, indicating that a transfer to/from slave *n* has finished.
	exthsn_hresp	I	Response signal of AHB slave *n*. This signal provides response status of the transfer.
	exthsn_hreadyout	O	AHB hready of slave *n*, indicating that the bus is ready to issue a new transfer.
	exthsn_haddr	O	Address bus of AHB slave *n*
	exthsn_hburst	O	Write data bus of AHB slave *n*
	exthsn_hprot	O	Protect signal of AHB slave *n*
	exthsn_hsel	O	Select signal of AHB slave *n*
	exthsn_hsize	O	Data size signal of AHB slave *n*, indicating the size of the transfer
	exthsn_htrans	O	Transfer type signal of AHB slave *n*, indicating the type of the current transfer
	exthsn_hwdata	O	Write data bus of AHB slave *n*
	exthsn_hwrite	O	Transfer direction of AHB slave *n*. HIGH indicates a write transfer and LOW indicates a read transfer.
APB External Slave (n=2~31)	pclk_o	O	APB clock
	presetn_o	O	APB reset signal (Active-Low)
	pprot	O	APB protect signal translated from the AHB master port.
	pstrb	O	APB write strobe signal translated from the AHB master port.
	paddr	O	APB address bus.
	penable	O	APB enable signal translated from the AHB master port
	pwrite	O	APB write signal translated from the AHB master port
	pwdata	O	APB write data bus translated from the AHB master port
	extpsn_psel	O	APB select signal This signal indicates that APB slave *n* is selected
	extpsn_prdata	I	APB read data of slave *n* The read data bus is driven by APB slave *n* device during the read cycles
	extpsn_pready	I	APB ready of slave *n* The slave uses this signal to extend an APB transfer

Interface	Signal Name	Type	Description
	extpsn_pslverr	I	This signal indicates a transfer failure of the slave ***n***
DMA External Req	dma_ack_ext	O	External burst transfer acknowledge
	dma_req_ext	I	External burst transfer request
Interrupt External Source	ext_int_src	I	External interrupt sources.

20.2 AE250 Memory Map

Table 20-2 AE250 Memory Map

Base Address	Descripton
0x00000000	RAM Bridge
0x80000000	SPI1 AHB Memory
0xA0000000	Local Memory Slave Port: ILM
0xA0200000	Local Memory Slave Port: DLM
0xE0000000	BMC
0xE0E00000	DMAC
0xE4000000	PLIC
0xE6000000	Machine Timer
0xE6400000	PLIC-SW
0xE6800000	Debug Module
0xF0000000	APBBRG
0xF0100000	SMU
0xF0200000	UART1
0xF0300000	UART2
0xF0400000	PIT1
0xF0500000	WDT
0xF0600000	RTC
0xF0700000	GPIO
0xF0A00000	I2C1
0xF0B00000	SPI1
0xF0F00000	SPI2
0xF1100000	SPI3
0xF1200000	SPI4
0xF1300000	I2C2
0xF1400000	I2C3
0xF1500000	I2C4
0xF1600000	PIT2
0xF1A00000	UART3

Base Address	Descripton
0xF1B00000	UART4

Note!

- The RAM bridge space indicates the size of the RAM behind this bridge. It can be different from the size of the address space allocated to the bridge on the bus. The default setting allocates a 2GiB space (0x00000000 – 0x7FFFFFFF) to the bridge on the bus. When the bridge sees a transaction for addresses outside of the addressable RAM, it will return an error response.

- In addition to the bus view described here, ILM/DLM are accessible by the processor through private address spaces visible only to the processor:

  - The address range of these private address spaces are controlled by $ILMB and $DLMB control registers.

  - The private address spaces have higher priority than the bus address spaces in the processor. Accesses will be directed to go through the local memory interfaces and bypass the bus address spaces if they hit the private address spaces.

  - If overlapping of address spaces is not desired, the $ILMB/$DLMB control registers could be programmed to avoid overlapping.

  - ILM is visible to the processor at 0x00000000 in the default setting.

  - DLM is visible to the processor at 0x00200000 in the default setting.

- APB and FIO Simulation Model is used in ae250 simulation environment and only for simulation. The base and size of APB and FIO regions can be customized to integrate devices. Please note that the changed base and size of these two regions should not overlap with those of the existing device regions. Otherwise, it may cause unexpected results in platform simulation

20.3 Interrupt Assignment

Interrupts in a RISC-V platform are classified into two types: local interrupts and global interrupts. Local interrupts are interrupts that go directly into a RISC-V processor, and global interrupts get arbitrated through a platform-level interrupt controller (PLIC) before going into the RISC-V processor as the external interrupts.

Local interrupts supported by N25(F) include non-maskable interrupts (nmi), machine timer interrupts (mtip) and software interrupts (msip). Additionally, an external interrupt pins (meip) and (seip) accept arbitrated interrupt signaling from PLIC. Table 20-3 AE250 Interrupt Assignment summarizes the interrupt source connectivity for local interrupts.

In the AE250 platform, the PLIC module is instantiated a second time with all interrupt sources tied to zero as the software interrupt controller (PLIC_SW). The capability of the PLIC controller to generate interrupts through its programming registers is used for generating software

interrupts.

The global interrupt sources in the AE250 platform and their connectivity to PLIC are summarized in Table 20-4 PLIC Interrupt Source.

Table 20-3 AE250 Interrupt Assignment

Interrupt Signal	Description
nmi	WDT
mtip	Machine Timer
meip	PLIC
seip	PLIC
msip	PLIC_SW

Table 20-4 PLIC Interrupt Source

Interrupt Source	Description
1	RTC Period
2	RTC Alarm
3	PIT1
4	SPI1
5	SPI2
6	I2C1
7	GPIO
8	UART1
9	UART2
10	DMAC
11	BMC
12	UART3
13	UART4
14	SPI3
15	SPI4
16	PIT2
17	SMU_interrupt_pd
18	SMU_standby_req
19	SMU_wakeup_ok
20	I2C2
21	I2C3
22	I2C4

20.4 Functional Description

20.4.1 ATCBMC200 – AHB Bus Matrix

The Bus Matrix Controller (BMC) switches the matrix connections according to the hardware configuration. In addition, the controller has slave information registers for software to enumerate the AHB bus components. Features of the controller include:

- Support of 32 bits address width
- Support of up to 16 AHB masters
- Support of up to 16 AHB slaves
- Including one internal slave for slave information registers and register programming
- Configurable connectivity between masters and slaves
- Programmable two-level priority arbitration scheme

20.4.2 ATCAPBBRG100 – AHB-to-APB Bridge

The AHB-to-APB bridge translates AHB transactions to APB transactions targeting a specific APB slave according to the slave base address and address space size configurations. Features of the bridge include:

- Compliant with AMBA 4 APB
- Support of 32 bits address width
- Support of up to 31 APB slaves
- Including one internal slave for slave information registers and register programming
- Configurable base/size for each downstream APB slave
- Support of various synchronous AHB/APB clock ratios (N:1, N = 1, 2, 3, . . .)
- Support of write buffering

20.4.3 ATCUART100 – UART Controller

The UART controller handles communications to the Universal Asynchronous Receiver/Transmitter (UART) serial interface. It has the following features:

- Compatible with the 16C550A register structure
- Support of the hardware flow control (CTS/RTS)
- Support of hardware handshaking to the DMA controller

- Option of by-8 or by-16 over-sampling frequency
- Configurable transmit/receive FIFO depth: 16/32/64/128 entries

20.4.4 ATCSPI200 – SPI Controller

The SPI controller handles communications to the Serial Peripheral Interface (SPI). The supported serial data formats range from 4 bits to 32 bits in length. Features of the SPI controller include:

- Support of both master and slave mode operations
- Support of memory-mapped accesses (read-only) through the AHB bus
- Support of hardware handshaking to the DMA controller
- Support of the dual I/O and quad I/O modes
- Support of the 3-line mode

20.4.5 ATCIIC100 – I2C Controller

The I2C controller handles communications to the Inter-Integrated Circuit (IIC or I2C) serial interface. Features of the I2C controller include:

- Programmable to be either a master or a slave device
- Programmable clock/data timing
- Support of the I2C-bus Standard-mode (100 kb/s), Fast-mode (400 kb/s) and Fast-mode plus (1 Mb/s)
- Support of hardware handshaking to the DMA controller
- Support of the master-transmit, master-receive, slave-transmit and slave-receive modes
- Support of the multi-master mode
- Support of 7-bit and 10-bit addressing modes
- Support of general call addressing mode
- Support of auto clock stretch

20.4.6 ATCWDT200 – Watchdog Timer

The Watchdog Timer (WDT) controller prevents the system from hanging if software is trapped in a deadlock condition. A decrementing counter (the watchdog timer) is maintained in WDT and a watchdog interrupt will be generated once the watchdog timer reaches zero. The timer should be restarted in the watchdog interrupt service routine. A secondary timer called system reset timer starts ticking after the watchdog interrupt, and it gets canceled upon restart of the watchdog timer. Should the watchdog timer be not restarted in time after the watchdog interrupt is triggered, system reset will be triggered by the system reset timer to reset the system. Features of WDT include:

- Internal/external clock source selection
- Separate timers for the watchdog interrupt and the system reset
  - Eight choices of watchdog timer intervals
  - Four choices of reset timer intervals
- Register write protection for watchdog timer control register and restart register
  - Configurable magic number for register write protection
  - Configurable magic number to restart the watchdog timer

20.4.7 ATCRTC100 – Real-Time Clock

Real-time clock (RTC) keeps track of current time relative to a base time. The time is stored in a RTC counter which records the amount of elapsed time since RTC is enabled. Features of RTC include:

- The frequency of clock source (before the clock divider) for the counter is 32.768KHz.
- Separate second, minute, hour and day counters.
- Periodic interrupts: half-second, second, minute, hour and day interrupts.
- Programmable alarm interrupt with specified second, minute and hour values.

RTC duplicates in functionality the RISC-V Machine Timer (Machine Timer). The RTC module is offered for compatibility of existing Andes platform software environment. It may be configured out for a pure RISC-V platform.

20.4.8 ATCPIT100 – PIT Controller

The Programmable Interval Timer (PIT) controller is a set of compact multi-function timers, which can be used as pulse width modulators (PWM) or simple timers. Each multi-function timer provides the following 6 usage scenarios:

- One 32-bit timer
- Two 16-bit timers
- Four 8-bit timers
- One 16-bit PWM
- One 16-bit timer and one 8-bit PWM
- Two 8-bit timers and one 8-bit PWM

Features of the PIT controller include:

- Support of AMBA 2.0 APB bus protocol

- Support of up to 4 multi-function timers
- Six usage scenarios (combination of timer and PWM) for each multi-function timer
- Programmable source of timer clock

20.4.9 ATCGPIO100 – GPIO Controller

The General Purpose I/O (GPIO) controller supports up to 32 channels with independently programmable input/output control. Features of the GPIO controller include:

- Configurable number of GPIO channels (pins): 1–32
- Independent control of each channel
- Programmable I/O direction
- Optional pull-up/down control
- Optional support of interrupt trigger control
- Flexible combination of interrupt trigger modes: high/low level trigger and rising/falling/both edge trigger
- Optional de-bounce functionality for input channels

20.4.10 ATCRAMBRG200 – RAM Bridge

The RAM bridge controller allows standard SRAMs to be accessed on the AHB bus.

- Configurable address width: 24 – 64 bits
- Configurable data width: 32 or 64 bits

20.4.11 ATCDMAC100 – DMA Controller

The Direct Memory Access Controller (DMAC) enhances system performance by transferring large data blocks between devices in background to offload the processor. Features of DMAC include:

- Up to 8 configurable DMA channels
- Up to 32 DMA request/acknowledge for hardware handshake
- Group round-robin arbitration scheme with 2 priority levels
- Support of 8, 16, and 32-bit data transfers
- Support of chain transfers

The source/destination IDs are needed for programming the handshake pairs when initiating DMA transfers. Table 20-5 DMA Hardware Handshake ID assigns the handshake ID for all devices of the AE250 platform. The source and destination ID should not be the same for a handshake pair since the DMA controller does not support transferring data

back to the same device.

Table 20-5 DMA Hardware Handshake ID

DMA Handshake ID	Devices
0	SPI1 TX
1	SPI1 RX
2	SPI2 TX
3	SPI2 RX
4	UART1 TX
5	UART1 RX
6	UART2 TX
7	UART2 RX
8	I2C1
9	SPI3 TX
10	SPI3 RX
11	SPI4 TX
12	SPI4 RX
13	UART3 TX
14	UART3 RX
15	UART4 TX
16	UART4 RX
17	I2C2
18	I2C3
19	I2C4

20.5 System Management Unit

20.5.1 Summary of Registers

SMU registers are summarized as follows:

Table 20-6 SMU Register Summary

Address Offset	Name	Description
0x00	SYSTEMVER	SYSTEM ID & Revision
0x10	WRSR	Wake-up and reset status register
0x14	SMUCR	SMU command register
0x1C	WRMASK	Wake-up and reset mask
0x20	CER	Clock enable register
0x24	CRR	Clock ratio register
0x40	SCRATCH	Scratch pad register
0x50	RESET_VECTOR	CPU Reset Vector

20.5.2 SYSTEM ID & Revision Register (SYSTEMVER) (0x00)

Field Name	Bits	Description	Type	Reset
MINOR	[3:0]	Minor revision number	RO	0x0
MAJOR	[7:4]	Major revision number	RO	0x0
ID	[31:8]	ID for AE250	RO	0x414525

20.5.3 Wake-up and Reset Status Register (WRSR) (0x10)

Field Name	Bits	Description	Type	Reset
APOR	[0]	AOPD Power-On Reset	W1C	Note1

Value	Meaning
0	No action
1	Reset has occurred

Field Name	Bits	Description		Type	Reset
MPOR	[1]	MPD Power-On Reset		W1C	Note2
		Value	Meaning		
		0	No action		
		1	Reset has occurred		
HW	[2]	Hardware Reset		W1C	Note3
		Value	Meaning		
		0	Reset didn't occur		
		1	Reset has occurred		
WDT	[3]	Watchdog Reset		W1C	Note3
		Value	Meaning		
		0	Reset didn't occur		
		1	Reset has occurred		
SW	[4]	Software Reset		W1C	Note3
		Value	Meaning		
		0	Reset didn't occur		
		1	Reset has occurred		
WI	[8]	Wake-up by external events		W1C	0
		Value	Meaning		
		0	Wake-up event didn't occur		
		1	Wake-up event has occurred		
ALM	[9]	Wake-up by RTC alarm events		W1C	0
		Value	Meaning		
		0	Wake-up event didn't occur		
		1	Wake-up event has occurred		
DBG	[10]	Wake-up by debug requests		W1C	0
		Value	Meaning		
		0	Wake-up event didn't occur		
		1	Wake-up event has occurred		

Note

- APOR is reset to 1 during the AOPD power-on reset.
- MPOR is reset to 1 during the MPD power-on reset.
- HW, WDT, and SW are reset to 0 during the AOPD power-on reset.

20.5.4 SMU Command Register (SMUCR) (0x14)

Field Name	Bits	Description			Type	Reset
SMUCMD	[7:0]	SMU command			WO	0
		Value	**Meaning**			
		0x3c	Software reset to reset the whole system.			
		0x5a	Power off the main power domain. The register would be cleared to 0 when the wake-up event happens.			
		0x55	Standby command that triggers the standby request to the processor. If the Clock Ratio Register (CRR) has been modified, SMU waits for the clock ratio change to take effect and then directly wakes up the processor. Otherwise, SMU waits for a wake-up event before waking up the processor.			

20.5.5 Wake-up and Reset Mask Register (WRMASK) (0x1c)

Field Name	Bits	Description			Type	Reset
WIMASK	[8]	Indicating whether external events will trigger wake-ups.			RW	0
		Value	**Meaning**			
		0	Don't mask			
		1	Mask the wake-up event			

Field Name	Bits	Description	Type	Reset
ALMMASK	[9]	Indicating whether RTC events will trigger wake-ups.	RW	0

Value	Meaning
0	Don't mask
1	Mask the wake-up event

Field Name	Bits	Description	Type	Reset
DBGMASK	[10]	Indicating whether debug requests will trigger wake-ups.	RW	0

Value	Meaning
0	Don't mask
1	Mask the wake-up event

20.5.6 Clock Enable Register (CER) (0x20)

This register controls all clocks in the platform. The processor clock, AHB/APB bus clock should be turned on/off following the Processor/Bus Clock Operation Sequences described after the bit fields description.

Field Name	Bits	Description	Type	Reset
CCLK_EN	[0]	Processor clock enable. The processor clock disable/enable operation must follow the *Processor Clock Operation Sequence*.	RW	1

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
HCLK_EN	[1]	AHB bus clock enable. The AHB bus clock disable/enable operation must follow the *Bus Clock Operation Sequence*.	RW	1

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_EN	[2]	Main APB bus clock enable. The APB bus clock disable/enable operation must follow the *Bus Clock Operation Sequence*.	RW	1

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_UART1_EN	[3]	UART1 APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_UART2_EN	[4]	UART2 APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_SPI1_EN	[5]	SPI1 APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_SPI2_EN	[6]	SPI2 APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_GPIO_EN	[7]	GPIO APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_I2C_EN	[8]	I2C APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_WDT_EN	[9]	WDT APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

Field Name	Bits	Description	Type	Reset
PCLK_PIT_EN	[10]	PIT APB bus clock enable	RW	0

Value	Meaning
0	Disable clock
1	Enable clock

PROCESSOR CLOCK OPERATION SEQUENCE

1. Set RTC alarm in the RTC programming register (optional).

2. Set CCLK_EN to 0.

3. Set standby command in the SMU command register.

   – SMU issues an external interrupt to the processor notifying the standby request.

   – The processor should execute the WFI instruction to make the processor go into the WFI mode.

4. The Processor clock is disabled after the processor enters the WFI mode.

5. The Processor clock is enabled and the processor is waked up when a wake-up event arrives.

6. Clear the SMU command and status registers.

BUS CLOCK OPERATION SEQUENCE

7. Set RTC alarm in the RTC programming register (optional).

8. Set PCLK_EN, HCLK_EN or CCLK_EN to 0.

9. Set standby command in the SMU command register.

   – SMU issues an external interrupt to the processor notifying the standby request.

   – The processor should execute the WFI instruction to make the processor go into the WFI mode.

10. The bus clock or processor clock is disabled after the processor enters the WFI mode, depending on the PCLK_EN, HCLK_EN and CCLK_EN setting.

11. The bus clock is enabled and the processor is waked up when a wake-up event arrives

12. Clear the SMU command and status registers.

20.5.7 Clock Ratio Register (CRR) (0x24)

Field Name	Bits	Description		Type	Reset
CCLKSEL	[0]	Processor clock select		RW	0
		Value	Meaning		
		0	OSCH (Default)		
		1	Divide OSCH by 2		
HPCLKSEL	[3:1]	HCLK and PCLK clock ratio select		RW	0
		Value	core_clk : hclk : pclk (frequency)		
		0	1:1:1		
		1	1:1:1/2		
		2	1:1:1/4		
		3	1:1/2:1/2		
		4	1:1/2:1/4		
		5-7	Reserved		

20.5.8 Scratch Pad Register (SCRATCH) (0x40)

This register is a scratch register and it retains values when the rest of the system is powered down. It could be used to hold some parameters during the power down period.

Field Name	Bits	Description	Type	Reset
SCRATCH	[31:0]	Scratch register	RW	0

20.5.9 Reset Vector (RESET_VECTOR) (0x50)

This register controls the value driven to the reset_vector[31:0] input signal of the N25(F) processor.

Field Name	Bits	Description	Type	Reset
RESET_VECTOR	[31:0]	Entry address upon processor reset.	RW	0x80000000

21 Platform-Level Interrupt Controller (PLIC)

21.1 Introduction

Andes Platform-Level Interrupt Controller (NCEPLIC100) prioritizes and distributes global interrupts. It is compatible with RISC-V PLIC with the following features:

- Configurable interrupt trigger types
- Software-programmable interrupt generation
- Preemptive priority interrupt extension
- Vectored interrupt extension

The block diagram of NCEPLIC100 is shown in Figure 21-1 NCEPLIC100 Block Diagram. Interrupt sources (e.g., devices) send interrupt requests to NCEPLIC100 through the int_src signals. The signals can be level-triggered or edge-triggered, and they are converted to interrupt requests by the interrupt gateway. Interrupt requests are prioritized and routed to interrupt targets (e.g., AndesCore processors) according to interrupt settings. Interrupt settings include enable bits, priorities, and priority thresholds, and these settings are programmable through the bus interface.

The tx_eip (x stands for the target number) signal is an external interrupt pending notification signal to the targets. It is a level signal summarizing the interrupt pending (IP) status of all interrupt sources to target x. When a target takes the external interrupt, it should send an interrupt claim request (bus read request) to retrieve the interrupt ID, upon which the corresponding interrupt pending status bit will be cleared and tx_eip will be deasserted. tx_eip is guaranteed to be deasserted for at least one cycle even if there are pending interrupt sources still remaining. This is done to ensure that the interrupt detection logic of the target processor

could see the remaining interrupt pending status.

The interrupt gateway stops processing newer interrupt requests from its interrupt sources once it reports an interrupt request. When the target has serviced the interrupt, it should send the interrupt completion message (bus write request) to NCEPLIC100 such that the interrupt gateway resumes processing newer interrupt requests.

The interrupt pending bit array of the PLIC registers provides a summary of all interrupt sources status. In addition, it is also writable for setting software-programmed interrupts for the corresponding interrupt sources. See Interrupt Pending for more information.

NCEPLIC100 implements two Andes specific extensions. See Support for Preemptive Priority Interruptfor information regarding the Andes preemptive priority interrupt extension and Vectored Interrupts for information regarding the Andes vectored PLIC extension.

Figure 21-1 NCEPLIC100 Block Diagram
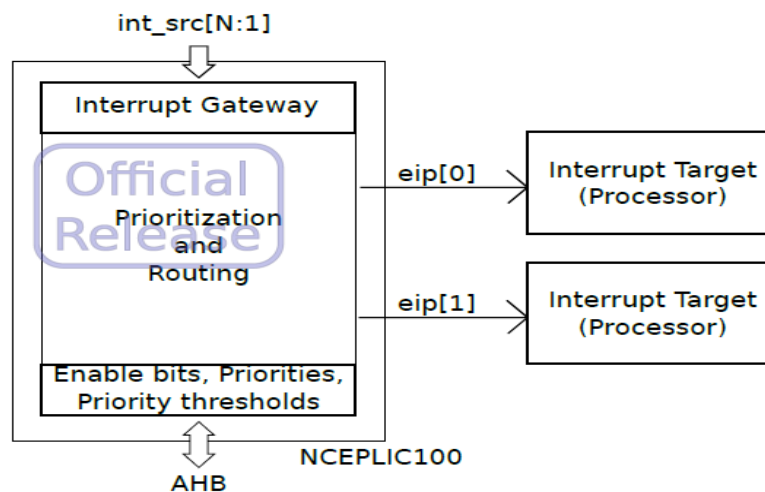
21.2 Support for Preemptive Priority Interrupt

NCEPLIC100 implements the Andes preemptive priority interrupt extension which enables faster responses for high-priority interrupts. With this extension, if a high-priority interrupt arrives and the global interrupt is enabled (i.e., mstatus.MIE is 1), the processor will stop servicing the current low-priority interrupt and begin servicing this new high-priority interrupt. The handling of the suspended lower-priority interrupts will resume only after the handling of the higher-priority interrupt ends. Interrupts of same or lower priorities will not cause preemption to take effect and interfere the handling of the current interrupt. They have to wait until the handling of the current interrupt finishes.

To support this feature, the PLIC core is enhanced with a preempted

priority stack for each target. The stack saves and restores priorities of the nested/preempted interrupts for the target it is associated. The operation of the preempted stack is implicitly performed through two regular PLIC operations (Interrupt Claim and Interrupt Completion). See the next two subsections for more information.

This feature is enabled by setting the PREEMPT field (bit 0) of the Feature Enable Register (offset: 0x0000) to 1.

21.2.1 Interrupt Claims with Preemptive Priority

When a target sends an interrupt claim message to the PLIC core, the PLIC core will atomically determine the ID of the highest-priority pending interrupt for the target and then deassert the corresponding source's IP bit. The PLIC core will then return the ID to the target.

At the same time, the priority number in the target's Priority Threshold Register will be saved to a preempted priority stack for that target and the new priority number of the claimed interrupt will be written to the Priority Threshold Register.

21.2.2 Interrupt Completion with Preemptive Priority

When a target sends an interrupt completion message to the PLIC core, in addition to forwarding the completion message to the associated gateway, the PLIC core will restore the highest priority number in the preempted priority stack back to the Priority Threshold Register of the target.

Note that out-of-order completion of interrupts is not allowed when this feature is turned on — the latest claimed interrupt should be completed first.

21.3 Vectored Interrupts

NCEPLIC100 enhances the RISC-V PLIC functionality with the vector mode extension to allow the interrupt target to receive the interrupt source ID without going through the target claim request protocol. This feature can shorten the latency of interrupt handling by enabling the interrupt target to run the corresponding interrupt handler directly upon accepting the external interrupt. It is enabled by setting the VECTORED field (bit 1) of the Feature Enable Register (offset: 0x0000) to 1.

Two extra interface signals, tx_eiid and tx_eiack, are added to facilitate interrupt handling in the vector mode. When a valid interrupt is sent to PLIC, PLIC would send tx_eip with tx_eiid. Upon accepting the interrupt, the

target asserts tx_eiack to PLIC and takes tx_eiid as the interrupt source ID. The assertion of tx_eiack would cause the deassertion of tx_eip and clearing of the tx_IP bit of corresponding interrupt source as does the handling of the interrupt claim request.

Note that interrupt completion messages are still required to notify the interrupt gateway the completion of interrupt handling and to forward additional interrupts to the PLIC core.

The interrupt priority arbitration works differently under the vector mode. In the non-vector mode, PLIC continues to arbitrate among all active interrupts even after the target is notified of occurrence of some interrupts (tx_eip sent to the target). Arbitration is not final until the claim request message is processed. In the vector mode, interrupt arbitration is final as soon as tx_eip is posted to the interrupt target. Interrupt arbitration resumes after tx_eiack is replied to PLIC and tx_eip is deasserted. The protocol does not change tx_eiid on the fly to allow PLIC and the interrupt target to operate at different clock domains.

The vector mode extension is designed such that each interrupt source is statically assigned to a single target. No two targets should compete servicing (claiming) the same interrupt source through the handshaking interface signals (tx_eiack). Otherwise, unpredictable results may occur. Despite automatic dispatching, the PLIC interrupt claim request protocol still works under the vector mode for the interrupt handler to claim additional interrupts.

21.3.1 Vector Mode Protocol

Figure 21-2 NCEPLIC100 Vector Mode Protocol
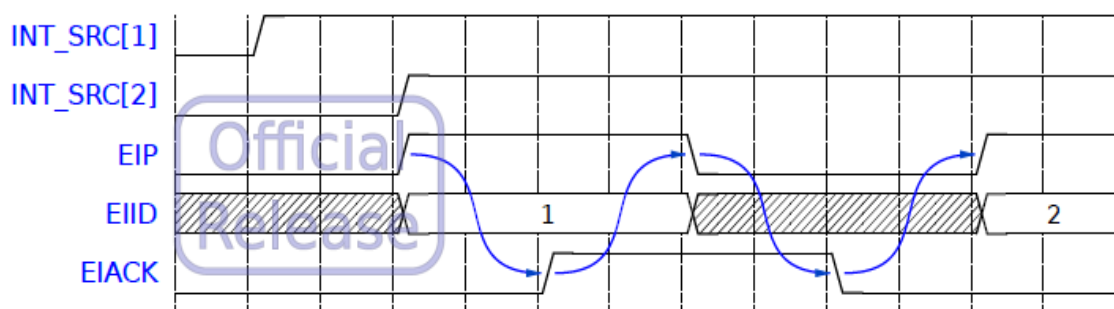
- tx_eiid remains stable when tx_eip is asserted.
- When tx_eip is asserted, it remains asserted until tx_eiack is asserted or an interrupt claim request is sent to PLIC.
- The assertion of tx_eiack causes the deassertion of tx_eip, which in turn causes the deassertion of tx_eiack.
- If there are more pending interrupts, tx_eip is asserted again after

deassertion of tx_eiack.

21.4 PLIC Configuration Options

Table 21-1 PLIC Configuration Parameters summarizes all supported configuration parameters and the subsections describe the parameters in detail.

Table 21-1 PLIC Configuration Parameters

Parameter Name	Type	Valid Values	Default Value
INT_NUM	Integer	1–1023	63
TARGET_NUM	Integer	1–16	1
MAX_PRIORITY	Integer	3/7/15/31/63/127/255	15
EDGE_TRIGGER	Integer	See Section 19.4.4	0
ASYNC_INT	Integer	See Section 19.4.5	0
ADDR_WIDTH	Integer	≥ 22	32
DATA_WIDTH	Integer	32/64	32
VECTOR_PLIC_SUPPORT	String	"yes"/"no"	"yes"
PLIC_BUS	String	"ahb"/"axi"	"axi"

21.4.1 Number of Interrupts

INT_NUM determines the number of interrupts.

21.4.2 Number of Targets

TARGET_NUM determines the number of interrupt targets.

21.4.3 Maximum Interrupt Priority

MAX_PRIORITY determines the valid priority levels of the interrupt sources and the target threshold. The priority value 0 is reserved to mean "never interrupt", and the larger the priority value, the higher the interrupt priority.

21.4.4 Edge Trigger

EDGE_TRIGGER is regarded as a bit vector and each bit controls whether the corresponding interrupt source is level-triggered or edge-triggered:

● Value 0 means level-triggered; and
● Value 1 means edge-triggered.

The bit width of EDGE_TRIGGER should be (INT_NUM+1).

21.4.5 Asynchronous Interrupt Source

ASYNC_INT is a bit vector where each bit indicates whether the corresponding interrupt source is asynchronous or synchronous.

- Value 0 means the interrupt source is synchronous.
- Value 1 means the interrupt source is asynchronous.

The bit width of ASYNC_INT should be (INT_NUM+1).

21.4.6 Address Width of PLIC Bus Interface

ADDR_WIDTH determines the address width of PLIC bus. The address width should be greater than or equal to 22 to encompass all addressable PLIC memory space.

21.4.7 Data Width of PLIC Bus Interface

DATA_WIDTH determines the data width of PLIC bus. PLIC AHB interface only supports 32-bit data width.

21.4.8 Support For Vectored PLIC Extension

VECTOR_PLIC_SUPPORT controls whether to include Andes Vectored PLIC Extension or not.

21.4.9 Bus Type of PLIC

PLIC_BUS determines the bus type of PLIC.

String "ahb" indicates PLIC is an AHB slave device.

21.5 PLIC Registers

21.5.1 Memory Map

NCEPLIC100 registers are accessed through bus transfers, and the memory map is shown in Table 21-2 AndeStar-V5 PLIC Memory Map. Please note that NCEPLIC100 supports only 32-bit transfers. Behaviors of 8-bit and 16-bit transfers are UNDEFINED, and the transfers might be ignored or result in error responses.

Table 21-2 AndeStar-V5 PLIC Memory Map

Address Offset		Description
Begin	**End**	
0x000000	0x000003	Feature enable register
0x000004	0x000007	Source 1 priority
0x000008	0x00000B	Source 2 priority
…	…	…
0x000FFC	0x000FFF	Source 1023 priority
0x001000	0x00107F	Pending array
0x001080	0x0010FF	Trigger type array
0x001100	0x001103	Number of interrupts and targets
0x001104	0x001107	Version and max priority register
0x002000	0x00207F	Target 0 interrupt enable bits
0x002080	0x0020FF	Target 1 interrupt enable bits
…	…	…
0x002780	0x0027FF	Target 15 interrupt enable bits
0x200000	0x200003	Target 0 priority threshold
0x200004	0x200007	Target 0 claim/complete
0x200400	0x20041F	Target 0 preempted priority stack
0x201000	0x20141F	Target 1 priority threshold, claim/complete, preempted priority stack
…	…	…
0x20F000	0x20F41F	Target 15 priority threshold, claim/complete, preempted priority stack

21.5.2 Feature Enable Register

Offset: 0x0

31		2	1	0
	0		VECTORED	PREEMPT
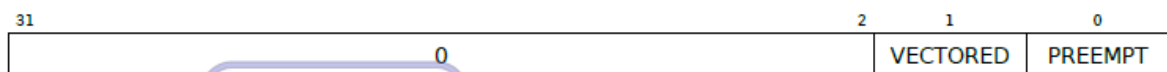
This register enables preemptive priority interrupt feature and the vector mode.

Field Name	Bits	Description	Type	Reset
PREEMPT	[0]	Preemptive priority interrupt enable	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
VECTORED	[1]	Vector mode enable	RW	0

Value	Meaning
0	Disabled
1	Enabled

Please note that both this bit and the mmisc_ctl.VEC_PLIC bit of the processor should be turned on for the vectored interrupt support to work correctly. See Section 15.8.7 for the definition of the VEC_PLIC bit.

21.5.3 Interrupt Source Priority

Offset: n*4

This register determines the priority for interrupt source n (1 $\leqslant$ n $\leqslant$ 1023).

Field Name	Bits	Description	Type	Reset
PRIORITY	[31:0]	Interrupt source priority. The valid range of this field is determined by the MAX_PRIORITY field of the Version & Maximum Priority Configuration Register.	RW	0

Value	Meaning
0	Never interrupt.
1–255	Interrupt source priority. The larger the value, the higher the priority.

21.5.4 Interrupt Pending

Offset: 0x1000 to 0x107F

These registers provide the interrupt pending status of interrupt sources, and a way for software to trigger an interrupt without relying on external devices. Every interrupt source occupies 1 bit. There are a total of 32 registers, each 32-bit wide, for 1023 interrupt sources.

When these registers are read, the interrupt pending status of interrupt sources are returned. The pending bits could be set by writing a bit mask that specifies the bit positions to be set, and this action would result in software-programmed interrupts of the corresponding interrupt sources. The pending bits could only be cleared through the Interrupt Claim requests.

The location of the interrupt pending bit for interrupt source n can be determined by the following equations:

- Word offset address: 0x1000 + 4 * floor(n/32)
- Bit Position: n modulo 32

21.5.5 Interrupt Trigger Type

Offset: 0x1080 to 0x10FF

These registers are read-only and indicate the configured interrupt trigger type of interrupt sources. Every interrupt source occupies 1 bit. There are a total of 32 registers, each 32-bit wide, for 1023 interrupt sources. The location of the interrupt trigger type bit for interrupt source n can be determined by the following equations:

- Word offset address: 0x1080 + 4 * floor(n/32)
- Bit Position: n modulo 32

The meaning of each bit is shown in Table 21-3 Meaning of Trigger Type.

Table 21-3 Meaning of Trigger Type

Value	Meaning
0	Level-triggered interrupt
1	Edge-triggered interrupt

21.5.6 Number of Interrupt and Target Configuration Register

Offset: 0x1100

31	16	15	0
NUM_TARGET		NUM_INTERRUPT	
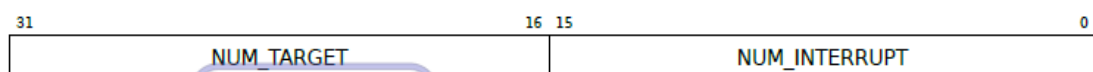
This register indicates the number of supported interrupt sources and supported targets.

Field Name	Bits	Description	Type	Reset
NUM_INTERRUPT	[15:0]	The number of supported interrupt sources	RO	IM
NUM_TARGET	[31:16]	The number of supported targets	RO	IM

21.5.7 Version & Maximum Priority Configuration Register

Offset: 0x1104

31	16	15	0
MAX_PRIORITY		VERSION	
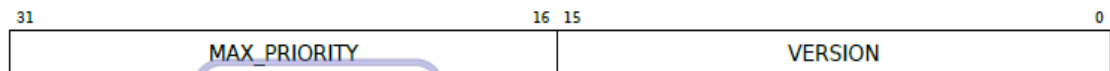
This register indicates the version and the maximum priority of PLIC implementation.

Field Name	Bits	Description	Type	Reset
VERSION	[15:0]	The version of the AndeStar-V5 PLIC design	RO	IM
MAX_PRIORITY	[31:16]	The maximum priority supported	RO	IM

21.5.8 Interrupt Enable Bits for Target m

Offset: (0x2000+m*128) to (0x207F+m*128)

These registers control the routing of interrupt source n to target m. Every interrupt source occupies 1 bit for each target to enable interrupt processing for that target. There are a total of 32 registers for 1023 interrupt sources for each target. The location of the interrupt enable bit for interrupt source n to target m can be determined by the following equations:

● Word offset address: 0x2000 + 128 * m + 4 * floor(n/32)
● Bit position: n modulo 32

21.5.9 Priority Threshold for Target m

Offset: 0x200000+4096*m

31	0
THRESHOLD	
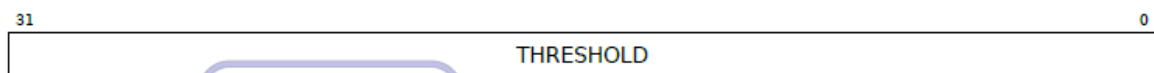
Each interrupt target is associated with one Priority Threshold Register. Only active interrupts with priorities strictly greater than the threshold will cause an interrupt notification to be sent to the target.

Field Name	Bits	Description	Type	Reset
THRESHOLD	[31:0]	Interrupt priority threshold. The valid range of this field is determined by the MAX_PRIORITY field of the Version & Maximum Priority Configuration Register.	RW	0

21.5.10 Claim and Complete Register for Target m

Offset: 0x200004+4096*m

There is one Claim and Complete Register for each interrupt target. Reading this register claims an interrupt source and returns the ID of that interrupt source.

The interrupt gateway stops processing newer interrupt requests from its interrupt sources until the earlier interrupt request is completed. Writing this register with an interrupt ID serves as the interrupt completion message acknowledging to PLIC that the handling of the claimed interrupt has been serviced in target m and the associated interrupt gateway may resume processing newer interrupt requests.

The interrupt gateway only resumes processing of newer interrupt requests if the enable bit of the interrupt source for Target m is set. If the enable bit is not set, the interrupt completion message is ignored.

Generally there are no limitations to the order of interrupt claims and completions except when the preemptive priority mode is enabled. When PLIC is in the preemptive priority mode, the latest claimed interrupt should be completed first.

Field Name	Bits	Description	Type	Reset
INTERRUPT_ID	[9:0]	On reads, indicating the interrupt source that has being claimed. On writes, indicating the interrupt source that has been handled (completed).	RW	0

21.5.11 Preempted Priority Stack Registers for Target m

Offset: (0x200400+4096*m) to (0x20041F+4096*m)

These registers are read/writable registers for accessing the preempted priority stack for target m. The purpose of these registers are for saving and restoring priorities of the nested/preempted interrupts for a particular target.

There are a total of 8 registers, each 32-bit wide, for 255 priority levels. Each bit in these registers indicates if the corresponding priority level has been preempted by a higher-priority interrupt. The location of the priority level bit for priority p of target m (Word offset Address, Bit Position) can be determined by the following equations:

- Word offset Address: 0x20_0400 + 4096 * m + 4 * floor(p/32)
- Bit Position: P modulo 32

21.6 Interrupt Latency

Figure 21-3 Minimum Interrupt Latency illustrates the minimum timing for the processor to execute the first instruction.

When a device asserts INT_SRC[n], it takes 3 BUS_CLK cycles for NCEPLIC100 to arbitrate interrupts and assert its MEIP output signal, where BUS_CLK is the clock source of NCEPLIC100. When the MEIP signal is asserted, it takes one CORE_CLK cycle for the processor to latch the signal into the mip register. The interrupt is usually taken immediately in the same cycle, unless current operations cannot be interrupted (e.g., load to device regions).

How the processor fetches the trap handler for handling the associated external interrupt depends on its vector interrupt setting. When mmisc_ctl.VEC_PLIC is 0, the processor fetches the instruction pointed by mtvec. When mmisc_ctl.VEC_PLIC is 1 (vector mode), mtvec points to a table of entry point addresses, one entry for each external interrupt. It takes the processor 3 additional CORE_ CLK cycles to fetch the entry point address, before fetching the first instruction of the associated trap handler. The waveform assumes that instruction fetch returns immediately without wait states.

The processor implements a 5-stage pipeline, so it takes at least 5 CORE_CLK cycles for the first instruction of the trap handler to execute and retire.

In summary, the minimum latency is 3 BUS_CLK and 6 CORE_CLK cycles. The latency is counted from assertion of INT_SRC[n] to the end of execution of the first instruction of the trap handler.

Figure 21-3 Minimum Interrupt Latency
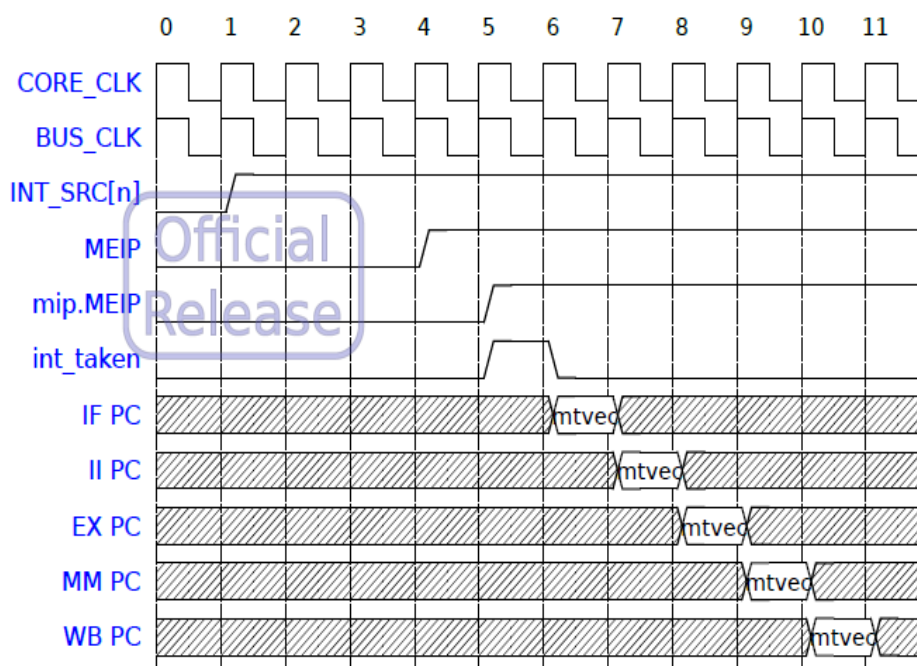
21.7 Interface Signals

NCEPLIC100 offers AHB bus interfaces. The interfaces are selected by the PLIC_BUS parameter. The interface signals to both interfaces are simultaneously present on its module port list and only the selected one will be used. The other group of signals will be unused and left floating.

The tables below describe the interface signals of NCEPLIC100 and the clock to NCEPLIC100 should be synchronous to that of AE250.

Table 21-6 Valid AHB Transactions for NCEPLIC1006 summarizes valid transactions that NCEPLIC100 accepts. It responds to invalid transactions by returning undefined values for read transactions and ignoring writes.

Table 21-4 General Signals of NCEPLIC100

Signal Name	Direction	Description
clk	input	Clock
reset_n	input	Reset (Active-Low)
int_src[INT_NUM:1]	input	Interrupt sources. The sources could be configured to be asynchronous inputs through the ASYNC_INT parameter. See Section 19.4.5 for detail.
tx_eip	output	External interrupt pending for target x.
tx_eiid[9:0]	output	External interrupt id for target x, see Section 19.3 for detail.
tx_eiack	input	Interrupt acknowledgment from target x, see Section 19.3 for detail.

Table 21-5 AHB Interface Signals of NCEPLIC100

Signal Name	Direction	Description
hsel	input	Slave Select
hrdata[31:0]	output	Read data bus
hready	input	Transfer done signal of AHB bus
hreadyout	output	Transfer done signal of PLIC
hresp[1:0]	output	Transfer response
haddr[ADDR_WIDTH-1:0]	input	Address bus
hburst[2:0]	input	Burst type
hlock	input	Locked transfer
hprot[3:0]	input	Protection control
hsize[2:0]	input	Transfer size
htrans[1:0]	input	Transfer type
hwdata[31:0]	input	Write data bus
hwrite	input	Transfer direction

Table 21-6 Valid AHB Transactions for NCEPLIC100

Transaction Type
Single WORD
INCR WORD
WRAP4 WORD
INCR4 WORD
WRAP8 WORD
INCR8 WORD
WRAP16 WORD
INCR16 WORD

22 Machine Timer

22.1 Introduction

The RISC-V architecture defines a machine timer that provides a real-time counter and generates timer interrupts. NCEPLMT100 is an implementation of the machine timer, and the block diagram is shown in the following figure.

This timer is not to be confused with the real-time clock timer of typical computing platforms. Per RISC-V privileged specification, the timer clock (mtime_clk) could be clocked at any arbitrary frequency as long as it is a fixed frequency clock that is not affected by clock gating or frequency scaling of clocks in the rest of the platform. On the other hand, real-time clock timer is usually clocked by a 32768Hz clock. The linux kernel expects microsecond resolution for the mtime, so it imposes an additional requirement that mtime_clk has to be greater than 1MHz. For non-linux applications, mtime_clk could share the same clock source as the real-time clock timer.

The RISC-V privileged specification expects that software discovers the frequency of mtime_clk through a platform specific mechanism. For linux kernels, this is achieved through the Device Tree specification.

NCEPLMT100 imposes a frequency limitation on the frequency of the mtime_clk clock. The mtime update synchronization logic requires that the frequency of the bus interface of NCEPLMT100 should be at least 2x faster than that of the mtime_clk clock when mtime is accessed.

On Andes evaluation platforms, the frequency of mtime_clk is set to the normal operating frequency of APB clocks to minimize the number of clock sources in FPGA.

NCEPLMT100 primarily consists of two registers: mtime and mtimecmp. The mtime register is a 64-bit real-time counter clocked by mtime_clk. The mtimecmp register stores a 64-bit value for comparing with

mtime. When the value in mtime is greater than or equal to the value in mtimecmp, the mtip signal is asserted for generating a timer interrupt. When mtimecmp is written, the interrupt is cleared and the mtip signal is deasserted.

Figure 22-1 NCEPLMT100 Block Diagram
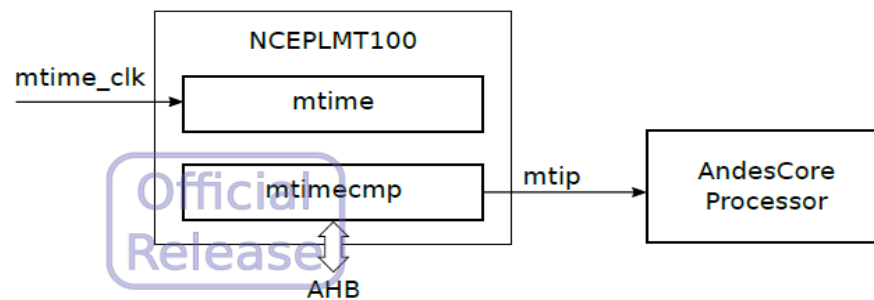
22.2 Machine Timer Registers

NCEPLMT100 registers are accessed through the bus interface, and their memory map is shown in Table 22-1 NCEPLMT100 Memory Map.

Please note that NCEPLMT100 supports only 32-bit transfers. Behaviors of 8-bit and 16-bit transfers are UNDEFINED, and these transfers might be ignored as well as result in error responses or unexpected register updates.

Table 22-1 NCEPLMT100 Memory Map

Address Offset	Description
0x0 – 0x3	mtime[31:0]
0x4 – 0x7	mtime[63:32]
0x8 – 0xB	mtimecmp0[31:0]
0xC – 0xF	mtimecmp0[63:32]
0x10 – 0x13	mtimecmp1[31:0]
0x14 – 0x17	mtimecmp1[63:32]
0x18 – 0x1B	mtimecmp1[31:0]
0x1C – 0x1F	mtimecmp1[63:32]
0x(n+8) – 0x(n+B)	mtimecmpn[31:0]
0x(n+C) – 0x(n+F)	mtimecmpn[63:32]

The mtime register is driven by mtime_clk, which is assumed to be slower than the clock of the bus interface. The mtime_shadow shadow register is maintained in the bus clock domain to reduce the latency of accessing the mtime register in the slow clock domain. The values of mtime and mtime_s hadow registers are constantly synchronized such that mtime_shadow maintains the most up-to-date values of mtime. The value

in mtime_shadow is instantly returned when reading mtime register. When writing the mtime register, bus write transactions finish when the values are written to the mti me_shadow register, and NCEPLMT100 handles the synchronization to mtime in the background.

22.3 Interface Signals

The interface ports of the NCEPLMT100 are described in this section. All signals are Active-High unless otherwise indicated.

NCEPLMT100 offers one type of bus interfaces: AHB interface. The interfaces are selected by the BUS_TYPE parameter. The interface signals to both interfaces are simultaneously present on its module port list and only the selected one will be used. The other group of signals will be unused and left floating.

The tables below describe the interface signals of NCEPLMT100 and the clock to NCEPLMT100 should be synchronous to that of AE250.

Table 22-5 summarizes valid transactions that NCEPLMT100 accepts.

Table 22-2 General Signals of NCEPLMT100

Signal Name	Direction	Description
clk	input	Clock for the bus interface
resetn	input	Reset for the bus interface (Active-Low)
mtime_clk	input	Clock for the mtime counter. See Section 20.1
por_rstn	input	Power on reset (Active-Low) for initializing the mtime counter
stoptime	input	Stop counting the mtime counter
mtip[NHART-1:0]	output	Timer interrupt pending

Table 22-3 AHB Interface Signals of NCEPLMT100

Signal Name	Direction	Description
hsel	input	Slave Select
hrdata[31:0]	output	Read data bus
hready	input	Transfer done signal of AHB bus
hreadyout	output	Transfer done signal of Machine Timer
hresp[1:0]	output	Transfer response
haddr[ADDR_WIDTH-1:0]	input	Address bus
hburst[2:0]	input	Burst type
hlock	input	Locked transfer
hprot[3:0]	input	Protection control
hsize[2:0]	input	Transfer size
htrans[1:0]	input	Transfer type
hwdata[31:0]	input	Write data bus
hwrite	input	Transfer direction

Table 22-5 Valid AHB Transactions for NCEPLMT100

Transaction Type
Single WORD

23 Debug Subsystem

23.1 Overview

The AE250 debug subsystem implements RISC-V External Debug Support (TD003) V0.13. Figure 23-1 Debug Subsystem Block Diagram shows the block diagram of the debug subsystem, which contains two components: NCEPLDM200 and NCEJDTM200. NCEPLDM200 is the debug module, which could be accessed through its two AHB slave ports. One is the system interface port, which is for an AE250 processor to access the debug module through the system bus. The other one is the Debug Memory Interface (DMI) port, which is accessed by NCEJDTM200 (JTAG Debug Transport Module). NCEJDTM200 converts debug commands in JTAG interfaces of external debuggers to bus read/write requests to the DMI port.

Figure 23-1 Debug Subsystem Block Diagram
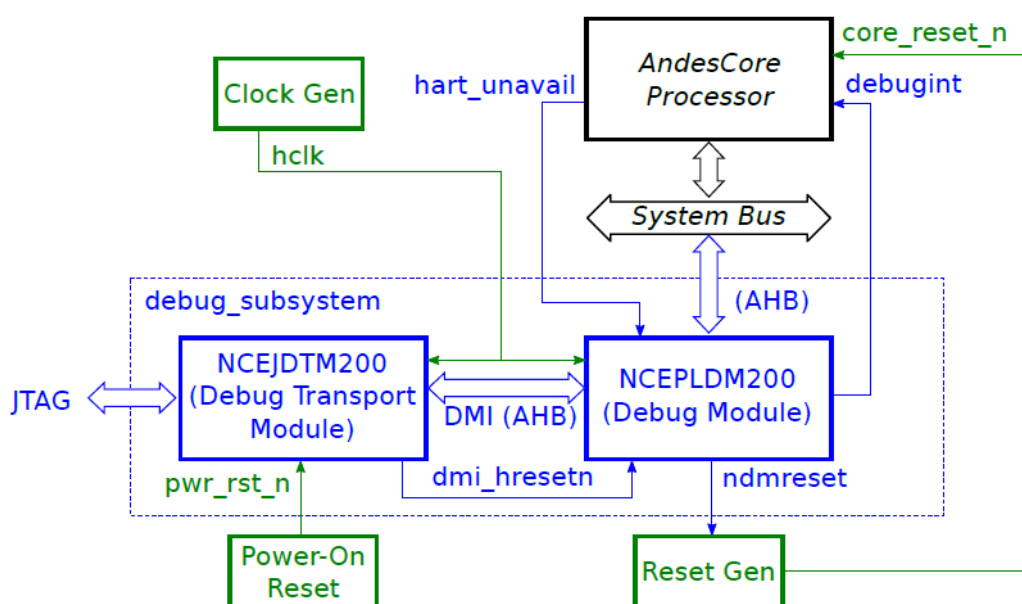
Debug interrupts cause AE250 to enter the debug mode and redirect the instruction fetch to the debug exception handler, whose entry point should be the base address of NCEPLDM200 and defined by the DEBUG_VEC parameter of AE250.

NCEPLDM200 includes a Debug ROM at its base address that defines the debug exception handler. When invoked, the debug handler polls NCEPLDM200 internal registers to process commands issued by the external debugger through the NCEJDTM200 module. Typical debug commands include accessing processor registers, accessing memories, and executing programs written in the Program Buffer, which is a memory region writable by the external debugger.

Correct operations of the debug subsystem require clocks, resets and I/O interfaces to be connected in a certain way, as well as making sure the DEBUG_VEC parameter is within a device region (Device Regions). Please see the next subsection for information.

23.2 Integration Requirements

For proper operations of the debug subsystem, the following platform-level requirements of reset, clock and I/O signals should be met:

- NCEJDTM200 should be reset by power-on resets, which will not be triggered by other resets in the system.
- NCEPLDM200 should be reset by dmi_hresetn that is generated by NCEJDTM200.
- ndmreset (non-debug-module reset) should connect to the platform reset generator for triggering platform-wide resets, which shall include core_reset_n that drives AndesCore processors. As the name suggests, ndmreset should not reset both NCEJDTM200 and NCEPLDM200. Please also note that the platform reset should not affect the pin-muxing of the external debug interface (JTAG) pins to preserve connectivity to the external debugger throughout resets.
- ndmreset should not cause assertion of power-on resets (pwr_rst_n).
- The clock signal (clk) to NCEJDTM200 and NCEPLDM200 should keep running during the assertion of ndmreset. Stopped clocks will impede the deassertion of ndmreset.
- The bus interfaces of NCEJDTM200 and NCEPLDM200 should be in the same clock domain.
- Both tck and tms pins of the JTAG interface should not be floating when the external debugger is not attached:
- tck can be either pulled up to HIGH or pulled down to LOW.
- tms should be pulled up to HIGH.

23.3 Debug Subsystem Configuration Options

Please see Section 5.4 for the description of Debug Subsystem Configuration Options.

23.4 NCEPLDM200

Table 23-1 System Memory Map of NCEPLDM200 summarizes the memory map within the NCEPLDM200 address space as viewed from the system bus interface. Please note that the offset for Program Buffer could be discovered by the external debugger through execution of the AUIPC instruction as the first instruction in Program Buffer, and the starting offset of Abstract Data is defined as hartinfo.DATAADDR. The offsets could be used as offsets of load/store instructions with the zero register as the base register to access this memory space. The zero register is automatically mapped to the base of the NCEPLDM200 for load/store instructions in Debug Mode.

This system bus address space of NCEPLDM200 is defined through the DEBUG_VEC parameter. It should be within a device region for the proper operation of the Debug ROM and the external debugger support. Please see Device Regions for information on how to setup this address space as device regions.

Table 23-2 DMI Memory Map of NCEPLDM200 summarizes the memory map as viewed from the DMI interface. The address[8:2] value in the table follows the address value assignment of the debug module debug bus registers as described in RISC-V External Debug Support (TD003) V0.13.

Table 23-1 System Memory Map of NCEPLDM200

Address Offset	Description
0x0000 – 0x007F	Debug ROM
0x0080 – 0x00BF	Program Buffer
0x00C0 – 0x00CF	Abstract Data 0–3
0x00D0 – 0x01FF	Reserved for internal use

Table 23-2 DMI Memory Map of NCEPLDM200

Address	Description
0x04 – 0x07	Abstract Data 0–3
0x10	Debug Module Control
0x11	Debug Module Status
0x12	Hart Info
0x13	Halt Summary
0x14	Hart Array Window Select
0x15	Hart Array Window
0x16	Abstract Control and Status
0x17	Abstract Command
0x18	Abstract Command Autoexec
0x19 – 0x1C	Device Tree Addr 0–3
0x20 – 0x2F	Program Buffer 0–15
0x30	Authentication Data
0x38	System Bus Access Control and Status
0x39 – 0x3B	System Bus Address
0x3C – 0x3F	System Bus Data

23.4.1 Abstract Data 0–3 (data0–3)

Basic read/write registers that may be read or changed by abstract commands.

The registers are accessible from both the DMI interface and the system bus interface to support data exchanges between the external debugger and the processor (i.e., instructions in the Program Buffer).

23.4.2 Debug Module Control (dmcontrol)

31	30	29	28	27	26	25	16	15	2	1	0
haltreq	resumereq	0	ackhavereset	0	hasel	hartsel		0		ndmreset	dmactive

Field Name	Bits	Description		Type	Reset
dmactive	[0]	Controlling reset signal for Debug Module itself.		RW	0x0
		Value	**Meaning**		
		0	The Debug Module's state takes its reset values.		
		1	The Debug Module functions normally.		
ndmreset	[1]	Controlling reset signal from the Debug Module to the rest of the system.		RW	0x0
		Value	**Meaning**		
		0	Deassert system reset signal.		
		1	Assert system reset signal.		
hartsel	[25:16]	Selecting the target hart to be debugged.		RW	0x0
hasel	[26]	How harts are selected. This field is 0x0 for N25(F) indicating that there is only one hart selected through `hartsel`.		RO	0x0
ackhavereset	[28]	Writing 1 to this bit clears the havereset bits for any selected harts. Harts are selected based on the new value of hartsel and hasel being written.		RW	0x0
resumereq	[30]	Writes the resume request bit for all currently selected harts. When set to 1, each selected hart will resume if it is currently halted. The resume request bit is ignored while the halt request bit is set. Harts are selected based on the new value of hartsel and hasel being written.		WO	0x0
haltreq	[31]	Writes the halt request bit for all currently selected harts. When set to 1, each selected hart will halt if it is not currently halted. Writing 1 or 0 has no effect on a hart which is already halted, but the bit must be cleared to 0 before the hart is resumed. Harts are selected based on the new value of hartsel and hasel being written.		WO	0x0

23.4.3 Debug Module Status (dmstatus)

9	8	7	6	5	4	3	0
allhalted	anyhalted	authenticated	authbusy	0	devtreevalid	version	

15	14	13	12	11	10
allnonexistent	anynonexistent	allunavail	anyunavail	allrunning	anyrunning

31 23	22	21 20	19	18	17	16
0	impebreak	0	allhavereset	anyhavereset	allresumeack	anyresumeack

Field Name	Bits	Description	Type	Reset
Version	[3:0]	Version of the implemented RISC-V External Debug Support. 0x2 indicates that the current implemented version is 0.13.	RO	0x2
devtreevalid	[4]	Whether the information in devtreeaddr0 – devtreeaddr3 registers hold the address of the Device Tree.	RO	0x0
authbusy	[6]	<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>The authentication module is ready to process the next read/write to *authdata*.</td></tr><tr><td>1</td><td>The authentication module is busy.</td></tr></table>	RO	0x0
authenticated	[7]	<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Authentication is required before using the Debug Module.</td></tr><tr><td>1</td><td>Authentication check has passed.</td></tr></table>	RO	0x1
anyhalted	[8]	Indicating whether any currently selected hart is halted.	RO	0x0
allhalted	[9]	Indicating whether all currently selected harts are halted.	RO	0x0
anyrunning	[10]	Indicating whether any currently selected hart is running.	RO	0x0

Field Name	Bits	Description	Type	Reset
allrunning	[11]	Indicating whether all currently selected harts are running.	RO	0x0
anyunavail	[12]	Indicating whether any currently selected hart is unavailable.	RO	0x0
allunavail	[13]	Indicating whether all currently selected harts are unavailable.	RO	0x0
anynonexistent	[14]	Indicating whether any currently selected hart does not exist in this system.	RO	0x0
allnonexistent	[15]	Indicating whether all currently selected harts do not exist in this system.	RO	0x0
anyresumeack	[16]	Indicating whether any currently selected hart has acknowledged the previous resume request.	RO	0x0
allresumeack	[17]	Indicating whether all currently selected harts have acknowledged the previous resume request.	RO	0x0
anyhavereset	[18]	Indicating whether any currently selected hart has been reset but the reset has not been acknowledged	RO	0x0
allhavereset	[19]	Indicating whether all currently selected harts have been reset but the reset has not been acknowledged	RO	0x0
impebreak	[22]	Indicating whether there is an implicit ebreak instruction at the non-existent word immediately after the Program Buffer.	RO	0x1

23.4.4 Hart Info (hartinfo)

31 24	23 20	19 17	16	15 12	11 0
0	nscratch	0	dataaccess	datasize	dataaddr

Field Name	Bits	Description	Type	Reset
dataaddr	[11:0]	Signed offset for accessing the shadowed *data* registers by the processor, to be used as offsets for load/store instructions with the `zero` register as the base register in Debug Mode.	RO	0xC0
datasize	[15:12]	Number of 32-bit words in the memory map dedicated to shadowing the data registers.	RO	0x4
dataaccess	[16]	The method for accessing the shadowed *data* registers. The value of this field is 0x1 for N25(F), indicating that the *data* registers are shadowed in the memory map under Debug Mode.	RO	0x1
nscratch	[23:20]	Number of dscratch registers available for the debugger to use during program buffer execution, starting from dscratch0.	RO	0x2

For the dataaccess field:

Value	Meaning
0	The *data* registers are shadowed in the hart by CSR registers.
1	The *data* registers are shadowed in the hart's memory map. Each register takes up 4 bytes in the memory map.

23.4.5 Halt Summary (haltsum)

31		1	0
	0		halt31_0
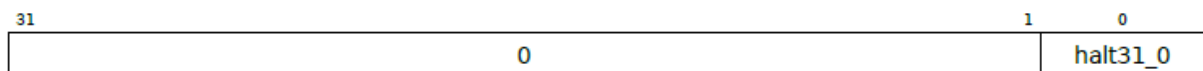
Bit 0 contains the logical OR of 32 halt bits for hart 0 – hart 31.

23.4.6 Hart Array Window Select (hawindowsel)

31		15	14		0
	0			hawindowsel	
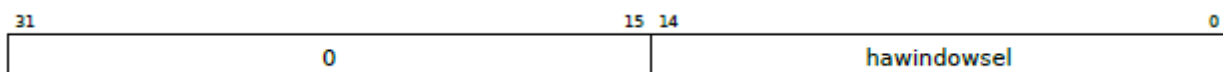
Field Name	Bits	Description	Type	Reset
hawindowsel	[14:0]	This register selects which of the 32-bit portion of the hart array mask register is accessible in hawindow.	RW	0x0

23.4.7 Hart Array Window (hawindow)

31		0
	hawindow	
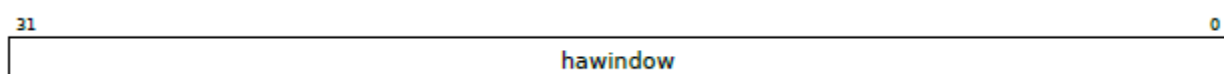
Field Name	Bits	Description	Type	Reset
hawindow	[31:0]	This register provides R/W access to a 32-bit portion of the hart array mask register The position of the window is determined by `hawindowsel`. I.e. bit 0 refers to hart (`hawindowsel` * 32), while bit 31 refers to hart (`hawindowsel` * 32 + 31).	RW	0x0

23.4.8 Abstract Control and Status (abstractcs)

31	29 28	24 23	13	12	11	10	8 7	5 4	0
0	progbufsize	0		busy	0	cmderr		0	datacount

Field Name	Bits	Description	Type	Reset
datacount	[4:0]	Number of *data* registers that are implemented.	RO	0x4
cmderr	[10:8]	Error code indicating that an abstract command fails. The bits in this field remain set until they are cleared by writing 1 to them. No abstract command is started until the value is reset to 0.	R/W1C	0x0

Value	Meaning
0	None: no error
1	Busy: an abstract command was executing while command, abstractcs, abstractauto was written, or when one of the data or progbuf registers was read or written
2	Not supported: the requested command is not supported.
3	Exception: an exception occurred while executing the command.
4	Halt/resume: an abstract command couldn't execute because the hart wasn't in the expected state (running/halted).

Field Name	Bits	Description	Type	Reset
busy	[12]	Flag indicating an abstract command is currently being executed.	RO	0x0
progbufsize	[28:24]	Size of the program buffer, in 32-bit words.	RO	0x8

23.4.9 Abstract Command

31	24 23	0
cmdtype	control	

Field Name	Bits	Description	Type	Reset
control	[23:0]	The field is interpreted in a command-specific manner.	WO	0x0
cmdtype	[31:24]	Controlling the overall functionality of this abstract command.	WO	0x0

The Abstract Data registers hold the command arguments or returned values, as shown in Table23-1

Table 23-3 Use of Data Registers in PLDM

Argument Width	arg0/return value	arg1	arg2
32	data0	data1	data2
64	data0, data1	data2, data3	data4, data5

Access Register

31 24	23	22 20	19	18	17	16	15 0
cmdtype (0)	0	size	0	postexec	transfer	write	regno

This command gives the debugger access to CPU registers and allows CPU to execute the program buffer.

Field Name	Bits	Description		Type	Reset
regno	[15:0]	Number of the register to access.		WO	0x0
write	[16]	The direction of data transfer.		WO	0x0

Value	Meaning
0	Copy data from the specified register into arg0 portion of the Abstract Data registers.
1	Copy data from arg0 portion of Abstract Data registers into the specified register.

Field Name	Bits	Description		Type	Reset
transfer	[17]	Indicating whether to perform data transfer.		WO	0x0

Value	Meaning
0	Don't do the operation specified by write.
1	Do the operation specified by write.

Field Name	Bits	Description		Type	Reset
postexec	[18]	Indicating whether to execute the program in the Program Buffer. If this field is set, execute the program in the Program Buffer exactly once after performing the transfer, if any.		WO	0x0
size	[22:20]			WO	0x0

Value	Meaning
2	Access the lowest 32 bits of the register
3	Access the lowest 64 bits of the register.

Quick Access

31 24	23 0
cmdtype (1)	0

Perform the following sequence of operations:

- If the hart is halted already, the command sets cmderr to halt/resume and does not continue.
- Halt the hart. If the hart halts for some other reason (e.g., breakpoint), the command sets cmderr to halt/resume and does not continue.
- Execute the Program Buffer. If an exception occurs, cmderr is set to exception and the program buffer execution ends, but the quick access command continues.

● Resume the hart.

Access Memory

| cmdtype (2) | aamvirtual | aamsize | aampostincrement | 0 | write | 0 |

(bit positions: 31 · 24 23 · 22 20 · 19 · 18 17 · 16 · 15 · 0)

This command lets the debugger perform memory accesses with the memory view of the selected hart.

Field Name	Bits	Description	Type	Reset
write	[16]	The direction of data transfer. Value / Meaning: 0 — Copy data from the memory location into *arg0* portion of the Abstract Data registers. 1 — Copy data from *arg0* portion of Abstract Data registers into the memory location.	WO	0x0
aampostincrement	[19]	Increment *arg1* by the number of bytes encoded in `aamsize` after a memory access has completed.	WO	0x0
aamsize	[22:20]	Size of memory accesses. Value / Meaning: 0 — Access the lowest 8 bits of the memory location. 1 — Access the lowest 16 bits of the memory location. 2 — Access the lowest 32 bits of the memory location. 3 — Access the lowest 64 bits of the memory location.	WO	0x0
aamvirtual	[23]	Virtual or physical address access Value / Meaning: 0 — Addresses are physical 1 — No action	WO	0x0

23.4.10 Abstract Command Autoexec (abstractauto)

| 0 | autoexecprogbuf | 0 | 0 | autoexecdata |

(bit positions: 31 · 24 23 · 16 15 · 12 11 · 4 3 · 0)

Field Name	Bits	Description	Type	Reset
autoexecdata	[3:0]	When a bit in this field is 1, read or write accesses to the corresponding data word cause the command in command to be executed again.	RW	0x0
autoexecprogbuf	[23:16]	When a bit in this field is 1, read or write accesses to the corresponding progbuf word cause the command in command to be executed again.	RW	0x0

23.4.11 Device Tree Addr 0–3 (devtreeaddr0–3)

The devicetreeaddr registers are hard-wired to zeros in NCEPLDM200.

23.4.12 Program Buffer 0–15 (progbuf0–15)

The progbuf registers provide read/write access to the program buffer. NCEPLDM200 supports programbuffer 0–7 only. Program buffer 8–15 are hard-wired to 32'h00100073 (the EBREAK instruction).

These registers are read/write accessible from both the DMI interface and the system bus, in addition to being a valid region for instruction fetches. They hold small programs written by the external debugger and these small programs will be fetched and executed by the processor upon execution of the abstract commands which require execution of the program buffer.

Programs in progbuf must end with EBREAK or C.EBREAK instructions if the program size is less than 8 words (32 bytes).

23.4.13 Authentication Data (authdata)

The authdata register is hard-wired to zeros in NCEPLDM200.

23.4.14 System Bus Access Control and Status (sbcs)

15	14	12 11	5	4	3	2	1	0
sbreadondata	sberror	sbasize	sbaccess128	sbaccess64	sbaccess32	sbaccess16	sbaccess8

31	29 28	23	22	21	20	19	17	16
sbversion	0	sbbusyerror	sbbusy	sbreadonaddr	sbaccess	sbautoincrement

The *sbcs* register holds the control bits and status flags of System Bus Accesses. It is only valid when the SYSTEM_BUS_ACCESS_SUPPORT parameter of NCEPLDM200 is set. It is otherwise hardwired to zero.

Field Name	Bits	Description	Type	Reset
sbaccess8	[0]	This bit indicates the supported status of 8-bit system bus data accesses.	RO	0x1
sbaccess16	[1]	This bit indicates the supported status of 16-bit system bus data accesses.	RO	0x1
sbaccess32	[2]	This bit indicates the supported status of 32-bit system bus data accesses.	RO	0x1
sbaccess64	[3]	This bit indicates the supported status of 64-bit system bus data accesses. This bit is 1 if SYS_DATA_WIDTH == 64.	RO	Configuration Dependent
sbaccess128	[4]	This bit indicates the supported status of 128-bit system bus data accesses.	RO	0x0
sbasize	[11:5]	Address width of System bus addresses in bits. This field is 0 if there is no bus access support. Otherwise, it is the value of the SYS_ADDR_WIDTH parameter of NCEPLDM200.	RO	Configuration Dependent

Field Name	Bits	Description	Type	Reset
sberror	[14:12]	Error code indicating the failure type of the system bus accesses. Write 1 to clear the status. 	Value	Meaning
---	---
0	No bus error
1	Timeout error
2	Bad address
3	Alignment error
4	Unsupported size was requested
7	Others		R/W1C	0x0
sbreadondata	[15]	Every read from *sbdata0* automatically triggers a system bus read at the (possibly auto-incremented) address.	RW	0x0
sbautoincrement	[16]	*sbaddress* is incremented by the size specified in *sbaccess_* after every system bus access.	RW	0x0
sbaccess	[19:17]	Access size. 	Value	Meaning
---	---
0	8-bit
1	16-bit
2	32-bit
3	64-bit
4	128-bit		RW	0x2
sbreadonaddr	[20]	Every write to *sbaddress0* automatically triggers a system bus read at the new address.	RW	0x0
sbbusy	[21]	Indicate the system bus master is busy.	RO	0x0
sbbusyerror	[22]	Indicate the debugger attempts to execute system bus access before *sbbusy* is cleared.	R/W1C	0x0
sbversion	[31:29]	The version of the supported System Bus Access version. It is currently 1 for v0.13 of the RISC-V External Debug Support specification.	RO	0x1

Note!

The support of $sbaccess8 - sbaccess128$ is set when the SYS_ADDR_WIDTH parameter in NCEPLDM200 covers the relative access size.

23.4.15 System Bus Address (sbaddress0–2)

The sbaddress registers are only valid when the SYSTEM_BUS_ACCESS_SUPPORT parameter of NCEPLDM200 is set. They are otherwise hardwired to zeros.

Table 23-4 System Bus Address Register

Address register	Description
sbaddress0	bit[31:0] of the address in *sbaddress*
sbaddress1	bit[63:32] of the address in *sbaddress*
sbaddress2	bit[95:64] of the address in *sbaddress*
sbaddress3	bit[127:96] of the address in *sbaddress*

23.4.16 System Bus Data (sbdata0–3)

The *sbdata* register is supported as below when SYSTEM_BUS_ACCESS_SUPPORT parameter of NCEPLDM200 is set. Otherwise it is hardwired to zeros.

Table 23-5 System Bus Data Register

Address register	Description
sbdata0	bit [31:0] of the address in *sbdata*
sbdata1	bit [63:32] of the address in *sbdata*
sbdata2	bit [95:64] of the address in *sbdata*
sbdata3	bit [127:96] of the address in *sbdata*

23.4.17 Interface Signals

The interface ports of the NCEPLDM200 are described in this section. All signals are Active-High unless otherwise indicated.

The interface ports consist of four groups of interfaces: a DMI interface, a bus slave interface, a system bus master interface and the rest of signals (the General Signals interface).

The DMI interface is a dedicated bus interface for communicating with debug transport modules (NCEJDTM200).

The bus slave interface is the bus interface for the target processor and the Debug Module to exchange data.

The system bus master interface is an optional interface for supporting the direct System Bus Access feature of the RISC-V External Debug Specification. This feature is not enabled by default and it can be turned on through the SYSTEM_BUS_ACCESS_ SUPPORT parameter of NCEPLDM200. The interface signals, however, are always present on the

module port list regardless of configuration. They should be left floating and ignored when not enabled.

NCEPLDM200 offers AHB interfaces for the bus slave interface and the system bus interface. The interfaces are selected by the RV_BUS_TYPE and SYS_BUS_ TYPE parameters. The interface signals to both interfaces are simultaneously present on its module port list and only the selected one will be used. The other group of signals will be unused and left floating. The values of RV_BUS_TYPE and SYS_BUS_TYPE parameters should match that of the BIU_BUS type of AE250. Note that the bus protocol of the DMI interface is not configurable and unconditionally the AHB interface.

The rest of signals include the clock and reset signals, processor status and debug-interrupt signals. NCEPLDM200 support multi-hart (multi-target) debugging. The number of hart supported is controlled by its NHART parameter, and the width of all hart specific signals are NHART-wide, one bit per hart.

The tables below describes the interface signals of NCEPLDM200 and the clock to NCEPLDM200 should be synchronous to that of AE250. Note that the slave bus interface signals and the system bus master signals are listed in the same tables, grouped according to their bus interface types.

Please see also Integration Requirements for integration requirements of NCEPLDM200.

Table 23-6   General Signals of NCEPLDM200

Signal Name	Direction	Description
clk	input	Clock input. This clock should also drive the DMI interface of NCEJDTM200. Furthermore, this clock should not be stopped during ndmreset. See Section 21.2 for integration requirements.
reset_n	input	Reset (Active-Low). The reset signal should be driven by NCEJDTM200. It should not be active when ndmreset is asserted. See Section 21.2 for integration requirements.
hart_halted[NHART-1:0]	input	Hart halted, one bit per hart. Each bit will be 1 if the corresponding hart is halted (in the debug mode).
hart_unavail[NHART-1:0]	input	Hart unavailable, one bit per hart. Each bit will be 1 if the corresponding hart is not available for accesses by the external debugger. The hart could be in the reset or some kind of power-gating state.
hart_under_reset[NHART-1:0]	input	Hart under reset, one bit per hart. Each bit will be 1 if the corresponding hart is under reset.
debug_int[NHART-1:0]	output	Debug interrupt, one bit per hart. Each bit will be 1 if the external debugger makes a debug request to the corresponding hart. Each hart should respond to the request by entering the debug mode.
dmactive	output	Debug module active. This signal reflects the value of dmcontrol.DMACTIVE.
ndmreset	output	Non-debug module reset. This signal should be routed to the platform reset controller, so that the external debugger could trigger system reset for the platform. See Section 21.2 for integration requirements.

Table 23-7 DMI Interface Signals of NCEPLDM200

dmi_hrdata[31:0]	output	DMI read data bus
dmi_hreadyout	output	DMI transfer done of NCEPLDM200
dmi_hresp[1:0]	output	DMI transfer response
dmi_hsel	input	DMI selection
dmi_htrans[1:0]	input	DMI transfer type
dmi_haddr[9:0]	input	DMI address bus
dmi_hburst[2:0]	input	DMI burst type
dmi_hprot[3:0]	input	DMI protection control
dmi_hsize[2:0]	input	DMI transfer size
dmi_hready	input	DMI transfer done
dmi_hwrite	input	DMI transfer direction
dmi_hwdata[31:0]	input	DMI write data bus

Table 23-8 AHB Interface Signals of NCEPLDM200

Signal Name	Direction	Description
rv_hrdata[BIU_DATA_WIDTH-1:0]	output	Processor read data bus
rv_hreadyout	output	Processor transfer done of NCEPLDM200
rv_hresp[1:0]	output	Processor transfer response
rv_haddr[31:0]	input	Processor address bus
rv_hburst[2:0]	input	Processor burst type
rv_hprot[3:0]	input	Processor protection control
rv_hsize[2:0]	input	Processor transfer size
rv_htrans[1:0]	input	Processor transfer type
rv_hwdata[BIU_DATA_WIDTH-1:0]	input	Processor write data bus
rv_hwrite	input	Processor transfer direction
rv_hsel	input	Processor selection
rv_hready	input	Processor transfer done
sys_hrdata[BIU_DATA_WIDTH-1:0]	input	System bus read data bus
sys_hready	input	System bus transfer done
sys_hgrant	input	System bus bus grant
sys_hresp[1:0]	input	System bus transfer response
sys_haddr[31:0]	output	System bus address bus
sys_hburst[2:0]	output	System bus burst type
sys_hprot[3:0]	output	System bus protection control

23.4.18 Non-polling access to Debug Module

Under halt mode, a core keeps executing commands from NCEPLDM200. Non-polling mechanism reduces unnecessary polling access from a core to NCEPLDM200. The core enters wait-state and waits until there are new commands available in NCEPLDM200. NCEPLDM200 informs the target core to leave wait-state via debugint when there is any new command requested from debugger.

The NCEPLDM200 with non-polling access mechanism works well with the legacy cores without implementing wait-state. The core without wait-state could still poll for the availability of next abstract command.

23.5 NCEJDTM200

NCEJDTM200 implements JTAG debug transport module (DTM), as defined by the spec: RISC-V

External Debug Support (TD003) V0.13. It implements a IEEE 1149.1 style test access port controller (TAP). The supported commands are summarized in Table 23-6 TAP Instruction NCEJDTM200.

Table 23-6 TAP Instruction NCEJDTM200

Encoding	Instruction
b11111	BYPASS
b00001	IDCODE
b10000	dtmcs
b10001	dmi

23.5.1 Interface Signal

Table 23-11 NCEJDTM100 Interface Signals

Signal Name	Direction	Description
pwr_rst_n	Input	Power on reset for ncejdtm200
TCK	Input	JTAG TCK clock
TMS	Input	JTAG TMS signal
TDI	Input	JTAG TDI signal
TDO	Output	JTAG TDO signal
dmi_hresetn	Output	The reset signal for NCEPLDM200
dmi_hsel	Output	DMI selection
dmi_haddr	Output	DMI address bus
dmi_htrans	Output	DMI transfer type
dmi_hsize	Output	DMI transfer size
dmi_hburst	Output	DMI burst type
dmi_hprot	Output	DMI protection control
dmi_hwrite	Output	DMI transfer direction
dmi_hwdata	Output	DMI write data bus
dmi_hrdata	Input	DMI read data bus
dmi_hresp	Input	DMI transfer response
dmi_hready	Output	DMI transfer done of NCEJDTM200
dmi_hreadyout	Input	DMI transfer done

23.5.2 BYPASS

When TAP instruction is BYPASS, a single-bit register is connected to tdi and tdo. In Capture-DR state, the register is loaded by 0. In Shift-DR state, data is transfered from tdi to tdo through the single-bit register.

23.5.3 IDCODE

This register contains device identification code: 0x1000563D.

31 28	27 12	11 1	0
Version	PartNumber	ManufId	1

Field Name	Bits	Description	Type	Reset
ManufId	[11:1]	Identifies the designer/manufacturer of this part.	RO	0x31E
PartNumber	[27:12]	Identifies the designer's part number of this part.	RO	0x0005
Version	[31:28]	Identifies the release version of this part.	RO	0x1

23.5.4 DTM Control and Status (dtmcs)

31	18	17	16	15	14	12	11	10	9	4	3	0
0		dmihardreset	dmireset	0	idle		dmistat			abits		Version

Field Name	Bits	Description	Type	Reset
Version	[3:0]	Version of the implemented DTM. 0x1 indicates that the current implementation conforms to *RISC-V External Debug Support (TD003) V0.13.*	RO	0x1
abits	[9:4]	Bit width of DMI address is 7	RO	0x7
dmistat	[11:10]	State of DMI	RO	0x0

Value	Meaning
0	No error
1	Reserved
2	An operation failed (resulted in op of 2)
3	An operation was attempted while a DMI access was still in progress (resulted in op of 3)

Field Name	Bits	Description	Type	Reset
idle	[14:12]	This is a hint to the debugger of the minimum number of cycles a debugger should spend in RunTest/Idle after every DMI scan to avoid a *busy* return code (dmistat of 3).	RO	0x7
dmireset	[16]	Writing 1 to this bit clears the sticky error state and allows the DTM to retry or complete the previous transaction.	W1	0
dmihardreset	[17]	Writing 1 to this bit does a hard reset of the DTM, causing the DTM to forget about any outstanding DMI transactions. In general, this should only be used when the Debugger has reason to expect that the outstanding DMI transaction will never complete (e.g., a reset condition causes an inflight DMI transaction to be canceled).	W1	0

23.5.5 Debug Module Interface Access (dmi)

This register allows accesses to the Debug Module Interface (DMI).

40	34	33		2	1	0
address			data			op

Field Name	Bits	Description		Type	Reset
op	[1:0]	Write operation:		RW	2

Value	Meaning
0	Ignore data and address. (nop)
1	Read from address. (read)
2	Write data to address. (write)
3	Reserved

Read operation:

Value	Meaning
0	The previous operation completed successfully.
1	Reserved
2	A previous operation failed.
3	An operation was attempted while a DMI request is still in progress. The data scanned into dmi in this access will be ignored.

Field Name	Bits	Description	Type	Reset
data	[33:2]	The data to send to the DM over the DMI during Update-DR, and the data returned from the DM as a result of the previous operation.	RW	0
addr	[40:34]	Address used for DMI access. In Update-DR this value is used to access the DM over the DMI.	RW	0

24 ATCWDT200

24.1 Introduction

The ATCWDT200 watchdog timer prevents a system from lock-up when the program execution goes astray.

24.1.1 Features

- Supports AMBA 2.0 APB bus
- Provides combinations of interrupt and reset when the watchdog timer expires
- Provides a write protection mechanism for the Control/Restart Registers
- Programmable source of timer clock
- Configurable magic numbers for write protection of registers and restart of the timer
- Watchdog timer could be externally paused

24.1.2 Block Diagram

Figure 24-1 ATCWDT200 Block Diagram shows the block diagram of the ATCWDT200 watchdog timer.

Figure 24-1 ATCWDT200 Block Diagram
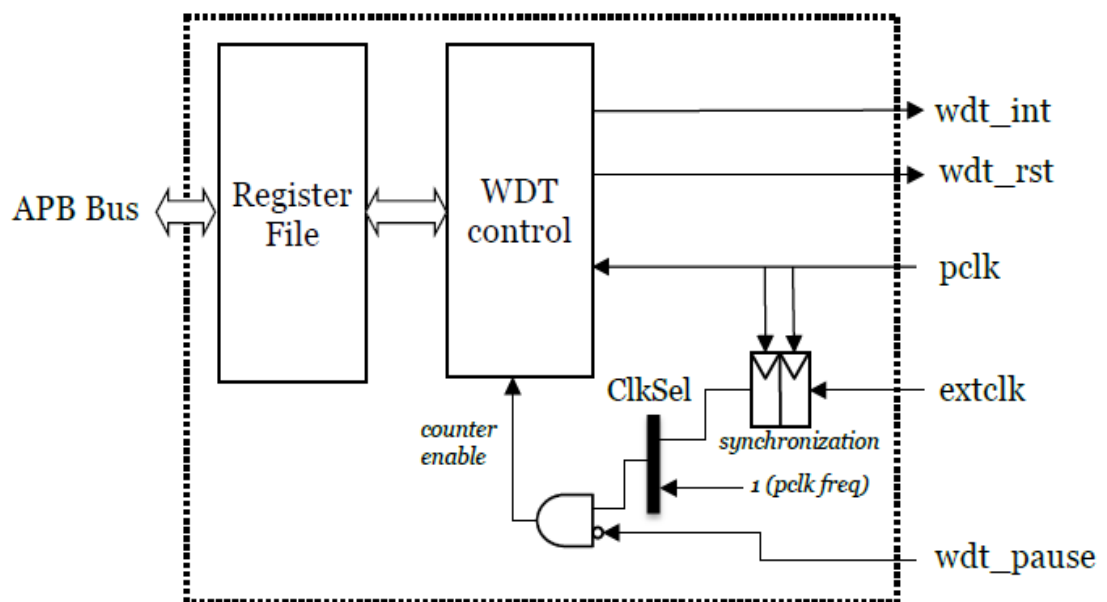
24.1.3 Function Description

The ATCWDT200 watchdog timer provides a two-stage mechanism to prevent a system from lock-up. The first stage is called "interrupt stage". If the watchdog interrupt is enabled and the watchdog timer is not restarted during the interrupt stage, the interrupt signal, wdt_int, will be asserted. The second stage, reset stage, begins right after the interrupt stage. If the watchdog reset is enabled and the watchdog timer is not restarted during the reset stage, the reset signal, wdt_rst, will be asserted.

Figure 24-2 Stages of the Watchdog Timer
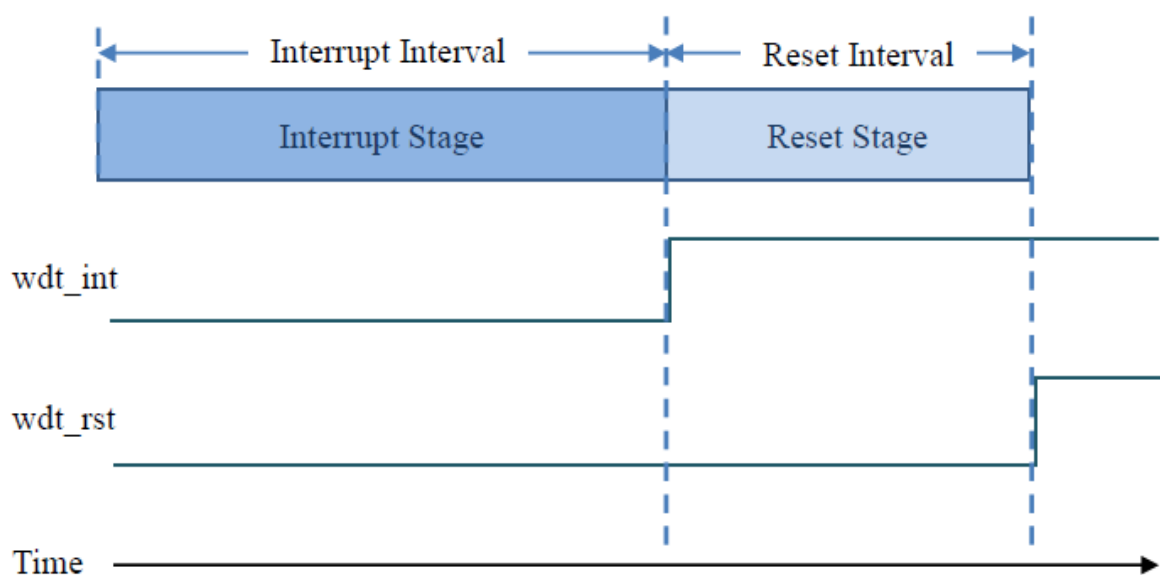
24.2 Signal Description

The input/output (I/O) signals of the ATCWDT200 watchdog timer are shown in Table 24-1 Signal Description of ATCWDT200.

Table 24-1 Signal Description of ATCWDT200

Name	I/O Type	Description
System signals		
extclk	I	External clock for the timer clock
		The watchdog timer is a single clock domain design clocked by PCLK and the EXTCLK signal does not drive any sequential logics. Instead, the signal is used as a data signal whose rising edge increases the counter of the watchdog timer. As a result, the frequency of EXTCLK must be at least 3 times slower than that of PCLK.
wdt_rst	O	Watchdog system reset
		Active high
wdt_int	O	Watchdog interrupt
		Active high
wdt_pause	I	Watchdog pause
		This signal causes the watchdog timer to pause and delays the triggering of the watchdog interrupt.
AMBA APB signals		
pclk	I	AMBA APB clock
presetn	I	AMBA APB reset signal; active low
psel	I	APB Slave select signal from the APB decoder
penable	I	AMBA APB enable signal
pwrite	I	AMBA APB transfer direction signal
		This signal indicates a write access when driven as HIGH and a read access when driven as LOW
paddr[4:2]	I	AMBA APB address bus
prdata[31:0]	O	AMBA APB read data bus
pwdata[31:0]	I	AMBA APB write data bus

24.3 Programming Model

The following sections describe ATCWDT200 registers in detail.

24.3.1 Summary of Registers

Table 24-2 Summary of registers shows the summary of ATCWDT200 programming registers.

Table 24-2 Summary of registers

Offset	Name	Description
+0x00	IdRev	ID and Revision Register
+0x04~ 0x0C	-	Reserved
+0x10	Ctrl	Control Register
+0x14	Restart	Restart Register
+0x18	WrEn	Write Enable Register
+0x1C	St	Status Register.

24.3.2 Register Description

The following sections describe ATCWDT200 registers in detail. The abbreviations for the Type column are summarized below.

RO: read only

WO: write only

R/W1C: readable and write 1 to clear

WP: write protected

R/WP: readable and write protected

DC: don't care

24.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 24-3 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:12	RO	ID number for ATCWDT200	0x03002
RevMajor	11:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

24.3.4 Control Register (0x10)

The Control Register is write-protected to protect against accidental overwrites. The Write Enable Register (Section 0) must be written before writing this register.

Table 24-4 Control Register

Name	Bit	Type	Description	Reset
–	31:11	–	Reserved	–
RstTime	10:8	R/WP	The time interval of the reset stage: 0: Clock period × 2^7 1: Clock period × 2^8 2: Clock period × 2^9 3: Clock period × 2^{10} 4: Clock period × 2^{11} 5: Clock period × 2^{12} 6: Clock period × 2^{13} 7: Clock period × 2^{14}	0x0
IntTime	7:4	R/WP	The timer interval of the interrupt stage: 0: Clock period × 2^6 1: Clock period × 2^8 2: Clock period × 2^{10} 3: Clock period × 2^{11} 4: Clock period × 2^{12} 5: Clock period × 2^{13} 6: Clock period × 2^{14} 7: Clock period × 2^{15} The following options are only valid when `ATCWDT200_32BIT_TIMER is defined: 8: Clock period × 2^{17} 9: Clock period × 2^{19}	0x0

Name	Bit	Type	Description	Reset
			10: Clock period $\times 2^{21}$	
			11: Clock period $\times 2^{23}$	
			12: Clock period $\times 2^{25}$	
			13: Clock period $\times 2^{27}$	
			14: Clock period $\times 2^{29}$	
			15: Clock period $\times 2^{31}$	
RstEn	3	R/WP	Enable or disable the watchdog reset 0: Disable 1: Enable	0x0
IntEn	2	R/WP	Enable or disable the watchdog interrupt 0: Disable 1: Enable	0x0
ClkSel	1	R/WP	Clock source of timer: 0: EXTCLK 1: PCLK	0x0
En	0	R/WP	Enable or disable the watchdog timer 0: Disable 1: Enable	0x0

24.3.5 Restart Register (0x14)

The Restart Register is write-protected to protect against accidental overwrites. The Write

Enable Register (Section 0) must be written before writing this register.

When the Restart Register is written with the preconfigured value

ATCWDT200_RESTART_NUM, it restarts the interrupt timer and cancels the system reset timer. When the value is not ATCWDT200_RESTART_NUM, the controller just ignores the write.

Table 24-5 Restart Register

Name	Bit	Type	Description	Reset
-	31:16	-	Reserved	-
Restart	15:0	WP	Write the magic number ATCWDT200_RESTART_NUM to restart the watchdog timer.	DC

24.3.6 Write Enable Register (0x18)

The Control Register and Restart Register should be programmed through a two-step write scheme. The Write Enable Register should be programmed with a magic number (ATCWDT200_WP_NUM) to disable the write protection before either of the two registers could be updated.

The register write-protection is enabled again upon the subsequent write to any of the watchdog registers is received.

Table 24-6 Write Enable Register

Name	Bit	Type	Description	Reset
-	31:16	-	Reserved	-
WEn	15:0	WO	Write the magic number ATCWDT200_WP_NUM to disable the write protection of the Control Register and the Restart Register.	DC

24.3.7 Status Register (0x1C)

Table 24-7 Status Register (0x1C)

Name	Bit	Type	Description	Reset
-	31:1	-	Reserved	-
IntExpired	0	R/W1C	The status of the watchdog interrupt timer 0: timer is not expired yet 1: timer is expired	0x0

24.4 Hardware Configuration Options

24.4.1 Magic Number for Write Protection

ATCWDT200_WP_NUM should be a 16-bit number. The value is the magic number for disabling the write protection. The following example defines the magic number to 0x5aa5:

define ATCWDT200_WP_NUM 16'h5aa5

24.4.2 Magic Number for Restarting the Watchdog Timer

ATCWDT200_RESTART_NUM should be a 16-bit number. The value is the magic number for restarting the watchdog timer. The following example defines the magic number to 0xcafe:

`define ATCWDT200_RESTART_NUM 16'hcafe

24.4.3 Duration of Watchdog Interrupt Timer Interval

The default watchdog interrupt timer is a 16-bit counter. The maximum watchdog timer duration would therefore be 215 cycles of either PCLK or EXTCLK periods. Given a 1MHz clock, that will be 32.768ms. It is expected that the use of a slower EXTCLK (e.g., from a 32768Hz Oscillator) may achieve longer intervals (one second). If the available frequency of PCLK/EXTCLK is too high or the desired interval duration needs to be much longer, 32-bit counter may be needed by defining the ATCWDT200_32BIT_TIMER macro:

`define ATCWDT200_32BIT_TIMER

24.5 Programming Sequence

24.5.1 Setup and Enable the Watchdog Timer:

13. Write the magic number ATCWDT200_WP_NUM to the Write Enable Register.

14. Write the Control Register: Select the clock source of timer, set the intervals of interrupt and reset, and enable the watchdog interrupt (IntEn=1), the watchdog reset (RstEn=1), and the watchdog time (En=1).

24.5.2 Restart the Watchdog Timer

Once the watchdog timer is enabled, it must be restarted by software to avoid the watchdog interrupt/reset. The restart procedure is as follows.

15. Write the magic number ATCWDT200_WP_NUM to the Write Enable Register.

16. Write "ATCWDT200_RESTART_NUM" to the Restart Register.

24.5.3 Disable the Watchdog Timer

17. Write the magic number ATCWDT200_WP_NUM to the Write Enable Register.

18. Write the Control Register: Disable the watchdog timer (En=0).

25 ATCRTC100

25.1 Introduction

25.1.1 Description

The ATCRTC100 is a low-power real-time clock (RTC) which keeps track of current time information and provides periodic and alarm interrupts.

25.1.2 Features

- AMBA 2.0 APB interface for register accesses
- Configurable counter size
- Periodic interrupts: half-second, second, minute, hour and day
- Programmable alarm interrupt
- Hardware digital trimming to compensate for inaccuracies of the external clock source

25.1.3 Block Diagram

Figure 25-1 ATCRTC100 Block Diagram shows the block diagram of ATCRTC100:

Figure 25-1 ATCRTC100 Block Diagram
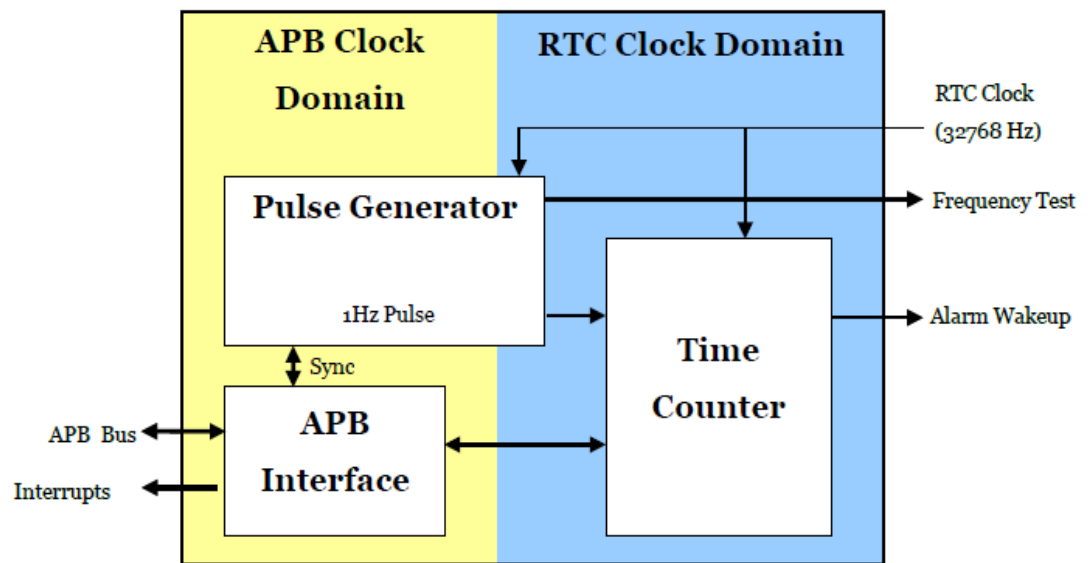
25.2 Signal Description

Table 25-1 Signal Description of ATCRTC100 describes the input/output (I/O) signals of ATCRTC100.

Table 25-1 Signal Description of ATCRTC100

Name	I/O Type	Description
APB Bus Interface		
pclk	I	AMBA APB clock
presetn	I	AMBA APB reset signal (Active-Low)
psel	I	AMBA APB slave select signal from the APB decoder
penable	I	AMBA APB enable signal
pwrite	I	AMBA APB transfer direction signal. This signal indicates a write access when driven as HIGH and a read access when driven as LOW.
paddr[4:2]	I	AMBA APB address bus
prdata[31:0]	O	AMBA APB read data bus
pwdata[31:0]	I	AMBA APB write data bus
RTC Signals		
rtc_clk	I	RTC clock source, 32.768 kHz
rtc_rstn	I	RTC reset (Active-Low)
rtc_int_hsec	O	Half-second interrupt signal
rtc_int_sec	O	Second interrupt signal
rtc_int_min	O	Minute interrupt signal
rtc_int_hour	O	Hour interrupt signal
rtc_int_day	O	Day interrupt signal
rtc_int_alarm	O	Alarm interrupt signal
alarm_wakeup	O	Alarm wakeup signal
Frequency Test Signals		
freq_test_out	O	512Hz frequency test output signal
freq_test_en	O	Frequency test output enable signal, to be connected to the output enable port of the output pad for the frequency test pin.

25.3 Programming Model

25.3.1 Summary of Registers

Table 25-2 Summary of Registers

Offset	Name	Description
+0x00	IdRev	ID and Revision Register
+0x04~ 0x0C	-	Reserved
+0x10	Cntr	Counter Register
+0x14	Alarm	Alarm Register
+0x18	Ctrl	Control Register
+0x1C	St	Status Register
+0x20	Trim	Digital Trimming Register

25.3.2 Register Description

The following sections describe ATCRTC100 registers in detail. The abbreviations for the Type column are summarized below.

RO: Read only

WO: Write only

R/W: Readable and writable

W1C: Write 1 to clear

25.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 25-3 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for ATCRTC100	0x030110
Major	7:4	RO	Major revision number	Revision dependent
Minor	3:0	RO	Minor revision number	Revision dependent

25.3.4 Counter Register (0x10)

The Counter Register records the current time.

Table 25-4 Counter Register

Name	Bit	Type	Description	Reset
Day	(N+17):17	R/W	Day passed after RTC enabled; N = (ATCRTC100_DAY_BITS − 1), see Section 4.1 for more information	0x0
Hour	16:12	R/W	Hour field of current time; range: 0 ~ 23	0x0
Min	11:6	R/W	Minute field of current time; range: 0 ~ 59	0x0
Sec	5:0	R/W	Second field of current time; range: 0 ~ 59	0x0

Note!

If the counters are set to out-of-range values, their behaviors are undefined.

25.3.5 Alarm Register (0x14)

The alarm interrupt and/or the alarm wakeup signal are triggered when the value of Counter Register matches the value of Alarm Register.

Control Register fields "Alarm_Int" and "Alarm_Wakeup" control the alarm interrupt and alarm wakeup signal.

Table 25-5 Alarm Register

Name	Bit	Type	Description	Reset
Hour	16:12	R/W	Hour field of alarm setting; range: 0 ~ 23	0x0
Min	11:6	R/W	Minute field of alarm setting; range: 0 ~ 59	0x0
Sec	5:0	R/W	Second field of alarm setting; range: 0 ~ 59	0x0

25.3.6 Control Register (0x18)

The Control Register controls the RTC enable and interrupts.

Table 25-6 Control Register

Name	Bit	Type	Description	Reset
Freq_Test_En	8	R/W	Enable the 512 Hz frequency test output	0x0
Hsec[1]	7	R/W	Enable half-second interrupt; half-second interrupt is generated when half a second passed	0x0
Sec	6	R/W	Enable "second" interrupt; "second" interrupt is generated when one second passed	0x0
Min	5	R/W	Enable minute interrupt; minute interrupt is generated when the seconds of RTC time changes from 59 to 0	0x0
Hour	4	R/W	Enable hour interrupt; hour interrupt is generated when the minutes of RTC time changes from 59 to 0	0x0
Day	3	R/W	Enable day interrupt; day interrupt is generated when the hours of RTC time changes from 23 to 0	0x0
Alarm_Int	2	R/W	Enable alarm interrupt	0x0
Alarm_Wakeup	1	R/W	Enable alarm wakeup signal	0x0
RTC_En	0	R/W	Enable RTC	0x0

Note!

1 Exists only if ATCRTC100_HALF_SECOND_SUPPORT is defined.

25.3.7 Status Register (0x1C)

The Status Register records the interrupt status and synchronization status of RTC register updates.

Table 25-7 Interrupt Status Register

Name	Bit	Type	Description	Reset
WriteDone	16	RO	This bit indicates the synchronization progress of RTC register updates. This bit becomes zero when any of RTC control registers (the Counter, Alarm and Control registers) are updated. It returns to one when all prior updates to these three registers have been successfully synchronized to the RTC clock domain. While an RTC register update is being synchronized to the RTC clock domain, a second update to the same register may be dropped. Each of the RTC registers is synchronized independently while their synchronization status are lumped into this single bit. Thus writes to different RTC registers can be done in a batch before checking this bit. Since the frequency of the RTC clock is quite slow when compared to the typical frequency of the APB clock, the synchronization period can be pretty long. The APB clock domain should not be shut down while the synchronization is still in progress.	1
Reserved	15:8	-	-	-
Hsec¹	7	W1C	Half-second interrupt status, write 1 to clear	0x0
Sec	6	W1C	Second interrupt status	0x0
Min	5	W1C	Minute interrupt status	0x0
Hour	4	W1C	Hour interrupt status	0x0
Day	3	W1C	Day interrupt status	0x0
Alarm_Int	2	W1C	Alarm interrupt status	0x0

25.3.8 Digital Trimming Register (0x20)

The Digital Trimming Register holds the trimming values and the respective directions for each RTC time period. This register should be programmed with RTC disabled. Please see Hardware Digital Trimming for usage of this register.

Table 25-8 Digital Trimming Register

Name	Bit	Type	Description	Reset
Day_Sign	31	R/W	Sign bit for the trimming value for the second on the day boundary: 1: Slow down the timer. 0: Speed up the timer.	0x0
Reserved	30:29	-	-	-
Day_Trim	28:24	R/W	Digital trimming value for the second on the day boundary.	0x0
Hour_Sign	23	R/W	Sign bit for the trimming value for the second on the hour boundary: 1: Slow down the timer. 0: Speed up the timer.	0x0
Reserved	22:21	-	-	-
Hour_Trim	20:16	R/W	Digital trimming value for the second on the hour boundary.	0x0
Min_Sign	15	R/W	Sign bit for the trimming value for the second on the minute boundary: 1: Slow down the timer. 0: Speed up the timer.	0x0
Reserved	14:13	-	-	-
Min_Trim	12:8	R/W	Digital trimming value for the second on the minute boundary.	0x0
Sec_Sign	7	R/W	Sign bit for the trimming value for the rest of seconds: 1: Slow down the timer. 0: Speed up the timer.	0x0
Reserved	6:5	-	-	-
Sec_Trim	4:0	R/W	Digital trimming value for the rest of seconds	0x0

25.4 Hardware Configuration Options

25.4.1 Day Counter Bits

Define ATCRTC100_DAY_BITS to configure the size of the day counter. The available range is 1–15.

For example, define ATCRTC100_DAY_BITS to 5 for counting up to 2^5 = 32 days.

`define ATCRTC100_DAY_BITS 5

25.4.2 Half-second Interrupt Support

Define ATCRTC100_HALF_SECOND_SUPPORT to support the half-second interrupt.

`define ATCRTC100_HALF_SECOND_SUPPORT

25.5 Programming Sequence

25.5.1 Adjust Time and Interrupts on the Hour

19. Wait until the WriteDone field of the Status Register equals 1.

20. Set the Counter Register to the current time; set the Day field to 0 to count from day 0.

21. Enable RTC and hourly interrupt: set the Control Register to 0x11.

25.5.2 Trigger an Alarm Interrupt at a Specific Time

22. Wait until the WriteDone field of the Status Register equals 1.

23. Set the Alarm Register to the time when an alarm will be issued.

24. Enable RTC and alarm interrupt: set the Control Register to 0x5.

25.5.3 Trigger an Alarm Wakeup Signal at a Specific Time

The behaviors of the alarm wakeup signal and the alarm interrupt are mostly the same except the following differences:

- Alarm interrupt is kept HIGH until writing 1 to the Alarm_Int field of the Status Register; alarm wakeup signal is a 1/32768 second-wide pulse.
- Alarm interrupt is an APB clock domain signal while the alarm wakeup signal is an RTC domain signal. The alarm interrupt signal will be disabled when the APB clock domain is off but the alarm wakeup signal will not be affected.

The following shows an example to trigger an alarm wakeup signal at a specific time:

25. Wait until the WriteDone field of the Status Register equals 1.

26. Set the Alarm Register to the time you want to wake up the system.

27. Enable RTC and alarm wakeup: set the Control Register to 0x3.

25.5.4 Hardware Digital Trimming

ATCRTC100 is primarily driven by the external 32768 Hz clock source, and its pulse generator generates an internal 1Hz pulse for incrementing its time counters. The 1Hz pulse is generated by counting the number of clock cycles of the external clock source. By default, the pulse generator outputs a one-cycle-wide pulse every 32768 clock cycles. Therefore the period of the 1Hz pulse will be one second and the second counter is incremented every second if the external clock source is exactly 32768Hz. However, the resulting period won't be exactly one second if the external clock cycle is inaccurate. The provided digital trimming function compensates these inaccuracies by increasing or decreasing the period to generate the 1 Hz pulses.

An adjustment value could be specified and adjustments are performed every second. However, the alternate values of adjustments could be specified depending on whether it is the second on the minute, hour or day boundary. Each adjustment value consists of a trimming value and a trimming sign bit in the Digital Trimming Register (0x20).

- Digital trimming value
- – The number of cycles to adjust for generating the 1Hz pulse.
- – Trimming is enabled if this value is not zero.
- Sign bit for the trimming value
- – Indicates the direction of digital trimming.
- – 1: Slow down the clock by enlarging the period of the 1Hz pulse by the trimming value.
- – 0: Speed up the clock by shortening the period of the 1Hz pulse by the trimming value.

The second trimming value (Sec_Trim) is applied every second if no other trimming values are specified. If any of the minute/hour/day trimming values (Min_Trim, Hour_Trim, and Day_Trim) is set, the specified trimming value is applied on the minute/day/hour boundary instead. The longer period trimming values will take priority on the second of multiple time period boundaries. That is, if both Sec_Trim and Min_Trim are set, the 1Hz pulse periods for the first 59 seconds are adjusted using Sec_Trim, while the Min_Trim is applied to the last second of one minute. If Hour_Trim is also set, it will be applied to the last second of the last minute of an hour.

The multitude of trimming values that ATCRTC100 supports allow better calibration of the clock source. The second trimming value may over/under-trim the clock source while the longer period trimming values could compensate it back by adjusting for the accumulated errors in the respective periods.

There are two methods to determine the trimming value. The first one is to measure the frequency of a 512 Hz output signal directly and calculate

the deviation from the ideal frequency. The second one is to observe the amount of time gained or lost over a calibrated wall clock time period.

25.5.5 Determining Trimming Value by Measuring Frequency

The following procedure shows how to decide the trimming value and sign bit by measuring the 512 Hz test frequency output:

1. Wait until the WriteDone field of the Status Register equals 1.

2. Set the Digital Trimming Register with an all-zero value (default).

3. Program the pin mux of the chip such that the RTC frequency test output is driven to the test pin.

4. Enable RTC and the frequency test output: set the Control Register to 0x81.

5. Measure the output frequency from the frequency test output pin with a calibration tool.

6. Calculate the trimming value from clock frequency deviation per second (see the following equation).

● Calculate the trimming value from the frequency deviation:
   trimming value = $32768 \times (F_{\mathrm{measured}} - F_{\mathrm{ideal}})/F_{\mathrm{ideal}}$

   Where:

   – Fmeasured is the measured frequency of the freq_test_out pin.
   – Fideal = 512 Hz.
● If the above "trimming value" is positive, the measured frequency is faster than the ideal frequency. The trimming sign bit needs to be set.
● If the above "trimming value" is negative, the measured frequency is slower than the ideal frequency. The trimming sign bit needs to be cleared.

7. Disable RTC: set the Control Register to 0x0.

8. Wait until the WriteDone field of the Status Register equals 1.

9. Set the Sec_Trim and Sec_Sign fields of the Digital Trimming Register with the computed trimming value and sign bit.

10. Enable RTC: set the Control Register to 0x1.

25.5.6 Determining Trimming Value by Observing Time Deviation

The following procedure shows how to decide the trimming value and the sign bit by observing the deviation over a period of time.

1. Wait until the WriteDone field of the Status Register equals 1.

2. Set the Digital Trimming Register with an all-zero value (default).

3. Enable RTC: set the Control Register to 0x1.

4. Compare the elapsed RTC timer against that of a calibrated wall clock.

5. Calculate the trimming value from time period deviation (see the following equation).

● Calculate the trimming value from time period deviation observation:
$$SEC_{dev} = SEC_{measured} - SEC_{ideal}$$
$$PPM_{dev} = (SEC_{dev}/SEC_{ideal}) \times 10^6$$
Where:

– SEC_{ideal} is the observed ideal time period in seconds.

– $SEC_{measured}$ is the elapsed RTC seconds according to ATCRTC100 during SEC_{ideal}.

– SEC_{dev} is the number of seconds gained or lost during SEC_{ideal}.

– If the PPM_{dev} is positive, ATCRTC100 is counting faster than the ideal clock. The trimming sign bit needs to be set.

– If the PPM_{dev} is negative, ATCRTC100 is counting slower than the ideal clock. The trimming sign bit needs to be cleared.

– The trimming value is determined as:
trimming value = $PPM_{dev} \times 32768/10^6$

6. Disable RTC: set the Control Register to 0x0.

7. Wait until the WriteDone field of the Status Register equals 1.

8. Set the Sec_Trim and Sec_Sign fields of the Digital Trimming Register with the computed trimming value and sign bit.

9. Enable RTC: set the Control Register to 0x1.

10. Please note that a longer observation period will improve the accuracy of measurement

25.5.7 Trimming Capabilities

The following table shows the trimming capability of all trimming values supported by ATCRTC100. The capabilities represent the amount of deviations of the external clock source that could be corrected by the respective trimming value alone, i.e., assuming the rest of trimming values are set to zero.

Table 25-9 Trimming Capabilities

Trimming Value	Capability (PPM)
Second	+/- 30.5 ~ 946
Minute	+/- 0.508 ~ 15.77
Hour	+/- 0.008 ~ 0.262
Day	+/- 0.000353 ~ 0.0109

The capability is computed by dividing the minimum and the maximum trimming value over the number of ideal clock counts within the respective time range. The range of trimming values is 1to 31, so the capability of the second trimming value is:

$$\text{Capability}_{MAX}(\text{Sec\_Trim}) = Sec\_Trim_{MAX} / 32768 = 31/32768 = 946 \times 10^{-6}$$

$$\text{Capability}_{MIN}(\text{Sec\_Trim}) = Sec\_Trim_{MIN}/ 32768 = 1/32768 = 30.56 \times 10^{-6}$$

And the capability of the minute trimming value is:

$$\text{CapabilityMAX}(\text{Min\_Trim}) = Min\_Trim_{MAX} / (32768 \times 60) = 31/ (32768 \times 60) = 15.77 \times 10^{-6}$$

$$\text{CapabilityMIN}(\text{Min\_Trim}) = Min\_Trim_{MIN}/ (32768 \times 60) = 1/ (32768 \times 60) = 0.508 \times 10^{-6}$$

Like-wise, the trimming capability for the hour and day trimming values are computed similarly.

26 ATCPIT100

26.1 Introduction

The ATCPIT100 Programmable Interval Timer (PIT) is a set of compact multi-function timers, which can be used as pulse width modulators (PWM) as well as simple timers.

26.1.1 Features

- Supports AMBA 2.0 APB bus
- Supports up to 4 multi-function timers
- Each multi-function timer provides 6 usage scenarios (combinations of timer and PWM)
- Programmable source of timer clock
- Timers can be externally paused

26.1.2 Block Diagram

Figure 26-1 ATCPIT100 Block Diagramshows the block diagram of ATCPIT100.

Figure 26-1 ATCPIT100 Block Diagram
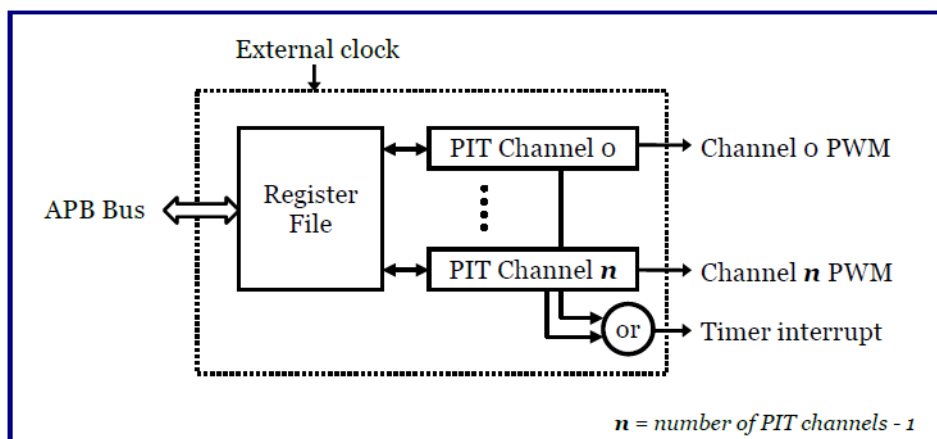
26.1.3 Function Description

ATCPIT100 supports up to 4 PIT channels. Each PIT channel is a multi-function timer and provides the following 6 usage scenarios:

- one 32-bit timer
- two 16-bit timers
- four 8-bit timers
- one 16-bit PWM
- one 16-bit timer and one 8-bit PWM
- two 8-bit timers and one 8-bit PWM

An n-bit timer means the timer contains an n-bit counter to generate periodic interrupts. An n-bit PWM means the PWM contains two n-bit counters to generate periodic square waves with programmable duty cycles. The more bits a counter contains, the larger period it can support.

Table 26-1 Effective Devices of Channel Modes shows the effective devices of the corresponding channel modes.

Table 26-1 Effective Devices of Channel Modes

Channel Mode	32-bit Timer	16-bit Timers	8-bit Timers	PWM	Mixed PWM/16-bit Timer	Mixed PWM/8-bit Timers
	32-bit Timer 0	16-bit Timer 0	8-bit Timer 0		16-bit Timer 0	8-bit Timer 0
		16-bit Timer 1	8-bit Timer 1			8-bit Timer 1
			8-bit Timer 2			
			8-bit Timer 3			
				16-bit PWM	8-bit PWM	8-bit PWM

26.2 Signal Description

ATCPIT100 input/output (I/O) signals are shown in Table 26-2 Signal Description of ATCPIT100.

Table 26-2 Signal Description of ATCPIT100

Name	I/O Type	Description
AMBA APB signals		
pclk	I	AMBA APB clock
presetn	I	AMBA APB reset signal; active low
psel	I	AMBA APB slave select signal from the APB decoder
penable	I	AMBA APB enable signal
pwrite	I	AMBA APB transfer direction signal This signal indicates a write access when driven as HIGH and a read access when driven as LOW
paddr[6:2]	I	AMBA APB address bus
prdata[31:0]	O	AMBA APB read data bus
pwdata[31:0]	I	AMBA APB write data bus
PIT Signals		
extclk	I	External counting reference signal extclk frequency must be 3 times slower than pclk
pit_intr	O	Timer interrupt
pit_pause	I	Timer pause This signal pauses timers and suspends all timer/PWM activities.
ch*n*_pwm	O	Channel *n* PWM output
ch*n*_pwmoe	O	Channel *n* PWM output enable

26.3 Programming Model

26.3.1 Summary of Registers

Table 26-3 Summary of registers shows the summary of ATCPIT100 programming registers.

Table 26-3 Summary of registers

Offset	Name	Description
+0x00	IdRev	ID and Revision Register
+0x04~ 0x0C	-	Reserved
+0x10	Cfg	Configuration Register
+0x14	IntEn	Interrupt Enable Register
+0x18	IntSt	Interrupt Status Register
+0x1C	ChEn	Channel Enable Register
+0x20	Ch0Ctrl	Channel 0 Control Register
+0x24	Ch0Reload	Channel 0 Reload Register
+0x28	Ch0Cntr	Channel 0 Counter Register
+0x2C	-	Reserved
+0x30	Ch1Ctrl	Channel 1 Control Register
+0x34	Ch1Reload	Channel 1 Reload Register
+0x38	Ch1Cntr	Channel 1 Counter Register
+0x3C	-	Reserved
+0x40	Ch2Ctrl	Channel 2 Control Register
+0x44	Ch2Reload	Channel 2 Reload Register
+0x48	Ch2Cntr	Channel 2 Counter Register
+0x4C	-	Reserved
+0x50	Ch3Ctrl	Channel 3 Control Register
+0x54	Ch3Reload	Channel 3 Reload Register
+0x58	Ch3Cntr	Channel 3 Counter Register

26.3.2 Register Description

The following sections describe ATCPIT100 registers in detail. The abbreviations for the Type column are summarized below.

RO: read only

R/W: readable and writable

W1C: write 1 to clear

26.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 26-4 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:12	RO	ID number for PIT	0x03031
RevMajor	11:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

26.3.4 Configuration Register (0x10)

Table 26-5 Configuration Register

Name	Bit	Type	Description	Reset
-	31:3	-	Reserved	0x0
NumCh	2:0	RO	Number of PIT channels	Configuration dependent

26.3.5 Interrupt Enable Register (0x14)

Table 26-6 Interrupt Enable Register

Name	Bit	Type	Description	Reset
-	31:16	-	Reserved	-
Ch3Int3En	15	R/W	Channel 3 Timer 3 interrupt enable	0x0
CH3Int2En	14	R/W	Channel 3 Timer 2 interrupt enable	0x0
Ch3Int1En	13	R/W	Channel 3 Timer 1 interrupt enable	0x0
CH3Int0En	12	R/W	Channel 3 Timer 0 interrupt enable	0x0
Ch2Int3En	11	R/W	Channel 2 Timer 3 interrupt enable	0x0
Ch2Int2En	10	R/W	Channel 2 Timer 2 interrupt enable	0x0
Ch2Int1En	9	R/W	Channel 2 Timer 1 interrupt enable	0x0
Ch2Int0En	8	R/W	Channel 2 Timer 0 interrupt enable	0x0
Ch1Int3En	7	R/W	Channel 1 Timer 3 interrupt enable	0x0
Ch1Int2En	6	R/W	Channel 1 Timer 2 interrupt enable	0x0
Ch1Int1En	5	R/W	Channel 1 Timer 1 interrupt enable	0x0
Ch1Int0En	4	R/W	Channel 1 Timer 0 interrupt enable	0x0
Ch0Int3En	3	R/W	Channel 0 Timer 3 interrupt enable	0x0
Ch0Int2En	2	R/W	Channel 0 Timer 2 interrupt enable	0x0
Ch0Int1En	1	R/W	Channel 0 Timer 1 interrupt enable	0x0
Ch0Int0En	0	R/W	Channel 0 Timer 0 interrupt enable 0: Disable 1: Enable	0x0

26.3.6 Interrupt Status Register (0x18)

Table 26-7 Interrupt Status Register

Name	Bit	Type	Description	Reset
-	31:16	-	Reserved	-
Ch3Int3	15	W1C	Channel 3 Timer 3 interrupt status	0x0
Ch3Int2	14	W1C	Channel 3 Timer 2 interrupt status	0x0
Ch3Int1	13	W1C	Channel 3 Timer 1 interrupt status	0x0
Ch3Int0	12	W1C	Channel 3 Timer 0 interrupt status	0x0
Ch2Int3	11	W1C	Channel 2 Timer 3 interrupt status	0x0
Ch2Int2	10	W1C	Channel 2 Timer 2 interrupt status	0x0
Ch2Int1	9	W1C	Channel 2 Timer 1 interrupt status	0x0
Ch2Int0	8	W1C	Channel 2 Timer 0 interrupt status	0x0
Ch1Int3	7	W1C	Channel 1 Timer 3 interrupt status	0x0
Ch1Int2	6	W1C	Channel 1 Timer 2 interrupt status	0x0
Ch1Int1	5	W1C	Channel 1 Timer 1 interrupt status	0x0
Ch1Int0	4	W1C	Channel 1 Timer 0 interrupt status	0x0
Ch0Int3	3	W1C	Channel 0 Timer 3 interrupt status 0: No effect 1: Timer 3 time up	0x0
Ch0Int2	2	W1C	Channel 0 Timer 2 interrupt status 0: No effect 1: Timer 2 time up	0x0
Ch0Int1	1	W1C	Channel 0 Timer 1 interrupt status 0: No effect 1: Timer 1 time up	0x0
Ch0Int0	0	W1C	Channel 0 Timer 0 interrupt status 0: No effect 1: Timer 0 time up	0x0

26.3.7 Channel Enable Register (0x1C)

Table 26-8 Channel Enable Register

Name	Bit	Type	Description	Reset
-	31:16	-	Reserved	-
Ch3TMR3En/ CH3PWMEn	15	R/W	ChMode = 1, 2, 3 Channel 3 Timer 3 enable ChMode = 4, 6, 7 Channel 3 PWM enable	0x0
Ch3TMR2En	14	R/W	Channel 3 Timer 2 enable	0x0
Ch3TMR1En	13	R/W	Channel 3 Timer 1 enable	0x0
Ch3TMR0En	12	R/W	Channel 3 Timer 0 enable	0x0
Ch2TMR3En/ CH2PWMEn	11	R/W	ChMode = 1, 2, 3 Channel 2 Timer 3 enable ChMode = 4, 6, 7 Channel 2 PWM enable	0x0
Ch2TMR2En	10	R/W	Channel 2 Timer 2 enable	0x0
Ch2TMR1En	9	R/W	Channel 2 Timer 1 enable	0x0
Ch2TMR0En	8	R/W	Channel 2 Timer 0 enable	0x0
Ch1TMR3En/ CH1PWMEn	7	R/W	ChMode = 1, 2, 3 Channel 1 Timer 3 enable ChMode = 4, 6, 7 Channel 1 PWM enable	0x0
Ch1TMR2En	6	R/W	Channel 1 Timer 2 enable	0x0
Ch1TMR1En	5	R/W	Channel 1 Timer 1 enable	0x0
Ch1TMR0En	4	R/W	Channel 1 Timer 0 enable	0x0
Ch0TMR3En/ Ch0PWMEn	3	R/W	ChMode = 1, 2, 3 Channel 0 Timer 3 enable ChMode = 4, 6, 7 Channel 0 PWM enable	0x0

Name	Bit	Type	Description	Reset
Ch0TMR2En	2	R/W	Channel 0 Timer 2 enable	0x0
Ch0TMR1En	1	R/W	Channel 0 Timer 1 enable	0x0
Ch0TMR0En	0	R/W	Channel 0 Timer 0 enable 0: Disable 1: Enable	0x0

Note that a timer or PWM cannot be enabled if the corresponding channel does not exist or it is not a valid device in the channel mode. For example, Timer 1 of Channel 0 cannot be enabled when Channel 0 is set to the 32-bit Timer mode.

26.3.8 Chn Control Register (0x20 + n * 0x10)

Table 26-9 Channel 0~3 Control Register (0x20 + n * 0x10)

Name	Bit	Type	Description	Reset
-	31:5	-	Reserved	-
PWMPark	4	R/W	PWM park value. When this channel is disabled, this bit reflects the output of PWM and writing to it will change the output of PWM. The value of this bit also governs how the PWM waveform is generated when this channel is enabled. 0: the PWM output is LOW when the channel is disabled; the low-period PWM counter will be counted first before toggling the output to HIGH and counting the high-period PWM counter when this channel is enabled. 1: the PWM output is HIGH when the channel is disabled; the high-period PWM counter will be counted first before toggling	0x0

Name	Bit	Type	Description	Reset
			the output to LOW and counting the low-period PWM counter when this channel is enabled.	
ChClk	3	R/W	Channel clock source: 0: External clock 1: APB clock	0x0
ChMode	2:0	R/W	Channel mode: 0: Reserved 1: 32-bit timer 2: 16-bit timers 3: 8-bit timers 4: PWM 5: Reserved 6: Mixed PWM/16-bit timer 7: Mixed PWM/8-bit timers	0x0

26.3.9 Chn Reload Register (0x24 + n * 0x10)

The Reload Register keeps the initial/reload value(s) for PWM/timer counter(s). The definition of the Reload Register varies according to the channel mode. The period of PWM/timer is equal to the reload value plus one. For example, in the 32-bit Timer mode, a timer interrupt will be generated every (TMR32_0 + 1) cycles. In the PWM mode, the high period is (PWM16_Hi + 1) cycles and the low period is (PWM16_Lo + 1) cycles.Table 26-10 Reload Register for 32-bit Timer Mode (ChMode=1) to TTable 26-15 Reload Register for Mixed PWM/8-bit Timers Mode (ChMode=7) shows the definition of Reload Register with respect to the channel mode.

Table 26-10 Reload Register for 32-bit Timer Mode (ChMode=1)

Name	Bit	Type	Description	Reset
TMR32_0	31:0	R/W	Reload value for 32-bit Timer 0	0x0

Table 26-11 Reload Register for 16-bit Timers Mode (ChMode=2)

Name	Bit	Type	Description	Reset
TMR16_1	31:16	R/W	Reload value for 16-bit Timer 1	0x0
TMR16_0	15:0	R/W	Reload value for 16-bit Timer 0	0x0

Table 26-12 Reload Register for 8-bit Timers Mode (ChMode=3)

Name	Bit	Type	Description	Reset
TMR8_3	31:24	R/W	Reload value for 8-bit Timer 3	0x0
TMR8_2	23:16	R/W	Reload value for 8-bit Timer 2	0x0
TMR8_1	15:8	R/W	Reload value for 8-bit Timer 1	0x0
TMR8_0	7:0	R/W	Reload value for 8-bit Timer 0	0x0

Table 26-13 Reload Register for PWM Mode (ChMode=4)

Name	Bit	Type	Description	Reset
PWM16_Hi	31:16	R/W	Reload value for PWM high period	0x0
PWM16_Lo	15:0	R/W	Reload value for PWM low period	0x0

Table 26-14 Reload Register for Mixed PWM/16-bit Timer Mode (ChMode=6)

Name	Bit	Type	Description	Reset
PWM8_Hi	31:24	R/W	Reload value for PWM high period	0x0
PWM8_Lo	23:16	R/W	Reload value for PWM low period	0x0
TMR16_0	15:0	R/W	Reload value for 16-bit Timer 0	0x0

Table 26-15 Reload Register for Mixed PWM/8-bit Timers Mode (ChMode=7)

Name	Bit	Type	Description	Reset
PWM8_Hi	31:24	R/W	Reload value for PWM high period	0x0
PWM8_Lo	23:16	R/W	Reload value for PWM low period	0x0
TMR8_1	15:8	R/W	Reload value for 8-bit Timer 1	0x0
TMR8_0	7:0	R/W	Reload value for 8-bit Timer 0	0x0

26.3.10 Chn Counter Register (0x28 + n * 0x10)

Chn Counter Register indicates the remaining cycles for the next timer interrupt or PWM toggle. Like the Reload Register, the field definition of Counter Register varies according to the channel mode.

Name	Bit	Type	Description	Reset
Counter	31:0	RO	The counter for the counting of Timer/PWM Refer to Table 10 ~ Table 15 for the field definitions with respect to the channel mode.	0x0

26.4 Hardware Configuration Options

26.4.1 Number of PIT Channels

Define ATCPIT100_NUM_CHANNEL_mto configure mPIT channels, where mis the number ranging from 1~4. For example, define "ATCPIT100_NUM_CHANNEL_3" to configure 3 PIT channels.

define ATCPIT100_NUM_CHANNEL_3

26.5 Programming Sequence

Each PIT channel may select either APB or external clock as its clock source. ChClk of PIT

Control Register determines the clock source selection.

The following sections give some usage examples.

26.5.1 Generate 2 Periodic Interrupts with Channel 0

Using channel 0 to generate 2 interrupts: one for every 1000 APB clock cycles and another for every 3000 APB cycles.

1. Set Channel 0 Control Register:
    ChMode = 2 (16-bit Timers)
    ChClk = 1 (APB clock)

2. Set Channel 0 Reload Register, refer to Table 11:
    TMR16_0 = 999
    TMR16_1 = 2999

3. Set Interrupt Enable Register to enable interrupt:
    Ch0Int0En = 1
    Ch0Int1En = 1

4. Enable Timer 0 and Timer 1 of channel 0.
    Ch0TMR0En = 1
    Ch0TMR1En = 1

26.5.2 Generate 2 Aligned PWM Output

Figure 26-2 PWM Example Waveform
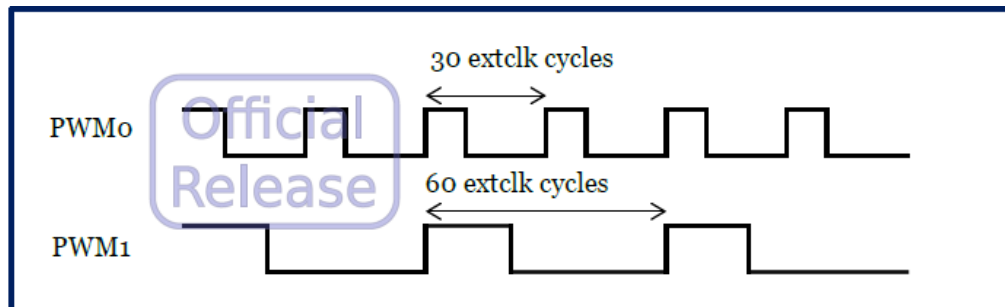
To drive the waveform illustrated in Figure 26-2 PWM Example Waveform:

PWM0: period = 30 external clock cycles, duty cycle = 1/3

PWM1: period = 60 external clock cycles, duty cycle = 1/3

Two PWMs are aligned at positive edge

1. Set Control Register:

    Channel 0: ChMode = 4 (PWM), ChClk = 0 (External Clock),

        PWMPark = 1

    Channel 1: ChMode = 4 (PWM), ChClk = 0 (External Clock),

      PWMPark = 1

2. Set Reload Register, see Table 13.

    Channel 0: PWM16_Hi = 9, PWM16_Lo = 19

    Channel 1: PWM16_Hi = 19, PWM16_Lo = 39

3. Enable PWM.

    Set Channel Enable Register: Ch0PWMEn = 1, Ch1PWMEn = 1.

26.5.3 Changing the PIT Modes

To change the mode of PIT channel that is currently working, the channel must be disabled first. Then set the channel to a new mode and enable it.

27 ATCBUSDEC200

27.1 Introduction

The ATCBUSDEC200 is an AMBA AHB-Lite decoder. It receives bus transactions from the upstream port and dispatches the transactions to the downstream ports according to the slave base address and space size configurations. This decoder also provides slave information registers for software to lookup the memory space assignment information.

27.1.1 Features

- Compliant with the AMBA AHB-Lite protocol
- Provides the slave base/size information through the internal slave device
- Supports one upstream port
- Supports up to 31 downstream ports
- Configurable base/size for each downstream port
- Configurable address width: 24, 32 bits
- Configurable data width: 32, 64, 128 bits

27.1.2 Block Diagram

Figure 27-1 ATCBUSDEC200 Block Diagram shows the block diagram of the ATCBUSDEC200 decoder, which consists of the interconnect logic, the internal slave, and the default slave.

Figure 27-1 ATCBUSDEC200 Block Diagram
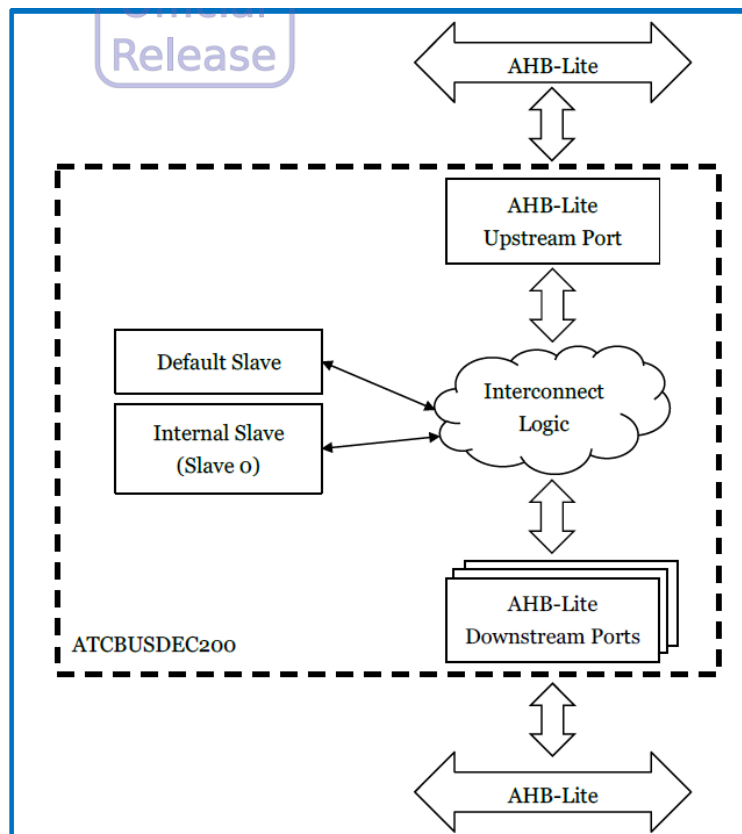
27.1.3 Interconnect Logic

The interconnect logic includes the address decoder and the downstream-to-upstream multiplexor. The address decoder dispatches the upstream transactions to the appropriate slave ports according to the slave base address and size configurations. The downstream-to-upstream multiplexor collects the returned data/responses to the upstream port.

27.1.4 Internal Slave

The internal slave contains registers recording the base/size information of downstream ports. The internal slave is designated the slave number 0. It is an optional slave that could be configured out by undefining the ATCBUSDEC200_SLV0_SUPPORT macro.

27.1.5 Default Slave

A transaction that does not hit any slave is directed to the default slave, which may be configured to generate ERROR as responses. See AHB-Lite Default Slave Error Response for detail.

27.2 Signal Description

Figure 27-2 Signal Diagram shows the signal overview and gives the detailed descriptions of I/O signals.

Figure 27-2 Signal Diagram
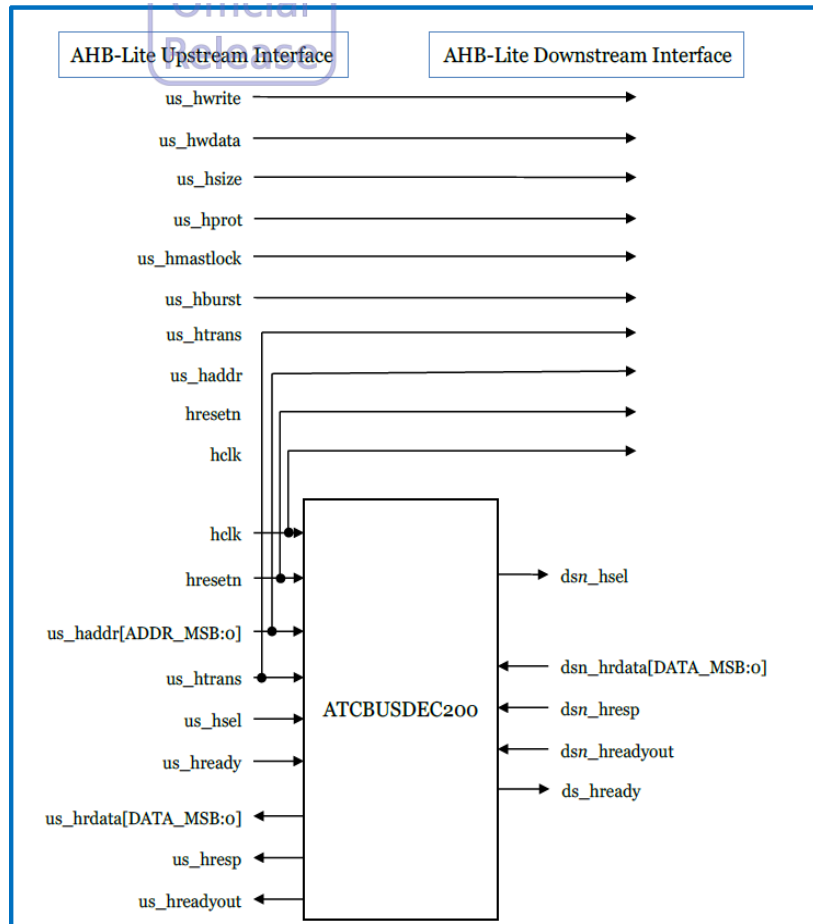
Table 27-1 ATCBUSDEC200 Signal Description

Signal Name	I/O Type	Description
AHB-Lite global signals		
hclk	I	AHB-Lite bus clock
hresetn	I	AHB-Lite bus reset
Upstream AHB-Lite interface		
us_haddr[ADDR_MSB:0]	I	AHB-Lite address bus
us_htrans	I	AHB-Lite transfer type
us_hsel	I	AHB-Lite slave select
us_hready	I	AHB-Lite input ready from upstream ports
us_hrdata[DATA_MSB:0]	O	AHB-Lite read data bus
us_hreadyout	O	AHB-Lite transfer done (internally connected with ds_hready)
us_hresp	O	AHB-Lite transfer response
Downstream AHB-Lite interface		
ds$n$_hsel	O	AHB-Lite slave select for the nth downstream port (n=1~31)
ds_hready	O	AHB-Lite input ready for all downstream ports (internally connected with us_hready)
ds$n$_hrdata[DATA_MSB:0]	I	AHB-Lite read data bus for the nth downstream port (n=1~31)
ds$n$_hreadyout	I	AHB-Lite transfer done for the nth downstream port (n=1~31)
ds$n$_hresp	I	AHB-Lite transfer response for the nth downstream port (n=1~31)

27.3 Programming Model

The registers for ATCBUSDEC200 are all informational only. They reside in the internal slave (slave 0) so they are only available when slave 0 is configured. That is, the hardware configuration option ATCBUSDEC200_SLV0_SUPPORT has to be defined for these registers to be available.

27.3.1 Summary of Registers

Table 27-2 Register Summary provides a summary of registers. All registers are read-only and writes to them are silently ignored. All undefined/reserved registers will be read as zero.

Table 27-2 Register Summary

Offset	Name	Description
+0x00	IdRev	ID and revision register
+0x04~0x1C	-	Reserved
+0x20~0x98	BaseSize$n$	Base/size register of downstream port n (1~31)

27.3.2 Register Description

Note that the registers are only available when the internal slave (slave

0) is defined. See beginning of this chapter for detail.

27.3.3 ID and Revision Register

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 27-3 ID and Revision Register (0x00)

Name	Bit	Description	Reset
ID	31:8	ID number for ATCBUSDEC200	0x000320
RevMajor	7:4	Major revision number	Revision dependent
RevMinor	3:0	Minor revision number	Revision dependent

27.3.4 Base/Size Register

These registers hold the base and size information for each downstream port.

Table 27-4 Base/Size Register of Downstream Port 1~n (0x1c + n*0x4) in 24-bit Address Mode

Name	Bit	Description	Reset
Reserved	23:M	M=ATCBUSDEC200_ADDR_DECODE_WIDTH	0x0
Base	(M-1):10	Base address offset	Depends on the configuration
Reserved	9:4	-	0x0
Size	3:0	Size of the slave address space 0x0: Invalid (for absent ports) 0x1: 1KB 0x2: 2KB 0x3: 4KB 0x4: 8KB 0x5: 16KB 0x6: 32KB 0x7: 64KB 0x8: 128KB 0x9: 256KB 0xA: 512KB 0xB: 1MB 0xC: 2MB 0xD: 4MB 0xE: 8MB 0xF: Reserved	Depends on the configuration

Table 27-5 Base/Size Register of Downstream Port 1~n (0x1c + n*0x4) in 32-bit Address Mode

Name	Bit	Description	Reset
Reserved	31:M	M=ATCBUSDEC200_ADDR_DECODE_WIDTH	0x0
Base	(M-1):20	Base address offset	Depends on the configuration
Reserved	19:4	-	0x0
Size	3:0	Size of the slave address space	Depends on the configuration
		0x0: Invalid (for absent ports)	
		0x1: 1MB	
		0x2: 2MB	
		0x3: 4MB	
		0x4: 8MB	
		0x5: 16MB	
		0x6: 32MB	
		0x7: 64MB	
		0x8: 128MB	
		0x9: 256MB	
		0xA: 512MB	
		0xB: 1GB	
		0xC: 2GB	
		0xE ~ 0xF: Reserved	

27.4 Hardware Configuration Options

27.4.1 AHB-Lite Bus Address Width

Define ATCBUSDEC200_ADDR_WIDTH_24 to set the address width to 24 bits. The default address width is 32 bits.

`define ATCBUSDEC200_ADDR_WIDTH_24

27.4.2 AHB-Lite Address Decode Width

The upstream address decoding logic only performs partial decoding of the address bits received from the AHB-Lite master. Define ATCBUSDEC200_ADDR_DECODE_WIDTH to specify the amount of partial address bits to use for decoding. This value also reflects the total address space size available to all downstream ports. For the 24-bit mode, this macro could be 11 – 24; for the 32-bit mode, this macro could be 21 – 32. For example,

define ATCBUSDEC200_ADDR_DECODE_WIDTH 22

In this case, the address decoding logic only decodes address bits [21:0] for selecting the target slave port.

27.4.3 AHB-Lite Bus Data Width

Define one of the following macro to set the data width to 64 bits or 128 bits. The default address width is 32 bits when neither of them is defined.

`define ATCBUSDEC200_DATA_WIDTH_64

`define ATCBUSDEC200_DATA_WIDTH_128

27.4.4 AHB-Lite Slave Ports

Define ATCBUSDEC200_SLVn_SUPPORT to enable the nth downstream port, where n = 0~31.   He macros do not need to be defined sequentially. For example,

`define ATCBUSDEC200_SLV0_SUPPORT

`define ATCBUSDEC200_SLV1_SUPPORT

`define ATCBUSDEC200_SLV15_SUPPORT

`define ATCBUSDEC200_SLV31_SUPPORT

Note!
- Slave 0 is the optional internal slave which provides accesses to the ATCBUSDEC200 programming registers.
- ATCBUSDEC200_SLV0_SUPPORT needs to be defined explicitly to enable slave 0.

27.4.5 AHB-Lite Slave Size

Define ATCBUSDEC200_SLVn_SIZE (n=1~31) to specify the address space size of the nth downstream port. The values are defined in Table 4 and Table 5 for the 24-bit mode and the 32-bit mode, respectively. For example, specify the address space size of the 2nd downstream port to be 4KB in the 24-bit mode:

`define ATCBUSDEC200_SLV2_SIZE 3

And the same value means 4MB in the 32-bit mode.

Note!

The address space size of slave 0 (the internal slave) is not configurable and it is predefined to be 4KB in the 24-bit mode and 1MB in the 32-bit mode.

27.4.6 AHB-Lite Slave Base Address Offset

Define ATCBUSDEC200_SLVn_OFFSET (n=0~31) to specify the base address offset of the nth downstream port. The value must be slave-size aligned. For example, suppose the size of the entire decoded address space is 2GB and the size of the 3rd downstream port is 8MB, specify the

address offset of the 3rd downstream port to be 0x10800000 as below:

`define ATCBUSDEC200_ADDR_DECODE_WIDTH 31

`define ATCBUSDEC200_SLV3_SIZE 4

`define ATCBUSDEC200_SLV3_OFFSET ATCBUSDEC200_ADDR_DECODE_WIDTH'h10800000

27.4.7 AHB-Lite Default Slave Error Response

Use ATCBUSDEC200_OOR_ERR_EN to control the responses of transactions that do not target any valid downstream port. When defined, the default slave will return the two-cycle ERROR response; otherwise, it will return the OKAY response.

`define ATCBUSDEC200_OOR_ERR_EN

27.5 Integration Guideline

Figure 27-3 ATCBUSDEC200 Connection Example shows an example use of ATCBUSDEC200, which connects an AHB-Lite master to two AHB-Lite slaves.

- An AHB-Lite master is connected to the upstream port of ATCBUSDEC200.
- Slave 3 and slave 9 are connected to the 3rd and 9th downstream ports of ATCBUSDEC200.
- misc indicates the AHB-Lite interface signals hwdata, hsize, hprot, hmastlock, and hburst from the upstream. These signals are connected to all slaves directly.

Figure 27-3 ATCBUSDEC200 Connection Example
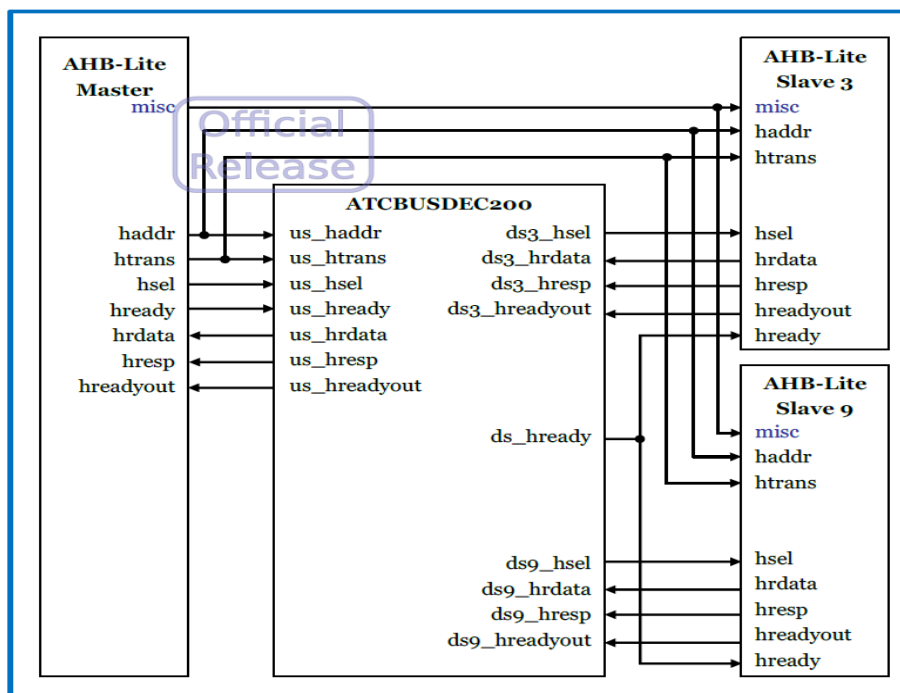
27.6 Access Latencies

Table 26-6 summarizes the ATCBUSDEC200 access latencies, the number of elapsed clock cycles on the upstream port between the moment a request is accepted and the moment the last response for the request is available. The table is made with the following assumptions:

- LEN stands for the value of the length indicated by hburst
- The downstream slave inserts no wait states

Table 27-6 ATCBUSDEC200 Access Latencies

Type of Request	Latency
Read	LEN
Write	LEN

28 ATCBMC200

28.1 Introduction

The ATCBMC200 Bus Matrix Controller is an AHB fabric that allows multiple simultaneous bus transactions. Programmable two-level priority scheme is used for arbitration.

28.1.1 Features

- Supports AMBA™ 2 AHB protocol specification
- – Split transfers are not supported
- – All transfers behave as locked transfers
- Supports 24/32-bit address bus
- Configurable number of AHB masters (1–16)
- Configurable number of AHB slaves (1–15)
- Configurable connectivity between AHB masters and slaves
- Programmable two level priority arbitration scheme

28.1.2 Block Diagram

Figure 28-1 ATCBMC200 Block Diagram shows the block diagram of the ATCBMC200 controller.

Figure 28-1 ATCBMC200 Block Diagram
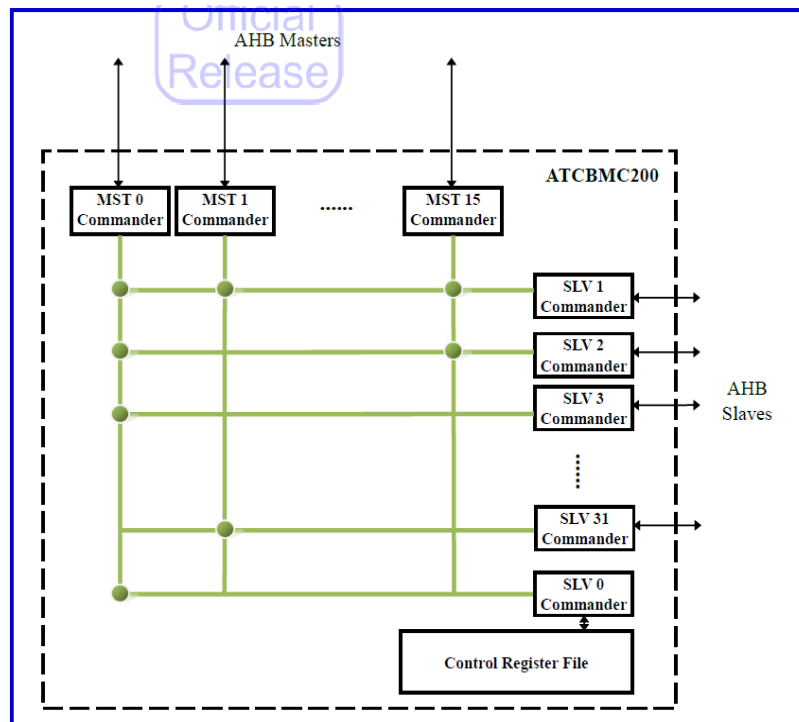
28.1.3 Function Description

The AndeShape™ ATCBMC200 bus matrix accepts requests from AHB masters and dispatches the transactions to the target AHB slaves. The transactions from multiple masters can proceed simultaneously as long as there is no resource conflict.

28.1.4 MST Commander

Each MST Commander interfaces one AHB master and it decides which AHB slave should process the bus transfer based on the address map information (as defined by the Base/Size Registers). The Bus Matrix will then route the transfer to the corresponding slave accordingly.

28.1.5 SLV Commander

Each SLV commander interfaces one AHB slave and it is responsible for arbitrating requests to that slave. The arbitration scheme is two-level priority arbitration.

28.1.6 Bus Matrix

The Bus Matrix routes the transfer between AHB masters and slaves according to the address decoding result provided by the MST Commander.

However, the connectivity inside the matrix is configurable and there might not be paths between all master/slave pairs. Please see Hardware Configuration Options Master and Slave Ports Connectivity for detail.

28.1.7 AHB Slave Grouping

AHB slaves are partitioned into two groups:

- Fixed address map group
- Programmable address map group

The difference is the programmability of the base address and the size as summarized in Table 28-1 Programmability of Base Address and Size Registers for AHB Slave 1–15. It is recommended to connect memory devices to AHB slave 11–15.

Table 28-1 Programmability of Base Address and Size Registers for AHB Slave 1–15

AHB slave number	Base Address	Slave Size	Usage
1–10	Non-programmable	Non-programmable	Memory (fixed address and size) / Device
11–15	Programmable	Programmable	Memory (adjustable address and size)

28.2 Signal Description

Figure 28-2 ATCBMC200 I/O Signals shows the input/output (I/O) signals of the ATCBMC200 controller.

Figure 28-2 ATCBMC200 I/O Signals
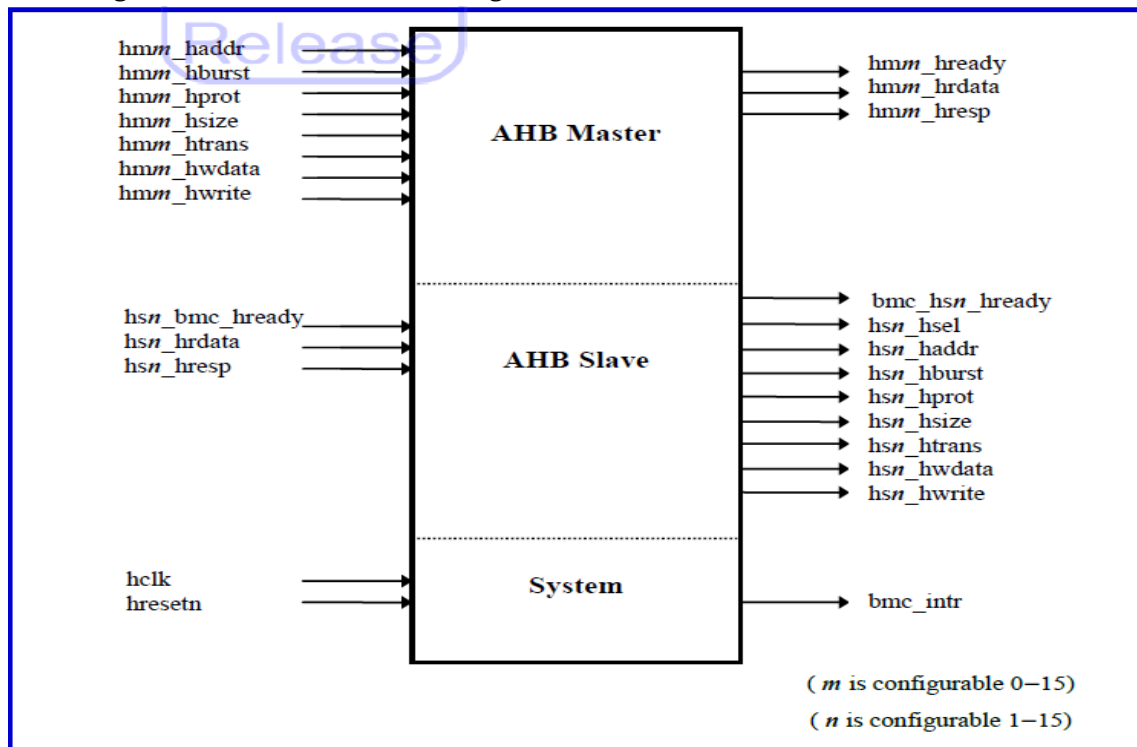
Table 28-2 ATCBMC200 Signal Definition gives the detailed descriptions of ATCBMC200 I/O signals.

Table 28-2 ATCBMC200 Signal Definition

Signal Name	I/O Type	Description
System		
hclk	I	AHB clock
hresetn	I	AHB reset signal; Active Low
bmc_intr	O	Interrupt signal; Active High
		The signal is asserted when there are no target slaves or multiple target slaves after decoding the address bus.
AHB Master Interface		
hm$m$_haddr[W:0]	I	AHB master m address bus
		W = 31 in 32-bit address mapping mode
		W = 23 in 24-bit address mapping mode
hm$m$_hburst[2:0]	I	AHB master m burst type signal
hm$m$_hprot[3:0]	I	AHB master m protect signal
hm$m$_hsize[2:0]	I	AHB master m data size signal
hm$m$_htrans[1:0]	I	AHB master m transfer type signal
hm$m$_hwdata[31:0]	I	AHB master m write data bus
hm$m$_hwrite	I	AHB master m read/write control signal
hm$m$_hrdata[31:0]	O	AHB master m read data bus
hm$m$_hready	O	AHB master m hready signal. Indicates that a transfer has been finished on the bus
hm$m$_hresp[1:0]	O	AHB master m response signal
AHB Slave Interface		
hs$n$_bmc_hready	I	AHB hready from slave n, indicating that a transfer to/from slave n has finished, n=1–15
		To avoid cyclic timing loop, the fan-in of hs$n$_bmc_hready must not include bmc_hs$n$_hready through combinational logic.

Signal Name	I/O Type	Description
hs$n$_hrdata[31:0]	I	AHB slave n read data bus. The read data bus transfers data from bus slaves to bus masters for read operations
hs$n$_hresp[1:0]	I	AHB slave n response signal. The signal provides response status of the transfer
bmc_hs$n$_hready	O	AHB hready for slave n, indicating that the bus is ready to issue a new transfer. n=1–15
hs$n$_haddr[W:0]	O	AHB slave n address bus W = 31 in 32-bit address mapping mode W = 23 in 24-bit address mapping mode
hs$n$_hburst[2:0]	O	AHB slave n write data bus
hs$n$_hprot[3:0]	O	AHB slave n protect signal
hs$n$_hsel	O	AHB slave n select signal
hs$n$_hsize[2:0]	O	AHB slave n data size signal, indicating the size of the transfer
hs$n$_htrans[1:0]	O	AHB slave n transfer type signal, indicating the type of the current transfer
hs$n$_hwdata[31:0]	O	AHB slave n write data bus
hs$n$_hwrite	O	AHB slave n transfer direction. HIGH indicates a write transfer and LOW indicates a read transfer

28.3 Programming Model

28.3.1 Summary of Registers

Table 28-3 ATCBMC200 Register Summary shows a summary of the ATCBMC200 registers.

Table 28-3 ATCBMC200 Register Summary

Offset	Name	Description
+0x00	IdRev	ID and revision register
+0x04–0x0C	-	Reserved
+0x10	Priority	Priority register
+0x14	Ctrl	Control register
+0x18	IntrStatus	Interrupt status register
+0x1C	-	Reserved
+0x20–e0x58	BaseSize$n$	Base/Size registers for AHB slave n, where n=1–15

28.3.2 Register Description

The abbreviations for the Type column are summarized below.

RO: read only

RW: readable and writable

RW1C: readable and write 1 to clear

28.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the revision fields are revision dependent.

Table 28-4 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:12	RO	ID number for ATCBMC200	0x00002
RevMajor	11:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

28.3.4 Priority Register (0x10)

The ATCBMC200 controller provides two arbitration modes — fixed priority mode and two level round-robin mode. The modes are controlled by the Priority field of the Priority Register. When all 16 bits of the Priority field equal zero, the arbitration mode is the fixed mode, where the master number determines the priority with the lower masters getting higher priorities. When the Priority field is not all zero, the arbiter works in the two-level round-robin mode where each bit of the Priority field indicates the priority group of the corresponding master. The masters are partitioned into two groups, group-0 and group-1, according to their corresponding Priority bit values. Group-1 masters are given higher priorities than group-0 masters, and the priorities within the same group are round-robin.

Under the two-level round-robin mode, Bit-0 of the Priority field indicates the priority of Master 0, bit-1 indicates that of Master 1, etc. If Master mdoes not exist, then bit-m is read as zero and write ignored.

Master 0 can be set to the top priority by setting the PHigh0 field to 1 regardless of the priority mode.

Table 28-5 Priority Register

Name	Bit	Type	Description	Reset
PHigh0	31	RW	Master 0 is the highest priority master regardless of the Priority mode 0: Disable 1: Enable	0
Reserved	30:16	-	-	-
Priority	15:0	RW	Priority Mode 0x0: Fixed mode. The priority from high to low is Master 0, Master 1, Master 2 … Master 15. Other values: Two-level round-robin mode. The behavior of two-level round-robin partitions the masters into high priority group and low priority group. Within each group, the priority policy is round-robin.	0xFFFF

28.3.5 Control Register (0x14)

Table 28-6 Control Register

Name	Bit	Type	Description	Reset
Reserved	31:2	-	-	-
IntrEn	1	RW	Interrupt enable 0: Disable interrupt 1: Enable interrupt	0
UnmappedResp	0	RW	The response type when there is no slave selected after decoding the address bus 0: OK response 1: ERROR response	0

28.3.6 Interrupt Status Register (0x18)

Table 28-7 Interrupt Status Register

Name	Bit	Type	Description	Reset
Reserved	31:16	-	-	-
Intr_Mst15SelErr	15	RW1C	Master15 address selection error 0: No error 1: The transfer address from Master15 does not map to any slaves	0
Intr_Mst14SelErr	14	RW1C	Master14 address selection error 0: No error 1: The transfer address from Master14 does not map to any slaves	0
Intr_Mst13SelErr	13	RW1C	Master13 address selection error 0: No error 1: The transfer address from Master13 does not map to any slaves	0
Intr_Mst12SelErr	12	RW1C	Master12 address selection error 0: No error 1: The transfer address from Master12 does not map to any slaves	0
Intr_Mst11SelErr	11	RW1C	Master11 address selection error 0: No error 1: The transfer address from Master11 does not map to any slaves	0
Intr_Mst10SelErr	10	RW1C	Master10 address selection error 0: No error 1: The transfer address from Master10 does not map to any slaves	0

.

Name	Bit	Type	Description	Reset
Intr_Mst9SelErr	9	RW1C	Master9 address selection error 0: No error 1: The transfer address from Master9 does not map to any slaves	0
Intr_Mst8SelErr	8	RW1C	Master8 address selection error 0: No error 1: The transfer address from Master8 does not map to any slaves	0
Intr_Mst7SelErr	7	RW1C	Master7 address selection error 0: No error 1: The transfer address from Master7 does not map to any slaves	0
Intr_Mst6SelErr	6	RW1C	Master6 address selection error 0: No error 1: The transfer address from Master6 does not map to any slaves	0
Intr_Mst5SelErr	5	RW1C	Master5 address selection error 0: No error 1: The transfer address from Master5 does not map to any slaves	0
Intr_Mst4SelErr	4	RW1C	Master4 address selection error 0: No error 1: The transfer address from Master4 does not map to any slaves	0
Intr_Mst3SelErr	3	RW1C	Master3 address selection error 0: No error 1: The transfer address from Master3 does not map to any slaves	0
Intr_Mst2SelErr	2	RW1C	Master2 address selection error 0: No error 1: The transfer address from Master2 does not map to any slaves	0
Intr_Mst1SelErr	1	RW1C	Master1 address selection error 0: No error 1: The transfer address from Master1 does not map to any slaves	0
Intr_Mst0SelErr	0	RW1C	Master0 address selection error 0: No error 1: The transfer address from Master0 does not map to any slaves	0

28.3.7 Base/Size Register of AHB Slave 1–10 (0x20–0x44)

These registers hold the base address and the size of AHB slave 1–10 and they are read only. The register format is shown in Table 28-8 Base/Size Register of AHB Slave 1–10 in 24-bit Mode and Table Table 28-9 Base/Size Register of AHB Slave 1–10 in 32-bit Mode for 24-bit and 32-bit addressing modes, respectively.

Table 28-8 Base/Size Register of AHB Slave 1–10 in 24-bit Mode

Name	Bit	Type	Description	Reset
Base	23:10	RO	Base address	Depends on the configuration
Reserved	9:4	-	-	-
Size	3:0	RO	Size of the address space 0x0: Invalid slave 0x8: 128K 0x1: 1K 0x9: 256K 0x2: 2K 0xA: 512K 0x3: 4K 0xB: 1M 0x4: 8K 0xC: 2M 0x5: 16K 0xD: 4M 0x6: 32K 0xE: 8M 0x7: 64K 0xF: Reserved	Depends on the configuration

Table 28-9 Base/Size Register of AHB Slave 1–10 in 32-bit Mode

Name	Bit	Type	Description	Reset
Base	31:20	RO	Base address	Depends on the configuration
Reserved	19:4	-	-	-
Size	3:0	RO	Size of the address space: 0x0: Invalid slave 0x8: 128M 0x1: 1M 0x9: 256M 0x2: 2M 0xA: 512M 0x3: 4M 0xB: 1G 0x4: 8M 0xC: 2G 0x5: 16M 0xD–0xF: Reserved 0x6: 32M 0x7: 64M	Depends on the configuration

28.3.8 Base/Size Register of AHB Slave 11–15 (0x48–0x58)

These register show the base address and the size of AHB slave 11–15. Contrary to the read only Base/Size registers of AHB Slave 1–10, these registers are readable and writable. The register format is shown in Table 28-10 Base/Size Register of AHB Slave 11–15 in 24-bit Mode and Table 28-11 Base/Size Register of AHB Slave 11–15 in 32-bit Mode for 24-bit and 32-bit addressing modes, respectively. Note that the base address must be aligned to multiple of the size of the corresponding AHB slave address space.

Table 28-10 Base/Size Register of AHB Slave 11–15 in 24-bit Mode

Name	Bit	Type	Description	Reset
Base	23:10	RW	Base address	Depends on the configuration
Reserved	9:4	-	-	-
Size	3:0	RW	Size of the address space: 0x0: Invalid slave 0x8: 128K 0x1: 1K 0x9: 256K 0x2: 2K 0xA: 512K 0x3: 4K 0xB: 1M 0x4: 8K 0xC: 2M 0x5: 16K 0xD: 4M 0x6: 32K 0xE: 8M 0x7: 64K 0xF: Reserved	Depends on the configuration

Table 28-11 Base/Size Register of AHB Slave 11–15 in 32-bit Mode

Name	Bit	Type	Description	Reset
Base	31:20	RW	Base address	Depends on the configuration
Reserved	19:4	-	-	-

Size	3:0	RW	Size of the address space:		Depends on the configuration
			0x0: Invalid slave	0x8: 128M	
			0x1: 1M	0x9: 256M	
			0x2: 2M	0xA: 512M	
			0x3: 4M	0xB: 1G	
			0x5: 16M	0xC: 2G	
			0x6: 32M	0xD–0xF: reserved	
			0x7: 64M		

28.4 Hardware Configuration Options

28.4.1 Address Width

Define the following macro to set the address width to 24-bit. Default address width is 32-bit.

`` `define ATCBMC200_ADDR_WIDTH_24 ``

28.4.2 Number of AHB Master Ports

Define ATCBMC200_AHB_MSTmto enable the respective AHB master port, where valid values for mrange from 1 to 15. For example, to enable AHB master 1 and 3, the following macros shall be defined:

`` `define ATCBMC200_AHB_MST1 ``

`` `define ATCBMC200_AHB_MST3 ``

28.4.3 Number of AHB Slave Ports

Define ATCBMC200_AHB_SLVn to enable the respective AHB slave port, where valid values for n range from 1 to 15. For example, to enable AHB slave 2 and 5, the following macros shall be defined:

`` `define ATCBMC200_AHB_SLV2 ``

`` `define ATCBMC200_AHB_SLV5 ``

28.4.4 Master and Slave Ports Connectivity

Define ATCBMC200_MSTm_SLVn to enable the bus matrix connection between AHB master mand slave n. For example, to create a bus matrix where slave 0, 1 and 2 are connected to master 0; slave 0 and 2 are connected to master 1; slave 1 and 2 are connected to master 3, the following macros shall be defined:

`define ATCBMC200_MST0_SLV0

`define ATCBMC200_MST0_SLV1

`define ATCBMC200_MST0_SLV2

`define ATCBMC200_MST1_SLV0

`define ATCBMC200_MST1_SLV2

`define ATCBMC200_MST3_SLV1

`define ATCBMC200_MST3_SLV2

28.4.5 Slave Base Address

The default base address of AHB slave n is defined by macro ATCBMC200_AHB_SLVn_BASE, where valid values of n ranges from 0 to 15. These are the reset values for the Base field of the Base/Size Registers. The values are read-only for slave 1–10 and the values for slave 11–15 are programmable. Note that the slave base address must be slave-size aligned.

In the 32-bit address mapping mode, the value should be 32-bit wide. In addition, bit[19:0] shall be 0 because the minimum space in 32-bit mode is 1MB. The following example assigns AHB slave 5 to 0x90300000:

`define ATCBMC200_AHB_SLV5_BASE 32'h9030_0000

In 24-bit address mapping mode, the value should be 24-bit wide. In addition, bit[9:0] shall be 0 because the minimum space in 24-bit mode is 1KB. The following example assigns AHB slave 6 to 0x7a0000:

`define ATCBMC200_AHB_SLV6_BASE 24'h7a_0000

28.4.6 Slave Size

The default size of AHB slave n is defined by macro ATCBMC200_AHB_SLVn_SIZE, where valid values of n ranges from 1 to 15. These are reset values for the Size field of the Base/Size Registers. The values are read-only for slave 1–10 and the values for slave 11–15 are programmable.

Table 28-12 Slave Sizes Encoding Tableshows the valid values for this macro. The encoding has different meaning depending on the address mapping mode (32-bit or 24-bit mode). The following example defines a 16MB space for AHB slave 3 in the 32-bit mode:

`define ATCBMC200_AHB_SLV3_SIZE 4'h5

Table 28-12 Slave Sizes Encoding Table

ATCBMC200_AHB_SLV*n*_SIZE	Address Space Size in 32-bit mode	Address Space Size in 24-bit mode
4'h0	No slave	No slave
4'h1	1M	1K
4'h2	2M	2K
4'h3	4M	4K
4'h4	8M	8K
4'h5	16M	16K
4'h6	32M	32K
4'h7	64M	64K
4'h8	128M	128K
4'h9	256M	256K
4'hA	512M	512K
4'hB	1G	1M
4'hC	2G	2M
4'hD	Reserved	4M
4'hE	Reserved	8M
4'hF	Reserved	Reserved

29 ATCAPBBRG100

29.1 Introduction

The ATCAPBBRG100 bridge converts transactions between an AHB bus and an APB bus.

29.1.1 Features

- Compliant with the AMBA 2 AHB protocol
- Compliant with the AMBA 4 APB protocol
- Support of various synchronous AHB/APB clock ratios (N:1, N = 1, 2, 3, ...)
- Support of up to 31 APB slaves
- Support of 32-bit or 24-bit addressing mode
- Support of write buffering
- Hardware-configurable base address and size for each APB slave
- Slave information registers for automatic bus component discovery

29.1.2 Block Diagram

Figure 29-1 ATCAPBBRG100 Block Diagram shows the block diagram of the ATCAPBBRG100 bridge.

Figure 29-1 ATCAPBBRG100 Block Diagram
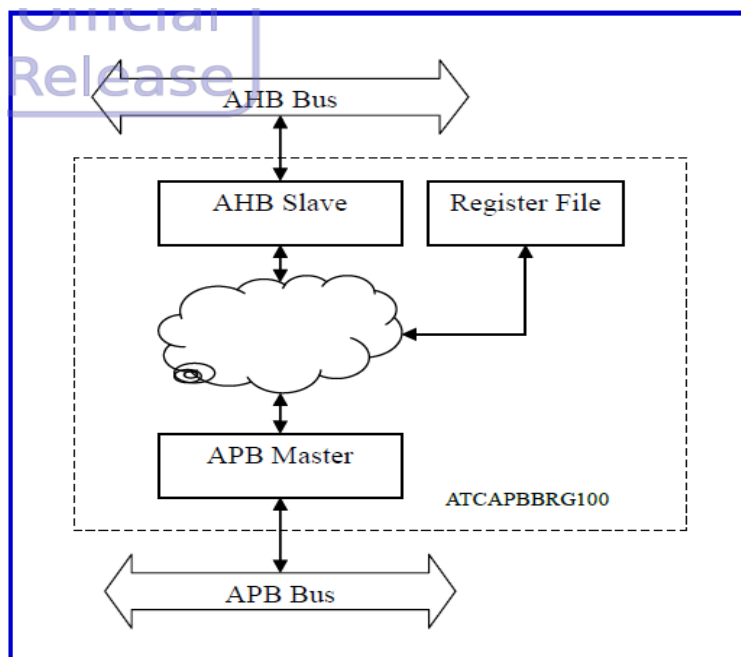
29.1.3 Function Description

The AndeShape ATCAPBBRG100 bridge converts transactions between an AHB bus and an APB bus. The bridge is compliant with AMBA 4 specifications. It is a slave on the AHB bus and is the only master on the APB bus.

AHB transactions are dispatched to APB slaves according to the slave base address and space size configurations. APB slave 0 represents the bridge for the register file access.

The register file keeps the configuration information for software to get the ATCAPBBRG100 ability and to enumerate the APB slaves. It also contains a field to control the write buffering scheme.

29.2 Signal Description

Figure 29-2 ATCAPBBRG100 Logic Symbol shows the logic symbol of the ATCAPBBRG100 bridge.

Figure 29-2 ATCAPBBRG100 Logic Symbol
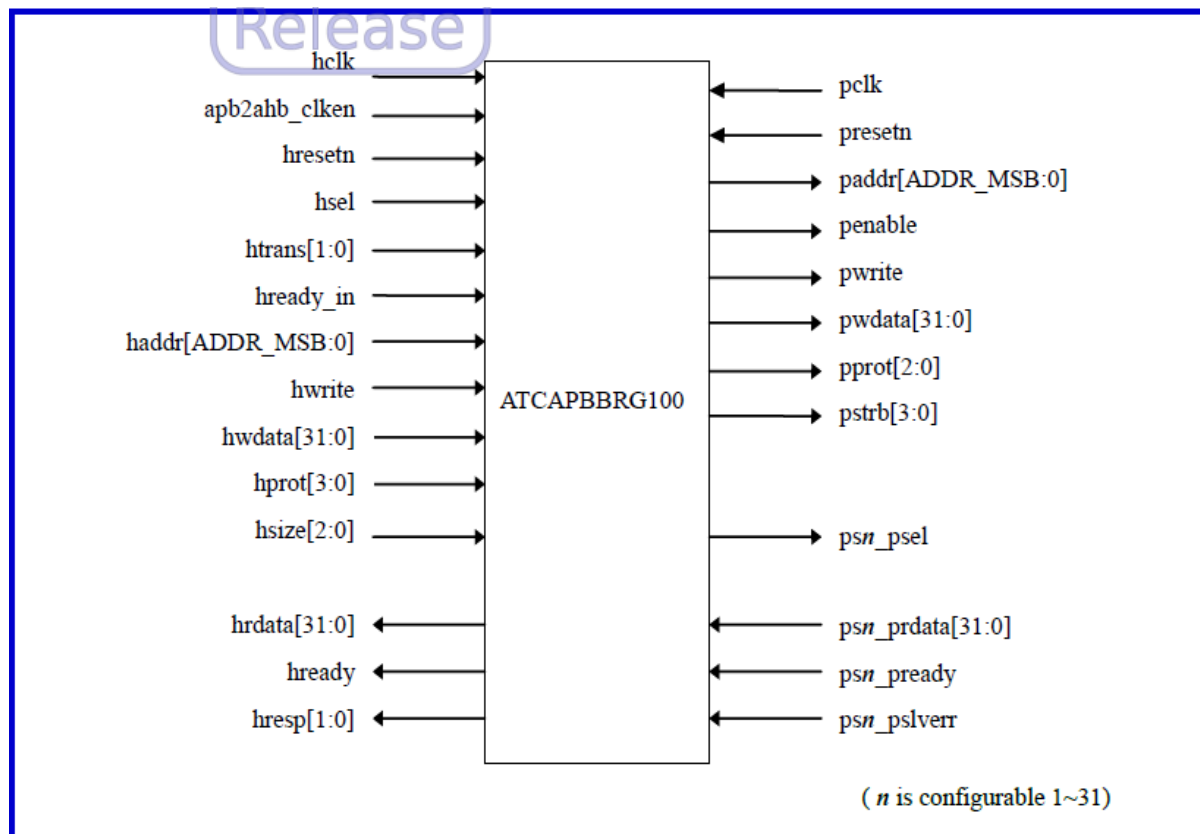
Table 29-1 ATCAPBBRG100 Signal Definition gives the detailed descriptions of ATCAPBBRG100 I/O signals.

Table 29-1 ATCAPBBRG100 Signal Definition

Signal Name	I/O Type	Description
AHB Bus Interface		
hclk	I	AHB clock
apb2ahb_clken	I	AHB clock enable is an AHB domain signal indicating the valid AHB clock cycles to sample and update APB domain signals when the AHB frequency is a multiple of the APB frequency.
hresetn	I	AHB reset signal (Active -Low)
hsel	I	AHB slave select signal
haddr[ADDR_MSB:0]	I	AHB address bus. The value of ADDR_MSB depends on the configuration.
hprot[3:0]	I	AHB protect signal
hsize[2:0]	I	AHB data size signal
htrans[1:0]	I	AHB transfer type signal
hready_in	I	AHB hready signal to indicate the previous transfer is complete
hwdata[31:0]	I	AHB write data bus
hwrite	I	AHB read/write control signal
hready	O	AHB hready signal is returned to the master
hrdata[31:0]	O	AHB read data bus is returned to the master
hresp[1:0]	O	AHB response signal is returned to the master
APB Bus Interface		
pclk	I	APB clock
presetn	I	APB reset signal (Active-Low)
ps$n$_psel	O	APB select signal This signal indicates that APB slave n is selected
ps$n$_prdata[31:0]	I	APB read data of slave n The read data bus is driven by APB slave n device during the read cycles

Signal Name	I/O Type	Description
ps*n*_pready	I	APB ready of slave *n*
		The slave uses this signal to extend an APB transfer
ps*n*_pslverr	I	This signal indicates a transfer failure of the slave *n*
paddr[ADDR_MSB:0]	O	APB address bus. The value of ADDR_MSB depends on the configuration.
		ADDR_MSB = 23 for 24-bit addressing mode;
		ADDR_MSB = 31 for 32-bit addressing mode.
penable	O	APB enable signal translated from the AHB master port
pprot[2:0]	O	APB protect signal translated from the AHB master port.
pstrb[3:0]	O	APB write strobe signal translated from the AHB master port.
pwrite	O	APB write signal translated from the AHB master port
pwdata[31:0]	O	APB write data bus translated from the AHB master port

29.2.1 Clock Enable Signal

The ATCAPBBRG100 design has an AHB clock enable input signal. The clock enable signal indicates the valid AHB clock cycles to sample and update APB domain signals when the AHB frequency is an integer multiple of the APB frequency. It must assert for one AHB clock cycle for each APB clock. An example is illustrated in Figure 29-3 ATCAPBBRG100 Clock Enable Signal for 4:1 AHB-to-APB Clock Ratio for the case that the AHB frequency is 4 times of the APB frequency. For the special case that AHB clock and APB clock are identical, the clock enable signal must be always high.

Figure 29-3 ATCAPBBRG100 Clock Enable Signal for 4:1 AHB-to-APB Clock Ratio
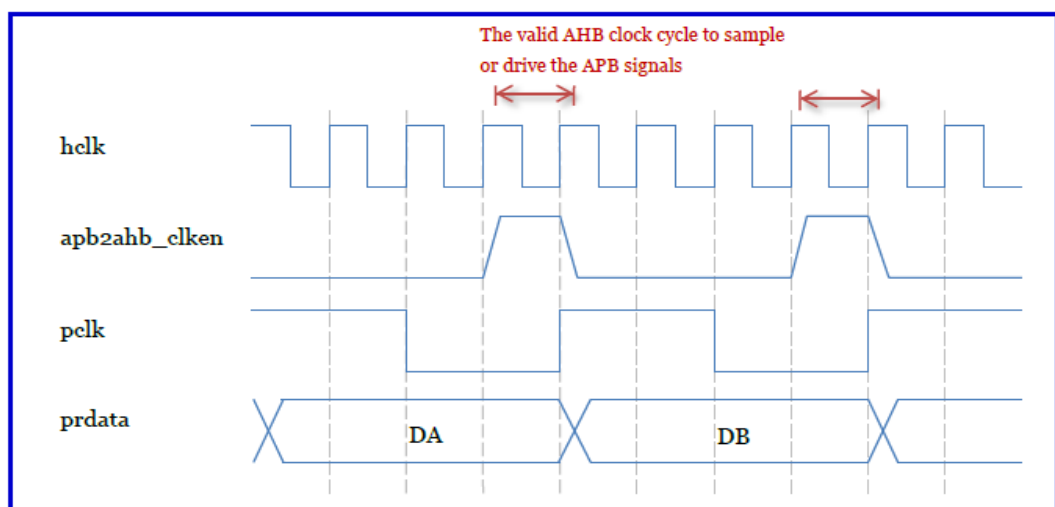
29.3 Programming Model

29.3.1 Summary of Registers

Table 29-2 ATCAPBBRG100 Register Summary shows a summary of the ATCAPBBRG100 registers.

Table 29-2 ATCAPBBRG100 Register Summary

Offset	Name	Description
+0x00	IdRev	ID and revision register
+0x04~0x10	-	Reserved
+0x14	Ctrl	Control register
+0x18~0x1C	-	Reserved
+0x20~0x98	BaseSize*n*	Base/size register of APB slave *n* (*n* = 1 ~ 31)

29.3.2 Register Description

The abbreviations for the Type column are summarized below.

RO: read only

RW: readable and writable

29.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 29-3 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for ATCAPBBRG100	0x000210
RevMajor	7:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

29.3.4 Control Register (0x14)

This register controls the behavior of the ATCAPBBRG100 bridge. The

WBUF_EN bit enables write buffering on the bridge. When write buffering is enabled, responses are returned to the master immediately before the write transaction is issued to the APB bus. This effectively shortens the write transaction latency. All buffered write transactions are issued to the APB bus in order and a subsequent read transaction waits until the buffered write transaction completes.

Table 29-4 Configuration Register

Name	Bit	Type	Description	Reset
Reserved	31:1	-	-	-
WBUF_EN	0	RW	Force the write buffering for all writes 1: Set this bit to force write buffering. All write transactions from the AHB master will be buffered in the bridge. 0: Writing buffering is based on AHB HPROT signal. Note: When toggling this bit, there should be no outstanding APB transactions.	0

29.3.5 Base/Size Register of Slave 1~n (0x20 + (n-1) * 0x4)

This register shows the base address offset and the size of APB slave n (n = 1 ~ 31). The register format is shown in Table 29-5 Base/Size Register of Slave 1~n in 24-bit modeand Table 29-6 Base/Size Register of Slave 1~n in 32-bit modefor 24-bit and 32-bit addressing modes, respectively.

Table 29-5 Base/Size Register of Slave 1~n in 24-bit mode

Name	Bit	Type	Description	Reset
Reserved	23:M	-	M = ATCAPBBRG100_ADDR_DECODE_WIDTH	-
Base	(M-1):10	RO	Base address offset	Depends on the configuration
Reserved	9:4	-	-	-
Size	3:0	RO	Size of the address space: 0x0: invalid slave 0x1: 1KiB 0x2: 2KiB 0x3: 4KiB 0x4: 8KiB 0x5: 16KiB 0x6: 32KiB 0x7: 64KiB 0x8: 128KiB 0x9: 256KiB 0xA: 512KiB 0xB: 1MiB 0xC: 2MiB 0xD: 4MiB 0xE: 8MiB 0xF: Reserved	Depends on the configuration

Table 29-6 Base/Size Register of Slave 1~n in 32-bit mode

Name	Bit	Type	Description	Reset
Reserved	31:M	-	M = ATCAPBBRG100_ADDR_DECODE_WIDTH	-
Base	(M-1):20	RO	Base address offset	Depends on the configuration
Reserved	19:4			-
Size	3:0	RO	Size of the address space: 0x0: invalid slave 0x1: 1MiB 0x2: 2MiB 0x3: 4MiB 0x4: 8MiB 0x5: 16MiB 0x6: 32MiB 0x7: 64MiB 0x8: 128MiB 0x9: 256MiB 0xA: 512MiB 0xB: 1GiB 0xC: 2GiB 0xD~0xF: Reserved	Depends on the configuration

29.4 Hardware Configuration Options

29.4.1 Address Width

Define the following macro to set the address width to 24-bit. Default address width is 32-bit.

`define ATCAPBBRG100_ADDR_WIDTH_24

29.4.2 Address Decode Width

Define ATCAPBBRG100_ADDR_DECODE_WIDTH to identify the width of the APB address decoding logic. The APB address decoding logic only decodes the partial address bits receiving from the AHB slave port. For 24-bit mode, this macro could be 11~24; for 32-bit mode, this macro could be 21~32. For example,

`define ATCAPBBRG100_ADDR_DECODE_WIDTH 16

In this case, the address decoding logic only decodes the address bit [15:0] to select the target APB slave.

29.4.3 APB Slave Ports

Define ATCAPBBRG100_SLV_n to enable the nth APB slave ports, where n=1~31. For example,

`define ATCAPBBRG100_SLV_1

`define ATCAPBBRG100_SLV_2

29.4.4 APB Slave Size

Define ATCAPBBRG100_SLVn_SIZE (n=1~31) to identify the space size of the nth APB slave based on Table 5 and Table 6 for 24-bit mode and 32-bit mode, respectively. For example, specify the space size of APB slave #1 to be 4KiB in 24-bit mode:

`define ATCAPBBRG100_SLV1_SIZE 3

29.4.5 APB Slave Base Address Offset

Define ATCAPBBRG100_SLVn_OFFSET (n=1~31) to identify the base address offset of the nth APB slave. The value of the offset must be slave-size aligned and it should be a Verilog hexadecimal constant number with bit width specified. The bit width should be equal to the Address Decode Width, which reflects the total address space size of the APB bus (see Address Decode Width).

For example, suppose the size of the entire APB bus address space is 64KiB and the size of the APB slave 1 is 4KiB. The offset value must be an integral multiple of 4KiB (e.g., 0x1000), and the width of the offset value should be log2(64K) = 16. Thus, the definition for specifying the address offset of the APB slave 1 at 0x1000 is:

`define ATCAPBBRG100_SLV1_OFFSET 16'h1000

29.5 Access Latencies

Table 29-7 ATCAPBBRG100 Access Latencies summarizes the ATCAPBBRG100 access latencies, the number of elapsed clock cycles on the upstream port between the moment an address is accepted and the moment the corresponding data/status is available.Table 29-7 ATCAPBBRG100 Access Latencies shows the minimum latency cycles, which are made with the following assumptions:

- The downstream APB interface inserts zero wait state.
- Zero arbitration wait state
- The frequency ratio of the AHB clock to the APB clock is n:1. The two clocks are synchronously related.

Table 29-7 ATCAPBBRG100 Access Latencies

Request Type	Latencies (AHB Clock Cycles)
AHB Interface (1:1 Identical Clock Configuration)	
Read	2
Non-bufferable Write	3
Bufferable Write (when buffer is NOT full)	1
Bufferable Write (when buffer is full)	2
AHB Interface (n :1 Synchronous Clock Configuration)	
Read	$2n+(n-1)$
Non-bufferable Write	$2n+1$
Bufferable Write (when buffer is NOT full)	1
Bufferable Write (when buffer is full)	$2n$

For bufferable writes, the upstream interface always receives an OKAY response if an internal buffer entry is available (not full). This means that the minimum access latency of bufferable writes is one AHB clock cycle. If the buffer is full, an additional access latency of at least 2n AHB clock cycles will occur, which is the amount of time for the downstream interface to drain one internal buffer entry.

The propagation delay of the specific bufferable write request from the upstream interface to the downstream interface is two AHB clock cycles minimally.

30 ATCAPBDEC100

30.1 Introduction

The ATCAPBDEC100 is an APB bus decoder dispatching bus transactions from an APB master to APB slaves according to the bus address mapping, which is hardware-configurable.

30.1.1 Features

- Compliant with AMBA™ 3 APB protocol specification
- Supports up to 31 APB slaves
- Supports 32-bit or 24-bit addressing mode
- Hardware-configurable base address and size for each APB slave
- Slave information registers for automatic bus component discovery

30.1.2 Block Diagram

Figure 30-1 ATCAPBDEC100 Block Diagram shows the block diagram of the ATCAPBDEC100 decoder.

Figure 30-1 ATCAPBDEC100 Block Diagram
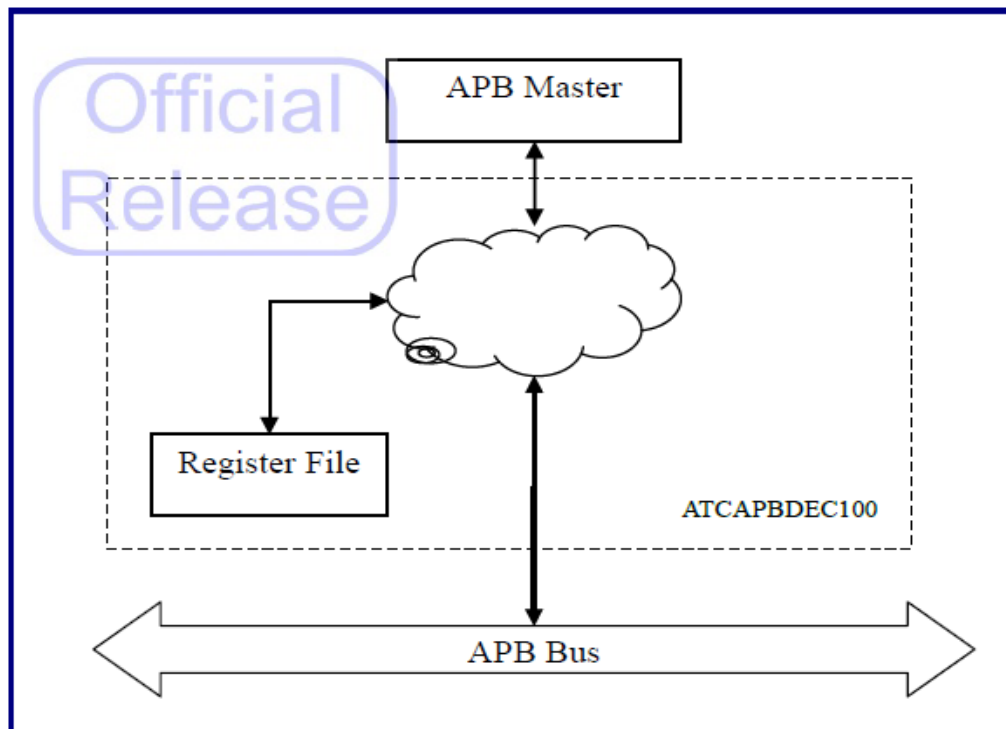
30.1.3 Function Description

AndeShape™ ATCAPBDEC100 is an APB bus decoder dispatching bus transactions from the APB master to APB slaves. The decoder is compliant with AMBA™ 3 specifications. It has the following blocks:

- an upstream APB port receiving transactions from an APB master,
- a combinational decoding logic for dispatching the transactions,
- a downstream APB port transmitting transactions to APB slaves,
- a register file recording informations about the slaves.

APB transactions from the mater are dispatched to APB slaves according to the slave base address and space size configurations. APB slave 0 is the ATCAPBDEC100 register file. The register file keeps the configuration information about the ATCAPBDEC100 capabilities and the slave address mappings.

30.2 Signal Description

The signal overview is illustrated in Figure 29-2 ATCAPBBRG100 Logic Symbol

Figure 30-2 ATCAPBDEC100 Input/Output Signals
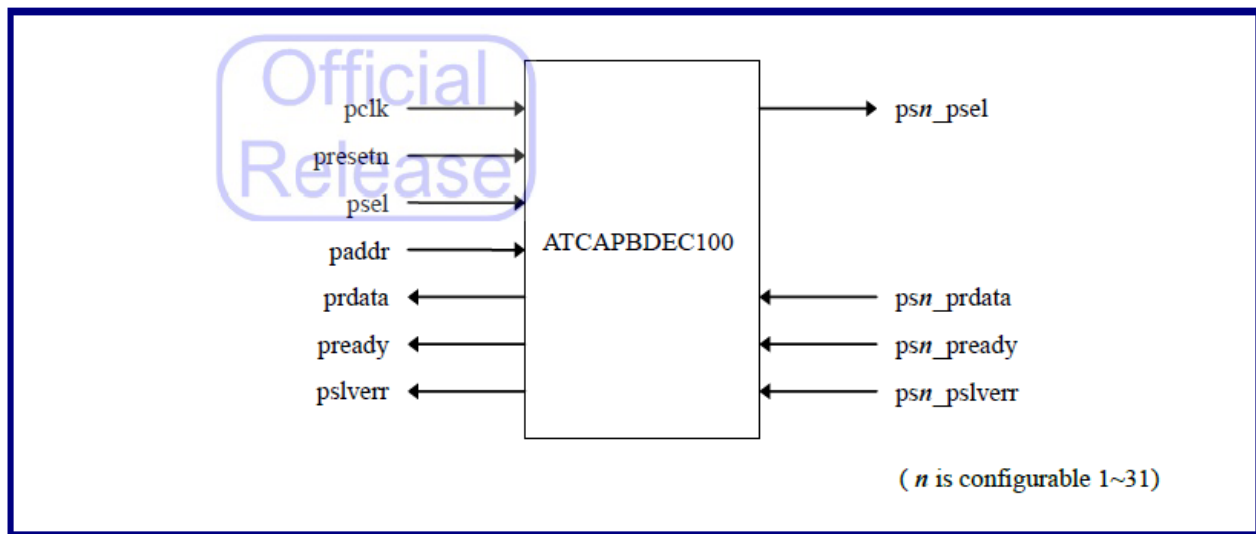
Table 30-1 ATCAPBDEC100 Signal Definition gives the descriptions of the ATCAPBDEC100 input/output (I/O) signals.

Table 30-1 ATCAPBDEC100 Signal Definition

Signal Name	I/O Type	Description
pclk	I	APB clock
presetn	I	APB reset signal. (active low)
psel	I	APB select signal from the APB master.
paddr[ADDR_MSB:0]	I	APB address bus from the APB master. ADDR_MSB = 23 for 24-bit addressing mode; ADDR_MSB = 31 for 32-bit addressing mode.
prdata[31:0]	O	APB read data bus for the APB master.
pready	O	APB ready for the APB master.
pslverr	O	APB slave error for the APB master.
ps*n*_psel	O	APB select signal for the slave *n*. This signal indicates that the APB slave *n* is selected.
ps*n*_prdata[31:0]	I	APB read data from the slave *n*. The read data bus is driven by the APB slave *n* during the read cycles.
ps*n*_pready	I	APB ready from the slave *n*. The slave uses this signal to extend an APB transfer.
ps*n*_pslverr	I	This signal indicates a transfer failure from the slave *n*.

30.3 Programming Model

30.3.1 Summary of Registers

Table 29-2 ATCAPBBRG100 Register Summary shows a summary of the ATCAPBDEC100 registers.

Table 30-2 ATCAPBDEC100 Register Summary

Offset	Name	Description
+0x00	IdRev	ID and revision register
+0x04~0x1C	-	Reserved
+0x20~0x98	BaseSize*n*	Base/size register of APB slave *n* (*n* = 1 ~ 31)

30.3.2 Register Description

The abbreviations for the Type column are summarized below.

RO: read only

RW: readable and writable

30.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 30-3 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:12	RO	ID number for ATCAPBDEC100	0x00031
RevMajor	11:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

30.3.4 Base/Size Register of Slave 1~ n (0x20 + (n-1) * 0x4)

This register holds the base address offset and the size of APB slave n (n = 1 ~ 31). The register format is shown in Table 29-4 Configuration Register and Table 29-5 Base/Size Register of Slave 1~n in 24-bit mode for 24-bit and 32-bit addressing mode, respectively.

Table 30-4 Base/Size Register of Slave 1~n in the 24-bit Mode

Name	Bit	Type	Description	Reset
Reserved	23:M	-	M = ATCAPBDEC100_ADDR_DECODE_WIDTH	-
Base	(M-1):10	RO	Base address offset	Depends on the configuration
Reserved	9:4	-	-	-
Size	3:0	RO	Size of the address space: 0x0: invalid slave 0x1: 1K 0x2: 2K 0x3: 4K 0x4: 8K 0x5: 16K 0x6: 32K 0x7: 64K 0x8: 128K 0x9: 256K 0xA: 512K 0xB: 1M 0xC: 2M 0xD: 4M 0xE: 8M 0xF: Reserved	Depends on the configuration

Table 30-5 Base/Size Register of Slave 1~n in the 32-bit Mode

Name	Bit	Type	Description	Reset
Reserved	31:M	-	M = ATCAPBDEC100_ADDR_DECODE_WIDTH	-
Base	(M-1):20	RO	Base address offset	Depends on the configuration
Reserved	19:4	-	-	-
Size	3:0	RO	Size of the address space: 0x0: invalid slave 0x1: 1M 0x2: 2M 0x3: 4M 0x4: 8M 0x5: 16M 0x6: 32M 0x7: 64M 0x8: 128M 0x9: 256M 0xA: 512M 0xB: 1G 0xC: 2G 0xD~0xF: Reserved	Depends on the configuration

30.4 Hardware Configuration Options

30.4.1 Hardware Configuration Options

Define the following macro to set the address width to 24-bit. Default address width is 32-bit.

`define ATCAPBDEC100_ADDR_WIDTH_24

30.4.2 Address Decode Width

Define ATCAPBDEC100_ADDR_DECODE_WIDTH to identify the partial address bits for APB address decoding. The APB address decoding logic only decodes the partial address bits received from the APB master. This value also reflects the total address space size of the APB bus. For the 24-bit mode, this macro could be 11~24; for the 32-bit mode, this macro could be 21~32. For example,

`define ATCAPBDEC100_ADDR_DECODE_WIDTH 16

In this case, the address decoding logic only decodes the address bit

[15:0] for selecting the target APB slave.

30.4.3 APB Slave Ports

Define ATCAPBDEC100_SLV_n to enable the nth APB slave port, where n=1~31. For example,

`define ATCAPBDEC100_SLV_1

`define ATCAPBDEC100_SLV_2

30.4.4 APB Slave Size

Define ATCAPBDEC100_SLVn_SIZE (n=1~31) to identify the address space size of the nth APB slave. The values are defined in Table 29-4 and Table 29-5 for the 24-bit mode and the 32-bit mode, respectively. For example, specify the address space size of APB slave #1 to be 4KB in the 24-bit mode:

`define ATCAPBDEC100_SLV1_SIZE 3

30.4.5 APB Slave Base Address Offset

Define ATCAPBDEC100_SLVn_OFFSET (n=1~31) to identify the base address offset of the nth APB slave. The value of the offset must be slave-size aligned and it should be a Verilog hexadecimal constant number with bit width specified. The bit width should be equal to the Address Decode Width, which reflects the total address space size of the APB bus (see Address Decode Width).

For example, suppose the size of the entire APB bus address space is 64KB and the size of the APB slave 1 is 4KB. The offset value must be integral multiple of 4K (e.g., 0x1000), and the width of the offset value should be $\log_2(64K) = 16$. Thus, the definition for specifying the address offset of the APB slave 1 at 0x1000 is:

`define ATCAPBDEC100_SLV1_OFFSET 16'h1000

31 ATCDMAC100

31.1 Introduction

The ATCDMAC100 is a direct memory access controller which transfers regions of data efficiently on bus.

31.1.1 Features

- Compliant with AMBA 2 AHB protocol specification
- Support of up to 8 DMA channels
- Support of up to 32 request/acknowledge pairs for hardware handshaking
- Support of round-robin arbitration with 2 priority levels
- Support of 8/16/32-bit wide data transfer
- Support of 24/32-bit address bus
- Support of chain transfer

31.1.2 Block Diagram

Figure 33-1 shows the block diagram of ATCDMAC100, which contains one AHB master interface for data transfer and one AHB slave interface for register programming.

Figure 31-1 ATCDMAC100 Block Diagram
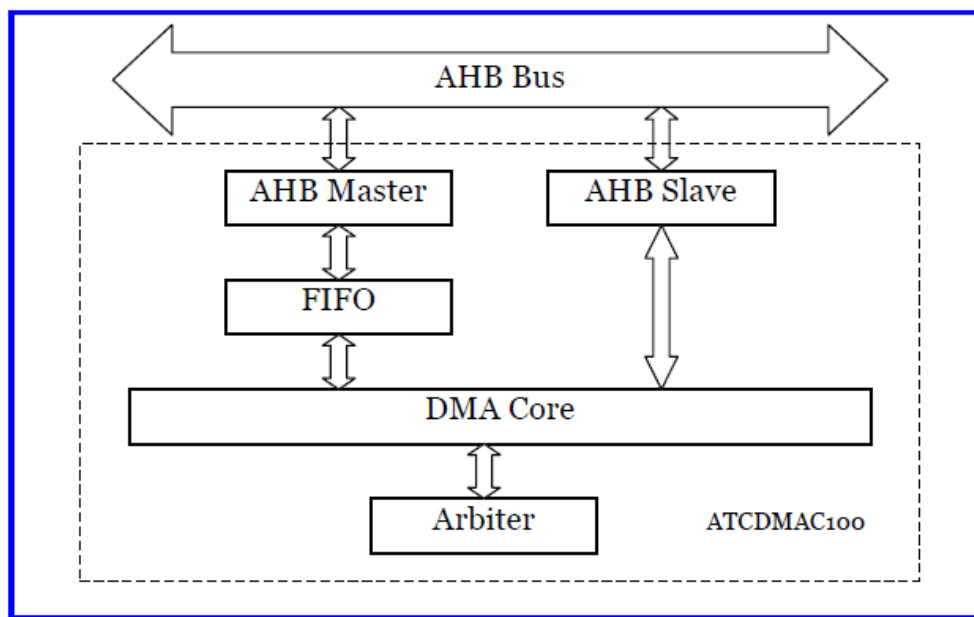
31.1.3 Function Description

ATCDMAC100 supports up to 8 DMA channels. Each DMA channel provides a set of registers to describe the intended data transfers. Multiple DMA channels can be enabled concurrently, but the DMA controller services one channel at a time.

Figure 31-2 Example of DMA Data Transfers shows an illustration of data transfer timing for a channel. To prevent channels from being starved, the DMA controller services all ready-channels alternatively, performing at most SrcBurstSize data transfers each time. Consequently, the data transfers of a channel may be split into several chunks when the total transfer size (TranSize) is larger than the source burst size (SrcBurstSize). When the overall data transfers of a channel complete, the DMA controller will update the interrupt status register, IntStatus, and assert the dma_int interrupt signal if the terminal count interrupt is enabled.

The data transfers of a channel will be stopped when an error occurs. The data transfers of a channel can also be aborted by software. In either case, the DMA controller will disable the channel, and assert dma_int if the corresponding interrupt is enabled.

Figure 31-2 Example of DMA Data Transfers
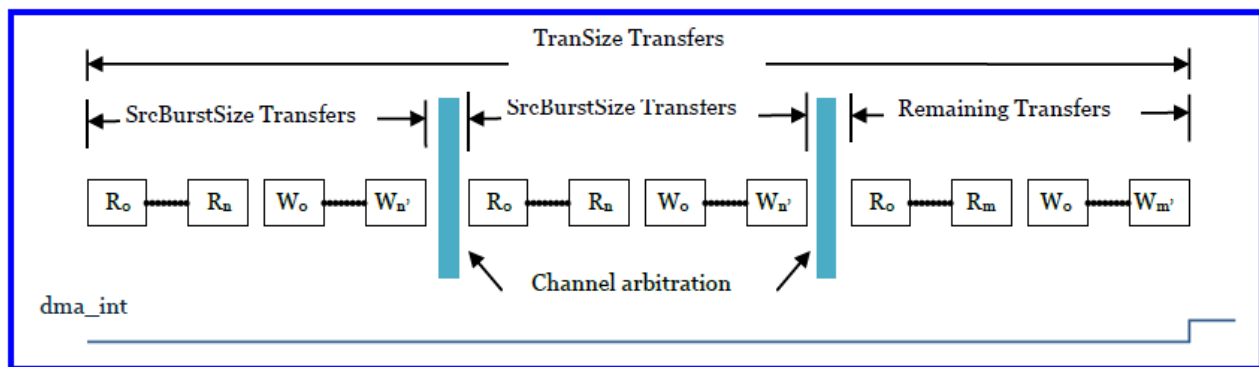
31.1.4 Channel Arbitration

ATCDMAC100 provides two priority levels for channel arbitration. Every channel is associated with a priority level by the Priority field of the channel control register, ChnCtrl. During the channel arbitration, the DMA controller selects a high priority channel first. A low priority channel is only selected if there is no high priority channel. Channels of the same priority level will be selected by the round-robin scheme.

31.1.5 Hardware Handshaking

ATCDMAC100 provides up to 32 pairs of hardware handshake signals (dma_req/dma_ack) for data transfers with low-speed devices. Figure 31-3 Example of Hardware Handshaking gives an example of hardware handshaking. The device should assert dma_req only when it prepares enough data to transfer or when it has enough empty space to receive the incoming data. The DMA controller only issues bus requests to read/write the data when it sees the dma_req asserted, avoiding holding the bus in the wait state indefinitely. The DMA controller asserts dma_ack when it completes SrcBurstSize data transfers from/to the device. The device should de-assert dma_req after detecting the assertion of dma_ack. The DMA controller should de-assert dma_ack after detecting the de-assertion of dma_req. If an error is encountered during the data transfers, the DMA controller will disable the channel without asserting dma_ack. The error handling software should reset both the source and destination of the DMA transfer to deassert dma_req.

Figure 31-3 Example of Hardware Handshaking
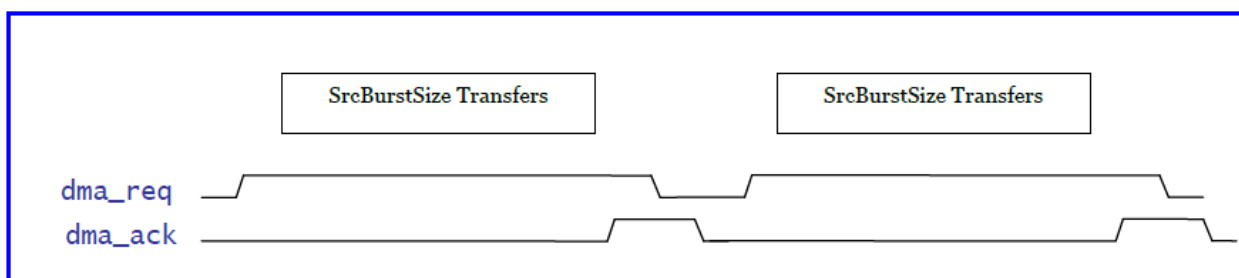
31.1.6 Chain Transfer

ATCDMAC100 provides the chain transfer function, with which multiple blocks of data can be transferred consecutively without the intervention of the main processor.

Before a chain transfer is started, a linked list structure must be built to describe the data blocks to move and the associated control setups. The first element of the list (the head of the list) is described by the channel control registers. The rest of elements of the list are specified by the linked list descriptors stored in the memory, where the linked list descriptor holds the control values to load to the channel control registers to continue the data transfer. Figure 31-4 Linked List Structure for Chain Transfers shows an example of the linked list structure.

When the channel is enabled, the DMA controller will first transfer data according to the channel control registers. After the data transfer completes, the DMA controller will continue the data transfer by following the ChnLLPointer. The content of the linked list descriptor pointed by ChnLLPointer will be loaded to the channel control registers if ChnLLPointer is not zero. The loaded descriptor becomes the new head of the list and this process repeats until the ChnLLPointer is zero.

Figure 31-4 Linked List Structure for Chain Transfers
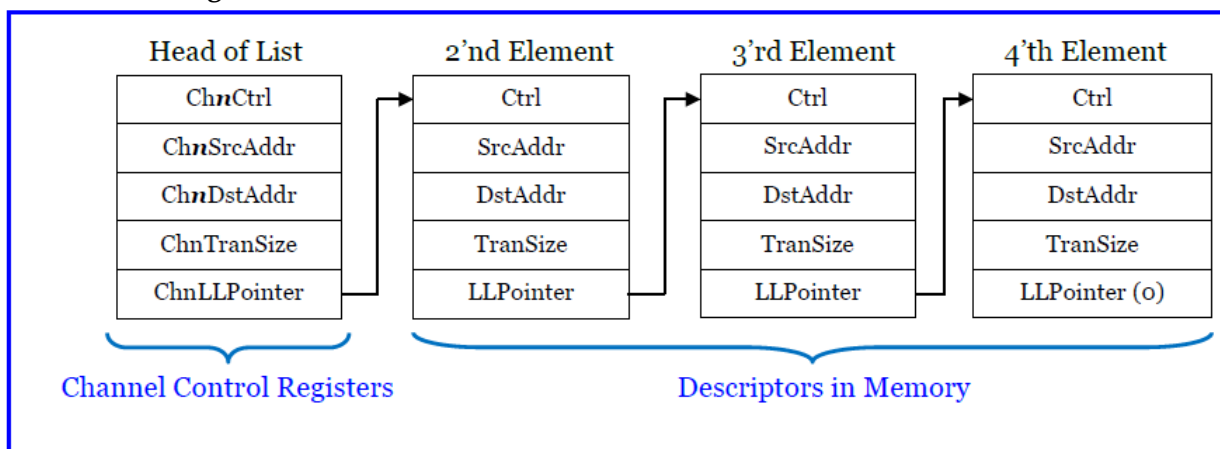
When the terminal count interrupt (IntTCMask) of a channel is enabled, the DMA controller will generate an interrupt and disable the channel when the data transfer for the head of the list isdone. If the ChnLLPointer is not zero, the channel control registers will be preloaded with the next descriptor before the interrupt is generated. The interrupt handling software could resume the chain transfer by just re-enabling the channel.

Table 31-1 Format of Linked List Descriptor shows the format of the linked list descriptor. The bit field definition of each descriptor word is the same as the corresponding channel control register except the channel enable bit, which is reserved in the linked list descriptor.

Table 31-1 Format of Linked List Descriptor

Name	Offset	Description	Format
Ctrl	0x00	Channel control	See Table 10
SrcAddr	0x04	Source address	See Table 11
DstAddr	0x08	Destination address	See Table 12
TranSize	0x0C	Total transfer size	See Table 13
LLPointer	0x10	Linked list pointer	See Table 14

31.1.7 Data Order

ATCDMAC100 provides three address control modes: increment mode, decrement mode, and fixed mode. At the increment mode, the address is increased after the DMA controller accesses a data of the source/destination. At the decrement mode, the address is decreased after the DMA controller accesses a data of the source/destination. At the fixed mode, the address remains unchanged after the DMA controller accesses a data of the source/destination.

When the address control mode of the source is the same as that of the destination, the DMA controller maintains the same byte order of the data between the source and the destination. When the address control mode of the source is opposite to that of the destination, the data written to the destination will be in the reverse byte order of that read from the source. The data order of the fixed mode is treated the same as that of the increment mode.

Figure 31-5 Data Order at the Destination when the Source Address Mode is the Increment Mode, Figure 31-6 Data Order at the Destination when the Source Address Mode is the Decrement Mode and

Figure 31-7 Data Order at the Destination when the Source Address Mode is
the Fixed Mode illustrate the byte order of the data at the destination when
the source address mode is increment, decrement, and fixed respectively.

Figure 31-5 Data Order at the Destination when the Source Address Mode is the Increment Mode
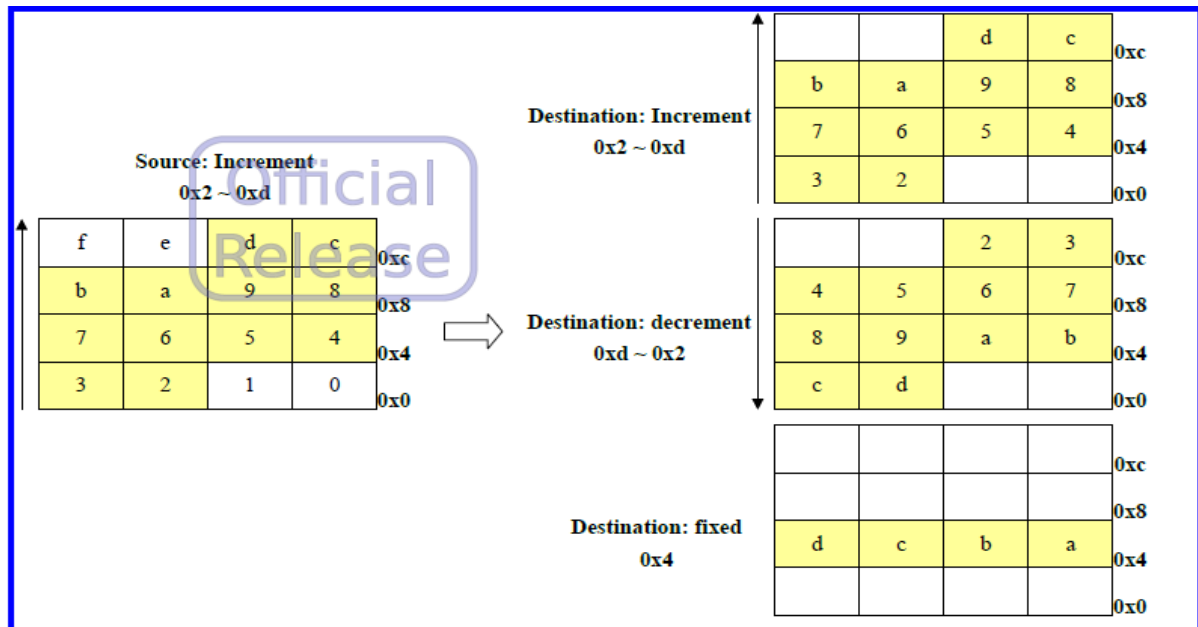
Figure 31-6 Data Order at the Destination when the Source Address Mode is the Decrement Mode
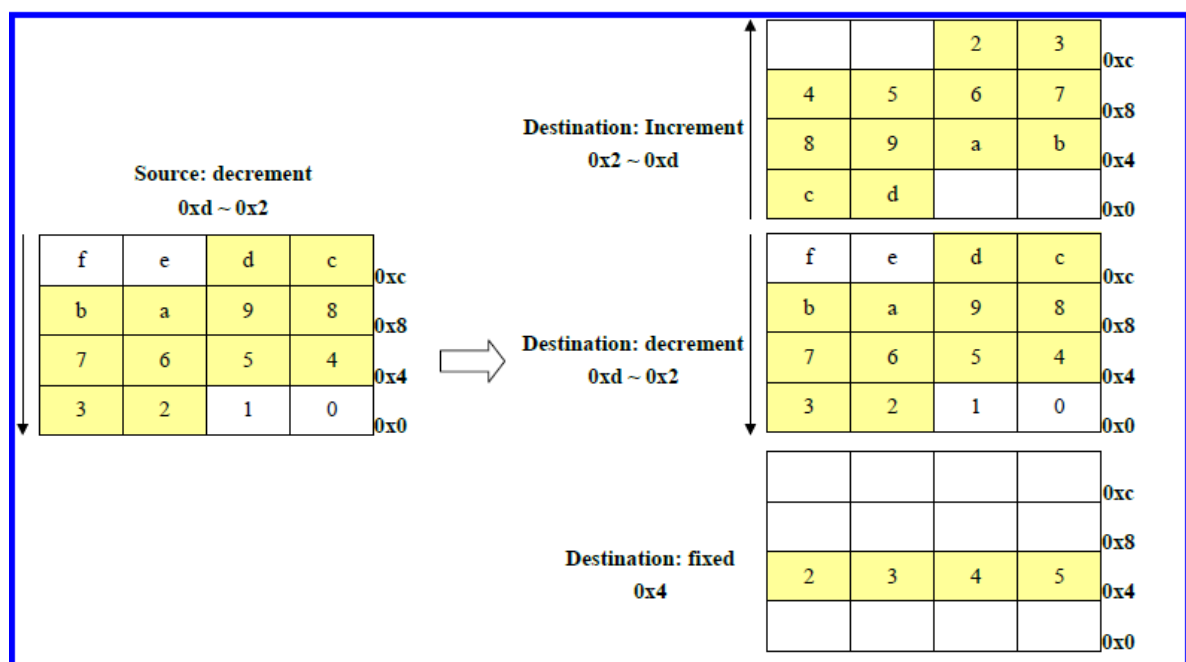
Figure 31-7 Data Order at the Destination when the Source Address Mode is the Fixed Mode
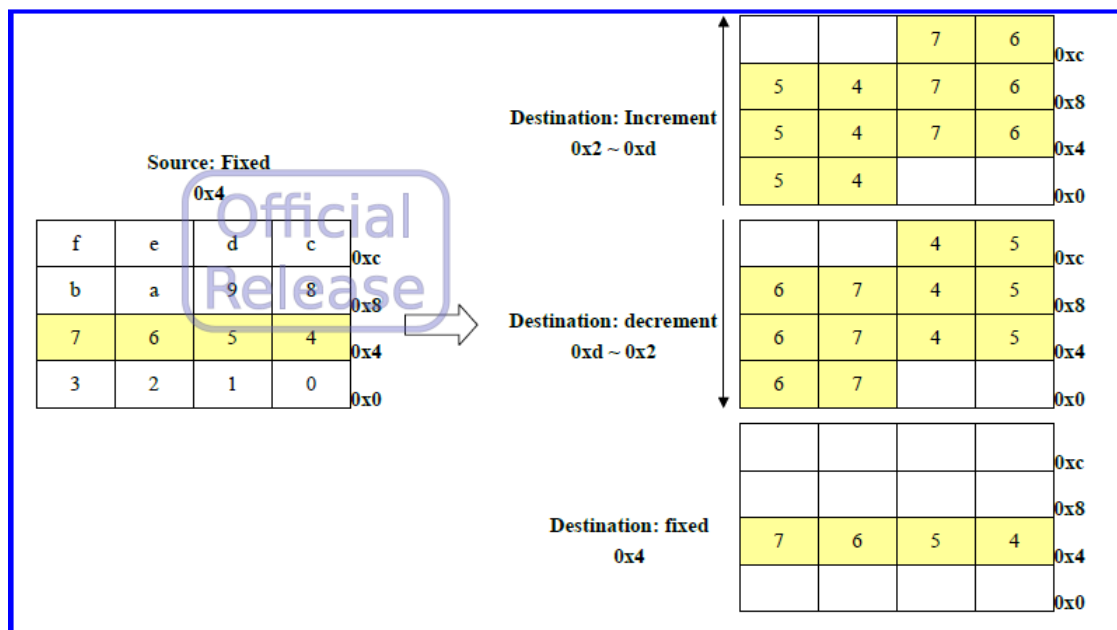
31.2 Signal Description

Figure 31-8 ATCDMAC100 Interfaces shows the interfaces of the ATCDMAC100.

Figure 31-8 ATCDMAC100 Interfaces
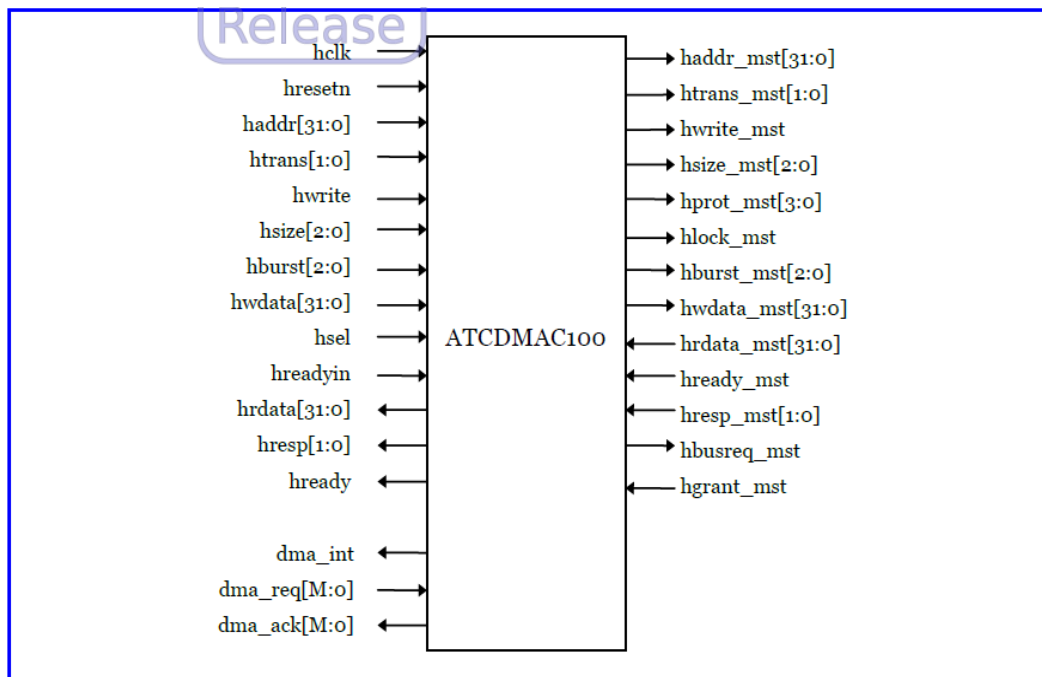
Table 30-2 gives the detailed descriptions of ATCDMAC100 I/O signals.

Table 31-2 ATCDMAC100 Signal Definition

Signal Name	I/O Type	Description
AHB global signals		
hclk	I	System bus clock
hresetn	I	System bus reset
DMA signals		
dma_int	O	Interrupt request
dma_req[M:0]	I	Burst transfer request (M is `ATCDMAC100_REQ_ACK_NUM – 1)
dma_ack[M:0]	O	Burst transfer acknowledge
AHB slave signals		
haddr[N:0]	I	AHB address bus (N is 23 and 31 for AHB 24-bit and 32-bit address, respectively)
htrans[1:0]	I	AHB transfer type
hwrite	I	AHB write signal
hsize[2:0]	I	AHB transfer size
hburst[2:0]	I	AHB burst type
hwdata[31:0]	I	AHB write data bus
hsel	I	AHB slave select signal
hreadyin	I	AHB ready input
hrdata[31:0]	O	AHB read data bus
hresp[1:0]	O	AHB transfer response
hready	O	AHB ready output
AHB master signals		
haddr_mst[N:0]	O	AHB address bus
htrans_mst[1:0]	O	AHB transfer type

Signal Name	I/O Type	Description
hwrite_mst	O	AHB write signal
hsize_mst[2:0]	O	AHB transfer size
hprot_mst[3:0]	O	AHB protection control
hlock_mst	O	AHB lock request
hburst_mst[2:0]	O	AHB burst type
hwdata_mst[31:0]	O	AHB write data bus
hrdata_mst[31:0]	I	AHB read data bus
hresp_mst[1:0]	I	AHB transfer response
hready_mst	I	AHB data ready input
hbusreq_mst	O	AHB bus request signal
hgrant_mst	I	AHB bus grant signal

31.3 Programming Model

31.3.1 Register Summary

ATCDMAC100 Register Summary shows a summary of the ATCDMAC100 registers.

Table 31-3 ATCDMAC100 Register Summary

Offset	Name	Description
ID and revision register		
+0x00	IdRev	ID and revision register
+0x04~0x0C	-	Reserved
Configuration register		
+0x10	DMACfg	DMAC configuration register
+0x14~0x1C	-	Reserved
Global control registers		
+0x20	DMACtrl	DMAC control register
+0x24~0x2C	-	Reserved
Channel status register		
+0x30	IntStatus	Interrupt status register
+0x34	ChEN	Channel enable register
+0x38~0x3c	-	Reserved
Channel control registers		
+0x40	ChAbort	Channel abort register
+0x44 + n*0x14	ChnCtrl	Channel n control register
+0x48 + n*0x14	ChnSrcAddr	Channel n source address register
+0x4c + n*0x14	ChnDstAddr	Channel n destination address register
+0x50 + n*0x14	ChnTranSize	Channel n transfer size register
+0x54 + n*0x14	ChnLLPointer	Channel n linked list pointer register

31.3.2 Register Description

The following sections describe ATCDMAC100 registers in detail. The abbreviations for the Type column are summarized below:

RO: Read only

WO: Write only

R/W: Readable and writable

R/W1C: Readable and write 1 to clear

31.3.3 ID and Revision Register (Offset 0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

<div align="center">

Table 31-4 ID and Revision Register

</div>

Name	Bit	Type	Description	Reset
ID	31:12	RO	ID number for DMAC	0x01021
RevMajor	11:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

31.3.4 DMAC Configuration Register (Offset 0x10)

<div align="center">

Table 31-5 DMAC Configuration Register

</div>

Name	Bit	Type	Description	Reset
ChainXfr	31	RO	Chain transfer 0x0: chain transfer is not configured 0x1: chain transfer is configured	Configuration dependent
ReqSync	30	RO	DMA request synchronization. The DMA request synchronization should be configured to avoid signal integrity problems when the request signal is not clocked by the system bus clock, which the DMA control logic operates in. If the request synchronization is not configured, the request signal is sampled directly without synchronization. 0x0: request synchronization is not configured 0x1: request synchronization is configured	Configuration dependent
Reserved	29:16	-	-	-
ReqNum	15:10	RO	Request/acknowledge number	Configuration dependent
FIFODepth	9:4	RO	FIFO depth	Configuration dependent
ChannelNum	3:0	RO	Channel number	Configuration dependent

31.3.5 DMAC Control Register (Offset 0x20)

Table 31-6 DMAC Control Register

Name	Bit	Type	Description	Reset
Reserved	31:1	-	-	-
Reset	0	WO	Software reset control. Set this bit to 1 to reset the DMA core and disable all channels.	0x0

31.3.6 Interrupt Status Register (Offset 0x30)

This register contains the terminal count, error, and abort status. The terminal count status of a channel is asserted when the channel encounters the terminal counter event. The error/abort status of a channel is asserted when the channel encounters the error/abort event. There is one bit of status for each channel and the status bit is zero when the corresponding channel is not configured.

Table 31-7 Interrupt Status Register

Name	Bit	Type	Description	Reset
Reserved	31:24	-	-	-
TC	23:16	R/W1C	The terminal count status of DMA channels, one bit per channel. The terminal count status is asserted when a channel transfer finishes without abort or error event. 0x0: channel N has no terminal count status 0x1: channel N has terminal count status	0x0
Abort	15:8	R/W1C	The abort status of channel, one bit per channel. The abort status is asserted when a channel transfer is aborted. 0x0: channel N has no abort status 0x1: channel N has abort status	0x0

Name	Bit	Type	Description	Reset
Error	7:0	R/W1C	The error status, one bit per channel. The error status is asserted when a channel transfer encounters the following error events: • Bus error • Unaligned address • Unaligned transfer width • Reserved configuration 0x0: channel N has no error status 0x1: channel N has error status	0x0

31.3.7 Channel Enable Register (Offset 0x34)

The register shows the DMA channel enable status. The status fields only exist when the corresponding channels are configured. This register is an alias of the Enable fields of all ChnCtrl registers.

Table 31-8 Channel Enable Register

Name	Bit	Type	Description	Reset
ChEN	N:0	RO	Alias of the Enable field of all ChnCtrl registers	0x0

31.3.8 Channel Abort Register (Offset 0x40)

The register controls the abortion of the DMA channel transfers, one-bit per channel. Write 1 to stop the current transfer of the corresponding channel. The abort bit is automatically cleared by hardware when the corresponding status bit in the interrupt status register is cleared.

Table 31-9 Channel Abort Register

Name	Bit	Type	Description	Reset
ChAbort	N:0	WO	Write 1 to this field to stop the channel transfer. The bits can only be set when the corresponding channels are enabled. Otherwise, the writes will be ignored for channels that are not enabled.	0x0

31.3.9 Channel n Control Register (Offset 0x44+n*0x14)

Table 31-10 Channel n Control Register

Name	Bit	Type	Description	Reset
SrcReqSelB5	31	R/W	Bit 5 of the source DMA request selection	0x0
DstReqSelB5	30	R/W	Bit 5 of the destination DMA request selection	0x0
Priority	29	R/W	Channel priority level 0x0: lower priority 0x1: higher priority	0x0
Reserved	28:25	-	-	-
SrcBurstSize	24:22	R/W	Source burst size. This field indicates the number of transfers before DMA channel re-arbitration. Total byte of a burst is SrcBurstSize * SrcWidth. 0x0: 1 transfer 0x1: 2 transfers 0x2: 4 transfers 0x3: 8 transfers 0x4: 16 transfers 0x5: 32 transfers 0x6: 64 transfers 0x7: 128 transfers	0x0

Name	Bit	Type	Description	Reset
SrcWidth	21:20	R/W	Source transfer width 0x0: byte transfer 0x1: half-word transfer 0x2: word transfer 0x3: reserved, setting the field with this value triggers error exception	0x2
DstWidth	19:18	R/W	Destination transfer width. Both the total transfer byte and the total burst bytes should be aligned to the destination transfer width; otherwise the error event will be triggered. For example, destination transfer width should be set as byte transfer if total transfer byte is not aligned to word or half-word. See SrcBurstSize field above for the definition of total burst byte and section 3.12 for the definition of the total transfer bytes. 0x0: byte transfer 0x1: half-word transfer 0x2: word transfer 0x3: reserved, set the field as this value triggers error exception	0x2
SrcMode	17	R/W	Source DMA handshake mode 0x0: normal mode 0x1: handshake mode	0x0
DstMode	16	R/W	Destination DMA handshake mode 0x0: normal mode 0x1: handshake mode	0x0

Name	Bit	Type	Description	Reset
SrcAddrCtrl	15:14	R/W	Source address control 0x0: increment address 0x1: decrement address 0x2: fixed address 0x3: reserved, setting the field with this value triggers the error exception	0x0
DstAddrCtrl	13:12	R/W	Destination address control 0x0: increment address 0x1: decrement address 0x2: fixed address 0x3: reserved, setting the field with this value triggers the error exception	0x0
SrcReqSel	11:8	R/W	Bits 4~0 of the source DMA request selection. {SrcReqSelB5, SrcReqSel} select the request/ack handshake pair that the source device is connected to.	0x0
DstReqSel	7:4	R/W	Bits 4~0 of the destination DMA request selection. {DstReqSelB5, DstReqSel} select the request/ack handshake pair that the destination device is connected to.	0x0
IntAbtMask	3	R/W	Channel abort interrupt mask 0x0: allow the abort interrupt to be triggered 0x1: disable the abort interrupt	0x0
IntErrMask	2	R/W	Channel error interrupt mask 0x0:allow the error interrupt to be triggered 0x1: disable the error interrupt	0x0
IntTCMask	1	R/W	Channel terminal count interrupt mask. 0x0: allow the terminal count interrupt to be triggerd 0x1: disable the terminal count interrupt	0x0
Enable	0	R/W	Channel enable bit 0x0: disable 0x1: enable	0x0

31.3.10 Channel n Source Address Register (Offset 0x48+n*0x14)

Table 31-11 Channel n Source Address Register

Name	Bit	Type	Description	Reset
SrcAddr	31:0	R/W	Source starting address. When a transfer completes, its value is updated to the ending address + sizeof(SrcWidth). This address must be aligned to the source transfer size; otherwise, an error event will be triggered.	0x0

31.3.11 Channel n Destination Address Register (Offset 0x4C+n*0x14)

Table 31-12 Channel n Destination Address Register

Name	Bit	Type	Description	Reset
DstAddr	31:0	R/W	Destination starting address. When a transfer completes, its value is updated to the ending address + sizeof(DstWidth). This address must be aligned to the destination transfer size; otherwise the error event will be triggered.	0x0

31.3.12 Channel n Transfer Size Register (Offset 0x50+n*0x14)

Table 31-13 Channel n Transfer Size Register

Name	Bit	Type	Description	Reset
Reserved	31:22	-	-	-
TranSize	21:0	R/W	Total transfer size from source. The total number of transferred bytes is TranSize * SrcWidth. The value is updated to zero when the DMA transfer is done. If a channel is enabled with zero total transfer size, the error event will be triggered and the transfer will be terminated.	0x0

31.3.13 Channel n Linked List Pointer Register (Offset 0x54+n*0x14)

Table 31-14 Channel Linked List Pointer Register

Name	Bit	Type	Description	Reset
LLPointer	31:2	R/W	Pointer to the next block descriptor. The pointer must be word aligned.	0x0
Reserved	1:0	-	-	-

31.4 Hardware Configuration Options

31.4.1 Number of DMA Channels

Define ATCDMAC100_CH_NUM_n to specify the number of DMA channels, where n=1~8.The following example configures ATCDMAC100 with 4 channels:

`define ATCDMAC100_CH_NUM_4

31.4.2 FIFO Size

Define ATCDMAC100_FIFO_DEPTH_n to specify the FIFO size as n entries (each entry is 32-bit). n could be 4, 8, 16, and 32. The following example configures the FIFO size as 8 entries:

`define ATCDMAC100_FIFO_DEPTH_8

31.4.3 DMA Request/Acknowledge Number

Define ATCDMAC100_REQ_ACK_NUM to specify the number of request/acknowledge pairs for hardware handshaking. The value could be 1 to 32. The following example configures ATCDMAC100 with 8 request/acknowledge pairs.

`define ATCDMAC100_REQ_ACK_NUM 8

31.4.4 DMA Request Synchronization Support

Define ATCDMAC100_REQ_SYNC_SUPPORT to add synchronizers at all DMA request input ports. Under this configuration, the DMA requests are allowed to be clocked by different clocks other than the system bus clock.

31.4.5 Chain Transfer Support

Define ATCDMAC100_CHAIN_TRANSFER_SUPPORT to support the

chain transfer.

31.4.6 Address Width

Define ATCDMAC100_ADDR_WIDTH_24 to set the address width to 24-bit. Default address width is 32-bit.

31.5 Programming Sequence

31.5.1 Transfer without Chain Transfer

The following sample programming sequence sets up channel 2 for

- Transfer of 32-words
- Source: 32-bit wide system memory
- Destination: fixed address, 32-bit wide device
- 32-tansfer burst
- Hardware handshaking is enabled. Source is connected to pair 1 and destination is connected to pair 2.
- All interrupts are enabled
- Check ChannelNum in the Configuration Register (0x10) bit[3:0] for the existence of channel 2.
- Set Channel Control Registers of channel 2 as Table 31-15 Register Setup Sample for Transfer without Chain Transfer. Ch2Ctrl should be programmed the last since the DMA channel becomes enabled once the register is programmed.
- Wait for the DMA interrupt and check the Interrupt Status Register (0x30).

The following example demonstrates DMA interrupt handling:

1. Check Interrupt Status Register (0x30) to determine the cause of the interruption.

2. Handle the interrupt accordingly.

3. Clear interrupt status by writing 1's to the Interrupt Status Register.

Table 31-15 Register Setup Sample for Transfer without Chain Transfer

Name	Offset	Value of the control register	
Ch2Ctrl	0x44+2*0x14	0x216B2121	
		Field	Value
		Priority	0x1
		SrcBurstSize	0x5
		SrcWidth	0x2
		DstWidth	0x2
		SrcMode	0x1
		DstMode	0x1
		SrcAddrCtrl	0x0
		DstAddrCtrl	0x2
		SrcReqSel	0x1
		DstReqSel	0x2
		IntAbtMask	0x0
		IntErrMask	0x0
		IntTCMask	0x0
		Enable	0x1
Ch2SrcAddr	0x48+2*0x14	The source address	
Ch2DstAddr	0x4c+2*0x14	The destination address	
Ch2TranSize	0x50+2*0x14	32	
Ch2LLPointer	0x54+2*0x14	0x0	

31.5.2 Chain Transfer

The following sample programming sequence sets up channel 0 for

- Transfer of 128 words split into four 32-word discontinuous data blocks
- Source: 32-bit wide memory
- Destination: fixed address, 32-bit wide device
- 32-transfer burst
- Hardware handshake is disabled
- All interrupts are enabled

1. Check ChannelNum in the Configuration Register (0x10) bit[3:0] for the existence of channel 0.

2. Create a linked list of block descriptors for the last three data blocks.

Each block descriptor describes the transfer for the respective 32-word data block.

3. Set Channel Control Registers according to Table 16 for the first data block and start the transfer. Ch0Ctrl should be programmed the last since the DMA channel becomes enabled once the register is programmed.

4. Wait for the DMA interrupt and check the Interrupt Status Register (0x30).

The following sequence demonstrates DMA interrupt handling for chain transfers:

5. Check Interrupt Status Register (0x30) to determine the cause of the interruption.

6. Handle the interrupt accordingly.

7. Clear interrupt status by writing 1's to the Interrupt Status Register.

8. Enable the channel if the interruption type is the terminal count interrupt and the chain transfer has more blocks to transfer (CnLLPointer !=0).

Table 31-16 Register Setup Sample for Transfer with Chain Transfer

Name	Offset	Value of the control register	
Ch0Ctrl	0x44+0*0x14	0x2168200F	
		Field	**Value**
		Priority	0x1
		SrcBurstSize	0x5
		SrcWidth	0x2
		DstWidth	0x2
		SrcMode	0x0
		DstMode	0x0
		SrcAddrCtrl	0x0
		DstAddrCtrl	0x2
		SrcReqSel	0x1
		DstReqSel	0x2
		IntAbtMask	0x1
		IntErrMask	0x1
		IntTCMask	0x1
		Enable	0x1
Ch0SrcAddr	0x48+0*0x14	The source address for the first data block	
Ch0DstAddr	0x4c+0*0x14	The destination address (fixed)	
Ch0TranSize	0x50+0*0x14	32	
Ch0LLPointer	0x54+0*0x14	Pointer to the subsequent link list descriptor in the memory	

32 ATCGPIO100

32.1 Introduction

32.1.1 Description

The ATCGPIO100 is a general purpose I/O (GPIO) controller which supports up to 32 channels with independently programmable input/output control.

32.1.2 Features

- Compliant with AMBA™ 2 APB protocol specification
- Up to 32 independent I/O channels
- Support of programmable pull-up/pull-down for each channel
- Support of programmable de-bounce function for each input channel
- Each input channel can be programmed as an interrupt input source
- The interrupt source could be level or edge triggered.

32.1.3 Block Diagram

Figure 32-1 ATCGPIO100 Block Diagram
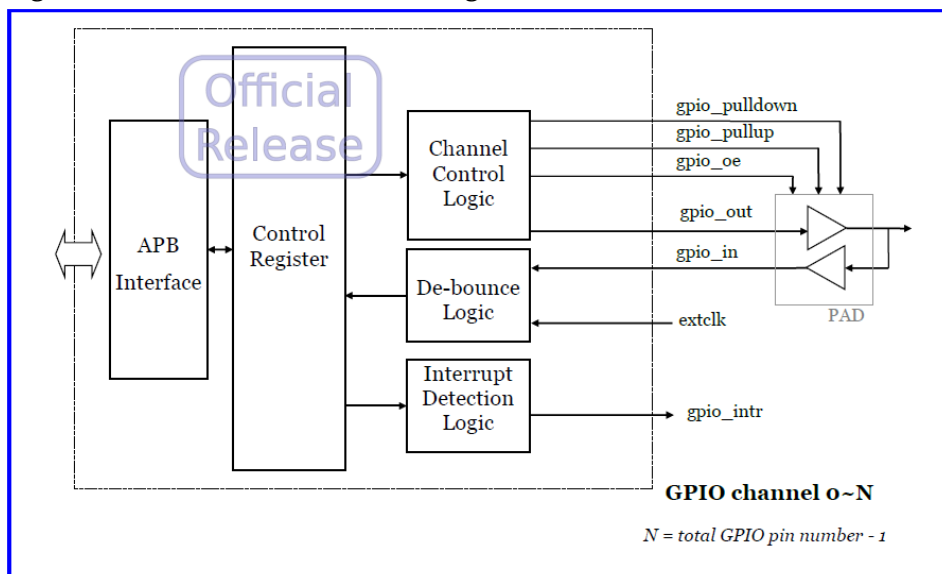
32.1.4 Function Description

The ATCGPIO100 controller provides up to 32 independent GPIO channels. Each channel can be individually programmed to be an input or an output port. As an input port, the controller samples the input signal and could generate an interrupt. As an output port, the controller could drive the output signal with pull-up/pull-down.

This controller also provides the de-bounce function, which filters out the glitches for input channels. The function can be enabled individually for each channel. The de-bounce duration is programmable as well.

32.2 Signal Description

Figure 32-2 ATCGPIO100 Pin Connection Diagram
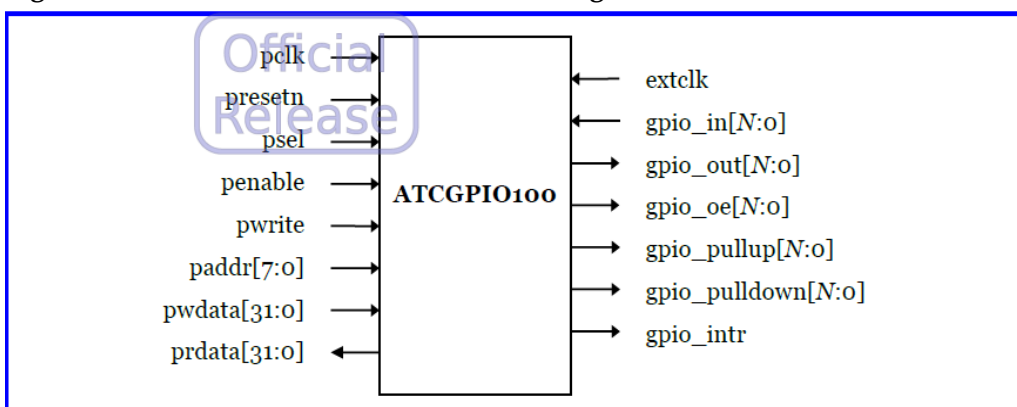
Table 32-1 ATCGPIO100 Pin Description

Name	Type	Source/Dest	Description
System signals			
extclk	Input	Clock generator	External de-bouncing clock source. The clock source must be at least 4x slower than pclk.
gpio_intr	Output	Interrupt controller	ATCGPIO100 interrupt signal
APB signals			
pclk	Input	APB	APB clock, used to time all the bus transfers
presetn	Input	APB	APB reset signal (Active-Low)
psel	Input	APB	APB select signal
			When set to 1, this signal indicates that the slave device has been selected by the APB bridge and that a data transfer is required.
penable	Input	APB	APB enable signal
pwrite	Input	APB	APB transfer direction signal

Name	Type	Source/Dest	Description
			This signal indicates a write access when high and a read access when low.
paddr[7:0]	Input	APB	APB address bus
prdata[31:0]	Output	APB	APB read data bus
pwdata[31:0]	Input	APB	APB write data bus
GPIO signals			
gpio_in[N:0]	Input	PAD	GPIO input data
gpio_oe[N:0]	Output	PAD	GPIO output data enable signal. Active high
gpio_out[N:0]	Output	PAD	GPIO output data
gpio_pullup[N:0]	Output	PAD	GPIO output pull-up enable signal
gpio_pulldown[N:0]	Output	PAD	GPIO output pull-down enable signal

Note!

* N = ATCGPIO100_GPIO_NUM – 1

32.3 Programming Model

32.3.1 Summary of Registers

Table 32-2 ATCGPIO100 Register Summary shows a summary of the ATCGPIO100 registers. Pull control registers, interrupt control registers and de-bounce control registers only exist when the corresponding option is configured.

Table 32-2 ATCGPIO100 Register Summary

Offset	Name	Description
ID and Revision Register		
+0x00	IdRev	ID and revision register
+0x04~0x0C	-	Reserved
Configuration Register		
+0x10	Cfg	Configuration register
+0x14~0x1C	-	Reserved
Channel Control Registers		
+0x20	DataIn	Channel data-in register
+0x24	DataOut	Channel data-out register
+0x28	ChannelDir	Channel direction register
+0x2C	DoutClear	Channel data-out clear register
+0x30	DoutSet	Channel data-out set register
+0x34~0x3C	-	Reserved
Pull Control Registers		
+0x40	PullEn	Pull enable register
+0x44	PullType	Pull type register
+0x48~0x4C	-	Reserved
Interrupt Control Registers		
+0x50	IntrEn	Interrupt enable register

Offset	Name	Description
+0x54	IntrMode0	Interrupt mode register (0~7)
+0x58	IntrMode1	Interrupt mode register (8~15)
+0x5C	IntrMode2	Interrupt mode register (16~23)
+0x60	IntrMode3	Interrupt mode register (24~31)
+0x64	IntrStatus	Interrupt status register
+0x68~0x6C	-	Reserved
De-bounce control registers		
+0x70	DeBounceEn	De-bounce enable register
+0x74	DeBounceCtrl	De-bounce control register
+0x78~0x7C	-	Reserved

32.3.2 Register Description

The abbreviations for the Type column are summarized below.

RO: read only

R/W: readable and writable

W1C: write 1 to clear

WO: write only (read as zero)

32.3.3 ID and Revision Register (Offset 0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

Table 32-3 and Revision Register

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for ATCGPIO100	0x020310
RevMajor	7:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

32.3.4 Configuration Register (Offset 0x10)

Table 32-4 Configuration Register

Name	Bit	Type	Description	Reset
Pull	31	RO	Pull option 0x0: Pull option is not configured 0x 1: Pull option is configured	Configuration dependent
Intr	30	RO	Interrupt option 0x 0: interrupt option is not configured 0x 1: interrupt option is configured	Configuration dependent
Debounce	29	RO	De-bounce option 0x 0: de-bounce option is not configured 0x 1: de-bounce option is configured	Configuration dependent
-	28:6	-	Reserved	-
ChannelNum	5:0	RO	Number of channels	Configuration dependent

32.3.5 Channel Data-In Register (Offset 0x20)

Table 32-5 Channel Data-In Register

Name	Bit	Type	Description	Reset
DataIn	N:0	RO	Channel data input register	0x0

32.3.6 Channel Data-Out Register (Offset 0x24)

Table 32-6 Channel Data-Out Register

Name	Bit	Type	Description	Reset
DataOut	N:0	R/W	GPIO data output	0x0

32.3.7 Channel Direction Register (Offset 0x28)

Table 32-7 Channel Direction Register

Name	Bit	Type	Description	Reset
ChannelDir	N:0	R/W	GPIO channel direction 0x0: input 0x1: output	0x0

32.3.8 Channel Data-Out Clear Register (Offset 0x2C)

Table 32-8 Channel Data-Out Clear Register

Name	Bit	Type	Description	Reset
DoutClear	N:0	WO	GPIO data-out clear; write 1 to clear the corresponding output channels.	0x0

32.3.9 Channel Data-Out Set Register (Offset 0x30)

Table 32-9 Channel Data-Out Set Register

Name	Bit	Type	Description	Reset
DoutSet	N:0	WO	GPIO data-out set; write 1 to set the corresponding output channels.	0x0

32.3.10 Pull Enable Register (Offset 0x40)

Table 32-10 Pull Enable Register

Name	Bit	Type	Description	Reset
PullEn	N:0	R/W	GPIO pull enable; write 1 to enable pull-up/pull-down of the corresponding channels.	0x0

32.3.11 Pull Type Register (Offset 0x44)

Table 32-11 Pull Type Register

Name	Bit	Type	Description	Reset
PullType	N:0	R/W	GPIO pull control 0x0: pull-up 0x1: pull-down	0x0

32.3.12 Interrupt Enable Register (Offset 0x50)

Table 32-12 Interrupt Enable Register

Name	Bit	Type	Description	Reset
IntEn	N:0	R/W	GPIO interrupt enable; write 1 to enable interrupts of the corresponding channels.	0x0

32.3.13 Interrupt Mode Register (Offset 0x54, 0x58, 0x5C, 0x60):

Table 32-13 Channel (0~7) Interrupt Mode Register (0x54) -

Table 32-16 Channel (24~31) Interrupt Mode Register (0x60) describe the interrupt trigger mode of the corresponding GPIO channel. See the Ch0IntrM field for the mode encodings. The field becomes reserved if the channel does not exist.

Table 32-13 Channel (0~7) Interrupt Mode Register (0x54)

Name	Bit	Type	Description	Reset
-	31	-	Reserved	-
Ch7IntrM	30:28	R/W	Channel 7 interrupt mode	0x0
-	27	-	Reserved	-
Ch6IntrM	26:24	R/W	Channel 6 interrupt mode	0x0
-	23	-	Reserved	-
Ch5IntrM	22:20	R/W	Channel 5 interrupt mode	0x0
-	19	-	Reserved	-
Ch4IntrM	18:16	R/W	Channel 4 interrupt mode	0x0
-	15	-	Reserved	-
Ch3IntrM	14:12	R/W	Channel 3 interrupt mode	0x0
-	11	-	Reserved	-
Ch2IntrM	10:8	R/W	Channel 2 interrupt mode	0x0
-	7	-	Reserved	-
Ch1IntrM	6:4	R/W	Channel 1 interrupt mode	0x0
-	3	-	Reserved	-
Ch0IntrM	2:0	R/W	Channel 0 interrupt mode 0x0: No operation 0x2: High-level 0x3: Low-level 0x5: Negative-edge 0x6: Positive-edge 0x7: Dual-edge 0x1, 0x4: Reserved	0x0

Table 32-14 Channel (8~15) Interrupt Mode Register (0x58)

Name	Bit	Type	Description	Reset
-	31	-	Reserved	-
Ch15IntrM	30:28	R/W	Channel 15 interrupt mode	0x0
-	27	-	Reserved	-
Ch14IntrM	26:24	R/W	Channel 14 interrupt mode	0x0
-	23	-	Reserved	-
Ch13IntrM	22:20	R/W	Channel 13 interrupt mode	0x0
-	19	-	Reserved	-
Ch12IntrM	18:16	R/W	Channel 12 interrupt mode	0x0
-	15	-	Reserved	-
Ch11IntrM	14:12	R/W	Channel 11 interrupt mode	0x0
-	11	-	Reserved	-
Ch10IntrM	10:8	R/W	Channel 10 interrupt mode	0x0
-	7	-	Reserved	-
Ch9IntrM	6:4	R/W	Channel 9 interrupt mode	0x0
-	3	-	Reserved	-
Ch8IntrM	2:0	R/W	Channel 8 interrupt mode	0x0

Table 32-15 Channel (16~23) Interrupt Mode Register (0x5C)

Name	Bit	Type	Description	Reset
-	31	-	Reserved	-
Ch23IntrM	30:28	R/W	Channel 23 interrupt mode	0x0
-	27	-	Reserved	-
Ch22IntrM	26:24	R/W	Channel 22 interrupt mode	0x0
-	23	-	Reserved	-
Ch21IntrM	22:20	R/W	Channel 21 interrupt mode	0x0
-	19	-	Reserved	-
Ch20IntrM	18:16	R/W	Channel 20 interrupt mode	0x0

Name	Bit	Type	Description	Reset
-	15	-	Reserved	-
Ch19IntrM	14:12	R/W	Channel 19 interrupt mode	0x0
-	11	-	Reserved	-
Ch18IntrM	10:8	R/W	GPIO channel 18 interrupt mode	0x0
-	7	-	Reserved	-
Ch17IntrM	6:4	R/W	Channel 17 interrupt mode	0x0
-	3	-	Reserved	-
Ch16IntrM	2:0	R/W	Channel 16 interrupt mode	0x0

Table 32-16 Channel (24~31) Interrupt Mode Register (0x60)

Name	Bit	Type	Description	Reset
-	31	-	Reserved	-
Ch31IntrM	30:28	R/W	Channel 31 interrupt mode	0x0
-	27	-	Reserved	-
Ch30IntrM	26:24	R/W	Channel 30 interrupt mode	0x0
-	23	-	Reserved	-
Ch29IntrM	22:20	R/W	Channel 29 interrupt mode	0x0
-	19	-	Reserved	-
Ch28IntrM	18:16	R/W	Channel 28 interrupt mode	0x0
-	15	-	Reserved	-
Ch27IntrM	14:12	R/W	Channel 27 interrupt mode	0x0
-	11	-	Reserved	-
Ch26IntrM	10:8	R/W	GPIO channel 26 interrupt mode	0x0
-	7	-	Reserved	-
Ch25IntrM	6:4	R/W	Channel 25 interrupt mode	0x0
-	3	-	Reserved	-
Ch24IntrM	2:0	R/W	Channel 24 interrupt mode	0x0

32.3.14 Channel Interrupt Status Register (Offset 0x64):

Table 32-17 Channel Interrupt Status Register

Name	Bit	Type	Description	Reset
IntrStatus	N:0	R/W1C	The interrupt status of the corresponding channel; write 1 to clear. 0x0: No interrupt 0x1: Interrupt	0x0

De-bounce Enable Register (Offset 0x70):

Table 32-18 De-bounce Enable Register

Name	Bit	Type	Description	Reset
DeBounceEn	N:0	R/W	Data-in de-bounce enable; write 1 to enable de-bouncing of the corresponding channels.	0x0

De-bounce Control Register (Offset 0x74):

Table 32-19 De-bounce Control Register

Name	Bit	Type	Description	Reset
DBClkSel	31	R/W	GPIO de-bounce clock source selection. Select pclk (the faster clock) as the de-bounce clock source would shorten the de-bounce latency. 0x0: extclk 0x1: pclk	0x0
-	30:8	-	Reserved	-
DBPreScale	7:0	R/W	GPIO pre-scale base, to scale the de-bounce clock source before it is used as the actual de-bounce clock. The de-bounce period would be multiplied by (DBPreScale+1); e.g., setting DBPreScale to 3 would filter out pulses which are less than 4 de-bounce clock period.	0x0

32.4 Hardware Configuration Options

32.4.1 GPIO Channel Number

Define ATCGPIO100_GPIO_NUM to specify the number of GPIO channels. The value ranges from 1 to 32. The following shows an example to define 16 GPIO channels.

`define ATCGPIO100_GPIO_NUM 16

32.4.2 Pull Support

Define ATCGPIO100_PULL_SUPPORT to support the GPIO pull-up/pull-down.

32.4.3 Interrupt

Define ATCGPIO100_INTR_SUPPORT to support the interrupt.

32.4.4 De-bounce

Define ATCGPIO100_DEBOUNCE_SUPPORT to support the de-bounce function.

32.5 Programming Sequence

32.5.1 Output Channel Programming

The following shows an example to program channel 0 as an output port.

- Set the channel direction register (0x28) bit[0] to 1.
- To drive 1 as the output value:
- Set the channel data-out register (0x24) bit[0] to 1, or
- Set the channel data-out set register (0x30) bit[0] to 1.
- To drive 0 as the output value:
- Set the channel data-out register (0x24) bit[0] to 0, or
- Set the channel data-out clear register (0x2c) bit[0] to 1.

32.5.2 Input Channel Programming

The following shows an example to program channel 0 as an input port.

32.5.3 Basic Input Channel

- Set the channel direction register (0x28) bit[0] to 0.
- Read the channel data-in register (0x20) bit[0] to get the input value.

32.5.4 Input Channel with Additional Functions

- Check the configuration register for pull, de-bounce, and interrupt support.
- Set the pull control registers if the pull function is supported.
- Set the pull type.
- – Set the pull type register (0x44) bit[0] to 0 to select the pull-up type, or
- – Set the pull type register (0x44) bit[0] to 1 to select the pull-down type.
- Set the pull enable register (0x40) bit[0] to 1 to enable pull.
- Set the de-bounce registers if the de-bounce function is supported.
- Set the de-bounce control register (0x74) bit[31] to 1 to select the external clock as the de-bounce clock source.
- Set the de-bounce control register (0x74) bit [7:0] to 0x3 to filter out the glitches which are shorter than 4 de-bounce clock periods.
- Set the de-bounce enable register (0x70) bit[0] to 1 to enable de-bounce.
- Set the interrupt registers if the interrupt function is supported.
- Set channel (0~7) interrupt mode register (ox54) bit [2:0]. See Channel n Linked List Pointer Register (Offset 0x54+n*0x14)for the interrupt mode setting.
- Set the interrupt enable register (0x50) bit[0] to 1 to enable interrupt.
- Set the channel direction register (0x28) bit[0] to 0.
- Check the interrupt status when the interrupt function is supported and enabled.
- Once a GPIO interrupt occurs, check the interrupt status register (ox64) bit[0], or
- Poll the interrupt status register (ox64) bit[0] directly.

33 ATCIIC100

33.1 Introduction

The ATCIIC100 controller is an I2C (Inter-Integrated Circuit) master/slave controller.

33.1.1 Features

- Supports AMBA 2.0 APB bus
- Supports Standard-mode (100 Kb/s), Fast-mode (400 Kb/s) and Fast-mode Plus (1 Mb/s) protocols
- Programmable Master/Slave mode
- Supports 7-bit and 10-bit addressing mode
- Supports general call address
- Auto clock stretching
- Programmable clock/data timing
- Supports direct memory access (DMA)

33.1.2 Block Diagram

Figure 33-1 ATCIIC100 Block Diagram
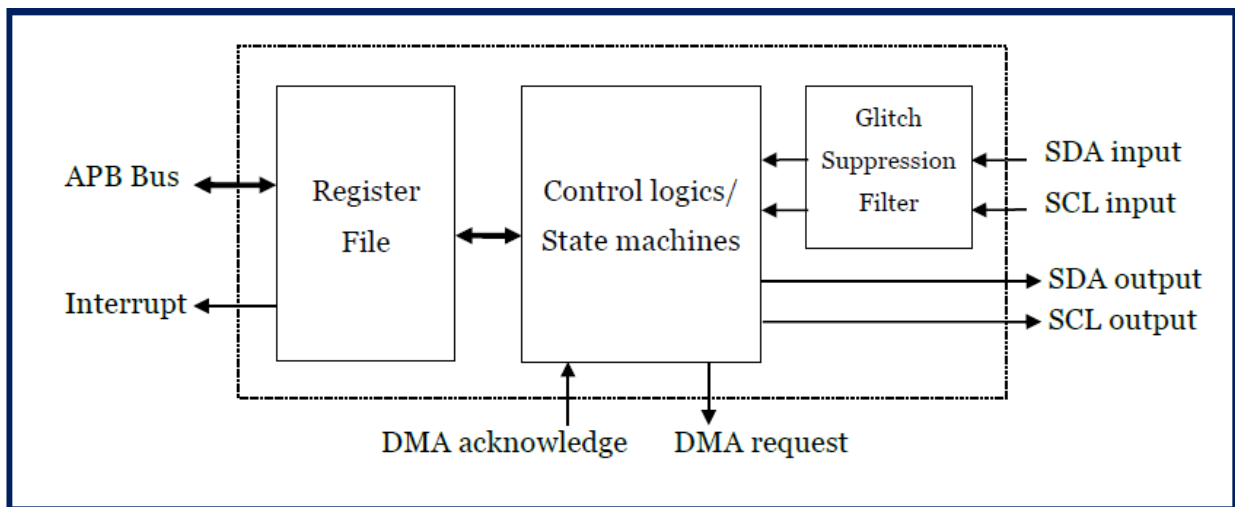
The ATCIIC100 controller can act as either an I^2C master device or an I^2C slave device, depending on the control register settings.

33.1.3 I²C Master

As an I^2C master, the controller provides an efficient way to initiate I^2C transactions. Every transaction can be delineated by four phases: Start, Address, Data and Stop. At the Start phase, a START condition is generated. At the Address phase, an address is sent. At the Data phase, one or more data bytes are transferred. At the Stop phase, a STOP condition is generated. The existence of each phase can be controlled independently.

33.1.4 I²C Slave

As an I^2C slave, the controller is addressed when the address byte of an I^2C transaction matches the Address Register. An Address Hit interrupt can be generated for the software to prepare for the subsequent operations.

33.1.5 General Call Address

The General Call Address is a special address to address all slave devices on the I^2C-bus. The ATCIIC100 controller at the slave mode will respond with an ACK to the general call address and set the GenCall field of the Status Register.

33.1.6 Auto Clock Stretch

The ATCIIC100 can automatically pause bus transactions by stretching clocks on the I^{2}C-bus when the software is not ready for the next byte of data or when the FIFO is full. Auto Clock Stretch is supported at both the master mode and the slave mode.

33.1.7 Auto-ACK

With Auto-ACK, the ATCIIC100 automatically generates proper acknowledgements for each byte received. Every received byte will be responded with an ACK, except for the last byte, which should be responded with a NACK according to the I^{2}C-bus protocol. On the other hand, if the software needs to determine each byte's acknowledgement status, Auto-ACK can be turned off by enabling the Byte Receive Interrupt.

33.1.8 Timing Parameter Multiplier

ATCIIC100 provides an I^{2}C Timing Parameter Multiplier Register (TPM), which enables ATCIIC100 to meet the I^{2}C-bus timing requirement while running under a relatively faster APB clock frequency. The actual values of all timing parameters in the Setup Register are multiplied by (TPM+1). Please see Timing Parameter Multiplierfor how to determine the multiplier value.

33.2 Signal Description

Table 33-1 ATCIIC100 Signal Descriptionshows the input/output (I/O) signal description of ATCIIC100.

Table 33-1 ATCIIC100 Signal Description

Name	I/O Type	Description
AMBA APB signals		
pclk	I	AMBA APB clock
presetn	I	AMBA APB reset signal; active low
psel	I	AMBA APB slave select signal from the APB decoder
penable	I	AMBA APB enable signal
pwrite	I	AMBA APB transfer direction signal
		This signal indicates a write access when driven as HIGH and a read access when driven as LOW
paddr[5:2]	I	AMBA APB address bus
prdata[31:0]	O	AMBA APB read data bus
pwdata[31:0]	I	AMBA APB write data bus
I²C controller signals		
i2c_int	O	Interrupt signal
i2c_req	O	DMA request
i2c_ack	I	DMA acknowledge
sda_o	O	I²C serial data output
scl_o	O	I²C serial clock output
sda_i	I	I²C serial data input
scl_i	I	I²C serial clock input

33.3 Programming Model

33.3.1 Summary of Registers

Table 33-2 ATCIIC100 Registers Summary shows the summary of ATCIIC100 registers.

<p style="text-align:center">Table 33-2 ATCIIC100 Registers Summary</p>

Offset	Name	Description
+0x00	IdRev	ID and Revision Register
+0x04 ~ +0x0C	-	Reserved
+0x10	Cfg	Configuration Register
+0x14	IntEn	Interrupt Enable Register
+0x18	Status	Status Register
+0x1C	Addr	Address Register
+0x20	Data	Data Register
+0x24	Ctrl	Control Register
+0x28	Cmd	Command Register
+0x2C	Setup	Setup Register
+0x30	TPM	I²C Timing Parameter Multiplier Register

The following sections describe ATCIIC100 registers in detail. Registers may have different behavior when the controller is at the master mode or the slave mode. The behaviors will be described separately.

The abbreviations for the Type column are summarized below:

RO: read only

WO: write only

R/W: readable and writable

R/W1C: readable and write 1 to clear

33.3.2 ID and Revision Register

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.

<p style="text-align:center">Table 33-3 ID and Revision Register</p>

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for ATCIIC100	0x020210
RevMajor	7:4	RO	Major revision number	Revision Dependent
RevMinor	3:0	RO	Minor revision number	Revision Dependent

33.3.3 Configuration Register

The Configuration Register keeps the size of data FIFO.

Table 33-4 Configuration Register

Name	Bit	Type	Description	Reset
Reserved	31:2	-	-	-
FIFOSize	1:0	RO	FIFO size: 0: 2 bytes 1: 4 bytes 2: 8 bytes 3: 16 bytes	Configuration Dependent

33.3.4 Interrupt Enable Register

The Interrupt Enable Register can enable or disable interrupts.

Table 33-5 Interrupt Enable Register

Name	Bit	Type	Description	Reset
Reserved	31:10	-	-	-
Cmpl	9	R/W	Set to enable the Completion Interrupt. Master: interrupts when a transaction is issued from this master and completed without losing the bus arbitration. Slave: interrupts when a transaction addressing the controller is completed.	0x0
ByteRecv	8	R/W	Set to enable the Byte Receive Interrupt. Interrupts when a byte of data is received Auto-ACK will be disabled if this interrupt is enabled, that is, the software needs to ACK/NACK the received byte manually.	0x0

Name	Bit	Type	Description	Reset
ByteTrans	7	R/W	Set to enable the Byte Transmit Interrupt. Interrupts when a byte of data is transmitted.	0x0
Start	6	R/W	Set to enable the START Condition Interrupt. Interrupts when a START condition/repeated START condition is detected.	0x0
Stop	5	R/W	Set to enable the STOP Condition Interrupt. Interrupts when a STOP condition is detected.	0x0
ArbLose	4	R/W	Set to enable the Arbitration Lose Interrupt. Master: interrupts when the controller loses the bus arbitration Slave: not available in this mode	0x0
AddrHit	3	R/W	Set to enable the Address Hit Interrupt. Master: interrupts when the addressed slave returned an ACK. Slave: interrupts when the controller is addressed.	0x0
FIFOHalf	2	R/W	Set to enable the FIFO Half Interrupt. Receiver: Interrupts when the FIFO is half-full, i.e. there is >= 1/2 entries in the FIFO. Transmitter: Interrupts when the FIFO is half-empty, i.e. there is <= 1/2 entries in the FIFO. This interrupt depends on the transaction direction; don't enable this interrupt unless the transfer direction is determined, otherwise unintended interrupts may be triggered.	0x0
FIFOFull	1	R/W	Set to enable the FIFO Full Interrupt. Interrupts when the FIFO is full.	0x0
FIFOEmpty	0	R/W	Set to enable the FIFO Empty Interrupt. Interrupts when the FIFO is empty.	0x0

33.3.5 Status Register

The Status Register keeps the interrupt status and I^2C-bus status.

Table 33-6 Status Register

Name	Bit	Type	Description	Reset
Reserved	31:15	-	-	-
LineSDA	14	RO	Indicates the current status of the SDA line on the bus. 1: High 0: Low	SDA line status
LineSCL	13	RO	Indicates the current status of the SCL line on the bus. 1: High 0: Low	SCL line status
GenCall	12	RO	Indicates that the address of the current transaction is a general call address. This status is only valid in slave mode. 1: General call 0: Not general call	0x0
BusBusy	11	RO	Indicates that the bus is busy. The bus is busy when a START condition is on bus and it ends when a STOP condition is seen on bus. 1: Busy 0: Not busy	0x0
ACK	10	RO	Indicates the type of the last received/transmitted acknowledgement bit. 1: ACK 0: NACK	0x0

Name	Bit	Type	Description	Reset
Cmpl	9	R/W1C	Transaction Completion Master: Indicates that a transaction has been issued from this master and completed without losing the bus arbitration. Slave: Indicates that a transaction addressing the controller has been completed. This status bit must be cleared to receive the next transaction; otherwise, the next incoming transaction will be blocked.	0x0
ByteRecv	8	R/W1C	Indicates that a byte of data has been received.	0x0
ByteTrans	7	R/W1C	Indicates that a byte of data has been transmitted.	0x0
Start	6	R/W1C	Indicates that a START Condition or a repeated START condition has been transmitted/received.	0x0
Stop	5	R/W1C	Indicates that a STOP Condition has been transmitted/received.	0x0
ArbLose	4	R/W1C	Indicates that the controller has lost the bus arbitration (master mode only).	0x0
AddrHit	3	R/W1C	Master: indicates that a slave has responded to the transaction. Slave: indicates that a transaction is targeting the controller (including the General Call).	0x0
FIFOHalf	2	RO	Transmitter: Indicates that the FIFO is half-empty. Receiver: Indicates that the FIFO is half-full.	0x0
FIFOFull	1	RO	Indicates that the FIFO is full.	0x0
FIFOEmpty	0	RO	Indicates that the FIFO is empty.	0x1

33.3.6 Address Register

The Address Register keeps the slave address. When programmed as a master, it is the target slave address for the next transaction. When programmed as a slave, it is the controller's address on the bus.

Table 33-7 Address Register

Name	Bit	Type	Description	Reset
Reserved	31:10	-	-	-
Addr	9:0	R/W	The slave address. For 7-bit addressing mode, the most significant 3 bits are ignored and only the least-significant 7 bits of Addr are valid.	0x0

33.3.7 Data Register

The Data Register is the data access port for the FIFO.

Table 33-8 Data Register

Name	Bit	Type	Description	Reset
Reserved	31:8	-	-	-
Data	7:0	R/W	Write this register to put one byte of data to the FIFO. Read this register to get one byte of data from the FIFO.	0x0

33.3.8 Control Register

The Control Register controls a transaction's phase choices and records the progress of Data phase.

Table 33-9 Control Register

Name	Bit	Type	Description	Reset
Reserved	31:13	-	-	-
Phase_start	12	R/W	Enable this bit to send a START condition at the beginning of transaction. Master mode only.	0x1
Phase_addr	11	R/W	Enable this bit to send the address after START condition. Master mode only.	0x1
Phase_data	10	R/W	Enable this bit to send the data after Address phase. Master mode only.	0x1
Phase_stop	9	R/W	Enable this bit to send a STOP condition at the end of a transaction. Master mode only.	0x1
Dir	8	R/W	Transaction direction Master: Set this bit to determine the direction for the next transaction. 0: Transmitter 1: Receiver Slave: The direction of the last received transaction. 0: Receiver 1: Transmitter	0x0
DataCnt	7:0	R/W	Data counts in bytes. Master: The number of bytes to transmit/receive. 0 means 256 bytes. DataCnt will be decreased by one for each byte transmitted/received. Slave: the meaning of DataCnt depends on the DMA mode: If DMA is not enabled, DataCnt is the number of bytes transmitted/received from the bus master. It is reset to 0 when the controller is addressed and then increased by one for each byte of data transmitted/received. If DMA is enabled, DataCnt is the number of bytes to transmit/receive. It will not be reset to 0 when the slave is addressed and it will be decreased by one for each byte of data transmitted/received.	0x0

33.3.9 Command Register

Table 33-10 Command Register

Name	Bit	Type	Description	Reset
Reserved	31:3	-	-	-
CMD	2:0	R/W	Write this register with the following values to perform the corresponding actions: 0x0: no action 0x1: issue a data transaction (Master only) 0x2: respond with an ACK to the received byte 0x3: respond with a NACK to the received byte 0x4: clear the FIFO 0x5: reset the I²C controller (abort current transaction, set the SDA and SCL line to the open-drain mode, reset the Status Register and the Interrupt Enable Register, and empty the FIFO) When issuing a data transaction by writing 0x1 to this register, the CMD field stays at 0x1 for the duration of the entire transaction, and it is only cleared to 0x0 after when the transaction has completed or when the controller loses the arbitration. Note: No transaction will be issued by the controller when all phases (Start, Address, Data and Stop) are disabled.	0x0

33.3.10 Setup Register

The Setup Register keeps the programmable configurations and the I²C-bus timing parameters.

For detail timing settings, see Timing Setup Guide Timing Setup Guide.

Table 33-11 Controller Setting Register

Name	Bit	Type	Description	Reset
Reserved	31:29	-	-	-
T_SUDAT	28:24	R/W	T_SUDAT defines the data setup time before releasing the SCL. Setup time = $(2 * t_{pclk}) + (2 + T\_SP + T\_SUDAT) * t_{pclk} * (TPM+1)$ t_{pclk} = PCLK period TPM = The multiplier value in Timing Parameter Multiplier Register	0x5
T_SP	23:21	R/W	T_SP defines the pulse width of spikes that must be suppressed by the input filter. Pulse width = $T\_SP * t_{pclk} * (TPM+1)$	0x1
T_HDDAT	20:16	R/W	T_HDDAT defines the data hold time after SCL goes LOW Hold time = $(2 * t_{pclk}) + (2 + T\_SP + T\_HDDAT) * t_{pclk} * (TPM+1)$	0x5

Name	Bit	Type	Description	Reset
Reserved	15:14	-	-	-
T_SCLRatio	13	R/W	The LOW period of the generated SCL clock is defined by the combination of T_SCLRatio and T_SCLHi values. When T_SCLRatio = 0, the LOW period is equal to HIGH period. When T_SCLRatio = 1, the LOW period is roughly two times of HIGH period. SCL LOW period = $(2 * t_{pclk}) + (2 + T\_SP + T\_SCLHi * ratio) * t_{pclk} * (TPM+1)$ 1: $ratio = 2$ 0: $ratio = 1$ This field is only valid when the controller is in the master mode.	0x1
T_SCLHi	12:4	R/W	The HIGH period of generated SCL clock is defined by T_SCLHi. SCL HIGH period = $(2 * t_{pclk}) + (2 + T\_SP + T\_SCLHi) * t_{pclk} * (TPM+1)$ The T_SCLHi value must be greater than T_SP and T_HDDAT values. This field is only valid when the controller is in the master mode.	0x10
DMAEn	3	R/W	Enable the direct memory access mode data transfer. 1: Enable 0: Disable	0x0
Master	2	R/W	Configure this device as a master or a slave. 1: Master mode 0: Slave mode	0x0
Addressing	1	R/W	I²C addressing mode: 1: 10-bit addressing mode 0: 7-bit addressing mode	0x0
IICEn	0	R/W	Enable the ATCIIC100 I²C controller. 1: Enable 0: Disable	0x0

33.3.11 Timing Parameter Multiplier Register

The Timing Parameter Multiplier Register holds a multiplier value that enlarges the I^{2}C-bus timing parameters defined in the Setup Register. The multiplier number helps the controller to meet timing requirements of the I^{2}C-bus interface when the controller is operating at a higher APB frequency. Please see Timing Parameter Multiplier for the procedures to compute its value.

Table 33-12 Timing Parameter Multiplier Register

Name	Bit	Type	Description	Reset
Reserved	31:5	-	-	-
TPM	4:0	R/W	A multiplication value for I2C timing parameters. All the timing parameters in the Setup Register are multiplied by (TPM+1).	0x0

33.4 RTL Configuration

33.4.1 Data FIFO Size

The ATCIIC100 controller uses a FIFO as buffer to the I^{2}C-bus. The data to be transmitted or received are stored in the FIFO. Define ATCIIC100_FIFO_DEPTH_n to configure an n-byte

FIFO, where n is the number 2, 4, 8 or 16.

33.5 Programming Sequence

33.5.1 Timing Setup Guide

Before enabling the ATCIIC100 controller, you must:

9. Setup the timing parameter multiplier by programming the Timing Parameter Multiplier Register

10. Setup the I^{2}C-bus timing parameters1 by programming the Setup Register

As an I^{2}C slave, the spike suppression width, the data setup time and the data hold time must be programmed properly according to the APB clock frequency and the speed of the I^{2}C-bus. As an I^{2}C master, the I^{2}C-bus clock frequency must be programmed as well.

By default, the I^{2}C-bus interface timings are specified by timing parameters in the Setup Register in units of APB clock cycles. Since the bit widths of timing parameters are limited, if the APB clock frequency is too high, the timing parameter settings may not meet the required I^{2}C-bus interface timing values. In this case, the Timing Parameter Multiplier Register could be set to enlarge the actual values of the timing parameters. It expands the effective range of timing parameters in the Setup Register and makes them fall into the valid parameter value ranges. The Timing Parameter Multiplier Register should be programmed while the I^{2}C bus is idle and ATCIIC100 is disabled (Setup.IICEn=0)

The following sub-sections show how to determine the Setup Register to meet the I^{2}C-bus timing parameters. All the examples assume that the APB clock frequency is 40MHz, i.e. the APB clock period is 25ns. If the APB clock frequency of your design is not 40MHz, please derive the register fields accordingly.

See NXP Semiconductors' "I^{2}C-bus specification and user manual" for details.

33.5.2 Spike Suppression Width

Table 33-13 Timing Parameters for Spike Suppression shows the pulse width of spikes that must be suppressed by the input filter. For the Fast-mode and the Fast-mode Plus, spikes less than 50ns must be suppressed (assuming TPM == 0). i.e.,

T_SP = 50ns / (25ns * (TPM + 1)) = 2

Table 33-13 Timing Parameters for Spike Suppression

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t$_{SP}$	Pulse width of spikes that must be suppressed by the input filter.	-	-	0	50	0	50	ns

33.5.3 Data Setup Time

Data setup time defines the time in which the SDA should be held steady before the SCL rising edge. Table 33-14 Timing Parameters for the Data Setup Time shows the timing parameters for the data setup time. The equation of data setup time shown in Table 11 is:

Setup time = (2 * tpclk) + (2 + T_SP + T_SUDAT) * tpclk * (TPM + 1)

For the Standard-mode (assuming TPM == 0),

250ns = 50ns + (2 + 2 + T_SUDAT) * 25ns

Then,

T_SUDAT = 4

For the other modes, T_SUDAT can be calculated similarly

Table 33-14 Timing Parameters for the Data Setup Time

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{SUDAT}	Data setup time	250	-	100	-	50	-	ns

33.5.4 Data Hold Time

Data hold time defines the time in which the SDA should be held steady after the SCL falling edge. Table 33-15 Timing Parameters for the Data Hold Time shows the timing parameters for the data hold time. The equation of data hold time shown in Table 33-11 Controller Setting Register is:

Hold time = $(2 * t_{pclk}) + (2 + T\_SP + T\_HDDAT) * t_{pclk} * (TPM + 1)$

For the Standard-mode (assuming TPM == 0),

300ns = 50ns + (4 + 2 + T_HDDAT) * 25ns

Then,

T_HDDAT = 4

For the other modes, T_HDDAT can be calculated similarly.

Table 33-15 Timing Parameters for the Data Hold Time

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{HDDAT}	Data hold time	300	-	300	-	0	-	ns

33.5.5 I²C-Bus Clock Frequency:

The I^{2}C-bus clock frequency is specified by the t_{HIGH} and t_{LOW} parameters, which are shown in Table 16 and can be achieved through the T_SCLHi and T_SCLRatio fields of the Setup Register.

Table 33-16 Timing Parameters for the SCL Clock

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{HIGH}	HIGH period of the SCL clock	4.0	-	0.6	-	0.26	-	µs
t_{LOW}	LOW period of the SCL clock	4.7	-	1.3	-	0.5	-	µs

For the Standard-mode, the minimum requirements of t_{HIGH} and t_{LOW} are close, so T_SCLRatio can be set to 0 (i.e., ratio = 1) to simplify the settings. The equations for the SCL periods shown in Table 33-11 Controller Setting Register are:

SCL HIGH period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) >= 4000ns

SCL LOW period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) >= 4700ns

Substitute 2 for T_SP, 1 for ratio and 25ns for t_{pclk}, the equations become (assuming TPM == 0):

50ns + (2 + 2 + T_SCLHi) * 25ns * (0 + 1) >= 4000ns

50ns + (2 + 2 + T_SCLHi * 1) * 25ns * (0 + 1) >= 4700ns

T_SCLHi >= 182

For the Fast-mode, the minimum requirement of t_{LOW} is about 2 times of t_{HIGH}, so T_SCLRatio can be set to 1 (i.e. ratio = 2). The equations for the SCL periods are:

SCL HIGH period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) >= 600ns

SCL LOW period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) >= 1300ns

Substitute 2 for T_SP, 2 for ratio and 25ns for t_{pclk}, the equations become (assuming TPM == 0):

50ns + (4 + T_SCLHi) * 25ns * (0 + 1) >= 600ns

50ns + (4 + T_SCLHi * 2) * 25ns * (0 + 1) >= 1300ns

T_SCLHi >= 23

For the Fast-mode Plus, T_SCLHi can be calculated using the same flow as the example in the Fast-mode.

33.5.6 Timing Parameter Multiplier

The Timing Parameter Multiplier Register could be adjusted to enlarge the effective values of the timing parameters in the Setup Register. This is only needed if the APB clock frequency is sohigh that it   is impossible to meet the required I^{2}C-bus interface timing specifications with the maximum values of timing parameters in the Setup Register alone (i.e., with TPM==0).

The procedure to determine the (non-zero) TPM value to meet the I^{2}C bus timing requirements is as follows:

11. Compute TPM by assigning maximum value (511) to T_SCLHi and minimum valid value (1) to T_SP. These values will be referred to as T_SCLHi' and T_SP' in discussions below to avoid confusion with the final T_SCLHi and T_SP values.

12. Compute T_SCLHi and T_SP based on the TPM value obtained above. Note that the final

    T_SCLHi will most likely not be 511, but some value smaller than 511.

For example, consider operating ATCIIC100 under 2ns APB-bus clock period (t_{pclk} = 2ns) with Standard-mode I2C bus timing requirements as described in Table 16 with additional requirement of 50ns spike suppression period. That is, t_{SP} = 50ns, t_{HIGH} = 4.0us and t_{LOW} = 4.7us.

First of all, based on equations for timing parameters in Table 11:

SCL HIGH period = 2 * t_{pclk} + (2 + T_SP' + T_SCLHi') * t_{pclk} * (TPM + 1) >= 4.0us

SCL LOW period = 2 * t_{pclk} + (2 + T_SP' + T_SCLHi'* ratio) * t_{pclk} * (TPM + 1) >= 4.7us

With T_SP' = 1, T_SCLHi' = 511, ratio = 1 and t_{pclk} = 2ns:

SCL HIGH period = 2 * 2ns + (2 + 1 + 511) * 2ns * (TPM + 1) >= 4.0us

SCL LOW period = 2 * 2ns + (2 + 1 + 511 * 1) * 2ns * (TPM + 1) >= 4.7us

Therefore,

2 * 2ns + (2 + 1 + 511) * 2ns * (TPM + 1) >= 4000ns

2 * 2ns + (2 + 1 + 511 * 1) * 2ns * (TPM + 1) >= 4700ns

So:

(TPM + 1) >= 4.57

TPM >= 3.57

Pick TPM = 4, and compute all timing parameters again based on formulae in 32-Table 11. That is, compute T_SP as:

T_SP >= 50ns / ((TPM+1) * t_{pclk}) = 50ns / ((4+1)*2ns) = 5

And T_SCLHi is computed by:

SCL HIGH period = 2 * t_{pclk} + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) >= 4000ns

SCL LOW period = 2 * t_{pclk} + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) >= 4700ns

So,

SCL HIGH period = 2 * 2ns + (2 + 5 + T_SCLHi) * 2ns * (4 + 1) >= 4000ns

SCL LOW period = 2 * 2ns + (2 + 5 + T_SCLHi) * 2ns * (4 + 1) >= 4700ns

Combine the two equations to reach:

2 * 2ns + (2 + 5 + T_SCLHi) * 2ns * (4 + 1) >= 4700ns

7 + T_SCLHi >= (4700 − 4) / (2*5)

T_SCLHi >= 462.6

So the timing parameters for the Standard-mode I2C bus timing requirement for 2ns APB-bus clock period with 50ns spike suppression would be: T_SCLHi = 463, T_SP = 5, and TPM = 4.

33.5.7 Master Mode

The following examples demonstrate how to initiate I^{2}C transactions at the Master mode.

33.5.8 Data Transmit without DMA

1. Setup the timing parameter multiplier by programming the Timing Parameter Multiplier Register if necessary.

2. Setup the controller by programming the Setup Register:

- Master = 1
- IICEn = 1
- timing parameters.

3. Set the data count, direction and phase choices in the Control Register:

- Phase_start = 1
- Phase_addr = 1
- Phase_data = 1
- Phase_stop = 1
- Dir = 0
- DataCnt = data counts in bytes.

4. Write the address of the target slave to the Address Register.

5. Enable the Completion Interrupt and FIFO Empty Interrupt in the Interrupt Enable Register:

- Cmpl = 1
- FIFOEmpty = 1.

6. Write 0x1 to the Command register to issue the transaction.

7. Wait for Interrupts:

– I. FIFO Empty Interrupt: push data into the FIFO by writing the data to the Data Register until the FIFO becomes full. If all data are pushed into the FIFO, disable the FIFO Empty Interrupt. Otherwise, repeat Step 6.

– II. Completion Interrupt: check the AddrHit field of the Status Register to make sure the targeted slave correctly receives the transaction. Write 1 to the Cmpl field of the Status Register to clear the completion status and go to Step 7.

8. Disable all interrupts and check the DataCnt field of the Control Register to know if all data are successfully transmitted.

33.5.9 Data Receive without DMA

1. Setup the timing parameter multiplier by programming the Timing Parameter Multiplier Register if necessary.

2. Setup the controller by programming the Setup Register:

- Master = 1
- IICEn = 1
- timing parameters.

3. Set the data count, direction and phase choices in the Control Register:

- Phase_start = 1
- Phase_addr = 1
- Phase_data = 1
- Phase_stop = 1

- Dir = 1
- DataCnt = data counts in bytes.

4. Write the address of the target slave to the Address Register.

5. Enable the Completion Interrupt and FIFO Full Interrupt in the Interrupt Enable Register:

- Cmpl = 1
- FIFOFull = 1.

6. Write 0x1 to the Command register to issue the transaction.

7. Wait for Interrupts:

- I. FIFO Full Interrupt: get data from the FIFO by reading the Data Register until the FIFO becomes empty; repeat Step 6.
- II. Completion Interrupt: check the AddrHit field of the Status Register to make sure the targeted slave correctly receives the transaction. Get all of the remaining data from the FIFO, write 1 to the Cmpl field of the Status Register to clear the completion status and go to Step 7.

8. Disable all interrupts and check the DataCnt of the Control Register to know if all data are successfully received.

   Data Transaction with DMA:

9. Setup the timing parameter multiplier by programming the Timing Parameter Multiplier Register if necssary.

10. Set the Controller by programming the Setup Register:

- Master = 1
- IICEn = 1
- DMAEn = 1
- timing parameters.

11. Setup the DMA controller.

12. Set data count, direction and phase choices in the Control Register:

- Phase_start = 1
- Phase_addr = 1
- Phase_data = 1
- Phase_stop = 1
- Dir = Transmitter/Receiver
- DataCnt = data counts in bytes

13. Write the address of the target slave to the Address Register.

14. Enable the Completion Interrupt in the Interrupt Enable Register:

- Cmpl = 1.

15. Write 0x1 to the Command Register to issue the transaction.

16. Wait for the Completion Interrupt: check the DataCnt of the Control Register to know if all data are successfully transferred.

33.5.10 Arbitration Lost in the Multi-Master Mode

When there are multiple masters on the I2C-bus, they may start transactions at the same time and the I2C-bus protocol defines an arbitration scheme to resolve the bus contention. Only one master will win the bus arbitration, and the other master(s) will lose the arbitration.

When the controller loses the arbitration, the arbitration lose interrupt will be triggered and the following steps should be done to restart the transaction.

1. If DMA is used, abort the DMA transfer and disable the I2C DMA request by setting the DMAEn field of the Setup Register to 0, then write 0x4 to the Command Register to clear the FIFO.

2. Wait for bus not busy by polling the BusBusy field of the Status Register.

3. Restart the transaction.

33.5.11 Slave Mode

The following examples demonstrate how to operate the I2C controller at the Slave mode.

33.5.12 Data Transaction without DMA:

1. Set the Address Register to the slave controller's I²C-bus address.

2. Setup the timing parameter multiplier by programming the Timing Parameter Multiplier Register if needed.

3. Setup the Controller by programming the Setup Register:

- Master = 0
- IICEn = 1
- timing parameters.

4. Enable the Address Hit Interrupt and Completion Interrupt in the Interrupt Enable Register:

- AddrHit = 1

- Cmpl = 1.

5. Wait for the Address Hit Interrupt: read the Dir field of the Control Register to identify the transaction direction. Read the Status Register if more transaction information is need, for example, the GenCall field to identify a General Call transaction. I. Receiver: enable the FIFOFull Interrupt and go to Step 5. II. Transmitter: enable the FIFOEmpty interrupt and go to Step 6.

6. Receiver:

- I. Wait for the FIFO full Interrupt until the Completion Interrupt asserted.
- II. If the FIFO Full Interrupt asserts: get data from the FIFO by reading the Data Register until the FIFO becomes empty, then go to (I).

7. Transmitter:

- I. Wait the FIFO Empty Interrupt until the Completion Interrupt asserted.
- II. If the FIFO Empty Interrupt asserts: push data to the FIFO by writing data to the Data Register until the FIFO becomes full, then go to (I).

8. During Step 5 or Step 6, if the Completion Interrupt asserts, clear the FIFO and go to step 8.

9. Check the DataCnt of the Control Register to know how many data are transferred. Clear the Completion Interrupt status of the Status Register when the software is ready to receive the next transaction.

33.5.13 Data transaction with DMA

The DMA should only be used when the next transaction's direction and data counts are known.

1. Set the Address Register to the controller's address.

2. Setup the timing parameter multiplier by programming the Timing Parameter Multiplier Register if necssary.

3. Setup the Controller by programming the Setup Register:

- Master = 0
- IICEn = 1
- timing parameters.

4. Enable the Address Hit Interrupt and Completion Interrupt in the Interrupt Enable Register:

- AddrHit = 1
- Cmpl = 1.

5. Wait for the Address Hit Interrupt: read the Dir field of the Control Register to check the direction. If it is not the designed direction, fall back to normal transfer without DMA (Data Transmit without DMA, Step 5 Read the Status Register if more transaction information is need, for example, the GenCall field to identify a General Call transaction.

6. Setup the DMA controller.

7. Set the data count of the the transaction in the Control Register: DataCnt = data counts in byte.

8. Enable the DMA by set the DMAEn of the Setup Register and wait for the Completion Interrupt.

9. After the Completion Interrupt asserted, check the DataCnt of the Control Register to know how many data are transferred.

10. 0 Disable the DMA.

11. 1 Clear the Completion Interrupt status of the Status Register when the software is ready to receive the next transaction.

34 ATCSPI200

34.1 Introduction

The ATCSPI200 is a Serial Peripheral Interface (SPI) controller which serves as a SPI master or a SPI slave. As a SPI master, the controller connects various SPI devices. As a SPI slave, the controller responds to the master requests for data exchange.

34.1.1 Features

Compliant with AMBA 2 AHB protocol specification

- Compliant with AMBA 3 APB protocol specification
- Support of MSB/LSB first transfer
- Support of Direct Memory Access (DMA) data transfer
- Support of programmable SPI SCLK
- Support of memory-mapped access (read-only) through AHB bus or EILM bus
- Support of SPI slave mode
- Configurable Dual and Quad I/O SPI interfaces
- Configurable TX/RX FIFO depth (The depth could be 2, 4, 8 or 16)
- Configurable programming port location on AHB/APB/EILM interfaces

34.1.2 Block Diagram

Figure 34-1 ATCSPI200 Block Diagram shows the block diagram of the ATCSPI200 controller.

Figure 34-1 ATCSPI200 Block Diagram
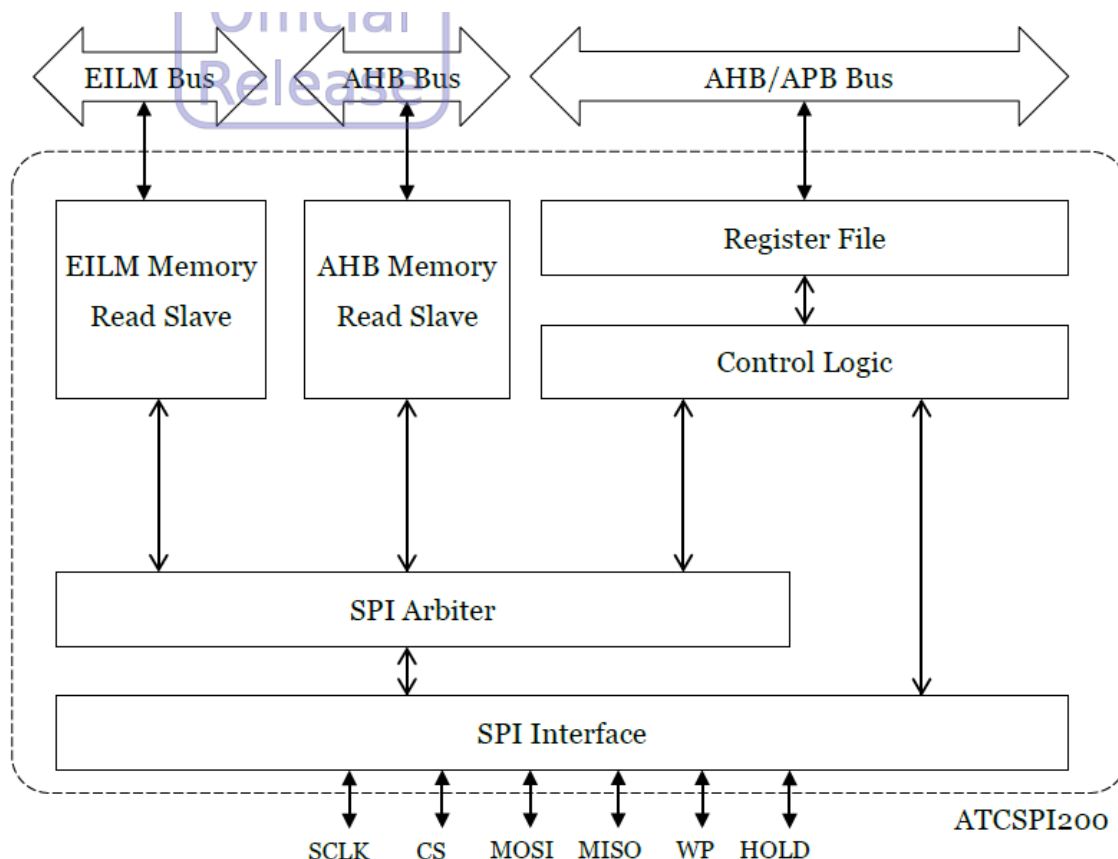
This section describes the SPI controller functions in four sub-sections for master mode, slave mode, dual I/O mode, and quad I/O mode. The master mode and slave mode sections describe the transfer initiation mechanism and the SPI frame formats. The dual mode and quad mode sections describe the other two transfer formats in comparison to the regular mode.

34.1.3 Master Mode

The ATCSPI200 controller can act as a SPI master initiating SPI transfers on the SPI bus. The SPI transfer format and interface timing are programmable via the AHB/APB programing port. The SPI transfers are initiated through the memory-mapped read access on the AHB/EILM bus or through register programming.

For SPI transfers initiated through the memory-mapped AHB/EILM read transactions, the read transactions are translated to the SPI interface based on the read command format setup in the SPI Memory Access Control Register. The default reset value of the register is hardware configurable to support booting from SPI ROMs. The size of the addressable memory-mapped space depends on the interface: up to 16MB

on the AHB bus and 4MB on the EILM bus.

The AHB/EILM address received from the memory-mapped interface is passed directly to the SPI slave since ATCSPI200 has no knowledge of the address and size of the slave. If addresses sent to the slave should be biased, the corresponding AHB/EILM address bits should be directly tied to the intended values.

For SPI transfers initiated by register programming, Figure 34-2 SPI Transfer Formatshows an example of SPI transfer format which includes command, address, and data phases for TX/RX data transfers. The controller provides dedicated registers to specify the contents of command, address, and data fields. The data register is shared by both TX and RX data transfers. Data transfers can be initiated through Programmed IO (PIO) or Direct Memory Access (DMA).

The ATCSPI200 controller provides TX/RX FIFO threshold interrupts to ease flow control under the PIO programming. The controller also has a programmable bit to issue an interrupt once the transfer completes.

In addition to the supported transfer format, the SPI controller allows direct control of the signals on the SPI interface. This capability enables communication with SPI devices which require special transfer formats. See SPI Direct IO Control Register (0x14) for more information.

Figure 34-2 SPI Transfer Format
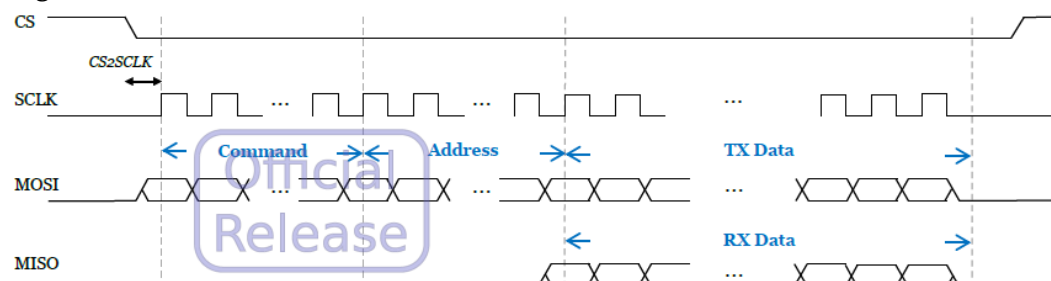
34.1.4 Slave Mode

The ATCSPI200 controller can also act as a SPI slave and accepts common commands as shown in Table 34-1 Supported Commands under the Slave Mode. In addition, the controller supports user-defined commands where the slave data field format is defined by the transfer control register.

The ATCSPI200 slave interprets the packet on the SPI I/O interface as three fields: slave command, slave dummy and slave data. The slave command field and slave dummy field are always 8-bit in length. The slave data field format is defined by the received command and the Transfer Control Register.

Table 34-1 Supported Commands under the Slave Mode

Slave Command Name	OP Code	Slave Data
Read Status Single IO	0x05	32-bit Status
Read Status Dual IO	0x15	32-bit Status
Read Status Quad IO	0x25	32-bit Status
Read Data Single IO	0x0B	Reply data from the Data Register in the FIFO manner
Read Data Dual IO	0x0C	Reply data from the Data Register in the FIFO manner
Read Data Quad IO	0x0E	Reply data from the Data Register in the FIFO manner
Write Data Single IO	0x51	Data saved to the Data Register in the FIFO manner
Write Data Dual IO	0x52	Data saved to the Data Register in the FIFO manner
Write Data Quad IO	0x54	Data saved to the Data Register in the FIFO manner
User-defined	Any 8-bit numbers other than the listed OP Codes	Depending on the Transfer Controller Register

For the status-reading commands, the slave returns the value of the Slave Status Register. The protocol format is illustrated in Figure 34-3 Timing Diagram of Status-Reading Commands (MSB First, DataMerge=0).

Figure 34-3 Timing Diagram of Status-Reading Commands (MSB First, DataMerge=0)
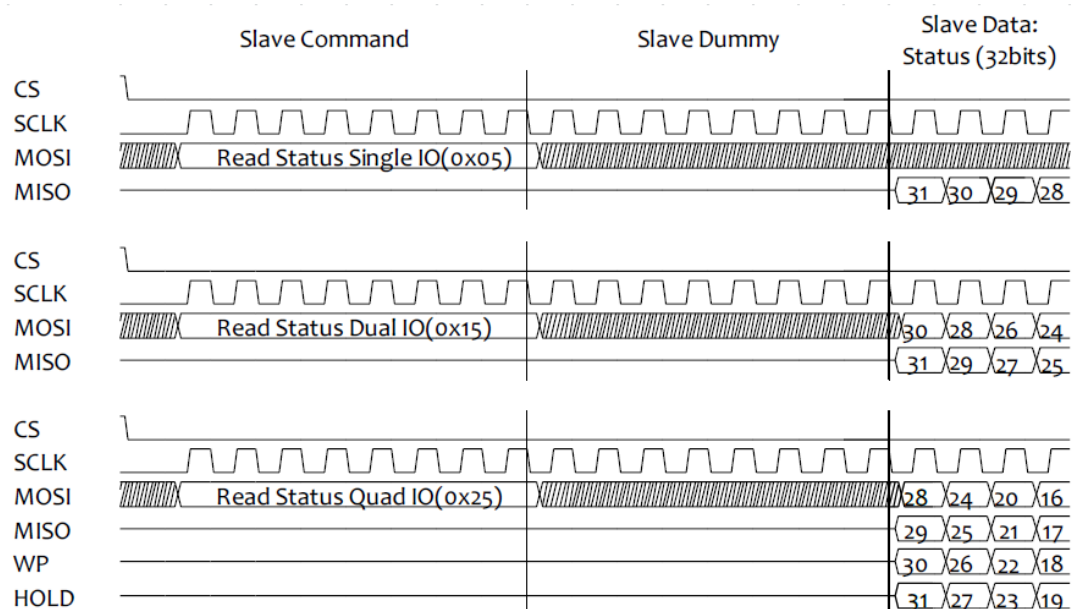
For the data-reading and data-writing commands, the protocol formats are illustrated in Figure 34-4 Timing Diagram of Data-Reading Commands (MSB First, Merge Mode), Figure 34-5 Timing Diagram of Data-Writing Commands (MSB First, Merge Mode), and Figure 34-6 Timing Diagram of

Data-Reading Commands (MSB First, Data Length = 16 Bits).

Figure 34-4 Timing Diagram of Data-Reading Commands (MSB First, Merge Mode)
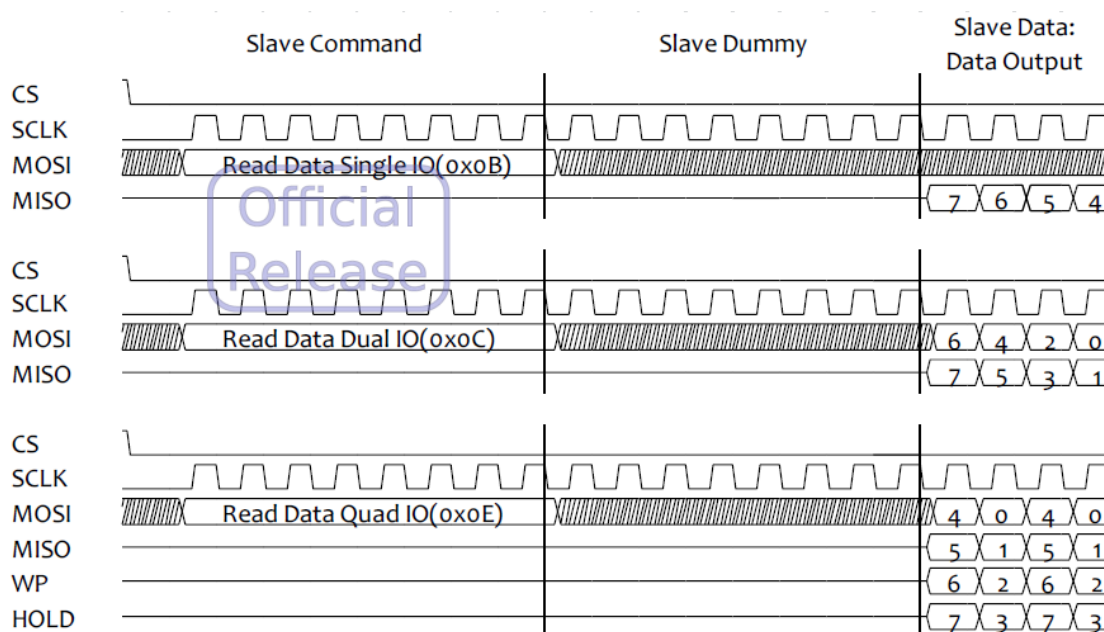
Figure 34-5 Timing Diagram of Data-Writing Commands (MSB First, Merge Mode)
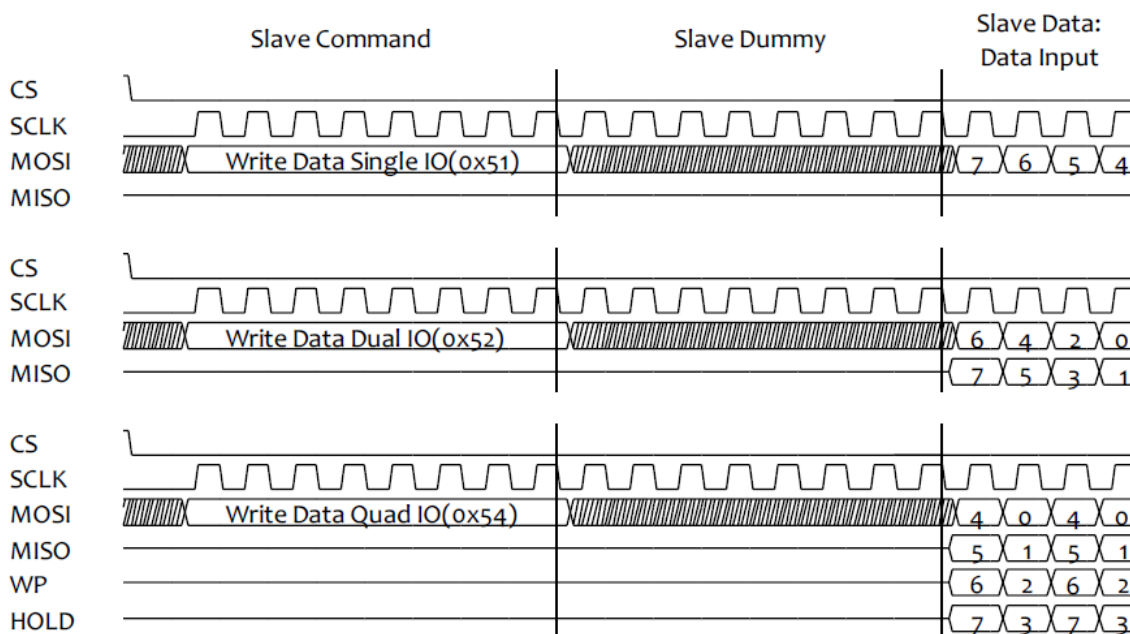
Figure 34-6 Timing Diagram of Data-Reading Commands (MSB First, Data Length = 16 Bits)
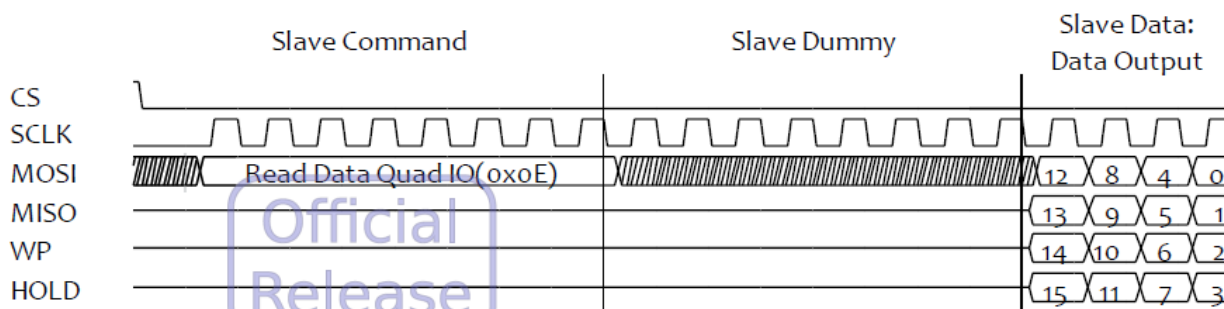
For user-defined commands, the slave data field format is defined by the SPI Transfer Control Register (0x20). For example, if the transfer mode is {Dummy, Write}, only the write field will be logged into the data register and the dummy field is dropped.

Figure 34-7 Timing Diagram of Slave User-Defined Command (MSB First, Merge Mode, TransMode = {Dummy,Write}, DualQuad = Quad, DummyCnt = 1, WrTranCnt = 3, Data length = 8 Bits)
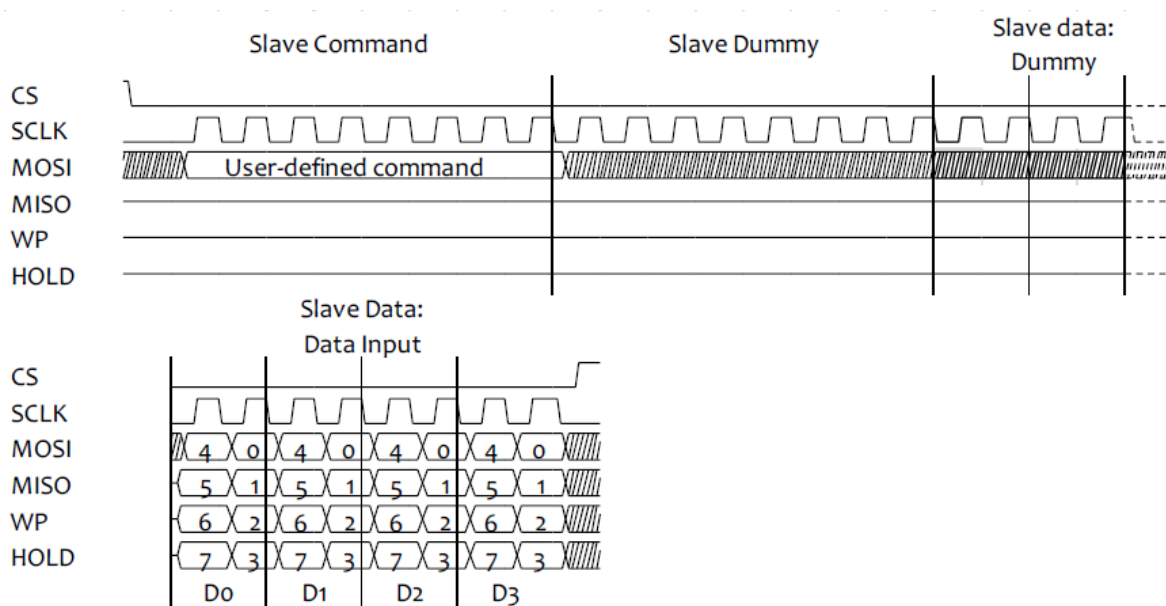
34.1.5 Dual I/O Mode

The dual I/O mode doubles the SPI bandwidth by treating the master-input/slave-output (MISO) and master-output/slave-input (MOSI) signals as bidirectional wires. The SPI controller provides two transfer formats at the dual I/O mode. In one format, both address phase and data phase make use of the two wires (MISO and MOSI). In the other format, only the data phase makes use of the two wires. See AddrFmt and DualQuad bits in the SPI Transfer Format Register (SPI Transfer Control

Register (0x20)) for more information. Figure 34-8 SPI Dual I/O Transfer (3-byte address) shows an example of dual I/O transfer.

Figure 34-8 SPI Dual I/O Transfer (3-byte address)
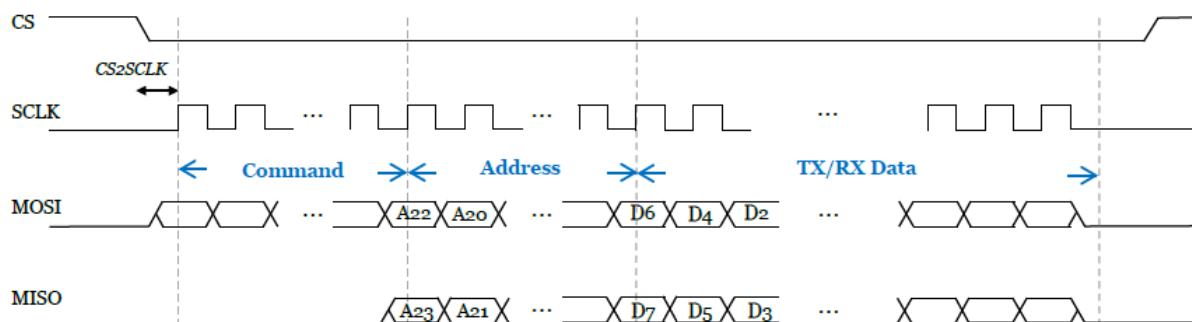
34.1.6 Quad I/O Mode

The quad I/O mode quadruples the SPI bandwidth by treating the master-input/slave-output (MISO), master-output/slave-input (MOSI), write protect (WP), and HOLD signals as bidirectional wires. The SPI controller provides two transfer formats at the quad I/O mode. In one format, both address phase and data phase make use of the four wires (MISO, MOSI, WP and HOLD). In the other format, only the data phase makes use of the four wires. See AddrFmt and DualQuad bits in the SPI Transfer Format Register (SPI Transfer Control Register (0x20)) for more information. Figure 34-9 SPI Quad I/O Mode Transfer (3-byte address) shows an example of quad I/O transfer.

Figure 34-9 SPI Quad I/O Mode Transfer (3-byte address)
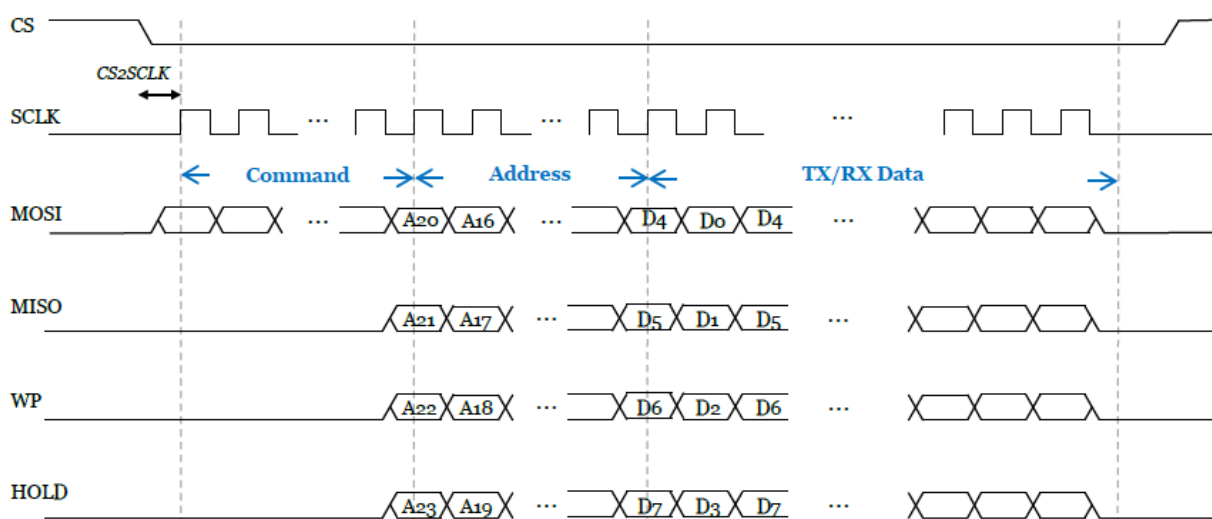
34.2 Signal Description

Table 34-2 ATCSPI200 Signal Definition gives the detailed descriptions of ATCSPI200 I/O signals.

Table 34-2 ATCSPI200 Signal Definition

Signal Name	I/O Type	Description
SPI Clock signals		
spi_clock	I	The clock source for the SPI interface. The SCLK signal of the SPI interface is generated by dividing this clock source with a programmable value.
spi_rstn	I	The reset signal for the spi_clock domain (Active-Low)
AHB Global Signals		
hclk	I	AHB clock
hresetn	I	AHB reset signal (Active-Low)
apb2ahb_clken	I	APB to AHB clock enable is an AHB domain signal indicating the valid AHB clock cycles to sample and update APB domain signals when the AHB frequency is a multiple of the APB frequency.
AHB Memory Read Port		
hsel_mem	I	AHB slave select
htrans_mem[1:0]	I	AHB transfer type
haddr_mem[N:0]	I	AHB address bus N is 23 if AHB address width is 24-bit. It is 31 if the AHB address width is 32-bit. Unused most significant bits must be padded with zero, otherwise undetermined behavior may occur.
hreadyin_mem	I	AHB bus ready
hwrite_mem	I	AHB transfer direction
hwdata_mem[31:0]	I	AHB write data bus
hreadyout_mem	O	AHB slave transfer done

Signal Name	I/O Type	Description
hresp_mem	O	AHB transfer response
hrdata_mem[31:0]	O	AHB read data bus
EILM Memory Read Port		
eilm_clk	I	EILM bus clock
ahb2eilm_clken	I	AHB to EILM clock enable is an EILM domain signal indicating the valid EILM clock cycles to sample and update AHB domain signals when the EILM frequency is a multiple of the AHB frequency.
apb2eilm_clken	I	APB to EILM clock enable is an EILM domain signal indicating the valid EILM clock cycles to sample and update APB domain signals when the EILM frequency is a multiple of the APB frequency.
eilm_resetn	I	EILM bus reset signal
eilm_req	I	EILM request
eilm_web[3:0]	I	EILM write enable
eilm_wait	O	EILM early wait state 0x0: Data will be ready at the next cycle 0x1: The interface should be in the wait state for the next cycle
eilm_wait_cnt[1:0]	I	Represents the static wait cycles for the CPU Note that eilm_wait_cnt should not be zero.
eilm_wdata[31:0]	I	EILM write data
eilm_addr[21:2]	I	EILM address bus
eilm_rdata[31:0]	O	EILM read data
AHB Register Programming Port		
hsel_reg	I	AHB slave select
htrans_reg[1:0]	I	AHB transfer type
haddr_reg[N:0]	I	AHB address bus N is 23 if AHB address width is 24-bit. It is 31 if the AHB address width is 32-bit.
hreadyin_reg	I	AHB bus ready

Signal Name	I/O Type	Description
hwrite_reg	I	AHB transfer direction
hwdata_reg[31:0]	I	AHB write data bus
hreadyout_reg	O	AHB slave transfer done
hresp_reg	O	AHB transfer response
hrdata_reg[31:0]	O	AHB read data bus
APB Register Programming Port		
pclk	I	APB clock (must be synchronous with the AHB/EILM clock)
presetn	I	APB reset signal (Active-Low)
paddr[31:0]	I	APB address bus
penable	I	APB enable signal
pwrite	I	APB write signal
psel	I	APB select signal
pwdata[31:0]	I	APB write data bus
prdata[31:0]	O	APB read data
pready	O	APB slave transfer done
SPI Interface		
spi_sclk_in	I	Status of the SPI SCLK signal
spi_cs_n_in	I	Status of the SPI CS (chip select) signal
spi_miso_in	I	Status of the SPI master-input/slave-output (MISO) signal
spi_mosi_in	I	Status of the SPI master-output/slave-input (MOSI) signal
spi_wp_n_in	I	Status of the SPI Flash write protect signal
spi_hold_n_in	I	Status of the SPI Flash hold signal
spi_sclk_out	O	Output value for the SPI SCLK signal
spi_cs_n_out	O	Output value for the SPI CS signal
spi_miso_out	O	Output value for the SPI MISO signal
spi_mosi_out	O	Output value for the SPI MOSI signal
spi_wp_n_out	O	Output value for the SPI Flash write protect signal

Signal Name	I/O Type	Description
spi_hold_n_out	O	Output value for the SPI Flash hold signal
spi_sclk_oe	O	Output enable for the SPI SCLK signal
spi_cs_n_oe	O	Output enable for the SPI CS signal
spi_miso_oe	O	Output enable for the SPI MISO signal
spi_mosi_oe	O	Output enable for the SPI MOSI signal
spi_wp_n_oe	O	Output enable for the SPI Flash write protect signal
spi_hold_n_oe	O	Output enable for the SPI Flash hold signal
Miscellaneous		
spi_default_mode3	I	Select the default SPI mode
		0x0: Default values of both CPOL and CPHA are 0x0
		0x1: Default values of both CPOL and CPHA are 0x1
spi_default_as_slave	I	Set the ATCSPI200 controller as a slave after reset
		0x0: Default as a master
		0x1: Default as a slave
spi_tx_dma_ack	I	TX FIFO DMA acknowledge
spi_rx_dma_ack	I	RX FIFO DMA acknowledge
spi_tx_dma_req	O	TX FIFO DMA request
spi_rx_dma_req	O	RX FIFO DMA request
spi_boot_intr	O	The SPI controller interrupt
scan_enable	I	Scan enable (Active-High during ATPG scan/shift phase)
scan_test	I	Scan test mode (Active-High during ATPG test)

34.3 Programming Model

34.3.1 Summary of Registers

Table 34-3 ATCSPI200 Register Summary shows the summary of ATCSPI200 registers.

Table 34-3 ATCSPI200 Register Summary

Offset	Name	Description
0x00	IdRev	ID and Revision Register
0x04~0x0C	-	Reserved
0x10	TransFmt	SPI Transfer Format Register
0x14	DirectIO	SPI Direct IO Control Register
0x18~0x1C		Reserved
0x20	TransCtrl	SPI Transfer Control Register
0x24	Cmd	SPI Command Register
0x28	Addr	SPI Address Register
0x2C	Data	SPI Data Register
0x30	Ctrl	SPI Control Register
0x34	Status	SPI Status Register
0x38	IntrEn	SPI Interrupt Enable Register
0x3C	IntrSt	SPI Interrupt Status Register
0x40	Timing	SPI Interface Timing Register
0x44 ~ 0x4C		Reserved
0x50	MemCtrl	SPI Memory Access Control Register
0x54 ~ 0x5c		Reserved
0x60	SlvSt	SPI Slave Status Register (Present only when ATCSPI200_SLAVE_SUPPORT is defined)
0x64	SlvDataCnt	SPI Slave Data Count Register (Present only when ATCSPI200_SLAVE_SUPPORT is defined)
0x68~0x78		Reserved
0x7C	Config	Configuration Register

34.3.2 Register Description

The abbreviations for the Type column are summarized below.

RO: read only

RW: readable and writable

W1C: readable and write 1 to clear

34.3.3 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset

values of the two revision fields are revision dependent.

Table 34-4 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for ATCSPI200	0x020020
RevMajor	7:4	RO	Major revision number	Revision dependent
RevMinor	3:0	RO	Minor revision number	Revision dependent

34.3.4 SPI Transfer Format Register (0x10)

This register defines the SPI transfer format.

Table 34-5 SPI Transfer Format Register

Name	Bit	Type	Description	Reset
Reserved	31:18	-	-	-
AddrLen	17:16	RW	Address length in bytes 0x0: 1 byte 0x1: 2 bytes 0x2: 3 bytes 0x3: 4 bytes	0x2
Reserved	15:13	-	-	-
DataLen	12:8	RW	The length of each data unit in bits The actual bit number of a data unit is ($DataLen + 1$)	0x07
DataMerge	7	RW	Enable Data Merge mode, which does automatic data split on write and data coalescing on read. This bit only takes effect when $DataLen$ = 0x7. Under Data Merge mode, each write to the Data Register will transmit all fourbytes of the write data; each read from the Data Register will retrieve four bytes of received data as a single word data. When Data Merge mode is disabled, only the least ($DataLen$+1) significant bits of the Data Register are valid for read/write operations; no automatic data split/coalescing will be performed.	0x1
Reserved	6:5	-	-	-
MOSIBiDir	4	RW	Bi-directional MOSI in regular (single) mode 0x0: MOSI is uni-directional signal in regular mode. 0x1: MOSI is bi-directional signal in regular mode. This bi-directional signal replaces the two	0x0

Name	Bit	Type	Description	Reset
			uni-directional data signals, MOSI and MISO.	
LSB	3	RW	Transfer data with the least significant bit first 0x0: Most significant bit first 0x1: Least significant bit first	0x0
SlvMode	2	RW	SPI Master/Slave mode selection 0x0: Master mode 0x1: Slave mode (Exist only when configuration ATCSPI200_SLAVE_SUPPORT is defined)	Depends on pin spi_default_as_ slave
CPOL	1	RW	SPI Clock Polarity 0x0: SCLK is LOW in the idle states 0x1: SCLK is HIGH in the idle states	Depends on pin spi_default_mod e3
CPHA	0	RW	SPI Clock Phase 0x0: Sampling data at odd SCLK edges 0x1: Sampling data at even SCLK edges	Depends on pin spi_default_mod e3

34.3.5 SPI Direct IO Control Register (0x14)

This register enables the direct control of the SPI interface signals. The register is valid only when the configuration ATCSPI200_DIRECT_IO_SUPPORT is defined.

Table 34-6 SPI Direct IO Control Register

Table 6. SPI Direct IO Control Register

Name	Bit	Type	Description	Reset
Reserved	31:25	-	-	-
DirectIOEn	24	RW	Enable Direct IO 0x0: Disable 0x1: Enable	0x0
Reserved	23:22	-	-	-
HOLD_OE	21	RW	Output enable for the SPI Flash hold signal	0x0
WP_OE	20	RW	Output enable for the SPI Flash write protect signal	0x0
MISO_OE	19	RW	Output enable fo the SPI MISO signal	0x0
MOSI_OE	18	RW	Output enable for the SPI MOSI signal	0x0
SCLK_OE	17	RW	Output enable for the SPI SCLK signal	0x0
CS_OE	16	RW	Output enable for SPI CS (chip select) signal	0x0
Reserved	15:14	-	-	-
HOLD_O	13	RW	Output value for the SPI Flash hold signal	0x1
WP_O	12	RW	Output value for the SPI Flash write protect signal	0x1
MISO_O	11	RW	Output value for the SPI MISO signal	0x0
MOSI_O	10	RW	Output value for the SPI MOSI signal	0x0
SCLK_O	9	RW	Output value for the SPI SCLK signal	0x0
CS_O	8	RW	Output value for the SPI CS (chip select) signal	0x1
Reserved	7:6	-	-	-
HOLD_I	5	RO	Status of the SPI Flash hold signal	Depends on pin spi_hold_n_in
WP_I	4	RO	Status of the SPI Flash write protect signal	Depends on pin spi_wp_n_in
MISO_I	3	RO	Status of the SPI MISO signal	Depends on pin spi_miso_in
MOSI_I	2	RO	Status of the SPI MOSI signal	Depends on pin spi_mosi_in
SCLK_I	1	RO	Status of the SPI SCLK signal	Depends on pin spi_clk_in
CS_I	0	RO	Status of the SPI CS (chip select) signal	Depends on pin spi_cs_n_in

34.3.6 SPI Transfer Control Register (0x20)

This register controls aspects of SPI transfers. Please see SPI Command Register (0x24) for starting a SPI transfer.

Table 34-7 SPI Transfer Control Register

Name	Bit	Type	Description	Reset
CmdEn	30	RW	The SPI command phase enable 0x0: Disable the command phase 0x1: Enable the command phase (Master mode only)	0x0
AddrEn	29	RW	The SPI address phase enable 0x0: Disable the address phase 0x1: Enable the address phase (Master mode only)	0x0
AddrFmt	28	RW	The SPI address phase format 0x0: Address phase is the regular (single) mode 0x1: The format of the address phase is the same as the data phase (DualQuad). (Master mode only)	0x0
TransMode	27:24	RW	The transfer mode The transfer sequence could be 0x0: Write and read at the same time 0x1: Write only 0x2: Read only 0x3: Write, Read 0x4: Read, Write 0x5: Write, Dummy, Read 0x6: Read, Dummy, Write 0x7: None Data (must enable CmdEn or AddrEn in master mode) 0x8: Dummy, Write 0x9: Dummy, Read 0xa~0xf: Reserved	0x0

Name	Bit	Type	Description	Reset
DualQuad	23:22	RW	The SPI data phase format 0x0: Regular (Single) mode 0x1: Dual I/O mode 0x2: Quad I/O mode 0x3: Reserved	0x0
TokenEn	21	RW	Append an one-byte special token following the address phase for SPI read transfers. The value of the special token should be selected in `TokenValue`. 0x0: Disable the one-byte special token 0x1: Enable the one-byte special token (Master mode only)	0x0
WrTranCnt	20:12	RW	Transfer count for write data `WrTranCnt` indicates the number of units of data to be transmitted to the SPI bus from the Data Register. The actual transfer count is (`WrTranCnt`+1). `WrTranCnt` only takes effect when `TransMode` is 0, 1, 3, 4, 5, 6 or 8. The size (bit-width) of a data unit is defined by the `DataLen` field of the Transfer Format Register. For `TransMode` 0, `WrTranCnt` must be equal to `RdTranCnt`.	0x0
TokenValue	11	RW	The value of the one-byte special token following the address phase for SPI read transfers. 0x0: token value = 0x00 0x1: token value = 0x69 (Master mode only)	0x0

Name	Bit	Type	Description	Reset
DummyCnt	10:9	RW	Dummy data count. The actual dummy count is (DummyCnt +1). The number of dummy cycles on the SPI interface will be (DummyCnt+1)* ((DataLen+1)/SPI IO width) The Data pins are put into the high impedance during the dummy data phase. DummyCnt is only used for TransMode 5, 6, 8 and 9, which has dummy data phases. The following table shows dummy cycle settings under some common transfer formats:	0x0
RdTranCnt	8:0	RW	Transfer count for read data RdTranCnt indicates the number of units of data to be received from SPI bus and stored to the Data Register. The actual received count is (RdTranCnt+1). RdTransCnt only takes effect when TransMode is 0, 2, 3, 4, 5, 6 or 9. The size (bit-width) of a data unit is defined by the DataLen field of the Transfer Format Register. For TransMode 0, WrTranCnt must equal RdTranCnt.	0x0

Table embedded in DummyCnt description:

DummyCnt+1	DataLen+1	DualQuad	#Dummy Cycles on the SPI Interface
1	8	Regular/ Single	8
1	8	Dual	4
1	8	Quad	2
2	8	Quad	4
3	8	Quad	6
1	32	Quad	8

34.3.7 SPI Command Register (0x24)

Writing operations on this register will trigger SPI transfers. This register must be written with a dummy value to start a SPI transfer even when the command phase is not enabled.When the ATCSPI200 controller is programmed to the slave mode, the command field of the last received SPI transaction is stored in this SPI Command Register.

Table 34-8 SPI Command Register

Name	Bit	Type	Description	Reset
Reserved	31:8	-	-	-
CMD	7:0	RW	SPI Command	0x0

34.3.8 SPI Address Register (0x28)

Table 34-9 SPI Address Register

Name	Bit	Type	Description	Reset
ADDR	31:0	RW	SPI Address (Master mode only)	0x0

34.3.9 SPI Data Register (0x2C)

When the controller is in the data merge mode, the byte endian of the SPI Data Register is little endian.

Table 34-10 SPI Data Register

Name	Bit	Type	Description	Reset
DATA	31:0	RW	Data to transmit or the received data	0x0
			For writes, data is enqueued to the TX FIFO. The least significant byte is always transmitted first. If the TX FIFO is full and the SPIActive bit of the status register is 1, the ready signal hready/pready will be deasserted to insert wait states to the transfer.	
			For reads, data is read and dequeued from the RX FIFO. The least significant byte is the first received byte. If the RX FIFO is empty and the SPIActive bit of the status register is 1, the ready signal hready/pready will be deasserted to insert wait states to the transfer.	
			The FIFOs decouple the speed of the SPI transfers and the software's generation/consumption of data. When the TX FIFO is empty, SPI transfers will hold until more data is written to the TX FIFO; when the RX FIFO is full, SPI transfers will hold until there is more room in the RX FIFO.	
			If more data is written to the TX FIFO than the write transfer count (WrTranCnt), the remaining data will stay in the TX FIFO for the next transfer or until the TX FIFO is reset.	

34.3.10 SPI Control Register (0x30)

Table 34-11 SPI Control Register

Name	Bit	Type	Description	Reset
Reserved	31:21	-	-	-
TXTHRES	20:16	RW	Transmit (TX) FIFO Threshold The TXFIFOInt interrupt or DMA request would be issued to replenish the TX FIFO when the TX data count is less than or equal to the TX FIFO threshold.	0x0
Reserved	15:13	-	-	-
RXTHRES	12:8	RW	Receive (RX) FIFO Threshold The RXFIFOInt interrupt or DMA request would be issued for consuming the RX FIFO when the RX data count is more than or equal to the RX FIFO threshold.	0x0
Reserved	7:5	-	-	-
TXDMAEN	4	RW	TX DMA enable	0x0
RXDMAEN	3	RW	RX DMA enable	0x0
TXFIFORST	2	RW	Transmit FIFO reset Write 1 to reset. It is automatically cleared to 0 after the reset operation completes.	0x0
RXFIFORST	1	RW	Receive FIFO reset Write 1 to reset. It is automatically cleared to 0 after the reset operation completes.	0x0
SPIRST	0	RW	SPI reset Write 1 to reset. It is automatically cleared to 0 after the reset operation completes.	0x0

34.3.11 SPI Status Register (0x34)

Table 34-12 SPI Status Register

Name	Bit	Type	Description	Reset
Reserved	31:24	-	-	-
TXFULL	23	RO	Transmit FIFO Full flag	0x0
TXEMPTY	22	RO	Transmit FIFO Empty flag	0x1
Reserved	21	-	-	-
TXNUM	20:16	RO	Number of valid entries in the Transmit FIFO	0x0
RXFULL	15	RO	Receive FIFO Full flag	0x0
RXEMPTY	14	RO	Receive FIFO Empty flag	0x1
Reserved	13	-	-	-
RXNUM	12:8	RO	Number of valid entries in the Receive FIFO	0x0
Reserved	7:1	-	-	-
SPIActive	0	RO	SPI register programming is in progress. In master mode, SPIActive becomes 1 after the SPI command register is written and becomes 0 after the transfer is finished. In slave mode, SPIActive becomes 1 after the SPI CS signal is asserted and becomes 0 after the SPI CS signal is deasserted. Note that due to clock synchronization, it may take at most two spi_clock cycles for SPIActive to change when the corresponding condition happens. Note this bit stays 0 when Direct IO Control or the memory-mapped interface is used.	0x0

34.3.12 SPI Interrupt Enable Register (0x38)

Table 34-13 SPI Interrupt Enable Register

Name	Bit	Type	Description	Reset
Reserved	31:6	-		-
SlvCmdEn	5	RW	Enable the Slave Command Interrupt. Control whether interrupts are triggered whenever slave commands are received. (Slave mode only)	0x0
EndIntEn	4	RW	Enable the End of SPI Transfer interrupt. Control whether interrupts are triggered when SPI transfers end. (In slave mode, end of read status transaction doesn't trigger this interrupt.)	0x0
TXFIFOIntEn	3	RW	Enable the SPI Transmit FIFO Threshold interrupt. Control whether interrupts are triggered when the valid entries are less than or equal to the TX FIFO threshold.	0x0
RXFIFOIntEn	2	RW	Enable the SPI Receive FIFO Threshold interrupt. Control whether interrupts are triggered when the valid entries are greater than or equal to the RX FIFO threshold.	0x0
TXFIFOURIntEn	1	RW	Enable the SPI Transmit FIFO Underrun interrupt. Control whether interrupts are triggered when the Transmit FIFO run out of data. (Slave mode only)	0x0
RXFIFOORIntEn	0	RW	Enable the SPI Receive FIFO Overrun interrupt. Control whether interrupts are triggered when the Receive FIFO overflows. (Slave mode only)	0x0

34.3.13 SPI Interrupt Status Register (0x3C)

Table 34-14 SPI Interrupt Status Register

Name	Bit	Type	Description	Reset
Reserved	31:6	-		-
SlvCmdInt	5	W1C	Slave Command Interrupt. This bit is set when Slave Command interrupts occur. (Slave mode only)	0x0
EndInt	4	W1C	End of SPI Transfer interrupt. This bit is set when End of SPI Transfer interrupts occur.	0x0
TXFIFOInt	3	W1C	TX FIFO Threshold interrupt. This bit is set when TX FIFO Threshold interrupts occur.	0x0
RXFIFOInt	2	W1C	RX FIFO Threshold interrupt. This bit is set when RX FIFO Threshold interrupts occur.	0x0
TXFIFOURInt	1	W1C	TX FIFO Underrun interrupt. This bit is set when TX FIFO Underrun interrupts occur. (Slave mode only)	0x0
RXFIFOORInt	0	W1C	RX FIFO Overrun interrupt. This bit is set when RX FIFO Overrun interrupts occur. (Slave mode only)	0x0

34.3.14 SPI Interface Timing Register (0x40)

This register controls the SPI interface timing to satisfy the SPI Slave interface timing requirements. Only the master needs to program this register.

Table 34-15 SPI Interface Timing Register

Name	Bit	Type	Description	Reset
Reserved	31:14	-	-	-
CS2SCLK	13:12	RW	The minimum time between the edges of SPI CS and the edges of SCLK. The actual duration is $$\frac{\text{SCLK period}}{2} \times (\text{CS2SCLK} + 1)$$	Configuration dependent
CSHT	11:8	RW	The minimum time that SPI CS should stay HIGH. The actual duration is $$\frac{\text{SCLK period}}{2} \times (\text{CSHT} + 1)$$	Configuration dependent
SCLK_DIV	7:0	RW	The clock frequency ratio between the clock source and SPI interface SCLK. SCLK period = $$\left((\text{SCLK\_DIV}+1) \times 2\right) \times (\text{Period of the SPI clock source})$$ The SCLK_DIV value 0xff is a special value which indicates that the SCLK frequency should be the same as the spi_clock frequency.	Configuration dependent

34.3.15 SPI Memory Access Control Register (0x50)

This register defines the SPI command issued for memory-mapped AHB/EILM read accesses.

The memory-mapped AHB/EILM read accesses should not be on-going while programming this register or the SPI Interface Timing Register (0x40). The AHB/EILM accesses could be resumed when the MemCtrlChg bit is cleared.

Table 34-16 SPI Memory Access Control Register

Name	Bit	Type	Description	Reset
Reserved	31:9	-	-	-
MemCtrlChg	8	RO	This bit is set when this register (0x50) or the SPI Interface Timing Register (0x40) is written; it is automatically cleared when the new programming takes effect.	0
Reserved	7:4	-	-	-
MemRdCmd	3:0	RW	Selects the SPI command for serving the memory-mapped reads on the AHB/EILM bus The command encoding table is listed in Table 17. The latency of each command is listed in Table 18.	Configuration dependent

Table 34-17 Supported SPI Read Commands for Memory-Mapped AHB/EILM Reads

MemRdCmd	Command	Address	Dummy	Data
0	0x03	3 bytes in Regular mode	N/A	Regular mode
1	0x0B	3 bytes in Regular mode	1 byte in Regular mode	Regular mode
2	0x3B	3 bytes in Regular mode	1 byte in Regular mode	Dual mode
3	0x6B	3 bytes in Regular mode	1 byte in Regular mode	Quad mode
4	0xBB	(3-byte address + 1-byte 0) in Dual mode	N/A	Dual mode
5	0xEB	(3-byte address + 1-byte 0) in Quad mode	2 bytes in Quad mode	Quad mode
6–7	Reserved	-	-	-
8	0x13	4 bytes in Regular mode	N/A	Regular mode
9	0x0C	4 bytes in Regular mode	1 byte in Regular mode	Regular mode
10	0x3C	4 bytes in Regular mode	1 byte in Regular mode	Dual mode
11	0x6C	4 bytes in Regular mode	1 byte in Regular mode	Quad mode
12	0xBC	(4-byte address + 1-byte 0) in Dual mode	N/A	Dual mode
13	0xEC	(4-byte address + 1-byte 0) in Quad mode	2 bytes in Quad mode	Quad mode
14-15	Reserved			

Table 34-18 Latency of a 4 Bytes Data Transfer through the AHB/EILM Memory Read Port

Command	Non-sequential	Sequential	Sequential (prefetched*)
0x03	8 BUS_CLK + 10 SPI_CLK + 64 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x0B	8 BUS_CLK + 10 SPI_CLK + 72 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x3B	8 BUS_CLK + 10 SPI_CLK + 56 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0x6B	8 BUS_CLK + 10 SPI_CLK + 48 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK
0xBB	8 BUS_CLK + 10 SPI_CLK + 40 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0xEB	8 BUS_CLK + 10 SPI_CLK + 28 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK
0x13	8 BUS_CLK + 10 SPI_CLK + 72 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x0C	8 BUS_CLK + 10 SPI_CLK + 80 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x3C	8 BUS_CLK + 10 SPI_CLK + 64 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0x6C	8 BUS_CLK + 10 SPI_CLK + 56 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK
0xBC	8 BUS_CLK + 10 SPI_CLK + 44 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0xEC	8 BUS_CLK + 10 SPI_CLK + 30 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK

BUS_CLK: bus (AHB or EILM) clock cycle; SPI_CLK: spi_clock clock cycle; SCLK: SCLK clock cycle

* The memory-mapped accesses are prefetched to speed up sequential accesses.

34.3.16 SPI Slave Status Register (0x60)

The Slave Status Register keeps slave statuses. An SPI master can get these statuses by issuing status-reading commands.

Table 34-19 SPI Slave Status Register

Name	Bit	Type	Description	Reset
Reserved	31:19	-	-	-
UnderRun	18	W1C	Data underrun occurs in the last transaction	0
OverRun	17	W1C	Data overrun occurs in the last transaction	0
Ready	16	RW	Set this bit to indicate that the ATCSPI200 is ready for data transaction. When an SPI transaction other than slave status-reading command ends, this bit will be cleared to 0.	0
USR_Status	15:0	RW	User defined status flags	0

34.3.17 SPI Slave Data Count Register (0x64)

This register shows the data count of the read/write transactions in the slave mode. Data register is accessed based on the data count information.

Table 34-20 SPI Slave Data Count Register

Name	Bit	Type	Description	Reset
Reserved	31:25	-	-	
WCnt	24:16	RO	Slave transmitted data count	0
Reserved	15:9	-	-	
RCnt	8:0	RO	Slave received data count	0

34.3.18 Configuration Register (0x7C)

Table 34-21 Configuration Register

Name	Bit	Type	Description	Reset
Reserved	31:15	-	-	-
Slave	14	RO	Support for SPI Slave mode	Configuration dependent
EILMMem	13	RO	Support for memory-mapped access (read-only) through EILM bus	Configuration dependent
AHBMem	12	RO	Support for memory-mapped access (read-only) through AHB bus	Configuration dependent
DirectIO	11	RO	Support for Direct SPI IO	Configuration dependent
Reserved	10	-	-	-
QuadSPI	9	RO	Support for Quad I/O SPI	Configuration dependent
DualSPI	8	RO	Support for Dual I/O SPI	Configuration dependent
Reserved	7:6	-	-	-
TxFIFOSize	5:4	RO	Depth of TX FIFO 0x0: 2 words 0x1: 4 words 0x2: 8 words 0x3: 16 words	Configuration dependent
Reserved	3:2	-	-	-
RxFIFOSize	1:0	RO	Depth of RX FIFO 0x0: 2 words 0x1: 4 words 0x2: 8 words 0x3: 16 words	Configuration dependent

34.4 Hardware Configuration Options

The ATCSPI200 controller provides the following configuration options to select the hardware features. The configuration result can be read from the Configuration Register (0x7C).

34.4.1 Address Width

Define the following macro to set the AHB address width to 24-bit. Default address width is 32-bit.

`define ATCSPI200_ADDR_WIDTH_24

34.4.2 Dual I/O Mode

Define ATCSPI200_DUALSPI_SUPPORT to support the dual I/O mode.

`define ATCSPI200_DUALSPI_SUPPORT

34.4.3 Quad I/O Mode

Define ATCSPI200_QUADSPI_SUPPORT to support both dual I/O mode and quad I/O mode.

`define ATCSPI200_QUADSPI_SUPPORT

34.4.4 TX FIFO Depth

Define ATCSPI200_TXFIFO_DEPTH_nW to identify the TX FIFO depth, where n could be 2, 4, 8, and, 16. The default depth is 2 words.

For example, define the TX FIFO depth to 4 words by:

`define ATCSPI200_TXFIFO_DEPTH_4W

34.4.5 RX FIFO Depth

Define ATCSPI200_RXFIFO_DEPTH_nW to identify the RX FIFO depth, where n could be 2, 4, 8, and, 16. The default depth is 2 words.

For example, define the RX FIFO depth to 4 words by:

`define ATCSPI200_RXFIFO_DEPTH_4W

34.4.6 Direct IO Control

Define ATCSPI200_DIRECT_IO_SUPPORT to enable Direct IO control.

`define ATCSPI200_DIRECT_IO_SUPPORT

34.4.7 Memory-Mapped Access Support

The ATCSPI200 provides a memory-mapped access interface. The interface can be either AHB bus or EILM bus.

Define ATCSPI200_AHB_MEM_SUPPORT to allow memory-mapped read accesses on the AHB bus.

`define ATCSPI200_AHB_MEM_SUPPORT

Define ATCSPI200_EILM_MEM_SUPPORT to allow memory-mapped read accesses on the EILM

bus.

`define ATCSPI200_EILM_MEM_SUPPORT

34.4.8 Slave Mode

Define ATCSPI200_SLAVE_SUPPORT to support the SPI Slave mode.

`define ATCSPI200_SLAVE_SUPPORT

34.4.9 AHB Register Port

Define ATCSPI200_REG_AHB to change the register file programming port to AHB. The default programming port is APB.

`define ATCSPI200_REG_AHB

34.4.10 Memory-Mapped AHB/EILM Read

Define ATCSPI200_MEM_RDCMD_DEFAULT to specify the reset value of the MemRdCmd field in the SPI Memory Access Control Register (0x50). For example, define the reset value to 1 by:

`define ATCSPI200_MEM_RDCMD_DEFAULT 4'd1

34.4.11 SPI Interface Timing Parameters

Define following macros to specify the reset values of the CS2SCLK, CSHT, and SCLK_DIV fields in the SPI Interface Timing Register (0x40). For example:

`define ATCSPI200_CS2CLK_DEFAULT 3'h0

`define ATCSPI200_CSHT_DEFAULT 3'h2

<div align="center">`define ATCSPI200_SCLKDIV_DEFAULT 8'h10</div>

34.5 Programming Sequence

This chapter describes the programming sequence to initiate SPI transfers via the register programming. The register programming should not be performed while the memory-mapped reads on the AHB/EILM bus is active. Similarly, the memory-mapped read operation should not be performed when the register programming is in progress.

34.5.1 SPI Write with DMA

The following sample programming sequence sets up the controller for

- Transmitting two-byte address and 8-bit data width,
- Total transmission count of 16,
- Merging data from four bytes to one word,
- DMA data transfer with hardware handshaking,
- Triggering interrupts at the end of the SPI transfer,
- SPI SCLK frequency being half of the SPI clock source frequency,
- Issuing the 'Page-Program' command (0x02) to the ROM.

SPI transfer format setup

1. Read TX/RX FIFO depth in the Configuration Register (0x70).

2. Wait for the previous SPI transfer to finish by waiting for the SPIActive bit of the SPI Status Register (0x34) to become zero.

3. Set the SPI Transfer Format Register (0x10) as follows:

   AddrLen = 1 (address length – 1)

   DataLen = 7 (data length – 1)

   DataMerge = 1

   Reset values for other fields

4. Set the SPI Transfer Control Register (0x20) as follows:

   CmdEn = 1

   AddrEn = 1

   TransMode = 1 (write only)

   WrTranCnt = 15 (total transmission count – 1)

   Reset values for other fields

5. Set the SPI Control Register (0x30) to enable DMA, reset the TX FIFO, and specify the TX FIFO threshold according to the TX FIFO configuration, e.g., half of the TX FIFO depth.

6. Set the SPI Interrupt Enable Register (0x38) to enable the EndIntEn interrupt.

7. Set the SPI Interface Timing Register (0x40) to set SCLK_DIV to 0.

SPI transfer execution

8. Set DMA controller to move data from memory to the SPI Data Register (0x2C).

9. Set the SPI Address Register (0x28).

10. Write the 'Page-Program' command (0x02) to the SPI Command Register (0x24) to trigger the SPI transfer. The encoding of the command may be different depending on SPI devices.

11. Wait the EndInt interrupt by checking the EndInt bit of the SPI Interrupt Status Register (0x3C).

12. Write one to clear the EndInt bit of the SPI Interrupt Status Register (0x3C).

34.5.2 SPI Read with DMA:

The following sample programming sequence sets up the controller for

- Receiving 16 bytes from a two-byte address,
- Merging data from four bytes to one word,
- DMA data transfer with hardware handshaking,
- Triggering interrupts at the end of SPI transfer,
- SPI SCLK frequency being half of the SPI clock source frequency,
- Issuing the 'Read-Data' command (0x03) to a ROM.

SPI transfer format setup

1. Check TX/RX FIFO depth in the Configuration Register (0x70).

2. Wait for the previous SPI transfer to finish by waiting for the SPIActive bit of the SPI

3. Status Register (0x34) to become zero.

4. Set the SPI Transfer Format Register (0x10) as follows:

AddrLen = 1 (address length – 1)

DataLen = 7 (data length – 1)

DataMerge = 1

Reset values for other fields

5. Set the SPI Transfer Control Register (0x20) as follows:

CmdEn = 1

AddrEn = 1

TransMode = 2 (read only)

RdTranCnt = 15 (total transmission count – 1)

Reset values for other fields

6. Set the SPI Control Register (0x30) to enable DMA and specify the RX FIFO threshold.

7. Set the SPI Interrupt Enable Register (0x38) to enable the EndIntEn interrupt.

8. Set the SPI Interface Timing Register (0x40) to set SCLK_DIV to 0.

    SPI transfer execution

9. Set DMA controller to move data from the SPI Data Register (0x2C) to memory.

10. Set the SPI Address Register (0x28).

11. Write the 'Read-Data' command (0x03) to the SPI Command Register (0x24) to trigger the SPI transfer. The encoding of this command may be different depending on SPI devices.

12. Wait for the EndInt interrupt by checking the EndInt bit of the SPI Interrupt Status Register (0x3C).

13. Write one to clear the EndInt bit of the SPI Interrupt Status Register (0x3C).

34.5.3 Stopping SPI Activities Initiated by the Memory-Mapped Interface

The memory-mapped interface (EILM/AHB) translates bus reads to SPI accesses on the SPI interface. The interface is optimized to speed up sequential accesses. The ATCSPI200 controller always prefetches sequential data words from SPI devices until RX FIFO is full, where it stops the activity by pausing SPI SCLK while keeping SPI CS still asserted. The data transfer would be resumed without restarting a new SPI request with the command and address phases, if subsequent accesses on the memory

interfaces are sequential and the RX FIFO full condition is cleared. Otherwise, the controller will end the current transfer to start a new transfer with the new non-sequential address. The controller also ends the current transfer if there is a new request made through the register programming interface.

To keep SPI CS from being asserted for too long without activity, the following actions can be explicitly performed to deassert SPI CS and make the controller leave the active memory-mapped access state when no more memory-mapped accesses will be performed soon. Please note that once the memory-mapped interface is active, the controller protects the FIFOs from being interfered by register programming, so a three-step procedure is required:

1. Read the current value of SPI Memory Access Control Register (0x50).

2. Write back the value to the SPI Memory Access Control Register (0x50).

3. Wait until MemCtrlChg becomes 0.

For product ID 0x02002044 or above, the SPI controller will automatically deassert SPI CS, clear RX FFIFO and return to the idle state if the memory-mapped interface has been quiet for some time. The exact condition is when RX FIFO and the internal serial-to-parallel shift registers are both full and there are no more incoming memory-mapped reads.

After ATCSPI200 is brought back to the idle state either automatically or by the three-step procedure, any normal operation can be performed later. Just follow the recommended flow described herein respectively.

34.5.4 Receiving Data from SPI Masters

Assuming that the SPI transfer format is:

- 8-bit data width,
- quad write command (0x54),
- 20 bytes are transferred.

Since the data merging is enabled by default, this will correspond to 5 data accesses to the Data Register.

1. Reset the RX FIFO by writing one to the RXFIFORST bit and wait the RXFIFORST bit to be cleared to 0.

2. Set the RX FIFO threshold (RXTHRES) in the SPI Control Register (0x30) according to the RX FIFO configuration, e.g., half of the RX FIFO depth.

3. Set the SPI Interrupt Enable Register (0x38) to enable the slave

command interrupt (SlvCmdEn), receive FIFO threshold interrupt (RXFIFOIntEn), and transfer end interrupt (EndIntEn).

4. In the interrupt service routine:

   a) Wait for the Slave Command interrupt by checking the SlvCmdInt bit of the SPI Interrupt Status Register (0x3C).

   – i. Prepare to handle the received SPI request as recorded in the SPI Command Register (0x24).

   – ii. Write one to clear the SlvCmdInt bit of the SPI Interrupt Status Register (0x3C).

   b) Wait for the RX FIFO interrupt by checking the RXFIFOInt bit of the SPI Interrupt Status Register (0x3C).

   – i. Pop words from the SPI Data Register (0x2C) according to the RX FIFO threshold (RXTHRES) setup.

   – ii. Write one to clear the RXFIFOInt bit of the SPI Interrupt Status Register (0x3C).

   c) Wait for transfer end interrupt by checking the EndInt bit of the SPI Interrupt Status Register (0x3C).

   – i. Check the RX FIFO entries (RXNUM) of the SPI Status Register (0x34).

   – ii. Pop all words from the SPI Data Register (0x2C).

   – iii. Write one to clear the EndInt bit of the SPI Interrupt Status Register (0x3C).

34.5.5 Transmitting Data to SPI Masters

The SPI master should arrange the slave to make data ready for transmission before submitting read commands. Assuming this is done through a user-defined command to indicate the address offset and data count of the next read transfer.

The user-defined command is followed by the read status command (0x05) to check for the progress of the slave in the processing of the user-defined commands (whether or not the data that the user-defined command asks for is ready).

Finally, the SPI master issues a data read command to initiate the data transfer. Assuming the data transmission format is:

- 8-bit data width,
- quad read command (0x0E),
- the user-defined command ask for 32 bytes of data to be transferred.

Since the data merging is enabled by default, this will correspond to 8 data accesses to the Data Register.

1. When receiving the user-defined command which asks for 32-bytes of data,

   Reset the TX FIFO by writing one to the TXFIFORST bit and wait the TXFIFORST cleared to 0.

   Set the TX FIFO threshold (TXTHRES) in the SPI Control Register (0x30) according to the TX FIFO configuration, e.g., half of the TX FIFO depth.

   Prepare the requested data (8 words = 32 bytes) and fill a number of words larger than the TXTHRES setup into the SPI Data Register (0x2C).

   Set the SPI Interrupt Enable Register (0x38) to enable the slave command interrupt (SlvCmdEn), transmit FIFO threshold interrupt (TXFIFOIntEn), transfer end interrupt (EndIntEn).

   Set the Ready bit in the SPI Slave Status Register (0x60).

2. In the interrupt service routine:

   Wait for the Slave Command interrupt by checking the SlvCmdInt bit of the SPI interrupt Status Register (0x3C).

   – i. Verify that the received command is a read command by checking the SPI command Register (0x24).
   – ii. Write one to clear the SlvCmdInt bit of the SPI Interrupt Status Register 0x3C).
   – Wait for the TX FIFO interrupt by checking the TXFIFOInt bit of the SPI Interrupt Status Register (0x3C).
   – i. Fill a number of words larger than the TXTHRES setup into the SPI Data Register (0x2C).
   – ii. If the data count of the read transfer has been completely filled into the Data Register, clear the transmit FIFO threshold interrupt (TXFIFOIntEn) in the SPI Interrupt Enable Register (0x38) to avoid a redundant interrupt.
   – iii. Write one to clear the TXFIFOInt bit of the SPI Interrupt Status Register (0x3C).
   – Wait for transfer end interrupt by checking the EndInt bit of the SPI Interrupt Status Register (0x3C).
   – i. Write one to clear the EndInt bit of the SPI Interrupt Status Register (0x3C).

34.6 Integration Guideline

This chapter describes the integration guideline for the ATCSPI200 controller. General integration guidelines are listed in the bullets below and special requirements for the slave mode, clock enable signals, clock gating cell, and DFT considerations are described in the subsections.

General integration guidelines for the ATCSPI200 controller are:

- eilm_clk, hclk, and pclk must be synchronous;
- eilm_clk frequency $\geqslant$ hclk frequency $\geqslant$ pclk frequency;
- In the chip-level environment, spi_clk_in, spi_clk_out and spi_clk_oe must be connected to the I/O pad of SCLK as shown in Figure 34-10 SCLK I/O Pad;
- In master mode, the slave device samples/returns data based on the observed clock SCLK in the chip-level environment (or spi_clk_out in the block-level environment). As a result, a generated clock derived from spi_clock is created on SCLK (or spi_clk_out) with PAD delay as the reference clock of set_input_delay/set_output_delay commands. (The set_input_delay/set_output_delay commands of Synopsys DesignCompiler, have the -reference_pin option for this kind of source synchronization design. However, to support Cadence RTL Compiler as well, the traditional generated clock approach is used here.)
- Like the previous step, another generated clock derived from the divided spi_clock is also created to analyze the SCLK DIVIDER path;
- Since spi_clk_in is looped back from SCLK (or spi_clk_out), spi_clk_in should be created as a generated clock derived from SCLK (or spi_clk_out) with PAD delay;
- In Figure 34-11 Relation Between SCLK Domain and spi_clock Domain, the path between spi_in_r and the shift register is only valid when SCLK DIVIDER is used. In this case, one SCLK period corresponds to ((SCLK_DIV+1)*2) periods of spi_clock. The shift register will only flop data at the correct spi_clock edge corresponding to the SCLK cycle. As a result, this path should be set as a ((SCLK_DIV_MIN+1)*2) multicycle path where SCLK_DIV_MIN represents the smallest possible SCLK_DIV value in the system;
- In Figure 34-11 Relation Between SCLK Domain and spi_clock Domain, d1_r is only used when SCLK_DIV = 0xff. As a result, the paths from the divided spi_clock domain to d1_r can be set false paths.
- (DataLen + 1)/($SPI\ IO\ width$) must be greater than 1, where the SPI IO width is 2 for the dual mode and 4 for the quad mode;
- The controller cannot be accessed concurrently by register programming and memory-mapped accesses.

Figure 34-10 SCLK I/O Pad

Figure 34-11 Relation Between SCLK Domain and spi_clock Domain

34.6.1 SCLK Frequency

Based on the current design, the frequency of SPI SCLK should be smaller than or equal to 1/4 of the spi_clock frequency. As a result, the remote SPI master should drive SCLK no faster than 1/4 of the spi_clock frequency:

$$\text{SCLK frequency} \leqslant \frac{1}{4}(\text{Slave spi\_clock frequency})$$

34.6.2 Time between the Edges of SPI CS to the First Edge of SCLK

The minimum time between the edges of SPI CS and the first edge of SCLK should be larger than 2 * period of spi_clock.

34.6.3 Clock Enable Signal

The ATCSPI200 design has several clock enable input signals. A Foo

to Bar clock enable signal indicates valid Bar-domain clock cycles to sample and update the Foo clock domain signals when the Bar clock frequency is a multiple of the Foo clock frequency. The clock enable signal must assert for one Bar clock cycle for each Foo clock cycle. An example is illustrated in Figure 34-12 ATCSPI200 Clock Enable Signal for 4:1 AHB-to-APB Clock Ratio for the case that the AHB frequency is 4 times of the APB frequency. For the special case that AHB clock and APB clock are identical, the clock enable signal must be a constant HIGH.

Figure 34-12 ATCSPI200 Clock Enable Signal for 4:1 AHB-to-APB Clock Ratio
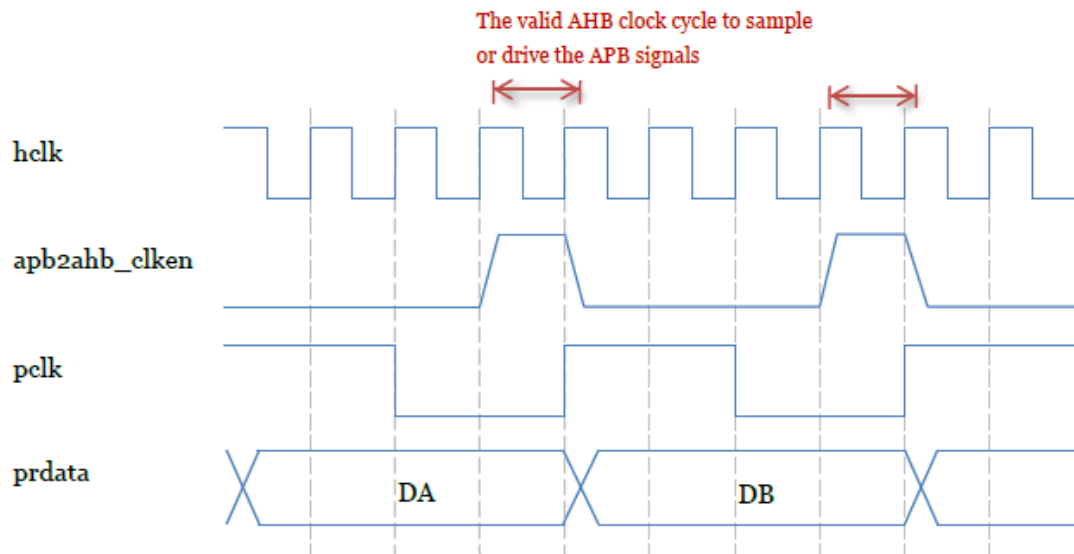
34.6.4 Clock Gating Cell

There is one clock gating cell model in the ATCSPI200 design. The model is located at $NDS_HOME/andes_ip/macro/hdl/gck.v. It is a latch based clock gating cell. The RTL modeling of gck.v is synthesizable. However, integrated clock gating cells from the standard library is preferred to better control clock skews.

The synthesis script automatically scans the standard cell library for an integrated clock gating cell that matches gck.v's behavior. A gck_autogen.v file will be created under the script directory if such a cell is found in the standard library. Please double check the correctness of the cell.

If the heuristic fails, it will fall back to let the synthesizer synthesize the gck.v. One could still modify gck.v to instantiate a proper integrated clock gating cell if the cell in question cannot be found by the synthesis script.

The synthesis script selects clock gating cells according to the following rule: it scans the library

to find all cells that have attribute clock_gating_integrated_cell and the value of the attribute is latch_posedge_precontrol. For example,

.lib file

cell (GCK_VENDOR_CELL) {

...

clock_gating_integrated_cell : "latch_posedge_precontrol";

...

}
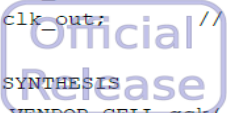
When there are multiple cells found in the library, the one with the smallest leakage power is selected. The script then connects the pins through these attributes in the library:

clock_gate_clock_pin, clock_gate_enable_pin, clock_gate_test_pin and

clock_gate_out_pin.

Figure 34-13 Clock Gating Logic for Simulation and Synthesis shows how the gck module should look like in case the synthesis script fails or you

want to generate gck.v manually.

Figure 34-13 Clock Gating Logic for Simulation and Synthesis

```verilog
module gck (clk_out, clk_en, clk_in, test_en);
input clk_in;          // clock input
input clk_en;          // gated clock signal
input test_en;         // enable shifting scan data
output clk_out;        // clock output

`ifdef SYNTHESIS
    GCK_VENDOR_CELL gck(
            .Q(clk_out),
            .E(clk_en),
            .TE(test_en),
            .CK(clk_in)
            );
`else
    // gated clock behavior code:
    // FPGA note: this code is synthesizable under
    // FPGA with FixGatedClock support.

    reg latch_out;

    always @(clk_in or clk_en or test_en)
        if (~clk_in)
            latch_out <= clk_en | test_en;

    // clock gating occurs at the negedge of clk_in.
    assign clk_out = clk_in & latch_out;
`endif //SYNTHESIS
endmodule
```

Figure 34-14 Clock Gating Cell Diagram with Waveform further depicts the requirement for the clock gating cell. The internal latch of the clock gating cell should be in the transparent mode when clock is low and in the latch mode when clock is high.

Figure 34-14 Clock Gating Cell Diagram with Waveform
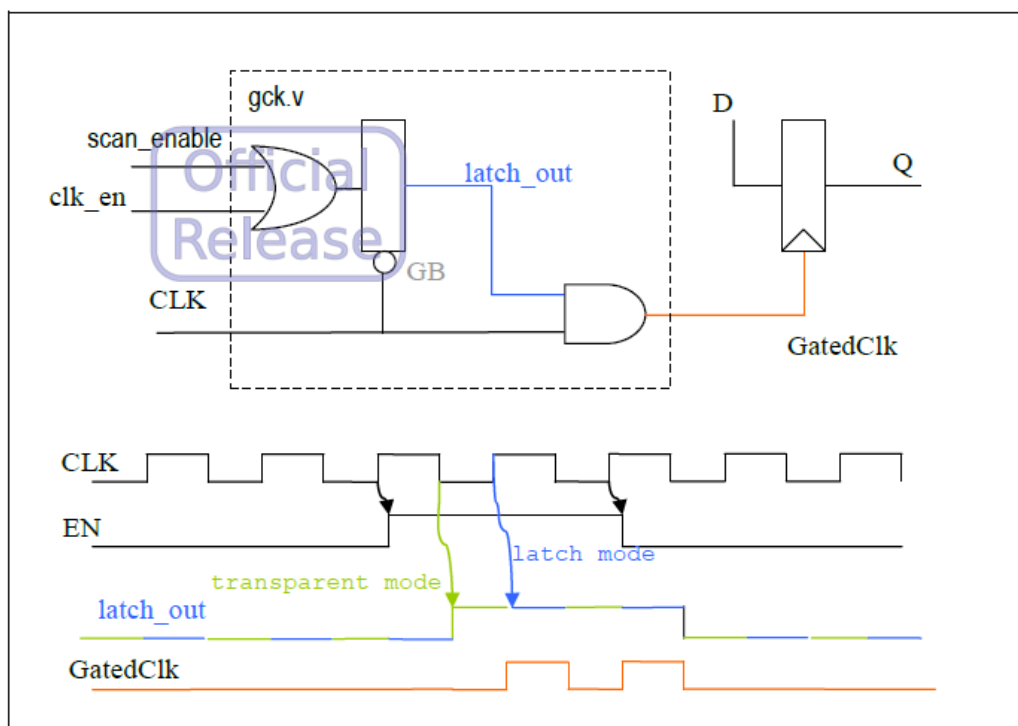
It is recommend that after the manual instantiation of the gck cell, a formal equivalence checking is performed to make sure that the selected cell matches the behavior code. However, unless gck cells with latch_posedge_precontrol is used, the equivalence checking will most likely not match.

34.6.5 DFT Considerations

The ATCSPI200 design provides scan_test and scan_enable input pins to control its behavior under ATPG test.

The scan_test pin should be asserted (Active-High) during the entire session of the ATPG test mode. There is a loop-back clock path in the ATCSPI200 design as shown in Figure 34-15 ATCSPI200 Design for ATPG Test. The loop-back clock should be controllable under the ATPG test mode to enhance the ATPG coverage. With the scan_test pin being HIGH for the entire duration of the ATPG test mode, the flops in the loop-back clock domain will be clocked by spi_clock and be immune to the changes of spi_clk_oe and spi_clk_out pins.

The scan_enable pin should only be active high during the ATPG

scan/shift phase, such that all clocks are turned on and ATPG test patterns and results can be shifted through the scan chains. The clock gating logic is disabled (clock running freely) when scan_enable is active. The scan_enable pin is off during ATPG capture phase to allow observabilities for clock gating enable pins.

The DFT tools need to be informed about the usage of these two signals using the sample commands listed below.

set_dft_signal -view existing_dft -port scan_enable -type ScanEnable

set_dft_signal -view existing_dft -port scan_test -type TestMode

Figure 34-15 ATCSPI200 Design for ATPG Test
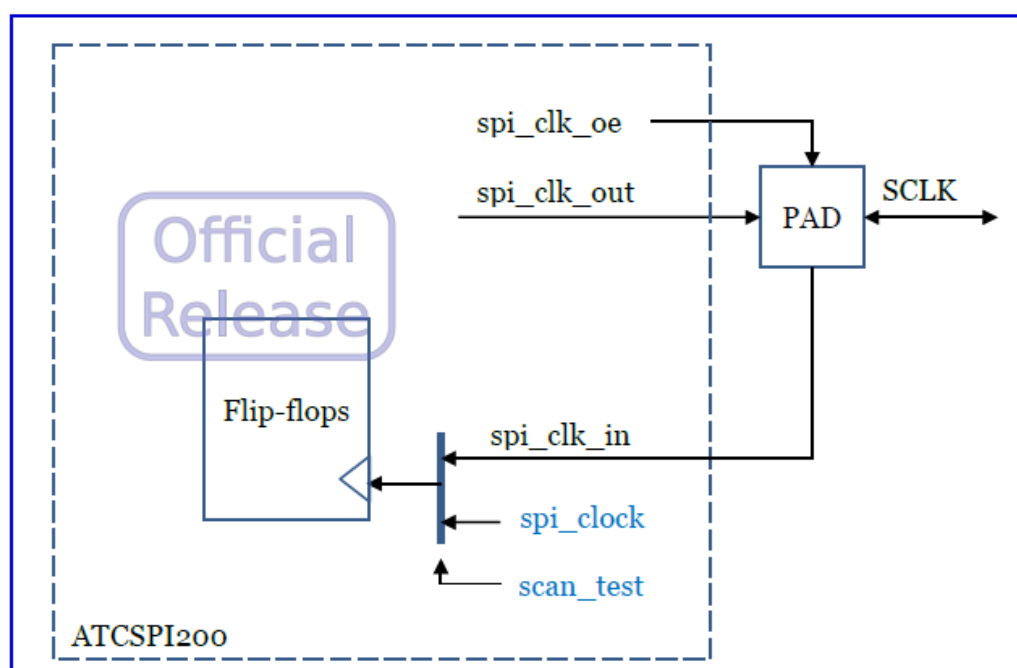
35ATCUART100

35.1 Introduction

The ATCUART100 is a Universal Asynchronous Receiver/Transmitter (UART) controller.

35.1.1 Features

- AMBA 2.0 APB interface for registers access
- Hardware configurable 16, 32, 64 and 128 bytes transmit/receive FIFOs
- Over-sampling frequency is programmable (even multiples ranging from 8x to 32x)
- Programming sequence compatible with the 16C550D UART
    - Supports 5 to 8 bits per character
    - Supports 1, 1.5 and 2 STOP bits
    - Supports even, odd and stick parity bits
    - Supports DMA function
    - Supports programmable baud rate
    - Supports modem control interface
    - Supports complete status reporting capabilities
    - Supports line breaks, parity errors, framing errors and data overrun detection

35.1.2 Block Diagram

Figure 35-1   ATCUART100 Block Diagram shows the block diagram of ATCUART100:

Figure 35-1   ATCUART100 Block Diagram
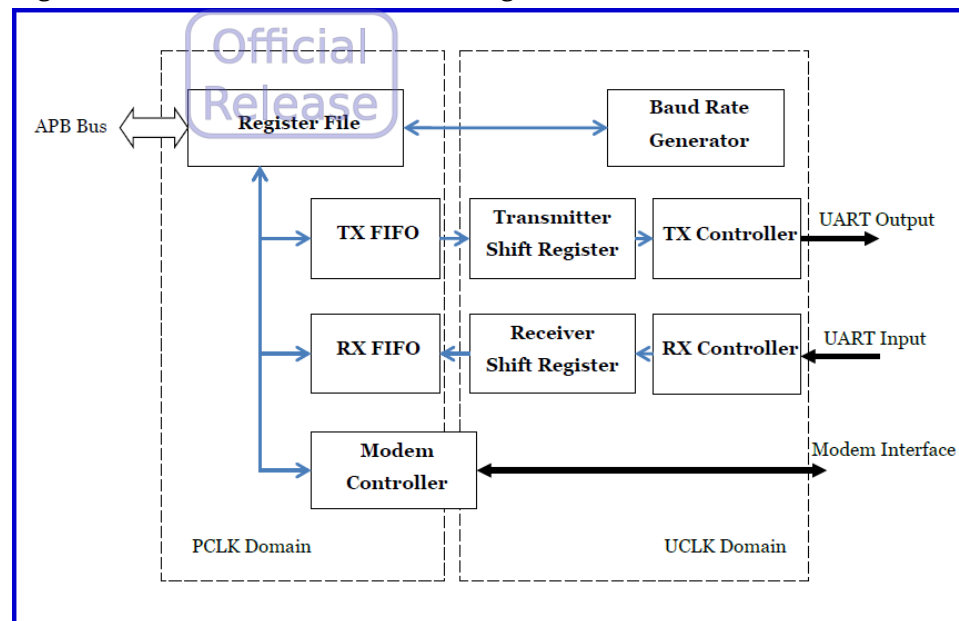
The ATCUART100 controller is a serial communication controller which provides asynchronous serial interface for a peripheral device or a modem. The controller comprises a transmitter, a receiver, a Baud Rate Generator, a Modem Controller, a Register File and an APB interface, as shown in Figure 34-1.

35.1.3 Transmitter

The transmitter comprises a Transmitter FIFO (TX FIFO), a Transmitter Shift Register (TSR), and a Transmitter Controller (TX controller). The TX FIFO holds data to be transferred through the serial interface. The TX FIFO can store up to 128 characters depending on hardware configurations and programming settings. The TSR reads a character from the TX FIFO for the next transmission. The TSR functions as a parallel-to-serial data converter, converting the outgoing character to serial bit streams. For each character transmission, the TX Controller generates a START bit, an optional parity bit, and some number of STOP bits. The generation of parity bit and STOP bit can be programmed by the Line Control Register. The TX FIFO is by default a one entry buffer called Transmitter Holding Register (THR). It needs to be enabled to work in multi-entry FIFO mode (FIFOE in the FIFO Control Register).

35.1.4 Receiver

The receiver comprises a Receiver FIFO (RX FIFO), Receiver Shift Register (RSR), and a Receiver Controller (RX Controller). The RX Controller uses the oversampling clock generated by Baud Rate Generator

to perform sampling at the center of each bit transmission. The received bits are shifted into the RSR for serial-to-parallel data conversion and the received character is stored into the RX FIFO. The RX FIFO is by default a one entry buffer called the Receiver Buffer Register (RBR). It needs to be enabled to work in multi-entry FIFO mode (FIFOE in the FIFO Control Register) with up to 128 characters depending on hardware configurations. The RX controller also detects some error conditions for each data transmission including parity error, framing error, data overrun, or line break.

35.1.5 Baud Rate Generator

The Baud Rate Generator takes the UART clock (uclk) as the source clock and divides it by a divisor. The divisor value is 16-bit in size and stored in two separate programming registers, each holding an 8-bit value. The most significant byte (MSB) is held in the Divisor Latch MSB (DLM) register and the least significant byte (LSB) is held in the Divisor Latch LSB (DLL) register.

The ratio of the sampling clock frequency to the baud rate is the oversampling ratio, which is stored in the Over Sample Control Register (OSCR). The default value for OSCR is 16. This is typically good enough and do not need further adjustment.

The formula for the divisor value is as follows:

The divisor value = the frequency of uclk / (desired baud rate × OSCR)

The oversampling clock is used by the RX controller to detect the leading edge of the START bit and to sample the data at the center of each bit transfer. Assume that the oversampling ratio is 16. The RX controller uses the oversampling clock to operate a counter, which starts counting from 1 to 16 after a falling edge of SIN (beginning of a START bit) is detected. The RX controller samples the value of SIN as a data bit when the counter value is 8. The counter will be reset to 1 after its value reaches 16 for sampling the next bit. The process repeats until the STOP bit is received. The oversampling clock is also used by the TX controller to generate its output data stream.

The oversampling technique provides better tolerance of clock variation. Suppose that:

- T is the period of one bit transmission as perceived by the ATCUART100 Rx Controller.
- $T_{transmitter}$ is the period of one bit transmission of the transmitter.
- N is the bit number for one frame of data – the START bit, data bits, parity bit (if any), and the STOP bit(s).

Then, the clock period tolerance for $T_{\text{transmitter}}$ is as follows:

$$\frac{(0.5-\frac{1}{OSCR})}{N} \times T \;\leqslant\; T_{\text{transmitter}} \;\leqslant\; (NN - 0.5)/(N - 1) \times T$$

Since T is the inverse of the baud rate, the actual baud rate generated by this controller in relation to the actual baud rate of the transmitter (the tolerance factor) can be within the range below:

$$\frac{(0.5-\frac{1}{OSCR})}{N} \leqslant \frac{Actual\ Baud\ Rate}{Actual\ Transmitter\ Baud\ Rate} \leqslant (N - 0.5)/(N - 1)$$

If the character has one START bit, 8 data bits, one parity bit and one STOP bit, then N is 11 (1 + 8 + 1 + 1). The tolerance factor is from 0.9602 to 1.05. The table below shows clock tolerance factors as percentage of the actual Transmitter Baud Rates for typical values of N and OSCR.

Table 35-1 Clock Variation Tolerance Factor

OSCR	N=9	N=10	N=11	N=12
8	95.83% - 106.25%	96.25% - 105.56%	96.59% - 105.00%	96.88% - 104.55%
16	95.14% - 106.25%	95.63% - 105.56%	96.02% - 105.00%	96.35% - 104.55%
32	94.79% - 106.25%	95.31% - 105.56%	95.74% - 105.00%	96.09% - 104.55%

35.1.6 Modem Controller

The Modem Controller provides the modem control function. Furthermore, an auto flow control function is provided to reduce the software management effort.

The flow control of UART can be achieved by the RTS/CTS handshaking. Without the flow control, overrun errors may occur when the data transmission rate exceeds the data consumption rate. The flow control guarantees that the data transmission will not proceed unless the receiver has enough space to accept the data.

The auto flow control function of ATCUART100 comprises auto-RTS and auto-CTS. The former is for the incoming data while the latter is for the outgoing data.

With auto-RTS, the RTS output of the controller should be connected to the CTS input of the other end of the UART connection.

- When the receiver FIFO $\geqslant$ n – 1, the RTS is de-asserted.
- When the receiver FIFO < n – 1, the RTS is automatically asserted to request the other end of the UART connection to send more data.
- n is the depth of the FIFO.

With auto-CTS, the CTS input of the controller is connected to the RTS output of the other end of the UART connection. The UART controller waits for the assertion of CTS before sending the next character. To stop the controller from transmitting the next character, CTS must be de-asserted before the STOP bit of the current character.

35.1.7 Loopback Mode

The ATCUART100 provides a loopback mode for diagnostic testing without connecting an external device. When the loopback mode is enabled, the behavior of the controller is as follows:

- The output signals (uart_sout, modem_rtsn, modem_dtrn, uart_out1n, uart_out2n) are disconnected from the TX Controller and driven HIGH to avoid confusing the other end of the serial connection in case the connection exists.
- The input signals (uart_sin, modem_ctsn, modem_dsrn, modem_rin, modem_dcsn) are disconnected from the RX Controller and ignored.
- The TX Controller output values originally intended for the uart_sout output signals is routed internally to replace the input signal of uart_sin for the RX Controller, so every bit sent by the TX Controller is looped back and received by the RX controller.
- The Modem Controller output values which are intended for the (modem_rtsn, modem_dtrn, uart_out1n, and uart_out2n) signals are routed internally to replace the input signals of (modem_ctsn, modem_dsrn, modem_rin, and modem_dcsn) for the Modem Controller.

The value written to the Transmitter Holding Register will be internally received by the Receiver Buffer Register and the value written to the Modem Control Register (bit3~bit0) will be routed to the Modem Status Register (bit7~bit4). The entire bit transmission path is exercised in the loopback mode, only the input/output port is isolated from the diagnostic activity. The relationship between the bit fields in the Modem Control Register and those in the Modem Status Register are illustrated in Figure 35-2 Relationship between MCR and MSR in the Loopback Mode.

Figure 35-2 Relationship between MCR and MSR in the Loopback Mode

Register	Bit 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Modem Control Register					OUT2	OUT1	RTS	DTR
Modem Status Register	DCD	RI	DSR	CTS				
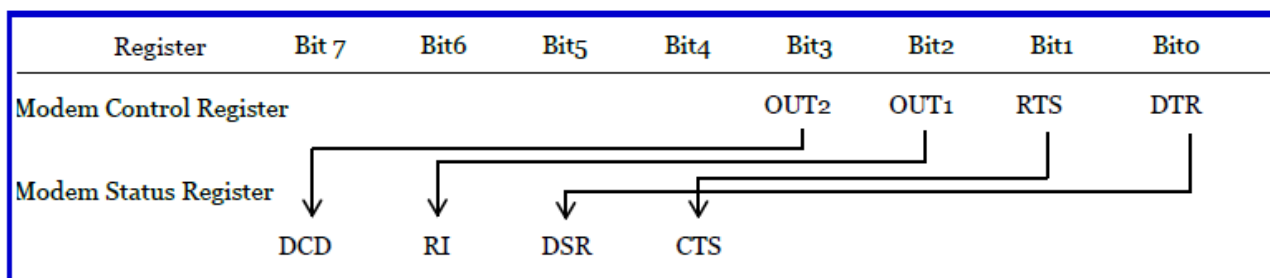
35.1.8 DMA Operation

The ATCUART100 controller provides two pairs of hardware handshake signals to work with a DMA controller (e.g. AndeShape™ ATCDMAC100) for data transfer. One pair is for data reception; the other pair is for data transmission.

When the RX FIFO reaches the threshold set by the Receiver FIFO Trigger Level (RFIFOT) field of the FIFO Control Register, the UART controller will assert dma_rx_req to request a data transfer. The DMA controller should then transfer a character1 from the RX FIFO followed by asserting dma_rx_ack. Next, the UART controller de-asserts dma_rx_req and the DMA controller de-asserts dma_rx_ack. The UART controller will assert dma_rx_req again unless the RX FIFO is empty.

When the TX FIFO reaches the threshold set by the Transmitter FIFO Trigger Level (TFIFOT) field of the FIFO Control Register, the UART controller will assert dma_tx_req to request a data transfer. The DMA controller should then transfer a character2 to the TX FIFO followed by asserting dma_tx_ack. Next, the UART controller de-asserts dma_tx_req and the DMA controller de-asserts dma_tx_ack. The UART controller will assert dma_tx_req again unless the TX FIFO is full.

Note！

- For each dma_rx_req, the DMA controller should read exactly one character to avoid RX FIFO underflow, i.e., the DMA burst size should be 1.
- For each dma_tx_req, the DMA controller should write exactly one character to avoid TX FIFO overflow, i.e., the DMA burst size should be 1.

35.2 Signal Description

Table 35-2 Signal Description of ATCUART100 describes the input/output (I/O) signals of ATCUART100.

Table 35-2 Signal Description of ATCUART100

Name	I/O Type	Description
APB Bus Interface		
pclk	I	AMBA APB clock
presetn	I	AMBA APB reset signal; active low
psel	I	AMBA APB slave select signal from the APB decoder
penable	I	AMBA APB enable signal
pwrite	I	AMBA APB transfer direction signal This signal indicates a write access when driven HIGH and a read access when driven LOW.
paddr[4:2]	I	AMBA APB address bus
pwdata[31:0]	I	AMBA APB write data bus
prdata[31:0]	O	AMBA APB read data bus
UART Signals		
uclk[3]	I	UART clock source
urstn	I	UART system reset; active low
uart_sin	I	UART serial input data When there is no communication, the uart_sin stays at 1.

uclk[3] and urstn only exist when ATCUART100_UCLK_PCLK_SAME is not defined. See Dual I/O Mode

Name	I/O Type	Description
uart_sout	O	UART serial output data When there is no communication, the uart_sout stays at 1.
uart_out1n	O	UART user-defined output 1; active low
uart_out2n	O	UART user-defined output 2; active low
uart_intr	O	UART interrupt signal
Modem Signals		
modem_rin	I	Modem ring indicator; active low
modem_dcdn	I	Modem data carrier detect; active low
modem_dsrn	I	Modem data set ready; active low
modem_ctsn	I	Modem clear to send (CTS); active low
modem_dtrn	O	Modem data terminal ready; active low
modem_rtsn	O	Modem request to send (RTS); active low
DMA Signals		
dma_tx_req	O	DMA TX request
dma_rx_req	O	DMA RX request
dma_tx_ack	I	DMA TX acknowledgement
dma_rx_ack	I	DMA RX acknowledgement

35.3 Programming Model

35.3.1 Summary of Registers

Table 35-3 Summary of Registers

Offset	Name	Description
+0x00	IdRev	ID and Revision Register
+0x04~ 0x0C	-	Reserved
+0x10	Cfg	Hardware Configure Register
+0x14	OSCR	Over Sample Control Register
+0x18~0x1C	-	Reserved
+0x20		DLAB = 0
	RBR	Receiver Buffer Register (Read only)
	THR	Transmitter Holding Register (Write only)
		DLAB = 1
	DLL	Divisor Latch LSB
+0x24		DLAB = 0
	IER	Interrupt Enable Register
		DLAB = 1
	DLM	Divisor Latch MSB
+0x28	IIR	Interrupt Identification Register (Read only)
	FCR	FIFO Control Register (Write only)
+0x2C	LCR	Line Control Register
+0x30	MCR	Modem Control Register
+0x34	LSR	Line Status Register
+0x38	MSR	Modem Status Register
+0x3C	SCR	Scratch Register

The following sections describe ATCUART100 registers in detail. The abbreviations for the Type column are summarized below:

RO: read only

WO: write only

R/W: readable and writable

W1C: write 1 to clear

RC: read clear

35.3.2 ID and Revision Register (0x00)

This register holds the ID number and the revision number. The reset values of the two revision fields are revision dependent.This register holds

the ID number and the revision

Table 35-4 ID and Revision Register

Name	Bit	Type	Description	Reset
ID	31:16	RO	ID number for ATCUART100	0x0201
Major	15:4	RO	Major revision number	Revision dependent
Minor	3:0	RO	Minor revision number	Revision dependent

35.3.3 Hardware Configure Register (0x10)

Table 35-5 Hardware Configure Register

Name	Bit	Type	Description	Reset
-	31:2	-	Reserved	0x0
FIFO_DEPTH	1:0	RO	The depth of RXFIFO and TXFIFO 0: 16-byte FIFO 1: 32-byte FIFO 2: 64-byte FIFO 3: 128-byte FIFO	Configuration dependent

35.3.4 Over Sample Control Register (0x14)

The Over Sample Control Register defines the clock ratio between the sampling clock and the baud rate. Please see Transmitter for details.

Table 35-6 Over Sample Control Register

Name	Bit	Type	Description	Reset
-	31:5	-	Reserved	0x0
OSC	4:0	R/W	Over-sample control The value must be an even number; any odd value writes to this field will be converted to an even value. **OSC = 0**: The over-sample ratio is 32 **OSC ≤ 8**: The over-sample ratio is 8 **8 < OSC < 32**: The over sample ratio is OSC	0x10

35.3.5 Receiver Buffer Register (when DLAB = 0) (0x20)

The RBR has two modes, the FIFO mode and the BUFFER mode. Bit0 of the FIFO Control Register (FIFOE) controls the selection between these two modes. When FIFOE is 1 (FIFO mode), the RBR is a RXFIFO. The

depth of RXFIFO is configurable by the macro ATCUART100_FIFO_DEPTH, see The Depth of FIFO for details. When FIFOE is 0 (BUFFER mode), the RBR is just a byte buffer.

Table 35-7 Receiver Buffer Register (when DLAB = 0)

Name	Bit	Type	Description	Reset
-	31:8	-	Reversed	0x0
RBR	7:0	RO	Receive data read port	0x0

35.3.6 Transmitter Holding Register (when DLAB = 0) (0x20)

The THR has two modes, the FIFO mode and the BUFFER mode. FCR bit0 (FIFOE) controls the selection between these two modes.When FIFOE is 1 (FIFO mode), the THR is a TXFIFO. The depth of TXFIFO is configurable by the macro ATCUART100_FIFO_DEPTH, see The Depth of FIFO for details. When FIFOE is 0 (BUFFER mode), the THR is a byte buffer.

Table 35-8 Transmitter Holding Register (when DLAB = 0)

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
THR	7:0	WO	Transmit data write port	0x0

35.3.7 Interrupt Enable Register (when DLAB = 0) (0x24)

The following table describes each field of the Interrupt Enable Register.

Table 35-9 Interrupt Enable Register (when DLAB = 0)

Name	Bit	Type	Description	Reset
-	31:4	-	Reserved	0x0
EMSI	3	R/W	Enable modem status interrupt	0x0
			The interrupt asserts when the status of one of the following occurs:	
			The status of modem_rin, modem_dcdn, modem_dsrn or modem_ctsn (If the auto-cts mode is disabled) has been changed.	
			If the auto-cts mode is enabled (MCR bit4 (AFE) = 1), modem_ctsn would be used to control the transmitter.	
ELSI	2	R/W	Enable receiver line status interrupt	0x0
ETHEI	1	R/W	Enable transmitter holding register interrupt	0x0
ERBI	0	R/W	Enable received data available interrupt and the character timeout interrupt	0x0
			0: Disable	
			1: Enable	

35.3.8 Divisor Latch LSB (when DLAB = 1) (0x20)

The Divisor Latch holds the divisor value for generating the sampling clock from the UART clock source (uclk). The size of the Divisor Latch is 16 bits (two bytes) and this register holds the least significant byte of the Divisor Latch. Please see Baud Rate Generator for details. The valid value of the Divisor Latch should be between 1 and 65535 (2^{16}-1), inclusive.

Table 35-10 Divisor Latch LSB (when DLAB = 1)

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
DLL	7:0	R/W	Least significant byte of the Divisor Latch	0x1

35.3.9 Divisor Latch MSB (when DLAB = 1) (0x24)

The Divisor Latch holds the divisor value for generating the sampling clock from the UART clock source (uclk). The size of the Divisor Latch is 16 bits (two bytes) and this register holds the most significant byte of the Divisor Latch. Please see Baud Rate Generator for details. The valid value of the Divisor Latch should be between 1 and 65535 (2^{16}-1), inclusive.

Table 35-11 Divisor Latch MSB (when DLAB = 1)

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
DLM	7:0	R/W	Most significant byte of the Divisor Latch	0x0

35.3.10 Interrupt Identification Register (0x28)

Table 35-12 Interrupt Identification Register

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
FIFOED	7:6	RO	FIFOs enabled These two bits are 1 when bit 0 of the FIFO Control Register (FIFOE) is set to 1.	0x0
-	5:4	-	Reserved	0x0
INTRID	3:0	RO	Interrupt ID See Table 13 for encodings	0x1

Table 35-13 Interrupt Control Table

Interrupt Identification Register					Interrupt Type	Interrupt Source Description	Interrupt Reset Method
Bit3	Bit2	Bit1	Bit0	Priority Level			
0	0	0	1	None	None	None	None
0	1	1	0	1	Receiver line status	Overrun errors, parity errors, framing errors, or line breaks	Read the Line Status Register (LSR)
0	1	0	0	2	Received data available	If FIFOE is disabled, there is one received data available in the RBR. If FIFOE is enabled, the numbers of received data available reach the trigger level (RFIFOT). The interrupt signal will stay active until the number of data available becoming smaller than the trigger level.	Read the Receiver Buffer Register (RBR)

1	1	0	0	2	Character timeout	When FIFOE is enabled and no character have been removed from or input to receive FIFO and there is at least one character in receive FIFO during the last four character times.	Read the Receiver Buffer Register (RBR)
0	0	1	0	3	Transmitter Holding Register empty	If FIFOE is disabled, the 1-byte THR is empty. If FIFOE is enabled, the whole 16-byte transmit FIFO is empty.	Write the Transmitter Holding Register (THR) or Read the Interrupt Identification Register (IIR).
0	0	0	0	4	Modem status	The Modem Status Register (MSR) bit[3:0] is not 0. One of the following events occurred: Clear To Send (CTS), Data Set Ready (DSR), Ring Indicator (RI), or Data Carrier Detect (DCD)	Read the Modem Status Register (MSR)

Note！

When multiple events would trigger interrupts at the same time, the priority level determines the value for the INTRID field.

35.3.11 FIFO Control Register (0x28)

Table 35-14 FIFO Control Register

Name	Bit	Type	Description	Reset
-	31:8		Reserved	0x0
RFIFOT	7:6	WO	Receiver FIFO trigger level Please refer to Table 15	0x0
TFIFOT	5:4	WO	Transmitter FIFO trigger level Please refer to Table 16	0x0
DMAE	3	WO	DMA enable 0: Disable 1: Enable	0x0
TFIFORST	2	WO	Transmitter FIFO reset Write 1 to clear all bytes in the TXFIFO and resets its counter. The Transmitter Shift Register is not cleared. This bit will automatically be cleared.	0x0
RFIFORST	1	WO	Receiver FIFO reset Write 1 to clear all bytes in the RXFIFO and resets its counter. The Receiver Shift Register is not cleared. This bit will automatically be cleared.	0x0
FIFOE	0	WO	FIFO enable Write 1 to enable both the transmitter and receiver FIFOs. The FIFOs are reset when the value of this bit toggles.	0x0

Table 35-15 Receive FIFO Trigger Level

RFIFOT Value	RXFIFO Trigger Level			
	16-byte RXFIFO	32-byte RXFIFO	64-byte RXFIFO	128-byte RXFIFO
0	Not empty	Not empty	Not empty	Not empty
1	More than 3	More than 7	More than 15	More than 31
2	More than 7	More than 15	More than 31	More than 63
3	More than 13	More than 27	More than 55	More than 111

Table 35-16 Transmit FIFO Trigger Level

TFIFOT	TXFIFO Trigger Level			
Value	16-byte TXFIFO	32-byte TXFIFO	64-byte TXFIFO	128-byte TXFIFO
0	Not full	Not full	Not full	Not full
1	Less than 12	Less than 24	Less than 48	Less than 96
2	Less than 8	Less than 16	Less than 32	Less than 64
3	Less than 4	Less than 8	Less than 16	Less than 32

35.3.12 Line Control Register (0x2C)

Table 35-17 Line Control Register

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
DLAB	7	R/W	Divisor latch access bit	0x0
BC	6	R/W	Break control	0x0
SPS	5	R/W	Stick parity	0x0
			1: parity bit is constant 0 or 1, depending on bit4 (EPS).	
			0: disable the sticky bit parity.	
			Please refer to Table 18.	
EPS	4	R/W	Even parity select	0x0
			1: even parity (an even number of logic-1 is in the data and parity bits)	
			0: old parity.	
			Please refer to Table 18.	
PEN	3	R/W	Parity enable	0x0
			When this bit is set, a parity bit is generated in transmitted data before the first STOP bit and the parity bit would be checked for the received data.	
			Please refer to Table 18.	
STB	2	R/W	Number of STOP bits	0x0
			0: 1 bits	
			1: the number of STOP bit is based on the WLS setting	
			When WLS = 0, STOP bit is 1.5 bits	
			When WLS = 1, 2, 3, STOP bit is 2 bits	
WLS	1:0	R/W	Word length setting	0x0
			0: 5 bits	
			1: 6 bits	
			2: 7 bits	
			3: 8 bits	

Table 35-18 Parity Bit Selection

PEN (bit3)	SPS (bit5)	EPS (bit4)	Parity Bit
0	X	X	No parity bit
1	0	0	Parity is odd
1	0	1	Parity is even
1	1	0	Parity bit is always 1
1	1	1	Parity bit is always 0

35.3.13 Modem Control Register (0x30)

This register controls the output of the modem status signals as well as the loopback mode and auto flow control.

Table 35-19 Modem Control Register

Name	Bit	Type	Description	Reset
-	31:6	-	Reserved	0x0
AFE	5	R/W	Auto flow control enable 0: Disable 1: the auto-CTS and auto-RTS setting is based on the 　RTS bit setting: 　　When RTS = 0, auto-CTS only 　　When RTS = 1, auto-CTS and auto-RTS	0x0
LOOP	4	R/W	Enable loopback mode 0: Disable 1: Enable	0x0
OUT2	3	R/W	User-defined output 2 This bit controls the uart_out2n output. 0: the uart_out2n output signal will be driven HIGH 1: the uart_out2n output signal will be driven LOW	0x0
OUT1	2	R/W	User-defined output 1 This bit controls the uart_out1n output. 0: the uart_out1n output signal will be driven HIGH 1: the uart_out1n output signal will be driven LOW	0x0
RTS	1	R/W	Request to send This bit controls the modem_rtsn output. 0: the modem_rtsn output signal will be driven HIGH 1: the modem_rtsn output signal will be driven LOW	0x0
DTR	0	R/W	Data terminal ready	0x0

Name	Bit	Type	Description	Reset
			This bit controls the modem_dtrn output.	
			0: the modem_dtrn output signal will be driven HIGH	
			1: the modem_dtrn output signal will be driven LOW	

35.3.14 Line Status Register (0x34)

This register reports the status of the transmitter and the receiver.

Table 35-20 Line Status Register

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
ERRF	7	RO	Error in RXFIFO In the FIFO mode, this bit is set when there is at least one parity error, framing error, or line break associated with data in the RXFIFO. It is cleared when this register is read and there is no more error for the rest of data in the RXFIFO.	0x0
TEMT	6	RO	Transmitter empty This bit is 1 when the THR (TXFIFO in the FIFO mode) and the Transmitter Shift Register (TSR) are both empty. Otherwise, it is zero.	0x1
THRE	5	RO	Transmitter Holding Register empty This bit is 1 when the THR (TXFIFO in the FIFO mode) is empty. Otherwise, it is zero. If the THRE interrupt is enabled, an interrupt is triggered when THRE becomes 1.	0x1

Name	Bit	Type	Description	Reset
LBreak	4	RO	Line break This bit is set when the uart_sin input signal was held LOW for longer than the time for a full-word transmission. A full-word transmission is the transmission of the START, data, parity, and STOP bits. It is cleared when this register is read. In the FIFO mode, this bit indicates the line break for the received data at the top of the RXFIFO.	0x0
FE	3	RO	Framing error This bit is set when the received STOP bit is not HIGH. It is cleared when this register is read. In the FIFO mode, this bit indicates the framing error for the received data at the top of the RXFIFO.	0x0
PE	2	RO	Parity error This bit is set when the received parity does not match with the parity selected in the LCR[5:4]. It is cleared when this register is read. In the FIFO mode, this bit indicates the parity error for the received data at the top of the RXFIFO.	0x0
OE	1	RO	Overrun error This bit indicates that data in the Receiver Buffer Register (RBR) is overrun.	0x0
DR	0	RO	Data ready. This bit is set when there are incoming received data in the Receiver Buffer Register (RBR). It is cleared when all of the received data are read.	0x0

35.3.15 Modem Status Register (0x38)

Table 35-21 Modem Status Register (0x38)

Name	Bit	Type	Description	Reset
-	31:8	–	Reserved	0x0
DCD	7	RO	Data carrier detect 0: the modem_dcdn input signal is HIGH. 1: the modem_dcdn input signal is LOW.	0x0
RI	6	RO	Ring indicator 0: The modem_rin input signal is HIGH. 1: the modem_rin input signal is LOW.	0x0
DSR	5	RO	Data set ready 0: The modem_dsrn input signal is HIGH. 1: The modem_dsrn input signal is LOW.	0x0
CTS	4	RO	Clear to send 0: The modem_ctsn input signal is HIGH. 1: The modem_ctsn input signal is LOW.	0x0
DDCD	3	RC	Delta data carrier detect This bit is set when the state of the modem_dcdn input signal has been changed since the last time this register is read. Otherwise, it is zero.	0x0
TERI	2	RC	Trailing edge ring indicator This bit is set when the state of the modem_rin input signal has been changed from LOW to HIGH since the last time this register is read.	0x0
DDSR	1	RC	Delta data set ready This bit is set when the state of the modem_dsrn input signal has been changed since the last time this register is read.	0x0
DCTS	0	RC	Delta clear to send This bit is set when the state of the modem_ctsn input signal has been changed since the last time this register is read.	0x0

35.3.16 Scratch Register (0x3C)

Table 35-22 Scratch Register

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	0x0
SCR	7:0	R/W	An one-byte storage register with no UART related function; available to software with no usage restrictions.	0x0

35.4 Hardware Configuration Options

35.4.1 The Depth of FIFO

Define the depth of TXFIFO and RXFIFO. The depth could be 16, 32, 64 and 128. The following example defines the depth of FIFO to 16-byte.

`define ATCUART100_FIFO_DEPTH_16

35.4.2 The Same Clock Source

If the uclk and the pclk are identical clocks, define this macro to remove the synchronization logic and the extra latencies associated with it. In addition, the uclk input port is omitted such that only the pclk input port remains (since the two clocks are the same one).

`define ATCUART100_UCLK_PCLK_SAME

35.5 Programming Sequence

35.5.1 UART Setup

The following programming sequence sets up the UART for:

● Baud rate: 38400
● Word length: 8-bit
● Parity: none
● Number of STOP bits: 1
● Flow control: none

1. Set baud rate.

Set the DLAB bit of the Line Control Register to 1.

Set DLL and DLM registers to select the desired baud rate for the UART.

- The divisor value = the frequency of uclk / (desired baud rate x OSCR)
- The default value for OSCR is 16

2. Set the DLAB bit of the Line Control Register back to 0.

3. Set the Line Control Register.

    Disable the parity: Set PEN = 0;

    Set STOP bits to 1: Set STB = 0;

    Set word length to 8-bit: Set WLS = 3.

    The following programming sequence transmits data through UART without/with DMA:

35.5.2 Data Transfer without DMA

1. Set FIFOE in the FIFO Control Register to enable the FIFO.

2. Wait for FIFO empty by looping until the THRE bit in the Line Status Register becomes 1.

3. Write at most FIFO_DEPTH bytes of data to THR.

4. If there are more data to send, go to .

35.5.3 Data Transfer with DMA

1. Program the FIFO Control Register with the following values:

    Specify the threshold: Set Transmitter FIFO Trigger Level (TFIFOT);

    Enable DMA: Set DMAE = 1;

    Enable FIFO: Set FIFOE = 1.

2. Set a DMA controller to write data to the THR:

    Set THR as the DMA destination.

    Set burst size to 1.

    Set destination data width to one byte.

    Enable hardware handshaking.

    Set DMA terminal count interrupt.

    Set other DMA transfer control registers.

3. The DMA data transfer performs in the background. You will receive a terminal count interrupt from the DMA controller once the data transfer completes.

4. If completion of the data transmission must be guaranteed, wait for the UART TXFIFO empty by checking the THRE bit in the Line Status Register.

35.5.4 Receiving Data

The following programming sequence receives data through UART without/with DMA:

35.5.5 Data Receive without DMA

1. Set FIFOE in the FIFO Control Register to enable the FIFO.

2. Wait until the Data Ready bit (the DR bit) of the Line Status Register becomes 1.

3. Read one byte of data from the RBR.

4. Go to read more data.

35.5.6 Data receive with DMA

5. Program the FIFO Control Register with the following values:

    Specify the threshold: Set Receiver FIFO Trigger Level (RFIFOT);

    Enable DMA: Set DMAE = 1;

    Enable FIFO: Set FIFOE = 1.

6. Set a DMA controller to read data from the RBR:

    Set RBR as the DMA source.

    Set burst size to 1.

    Set source data width to one byte.

    Enable hardware handshaking.

    Set DMA terminal count interrupt.

    Set other DMA transfer control registers.

7. The DMA data transfer performs in the background. You will receive a terminal count interrupt from the DMA controller once the data transfer completes.