Loongson 3A5000/3B5000 Processor Reference Manual

Multicore Processor Architecture, Register Descriptions and System Software Programming Guide

Loongson Technology Corporation Limited

Version 1.03

Table of Contents

List of Figures

List of Tables

- GPIO output enable register
- Address access method
- Configuration register instruction access method
- Register description
- Other function configuration register
- Node counter register
- Output enable register
- Input/Output Register
- Interrupt control register
- Table of GPIO multiplexing function
- Interrupt control register
- List of configuration information for the instruction set functions implemented in the 3A5000
- Shared Cache lock window register configuration
- Shared Cache configuration register (`SC_CONFIG`)
- Registers related to inter-processor interrupt and their functional descriptions
- List of inter-processor interrupt and communication registers for processor core 0
- List of inter-processor interrupt and communication registers for processor core 1
- List of inter-processor interrupt and communication registers for processor core 2
- List of inter-processor interrupt and communication registers for processor core 3
- List of inter-processor interrupt and communication registers for the current processor core
- Processor core inter-processor communication registers
- Processor core inter-processor communication registers
- Interrupt control register
- I/O control register address
- Description of the interrupt routing register
- Interrupt routing register address
- Processor core private interrupt status register
- Other function configuration register
- Extended I/O interrupt enable register
- Extended I/O interrupt auto-rotation enable register
- Extended I/O interrupt interrupt status register
- Extended I/O interrupt status register for each processor core
- Description of the interrupt pin routing register
- Interrupt routing register address
- Description of the interrupt destination processor core routing register
- Interrupt destination processor core routing register address
- Interrupt destination node mapping method configuration
- Extended I/O interrupt status register for the current processor core
- Extended I/O interrupt trigger register
- Description of temperature collection register

- Extended I/O interrupt trigger register
- Description of high/low temperature interrupt register
- Description of high-temperature underclock control register
- Description of temperature status detection and control register
- Description of temperature sensor configuration register
- Description of temperature sensor data register
- Software-visible parameter list of the memory controller
- Memory controller 0 error status observation register
- Memory controller 1 error status observation register
- Pin signals related to HyperTransport bus
- Commands that can be received by the HyperTransport receiver
- Commands that will be transported in both modes
- Default address window layout of the 4 HyperTransport interfaces
- Address window distribution inside the HyperTransport interface of the Loongson 3 processor
- Address window provided in the HyperTransport interface of the Loongson 3A5000 processor
- Software visible registers in the HT controller
- Bridge control register
- Definition of command, capabilities pointer, capability ID registers
- Definition of link config, link control registers
- Definition of revision id, link freq, link error, link freq capregisters
- Definition of feature registers
- Error Retry Control Register
- Retry Count Register
- Revision ID Register
- Interrupt capability register
- Interrupt dataport register
- Interrupt intrInfo register 1
- Interrupt intrInfo register 2
- Interrupt vectors mapping
- HT bus interrupt vector register definition 1
- HT bus interrupt vector register definition 2
- HT bus interrupt vector register definition 3
- HT bus interrupt vector register definition 4
- HT bus interrupt vector register definition 5
- HT bus interrupt vector register definition 6
- HT bus interrupt vector register definition 7
- HT bus interrupt vector register definition 8
- HT bus interrupt enable register definition 1
- HT bus interrupt enable register definition 2
- HT bus interrupt enable register definition 3
- HT bus interrupt enable register definition 4

- HT bus interrupt enable register definition 5
- HT bus interrupt enable register definition 6
- HT bus interrupt enable register definition 7
- HT bus interrupt enable register definition 8
- Link Train Register
- Definition of busreceive address window 0 enable (external access) register
- Definition of HT bus receive address window 0 base address (external access) register
- Definition of HT bus receive address window 1 enable (external access) register
- Definition of bus receive address window 1 base address (external access) register
- Definition of bus receive address window 2 enable (external access) register
- Definition of HT bus receive address window 2 base address (external Access) register
- Definition of HT bus receive address window 3 enable (external access) register
- Definition of HT bus receive address window 3 base address (external access) register
- Definition of HT bus receive address window 4 enable (external access) register
- Definition of HT bus receive address window 4 base address (external access) register
- Definition of configuration space extended address translation register
- Definition of extended address translation register
- HT bus POST address window 0 enable (internal access)
- HT bus POST address window 0 base address (internal access)
- HT bus POST address window 1 enable (internal access)
- HT bus POST address window 1 base address (internal access)
- HT bus prefetchable address window 0 enable (internal access)
- HT bus prefetchable address window 0 base address (internal access)
- HT bus prefetchable address window 1 enable (internal access)
- HT bus prefetchable address window 1 base address (internal access)
- HT bus uncache address window 0 enable (internal access)
- HT bus uncache address window 0 base address (internal access)
- HT bus uncache address window 1 enable (internal access)
- HT bus uncache address window 1 base address (internal access)
- HT bus uncache address window 2 enable (internal access)
- HT Bus uncache Address Window 2 Base Address (Internal Access)
- HT bus uncache address window 3 enable (internal access)
- HT Bus uncache address window 3 base address (internal access)
- Definition of HT bus P2P address window 0 enable (external access) register
- Definition of HT bus P2P address window 0 base address (external access) register
- Definition of HT bus P2P address window 1 enable (external access) register
- Definition of HT bus P2P address window 1 base address (external access) register
- Definition of controller parameter configuration register 0
- Definition of controller parameter configuration register 1
- Receive diagnostic register
- PHY status register

- Transport command Cache size register
- Transport data Cache size register
- Transport Cache debug register
- Receive buffer initialization configuration register
- Training 0 timeout short counter register
- Training 0 timeout long counter register
- Training 1 counter register
- Training 2 counter register
- Training 3 counter register
- Software frequency configuration register
- PHY impedance matching control register
- PHY configuration register
- Link initialization debug register
- LDT debug register 1
- LDT debug register 2
- LDT debug register 3
- LDT debug register 4
- LDT debug register 5
- LDT debug register 6
- HT TX POST ID WIN0
- HT TX POST ID WIN1
- HT TX POST ID WIN2
- HT TX POST ID WIN3
- HT RX INT TRANS Lo
- HT RX INT TRANS Hi
- Data transport register
- Interrupt enable register
- Interrupt identity register
- Table of interrupt control function
- FIFO control register
- Line control register
- MODEM Control Register
- MODEM state register
- Receive FIFO counter
- Transport FIFO counter
- Frequency division latchs 1
- Frequency division latchs 2
- Frequency division latchs 3
- Control register
- State register
- Transport data register

- External register
- Frequency division factor
- Parameter control register
- Chip select control register
- Timing control register
- Custom controller register
- Custom command register
- Custom data register 0
- Custom data register 1
- Custom timing register 0
- Custom timing register 1
- Custom timing register 2
- Frequency division latch low-order byte register
- Frequency division latch high-order byte register
- Control register
- Transport data register
- Receive data register
- Command control register
- State register
- Slave device controller register
- HT RX INT TRANS Lo
- HT RX INT TRANS Hi
- Other function configuration register
- Processor core inter-processor communication registers

About this manual

Copyright Statement

Disclaimer

This document provides only periodic information, and the contents contained may be updated at any time without notice, depending on the actual situation of the product. Loongson Technology Corporation Limited is not responsible for any direct or indirect damage aused by the improper use of the document.

Loongson Technology Corporation Limited

Building No.2, Loongson Industrial Park,
Zhongguancun Environmental Protection Park, Haidian District, Beijing

Tel: 010-62546668

Fax: 010-62600826

Reading Guide

This manual introduces the Loongson 3A5000/3B5000 multicore processor architecture and register descriptions. It provides detailed descriptions of the chip system architecture, functions and configurations of the main modules, register lists and bit fields.

Translator's Note

These documents were translated by Yanteng Si and Feiyang Chen.

This is the translation of https://github.com/loongson/LoongArch-Documentation/releases/latest/download/Loongson-3A5000-usermanual-v1.02-CN.pdf.

Due to the limited knowledge of the translators, there are some inevitable errors and omissions existing in this document, please feel free to correct.

License

Contributors

Since the release of the project, we have gotten several errata and content changes donated. Here are all the people who have contributed to LoongArch Documentation as an open source project. Thank you everyone for helping make this a better book for everyone.

The contributors are listed in alphabetical order.

```
Chenghua Xu <xuchenghua@loongson.cn>
Feiyang Chen <chenfeiyang@loongson.cn>
FreeFlyingSheep <fyang.168.hi@163.com>
qmuntal <quimmuntal@gmail.com>
wangguofeng <wangguofeng@loongson.cn>
Wu Xiaotian <wuxiaotian@loongson.cn>
Wu Xiaotian <yetist@gmail.com>
Xi Ruoyao <xry111@mengyan1223.wang>
Yang Yujie <yangyujie@alumni.sjtu.edu.cn>
Yang Yujie <yangyujie@loongson.cn>
Yanteng Si <siyanteng@loongson.cn>
```

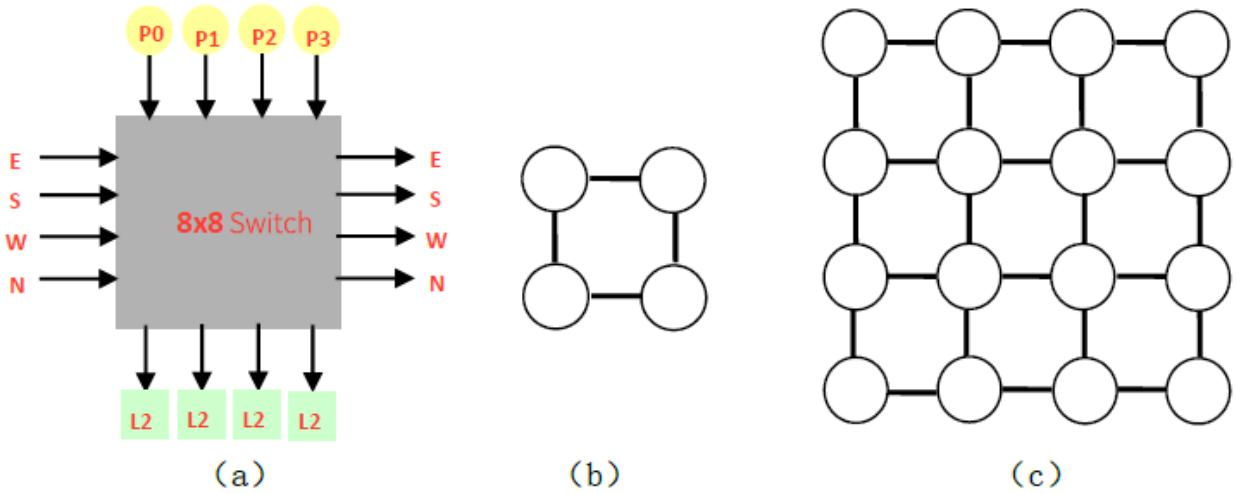
Chapter 1. Introduction

1.1. Introduction to the Loongson Family of Processors

Loongson processors mainly include three series. Loongson Series 1 processor adopts 32-bit processor cores and integrates various peripheral interfaces to form application-specific monolithic solutions, which are mainly applied to IOT terminals, instrumentation devices, data acquisition and other fields. Loongson Series 2 processor adopts 32-bit/64-bit processor cores and integrates various peripheral interfaces to form a high-performance low-power SoC chip for network devices, industrial terminals, intelligent manufacturing, etc. Loongson Series 3 processors integrate multiple 64-bit processor cores and necessary storage and IO interfaces on-chip, targeting high-end embedded computers, desktops, servers and other applications.

The Loongson 3 multi-core series processors are designed based on a scalable multi-core interconnect architecture, which integrates multiple high-performance processor cores and a large amount of Level 2 Cache on a single chip, and interconnects multiple chips through high-speed I/O interfaces to form a larger scale system.

The scalable interconnect architecture adopted by Loongson 3 is shown in the figure below. Each node consists of 8 × 8 cross-switches, with each cross-switch connecting four processor cores and four shared caches, and interconnecting with other nodes in four directions: East (E), South (N), West (W), and North (N).



Loongson 3 node and 2-D interconnect structure:
(a) node structure;
(b) 2*2 mesh network connecting 16 processors;
(c) 4*4 mesh network connecting 64 processors.

Figure 1. Loongson 3 System architecture

The node structure of the Loongson 3 is shown in the figure below. Each node has two levels of AXI cross-switches connecting the processor, the shared Cache, the memory controller, and the I/O controller. The first level AXI cross-switch (called X1 Switch) connects the processor and the shared Cache, and the second level cross-switch (called X2 Switch) connects the shared Cache and the memory controller.

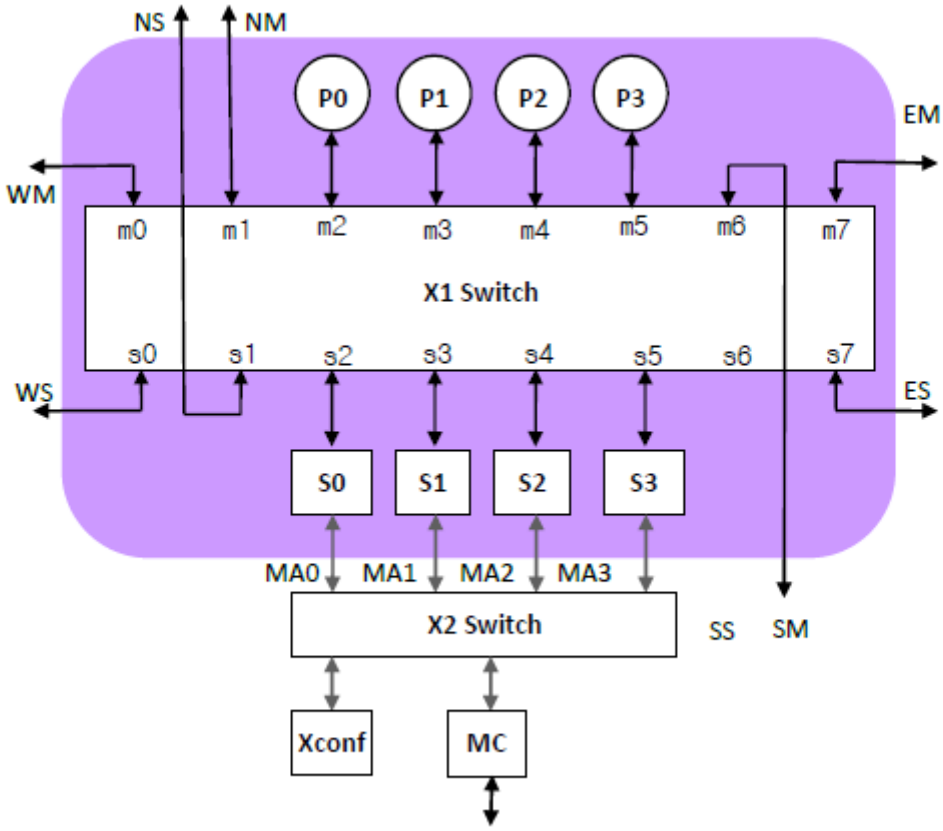


Figure 2. Loongson 3 node structure

In each node, up to 8 × 8 X1 cross switches are connected to four processor cores (P0, P1, P2, P3 in the figure) through four Master ports. The four interleave shared Cache blocks (S0, S1, S2, S3 in the figure) are universally addressed through the four slave ports. Other nodes or I/O nodes in the East, South, West and North directions are connected via four pairs of Master/Slave ports (EM/ES, SM/SS, WM/WS, NM/NS in the figure).

The X2 cross-switch connects four shared Caches via four Master ports, at least one Slave port to a memory controller, and at least one slave port to a configuration module (Xconf) of the cross-switch that is used to configure the address windows of X1 and X2 of this node. Additional memory controllers, I/O ports, can be connected as needed.

1.2. Introduction to Loongson 3A5000/3B5000

The Loongson 3A5000/3B5000 is a quad-core Loongson processor with a stable operating frequency of 2.0-2.5GHz.

The main technical features are as follows:

- On-chip integration of four 64-bit quad-launch superscalar LA464 processor cores.
- Peak floating-point computing power 160GFLOPS@2.5GHz.
- On-chip integration of 16MB of split shared tertiary Cache.
- Maintenance of Cache consistency for multi-core and I/O DMA accesses via directory protocol.
- On-chip integration of two 72-bit DDR4 controllers with ECC, supporting DDR4-3200.
- On-chip integration of two 16-bit HyperTransport controllers (hereinafter referred to as HT) with a maximum bus frequency of 3.2 GHz.

- Each group of 16-bit HT ports can be split into two groups of 8-bit HT ports for use.
- 2 I2C, 1 UART, 1 SPI, 16 GPIO interfaces on-chip

The architecture of the Loongson 3A5000/3B5000 is designed to increase the shared Cache capacity based on the 3A4000 and supports 16-way interconnect.

The Loongson 3B5000 supports consistent interconnects on the HT0 interface compared to the 3A5000, with special filtering based on server scenario requirements. There is no difference in the other parts from the hardware and software perspective, and they are collectively referred to as 3A5000.

The overall architecture of the Loongson 3A5000 chip is based on multi-level interconnects and is shown in the figure below.

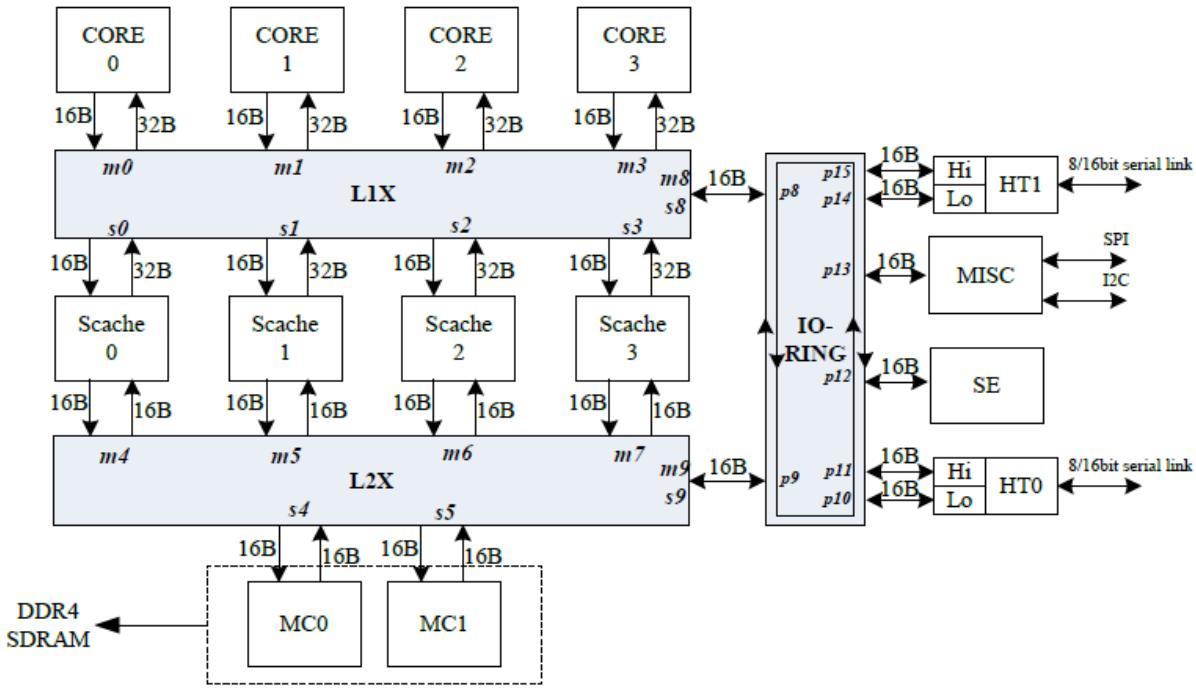


Figure 3. Loongson 3A5000 chip structure

The first level of interconnect uses a 5 × 5 frequency division switch to connect four LA464 cores (as masters), four shared Cache modules (as slaves), and one I/O port to I/O-RING (The I/O port uses one Master and one Slave).

The second level interconnect uses a 5 × 3 cross-switch to connect four shared Cache modules (as masters), two DDR3/4 memory controllers, and one I/O port to the I/O-RING.

The I/O-RING contains 8 ports and the connections include 4 HT controllers, MISC module, SE module and two level cross switches. The two HT controllers (lo/hi) share the 16-bit HT bus, which is used as two 8-bit HT buses, or lo can occupy the 16-bit HT bus exclusively. A DMA controller is integrated into the HT controller, which is responsible for the DMA control of the I/O and the maintenance of inter-chip consistency.

All of these interconnect structures use read/write separated data channels with a 128-bit data channel width operating at the same frequency as the processor core to provide high-speed on-chip data transport. In addition, a one-level cross-switch connects the four processor cores to the SCache with a 256-bit read data channel to increase the read bandwidth of the on-chip processor cores accessing the SCache.

Chapter 2. System Configuration and Control

2.1. Chip Operating Modes

Depending on the structure of the constituent systems, the Loongson 3A5000 consists of two main operating modes. Single-chip mode: The system contains only 1 chip of Loongson 3A5000, which is a symmetric multiprocessor system (SMP). Multi-chip interconnect mode: The system contains 2, 4, or 16 chips of the Loongson 3A5000 interconnected through HT ports to form a non-uniform access multiprocessor system (CC-NUMA).

2.2. Descriptions of Pins

Main control pins include DO_TEST, ICCC_EN, NODE_ID[2:0], CLKSEL[9:0], and CHIP_CONFIG[5:0].

Table 5. Descriptions of control pins

Signal	Pull-up or Pull-down	Description
DO_TEST	Pull-up	1'b1 indicates functional mode 1'b0 indicates test mode
ICCC_EN	Pull-down	1'b1 indicates multi-chip coherent interconnect mode 1'b0 indicates single chip mode
NODE_ID[2:0]		Indicates the processor number in multi-chip coherent interconnect mode

<table>
<tr><th>Signal</th><th>Pull-up or Pull-down</th><th>Description</th></tr>
<tr><td>CLKSEL[9:0]</td><td></td><td>
Table 1. HT clock control
<table>
<tr><th>Signal</th><th>Description</th></tr>
<tr><td>CLKSEL[9]</td><td>1'b1 indicates that HT PLL clock uses CLKSEL[7:6] to control

1'b0 indicates that initial frequency multiplier is 1X and can be reconfigured by software</td></tr>
<tr><td>CLKSEL[8]</td><td>1'b1 indicates that HT PLL uses the SYSCLK clock input

1'b0 indicates that HT PLL uses the differential clock input</td></tr>
<tr><td>CLKSEL[7:6]</td><td>2'b00 indicates that PHY clock frequency is 1.6GHz

2'b01 indicates that PHY clock frequency is 6.4GHz

2'b10 Reserved

2'b11 indicates that PHY clock frequency is 4.8GHz</td></tr>
<tr><td>CLKSEL[5]</td><td>Reserved</td></tr>
<tr><td>CLKSEL[4]</td><td>1'b1 - the reference clock is 25MHz

1'b0 - the reference clock is 100MHz</td></tr>
</table>
Table 2. MEM clock control (clock frequency should be 1/2 of the interface clock)
<table>
<tr><th>CLKSEL[3:2]</th><th>Output Frequency</th></tr>
<tr><td>2'b00</td><td>466MHz</td></tr>
<tr><td>2'b01</td><td>600MHz</td></tr>
<tr><td>2'b10</td><td>Software configuration (PLL clock multiplier is 1.6-3.2GHz)</td></tr>
<tr><td>2'b11</td><td>SYSCLK (100MHz/25MHz)</td></tr>
</table>
Table 3. Main clock control (network and maximum processor core frequency)
<table>
<tr><th>CLKSEL[1:0]</th><th>Output Frequency</th></tr>
<tr><td>2'b00</td><td>1GHz</td></tr>
<tr><td>2'b01</td><td>2GHz</td></tr>
<tr><td>2'b10</td><td>Software configuration (PLL clock multiplier is 4.8-6.4GHz)</td></tr>
<tr><td>2'b11</td><td>SYSCLK (100MHz/25MHz)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Signal</th><th>Pull-up or Pull-down</th><th>Description</th></tr>
<tr><td>CHIP_CONFIG[5:0]</td><td></td><td>
Table 4. Chip configuration control
<table>
<tr><th>CHIP_CONFIG[0]</th><th>SE function enable</th></tr>
<tr><td>CHIP_CONFIG[1]</td><td>Default HT Gen1 Mode</td></tr>
<tr><td>CHIP_CONFIG[2]</td><td>NodeID[3]</td></tr>
<tr><td>CHIP_CONFIG[3]</td><td>HT1-hi enters consistency mode by default</td></tr>
<tr><td>CHIP_CONFIG[4]</td><td>HT1-lo enters consistency mode by default and is used to support 8/16 way interconnects</td></tr>
<tr><td>CHIP_CONFIG[5]</td><td>On-chip clock debug enable (DCDL)</td></tr>
</table>
</td></tr>
</table>

Chapter 3. Physical Address Space Layout

The Loongson 3 Seriesprocessor has a globally accessible hierarchical addressing design for system physical address distribution to ensure extended system development compatibility. The physical address width of the entire system is 48 bits. The entire address space is evenly distributed over 16 nodes according to the high 4 bits of the address, i.e., 44 bits of address space per node.

3.1. Physical Address Space Layout Between Nodes

The Loongson 3A5000 processor can be directly connected with 2/4/8/16 3A5000 chips to build a CC-NUMA system, the processor number of each chip is determined by the pin NODEID, and the address space of each chip is distributed as follows:

Table 6. System global address layout at the node level

Chip Node ID (NODEID)	[47:44] Bits of the Address	Start address	End address
0	0	0x0000_0000_0000	0x0FFF_FFFF_FFFF
1	1	0x1000_0000_0000	0x1FFF_FFFF_FFFF
2	2	0x2000_0000_0000	0x2FFF_FFFF_FFFF
3	3	0x3000_0000_0000	0x3FFF_FFFF_FFFF
4	4	0x4000_0000_0000	0x4FFF_FFFF_FFFF
5	5	0x5000_0000_0000	0x5FFF_FFFF_FFFF
6	6	0x6000_0000_0000	0x6FFF_FFFF_FFFF
7	7	0x7000_0000_0000	0x7FFF_FFFF_FFFF
8	8	0x8000_0000_0000	0x8FFF_FFFF_FFFF
9	9	0x9000_0000_0000	0x9FFF_FFFF_FFFF
10	10	0xA000_0000_0000	0xAFFF_FFFF_FFFF
11	11	0xB000_0000_0000	0xBFFF_FFFF_FFFF
12	12	0xC000_0000_0000	0xCFFF_FFFF_FFFF
13	13	0xD000_0000_0000	0xDFFF_FFFF_FFFF
14	14	0xE000_0000_0000	0xEFFF_FFFF_FFFF
15	15	0xF000_0000_0000	0xFFFF_FFFF_FFFF

When the number of system nodes is less than 16 nodes, the nodemask field of the route setting register (0x1fe00400) should be set to ensure that a response can be obtained even if there is no physical node address when a guessed access occurs (2-way: 0x1; 4-way: 0x3; 8-way: 0x7; 16-way: 0xF).

3.2. Physical Address Space Layout Within Nodes

The Loongson 3A5000 uses a single node 4-core configuration, so the corresponding addresses of the DDR memory controller and HT bus integrated in the Loongson 3A5000 chip are contained in a 44-bit address space from 0x0 (inclusive) to 0x1000_0000_0000 (exclusive). Within each node, the 44-bit address space is further divided among all devices connected within the node, and requests are routed to the four shared Cache modules only when the access type is cached. Depending on the chip and system architecture

configuration, if there is no slave device connected on a port, the corresponding address space is reserved address space and access is not allowed.

The address space corresponding to each slave device side of the Loongson 3A5000 chip internal interconnect is as follows:

Table 7. Address layout within nodes

Device	the [43:40] bits of the address	Start address within nodes	End address within nodes
MC0	4	0x400_0000_0000	0x4FF_FFFF_FFFF
MC1	5	0x500_0000_0000	0x5FF_FFFF_FFFF
SE	c	0xC00_0000_0000	0xCFF_FFFF_FFFF
HT0 Lo controller	a	0xA00_0000_0000	0xAFF_FFFF_FFFF
HT0 Hi controller	b	0xB00_0000_0000	0xBFF_FFFF_FFFF
HT1 Lo controller	e	0xE00_0000_0000	0xEFF_FFFF_FFFF
HT1 Hi controller	f	0xF00_0000_0000	0xFFF_FFFF_FFFF

Unlike the directional port mapping relationship, the Loongson 3A5000 can determine the cross-addressing method of the shared Cache based on the access behavior of the actual application. The address space corresponding to the four shared Cache modules in the node is determined based on one or two select bits of the address bits, and can be dynamically configured and modified by software. A configuration register named SCID_SEL is set to determine the address selection bits, as shown in the following table. By default the [7:6] address hash is used for distribution, i.e., the two bits of address [7:6] determine the corresponding shared Cache number. This register is addressed as 0x1fe00400 and can also be accessed using the configuration register instruction (IOCSR).

Table 8. SCID_SEL Address bit configuration

SCID_SEL	Address Bit Selection	SCID_SEL	Address Bit Selection
4'h0	7: 6	4'h8	23:22
4'h1	9: 8	4'h9	25:24
4'h2	11:10	4'ha	27:26
4'h3	13:12	4'hb	29:28
4'h4	15:14	4'hc	31:30
4'h5	17:16	4'hd	33:32
4'h6	19:18	4'he	35:34
4'h7	21:20	4'hf	37:36

The default distribution of the internal 44-bit physical addresses for each node of the Loongson 3A5000 processor is shown in the table below:

Table 9. 44-bit physical address layout within nodes

Address Range	Access Properties	Destination
addr[43:40]==4'ha	Local node, uncache	HT0_LO
addr[43:40]==4'hb	Local node, uncache	HT0_HI

Address Range	Access Properties	Destination
addr[43:40]==4'hc	Local node, uncache	SE
addr[43:40]==4'he	Local node, uncache	HT1_LO
addr[43:40]==4'hf	Local node, uncache	HT1_HI
0x10000000-0x1fffffff, 0x3ff00000-0x3ff0ffff (can be turned off)	Local node, uncache	MISC
Mc interleave is 0 and is not the above address	Local node, uncache	MC0
Mc interleave is 1 and is not the above address	Local node, uncache	MC1
SCache interleave is 0 (address bit selection is determined by scid_sel)	Local node, Cache	Scache0
SCache interleave is 1 (address bit selection is determined by scid_sel)	Local node, Cache	Scache1
SCache interleave is 2 (address bit selection is determined by scid_sel)	Local node, Cache	Scache2
SCache interleave is 3 (address bit selection is determined by scid_sel)	Local node, Cache	Scache3

3.3. Address Routing Layout and Configuration

The routing of the loongson 3A5000 is mainly implemented through the system's two-level cross switch with IO-RING. The software can configure the routing of the requests received by each Master port. Each Master port has 8 address windows, and the target routing of 8 address windows can be completed. Each address window consists of three 64-bit registers, BASE, MASK and MMAP, with BASE aligned by K bytes; MASK adopts a format similar to network mask with high bit of 1; the lower four bits of MMAP indicate the number of the corresponding target Slave port; MMAP[4] indicates allow fetch instrustions; MMAP[5] indicates allow block read; MMAP[6] indicates allow interleaved access enable; MMAP[7] indicates window enable.

Table 10. Space access attributes corresponding to MMAP

[7]	[6]	[5]	[4]
Window enable	Allow interleaved access to SCache/memory	Allow to read blocks	Allow to fetch instructions

Window hit formula: (IN_ADDR & MASK) == BASE

Since Loongson 3 uses fixed routing by default, the configuration window is closed at power-up and requires system software to enable it for use.

When the SCache/memory interleaved access configuration is enabled, the slave number is only valid when it is 0 or 4. 0 indicates routing to SCACHE and SCID_SEL determines how interleaved access is performed

across the 4 SCaches. 4 indicates routing to memory and `interleave_bit` determines how interleaved accesses are performed across the 2 MCs.

The address window translation registers are shown in the table below. The base address is 0x1FE0_0000, or accessed via the IOCSR instruction.

Table 11. Table of address window registers

Address	Register	Address	Register
2000	CORE0_WIN0_BASE	2100	CORE1_WIN0_BASE
2008	CORE0_WIN1_BASE	2108	CORE1_WIN1_BASE
2010	CORE0_WIN2_BASE	2110	CORE1_WIN2_BASE
2018	CORE0_WIN3_BASE	2118	CORE1_WIN3_BASE
2020	CORE0_WIN4_BASE	2120	CORE1_WIN4_BASE
2028	CORE0_WIN5_BASE	2128	CORE1_WIN5_BASE
2030	CORE0_WIN6_BASE	2130	CORE1_WIN6_BASE
2038	CORE0_WIN7_BASE	2138	CORE1_WIN7_BASE
2040	CORE0_WIN0_MASK	2140	CORE1_WIN0_MASK
2048	CORE0_WIN1_MASK	2148	CORE1_WIN1_MASK
2050	CORE0_WIN2_MASK	2150	CORE1_WIN2_MASK
2058	CORE0_WIN3_MASK	2158	CORE1_WIN3_MASK
2060	CORE0_WIN4_MASK	2160	CORE1_WIN4_MASK
2068	CORE0_WIN5_MASK	2168	CORE1_WIN5_MASK
2070	CORE0_WIN6_MASK	2170	CORE1_WIN6_MASK
2078	CORE0_WIN7_MASK	2178	CORE1_WIN7_MASK
2080	CORE0_WIN0_MMAP	2180	CORE1_WIN0_MMAP
2088	CORE0_WIN1_MMAP	2188	CORE1_WIN1_MMAP
2090	CORE0_WIN2_MMAP	2190	CORE1_WIN2_MMAP
2098	CORE0_WIN3_MMAP	2198	CORE1_WIN3_MMAP
20a0	CORE0_WIN4_MMAP	21a0	CORE1_WIN4_MMAP
20a8	CORE0_WIN5_MMAP	21a8	CORE1_WIN5_MMAP
20b0	CORE0_WIN6_MMAP	21b0	CORE1_WIN6_MMAP
20b8	CORE0_WIN7_MMAP	21b8	CORE1_WIN7_MMAP
2200	CORE2_WIN0_BASE	2300	CORE3_WIN0_BASE
2208	CORE2_WIN1_BASE	2308	CORE3_WIN1_BASE
2210	CORE2_WIN2_BASE	2310	CORE3_WIN2_BASE
2218	CORE2_WIN3_BASE	2318	CORE3_WIN3_BASE
2220	CORE2_WIN4_BASE	2320	CORE3_WIN4_BASE

Address	Register	Address	Register
2228	CORE2_WIN5_BASE	2328	CORE3_WIN5_BASE
2230	CORE2_WIN6_BASE	2330	CORE3_WIN6_BASE
2238	CORE2_WIN7_BASE	2338	CORE3_WIN7_BASE
2240	CORE2_WIN0_MASK	2340	CORE3_WIN0_MASK
2248	CORE2_WIN1_MASK	2348	CORE3_WIN1_MASK
2250	CORE2_WIN2_MASK	2350	CORE3_WIN2_MASK
2258	CORE2_WIN3_MASK	2358	CORE3_WIN3_MASK
2260	CORE2_WIN4_MASK	2360	CORE3_WIN4_MASK
2268	CORE2_WIN5_MASK	2368	CORE3_WIN5_MASK
2270	CORE2_WIN6_MASK	2370	CORE3_WIN6_MASK
2278	CORE2_WIN7_MASK	2378	CORE3_WIN7_MASK
2280	CORE2_WIN0_MMAP	2380	CORE3_WIN0_MMAP
2288	CORE2_WIN1_MMAP	2388	CORE3_WIN1_MMAP
2290	CORE2_WIN2_MMAP	2390	CORE3_WIN2_MMAP
2298	CORE2_WIN3_MMAP	2398	CORE3_WIN3_MMAP
22a0	CORE2_WIN4_MMAP	23a0	CORE3_WIN4_MMAP
22a8	CORE2_WIN5_MMAP	23a8	CORE3_WIN5_MMAP
22b0	CORE2_WIN6_MMAP	23b0	CORE3_WIN6_MMAP
22b8	CORE2_WIN7_MMAP	23b8	CORE3_WIN7_MMAP
2400	SCACHE0_WIN0_BASE	2500	SCACHE1_WIN0_BASE
2408	SCACHE0_WIN1_BASE	2508	SCACHE1_WIN1_BASE
2410	SCACHE0_WIN2_BASE	2510	SCACHE1_WIN2_BASE
2418	SCACHE0_WIN3_BASE	2518	SCACHE1_WIN3_BASE
2420	SCACHE0_WIN4_BASE	2520	SCACHE1_WIN4_BASE
2428	SCACHE0_WIN5_BASE	2528	SCACHE1_WIN5_BASE
2430	SCACHE0_WIN6_BASE	2530	SCACHE1_WIN6_BASE
2438	SCACHE0_WIN7_BASE	2538	SCACHE1_WIN7_BASE
2440	SCACHE0_WIN0_MASK	2540	SCACHE1_WIN0_MASK
2448	SCACHE0_WIN1_MASK	2548	SCACHE1_WIN1_MASK
2450	SCACHE0_WIN2_MASK	2550	SCACHE1_WIN2_MASK
2458	SCACHE0_WIN3_MASK	2558	SCACHE1_WIN3_MASK
2460	SCACHE0_WIN4_MASK	2560	SCACHE1_WIN4_MASK
2468	SCACHE0_WIN5_MASK	2568	SCACHE1_WIN5_MASK
2470	SCACHE0_WIN6_MASK	2570	SCACHE1_WIN6_MASK

Address	Register	Address	Register
2478	SCACHE0_WIN7_MASK	2578	SCACHE1_WIN7_MASK
2480	SCACHE0_WIN0_MMAP	2580	SCACHE1_WIN0_MMAP
2488	SCACHE0_WIN1_MMAP	2588	SCACHE1_WIN1_MMAP
2490	SCACHE0_WIN2_MMAP	2590	SCACHE1_WIN2_MMAP
2498	SCACHE0_WIN3_MMAP	2598	SCACHE1_WIN3_MMAP
24a0	SCACHE0_WIN4_MMAP	25a0	SCACHE1_WIN4_MMAP
24a8	SCACHE0_WIN5_MMAP	25a8	SCACHE1_WIN5_MMAP
24b0	SCACHE0_WIN6_MMAP	25b0	SCACHE1_WIN6_MMAP
24b8	SCACHE0_WIN7_MMAP	25b8	SCACHE1_WIN7_MMAP
2600	SCACHE2_WIN0_BASE	2700	SCACHE3_WIN0_BASE
2608	SCACHE2_WIN1_BASE	2708	SCACHE3_WIN1_BASE
2610	SCACHE2_WIN2_BASE	2710	SCACHE3_WIN2_BASE
2618	SCACHE2_WIN3_BASE	2718	SCACHE3_WIN3_BASE
2620	SCACHE2_WIN4_BASE	2720	SCACHE3_WIN4_BASE
2628	SCACHE2_WIN5_BASE	2728	SCACHE3_WIN5_BASE
2630	SCACHE2_WIN6_BASE	2730	SCACHE3_WIN6_BASE
2638	SCACHE2_WIN7_BASE	2738	SCACHE3_WIN7_BASE
2640	SCACHE2_WIN0_MASK	2740	SCACHE3_WIN0_MASK
2648	SCACHE2_WIN1_MASK	2748	SCACHE3_WIN1_MASK
2650	SCACHE2_WIN2_MASK	2750	SCACHE3_WIN2_MASK
2658	SCACHE2_WIN3_MASK	2758	SCACHE3_WIN3_MASK
2660	SCACHE2_WIN4_MASK	2760	SCACHE3_WIN4_MASK
2668	SCACHE2_WIN5_MASK	2768	SCACHE3_WIN5_MASK
2670	SCACHE2_WIN6_MASK	2770	SCACHE3_WIN6_MASK
2678	SCACHE2_WIN7_MASK	2778	SCACHE3_WIN7_MASK
2680	SCACHE2_WIN0_MMAP	2780	SCACHE3_WIN0_MMAP
2688	SCACHE2_WIN1_MMAP	2788	SCACHE3_WIN1_MMAP
2690	SCACHE2_WIN2_MMAP	2790	SCACHE3_WIN2_MMAP
2698	SCACHE2_WIN3_MMAP	2798	SCACHE3_WIN3_MMAP
26a0	SCACHE2_WIN4_MMAP	27a0	SCACHE3_WIN4_MMAP
26a8	SCACHE2_WIN5_MMAP	27a8	SCACHE3_WIN5_MMAP
26b0	SCACHE2_WIN6_MMAP	27b0	SCACHE3_WIN6_MMAP
26b8	SCACHE2_WIN7_MMAP	27b8	SCACHE3_WIN7_MMAP

Address	Register	Address	Register
-	-	2900	IO_L2X_WIN0_BASE
-	-	2908	IO_L2X_WIN1_BASE
-	-	2910	IO_L2X_WIN2_BASE
-	-	2918	IO_L2X_WIN3_BASE
-	-	2920	IO_L2X_WIN4_BASE
-	-	2928	IO_L2X_WIN5_BASE
-	-	2930	IO_L2X_WIN6_BASE
-	-	2938	IO_L2X_WIN7_BASE
-	-	2940	IO_L2X_WIN0_MASK
-	-	2948	IO_L2X_WIN1_MASK
-	-	2950	IO_L2X_WIN2_MASK
-	-	2958	IO_L2X_WIN3_MASK
-	-	2960	IO_L2X_WIN4_MASK
-	-	2968	IO_L2X_WIN5_MASK
-	-	2970	IO_L2X_WIN6_MASK
-	-	2978	IO_L2X_WIN7_MASK
-	-	2980	IO_L2X_WIN0_MMAP
-	-	2988	IO_L2X_WIN1_MMAP
-	-	2990	IO_L2X_WIN2_MMAP
-	-	2998	IO_L2X_WIN3_MMAP
-	-	29a0	IO_L2X_WIN4_MMAP
-	-	29a8	IO_L2X_WIN5_MMAP
-	-	29b0	IO_L2X_WIN6_MMAP
-	-	29b8	IO_L2X_WIN7_MMAP
2a00	HT0_LO_WIN0_BASE	2b00	HT0_HI_WIN0_BASE
2a08	HT0_LO_WIN1_BASE	2b08	HT0_HI_WIN1_BASE
2a10	HT0_LO_WIN2_BASE	2b10	HT0_HI_WIN2_BASE
2a18	HT0_LO_WIN3_BASE	2b18	HT0_HI_WIN3_BASE
2a20	HT0_LO_WIN4_BASE	2b20	HT0_HI_WIN4_BASE
2a28	HT0_LO_WIN5_BASE	2b28	HT0_HI_WIN5_BASE
2a30	HT0_LO_WIN6_BASE	2b30	HT0_HI_WIN6_BASE
2a38	HT0_LO_WIN7_BASE	2b38	HT0_HI_WIN7_BASE
2a40	HT0_LO_WIN0_MASK	2b40	HT0_HI_WIN0_MASK
2a48	HT0_LO_WIN1_MASK	2b48	HT0_HI_WIN1_MASK

Address	Register	Address	Register
2a50	HT0_LO_WIN2_MASK	2b50	HT0_HI_WIN2_MASK
2a58	HT0_LO_WIN3_MASK	2b58	HT0_HI_WIN3_MASK
2a60	HT0_LO_WIN4_MASK	2b60	HT0_HI_WIN4_MASK
2a68	HT0_LO_WIN5_MASK	2b68	HT0_HI_WIN5_MASK
2a70	HT0_LO_WIN6_MASK	2b70	HT0_HI_WIN6_MASK
2a78	HT0_LO_WIN7_MASK	2b78	HT0_HI_WIN7_MASK
2a80	HT0_LO_WIN0_MMAP	2b80	HT0_HI_WIN0_MMAP
2a88	HT0_LO_WIN1_MMAP	2b88	HT0_HI_WIN1_MMAP
2a90	HT0_LO_WIN2_MMAP	2b90	HT0_HI_WIN2_MMAP
2a98	HT0_LO_WIN3_MMAP	2b98	HT0_HI_WIN3_MMAP
2aa0	HT0_LO_WIN4_MMAP	2ba0	HT0_HI_WIN4_MMAP
2aa8	HT0_LO_WIN5_MMAP	2ba8	HT0_HI_WIN5_MMAP
2ab0	HT0_LO_WIN6_MMAP	2bb0	HT0_HI_WIN6_MMAP
2ab8	HT0_LO_WIN7_MMAP	2bb8	HT0_HI_WIN7_MMAP
2c00	SE_WIN0_BASE	2d00	MISC_WIN0_BASE
2c08	SE_WIN1_BASE	2d08	MISC_WIN1_BASE
2c10	SE_WIN2_BASE	2d10	MISC_WIN2_BASE
2c18	SE_WIN3_BASE	2d18	MISC_WIN3_BASE
2c20	SE_WIN4_BASE	2d20	MISC_WIN4_BASE
2c28	SE_WIN5_BASE	2d28	MISC_WIN5_BASE
2c30	SE_WIN6_BASE	2d30	MISC_WIN6_BASE
2c38	SE_WIN7_BASE	2d38	MISC_WIN7_BASE
2c40	SE_WIN0_MASK	2d40	MISC_WIN0_MASK
2c48	SE_WIN1_MASK	2d48	MISC_WIN1_MASK
2c50	SE_WIN2_MASK	2d50	MISC_WIN2_MASK
2c58	SE_WIN3_MASK	2d58	MISC_WIN3_MASK
2c60	SE_WIN4_MASK	2d60	MISC_WIN4_MASK
2c68	SE_WIN5_MASK	2d68	MISC_WIN5_MASK
2c70	SE_WIN6_MASK	2d70	MISC_WIN6_MASK
2c78	SE_WIN7_MASK	2d78	MISC_WIN7_MASK
2c80	SE_WIN0_MMAP	2d80	MISC_WIN0_MMAP
2c88	SE_WIN1_MMAP	2d88	MISC_WIN1_MMAP
2c90	SE_WIN2_MMAP	2d90	MISC_WIN2_MMAP
2c98	SE_WIN3_MMAP	2d98	MISC_WIN3_MMAP

Address	Register	Address	Register
2ca0	SE_WIN4_MMAP	2da0	MISC_WIN4_MMAP
2ca8	SE_WIN5_MMAP	2da8	MISC_WIN5_MMAP
2cb0	SE_WIN6_MMAP	2db0	MISC_WIN6_MMAP
2cb8	SE_WIN7_MMAP	2db8	MISC_WIN7_MMAP
2e00	HT1_LO_WIN0_BASE	2f00	HT1_HI_WIN0_BASE
2e08	HT1_LO_WIN1_BASE	2f08	HT1_HI_WIN1_BASE
2e10	HT1_LO_WIN2_BASE	2f10	HT1_HI_WIN2_BASE
2e18	HT1_LO_WIN3_BASE	2f18	HT1_HI_WIN3_BASE
2e20	HT1_LO_WIN4_BASE	2f20	HT1_HI_WIN4_BASE
2e28	HT1_LO_WIN5_BASE	2f28	HT1_HI_WIN5_BASE
2e30	HT1_LO_WIN6_BASE	2f30	HT1_HI_WIN6_BASE
2e38	HT1_LO_WIN7_BASE	2f38	HT1_HI_WIN7_BASE
2e40	HT1_LO_WIN0_MASK	2f40	HT1_HI_WIN0_MASK
2e48	HT1_LO_WIN1_MASK	2f48	HT1_HI_WIN1_MASK
2e50	HT1_LO_WIN2_MASK	2f50	HT1_HI_WIN2_MASK
2e58	HT1_LO_WIN3_MASK	2f58	HT1_HI_WIN3_MASK
2e60	HT1_LO_WIN4_MASK	2f60	HT1_HI_WIN4_MASK
2e68	HT1_LO_WIN5_MASK	2f68	HT1_HI_WIN5_MASK
2e70	HT1_LO_WIN6_MASK	2f70	HT1_HI_WIN6_MASK
2e78	HT1_LO_WIN7_MASK	2f78	HT1_HI_WIN7_MASK
2e80	HT1_LO_WIN0_MMAP	2f80	HT1_HI_WIN0_MMAP
2e88	HT1_LO_WIN1_MMAP	2f88	HT1_HI_WIN1_MMAP
2e90	HT1_LO_WIN2_MMAP	2f90	HT1_HI_WIN2_MMAP
2e98	HT1_LO_WIN3_MMAP	2f98	HT1_HI_WIN3_MMAP
2ea0	HT1_LO_WIN4_MMAP	2fa0	HT1_HI_WIN4_MMAP
2ea8	HT1_LO_WIN5_MMAP	2fa8	HT1_HI_WIN5_MMAP
2eb0	HT1_LO_WIN6_MMAP	2fb0	HT1_HI_WIN6_MMAP
2eb8	HT1_LO_WIN7_MMAP	2fb8	HT1_HI_WIN7_MMAP

The secondary xbar mainly connects 2 memory controllers and IO-RING as slave devices, with 4 SCache (4, representing 4xxx, same as 5, 6, 7) and IO-RING (9) as master devices for window mapping, which can use these window configuration registers (4, 5, 6, 7, 9) for memory window configuration and address translation.

Each address window consists of three 64-bit registers, BASE, MASK, and MMAP, with BASE aligned in K bytes, MASK in a format similar to the network mask high bit 1, and MMAP containing the converted address, routing, and enable control bits, as shown in the following table:

Table 12. Description of MMAP register bit field

[63:48]	[47:10]	[7:4]	[3:0]
Reserved	Address after translation	Window enable	Slave device number

Among them, the devices corresponding to the slave device number are shown in the following table:

Table 13. Correspondence from the device number to the module it belongs to

Slave Device Number	Destination Device
0-3	Scache0-3
4-5	MC0-1
a	HT0_lo
b	HT0_hi
c	SE
d	MISC
e	HT1_lo
f	HT1_hi

The meaning of the window enable bits is shown in the table below:

Table 14. Space access attributes corresponding to MMAP

[7]	[6]	[5]	[4]
Window enable	Allow interleaved access to DDR. Valid when the slave device number is 0, to route requests for hit window addresses as configured by the interleave_bit (CSR0x0400). The interleave enable bit is required to be greater than 10	Allow to read blocks	Allow to fetch instructions

Note that the window configuration cannot perform address translation for Cache consistency requests, otherwise the address at the SCache will not match the address at the processor-level Cache, resulting in a Cache consistency maintenance error.

Window hit formula: `(IN_ADDR & MASK) == BASE`

New address conversion formula: `OUT_ADDR = (IN_ADDR & ~MASK) | {MMAP[63:10], 10'h0}`

According to the default register configuration, the CPU's address range of 0x00000000-0x0fffffff after the chip is booted (256M) mapped to the address interval 0x00000000-0x0fffffff of the DDR. 0x10000000-0x17ffffff are mapped to the PCI_MEM space of the bridge chip. 0x18000000-0x19ffffff are mapped to the PCI_IO space of the bridge chip. 0x1a000000-0x1affffff are mapped to the bridge chip's PCI configuration space (Type0). 0x1b000000-0x1bffffff are mapped to the bridge chip's PCI configuration space (Type1). 0x40000000-0x7fffffff are mapped to the bridge chip's PCI_MEM space. Software can implement the new address space routing and translation by modifying the corresponding configuration registers.

In addition, when there is a read access to an illegal address due to CPU guessing execution, all 8 address windows are not hit and random data is returned to prevent the CPU from dying, etc.

Chapter 4. Chip Configuration Register

The chip configuration registers in the Loongson3A5000 provide a mechanism for reading and writing configuration of various functions of the chip. The individual configuration registers are described in detail below.

The base address of each chip configuration register in this chapter is 0x1fe00000, which can also be accessed using the configuration register instruction (IOCSR).

CSR[A][B] in this document indicates bit B in the IOCSR register with offset address A, where B can be a range.

4.1. Version Register (0x0000)

The offset address is 0x0000.

Table 15. Version register

Bit Field	Name	Read/Write	Reset Value	Description
7:0	Version	R	8'h11	Configuration register version number

4.2. Chip Characteristics Register (0x0008)

This register identifies a number of software-related processor features for software to view before enabling a specific function. The offset address is 0x0008.

Table 16. Chip characteristics register

Bit Field	Name	Read/Write	Reset Value	Description
0	Centigrade	R	1'b1	1 indicates that CSR[0x428] is valid
1	Node counter	R	1'b1	1 indicates that CSR[0x408] is valid
2	MSI	R	1'b1	1 indicates that MSI is available
3	EXT_IOI	R	1'b1	1 indicates that EXT_IOI is available
4	IPI_percore	R	1'b1	1 indicates that IPI is sent via VSR private address
5	Freq_percore	R	1'b1	1 indicates that frequency is modulated via VSR private address
6	Freq_scale	R	1'b1	1 indicates that dynamic frequency division is available
7	DVFS_v1	R	1'b1	1 indicates that dynamic frequency adjustment v1 is available
8	Tsensor	R	1'b1	1 indicates that temperature sensor is available
9	Interrupt decoding	R	1'b1	Interrupt pin decoding mode is available
10	Flat Mode	R	1'b0	Legacy compatibility mode
11	Guest Mode	WR	1'b0	KVM Virtual Machine Mode

4.3. Manufacturer Name Register (0x0010)

This register is used to identify the vendor name. The offset address is 0x0010.

Table 17. Manufacturer name register

Bit Field	Name	Read/Write	Reset Value	Description
63:0	Vendor	R	0x6e6f7367_6e6f6f4c	string “Loongson”

4.4. Chip Name Register (0x0020)

This register is used to identify the chip name. The offset address is 0x0020.

Table 18. Chip name register

Bit Field	Name	Read/Write	Reset Value	Description
63:0	ID	R	0x00003030_30354133	String “3A5000”

4.5. Function configuration Register (0x0180)

The offset address is 0x0180.

Table 19. Function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
0		RW	1’b0	
1		RW	1’b0	
3:2		RW	2’b0	Reserved
4	MC0_disable_confspace	RW	1’b0	Whether to disable MC0 DDR configuration space
5	MC0_defult_confspace	RW	1’b1	Routing all memory accesses to the configuration space
6	MCA0 clock en	RW	1’b1	MCA0 clock enable
7	MC0_resetn	RW	1’b1	MC0 software reset (active low)
8	MC0_clken	RW	1’b1	Whether to enable MC0
9	MC1_disable_confspace	RW	1’b0	Whether to disable MC1 DDR configuration space
10	MC1_defult_confspace	RW	1’b1	Routing all memory accesses to the configuration space
11	MCA1 clock en	RW	1’b1	MCA1 clock enable
12	MC1_resetn	RW	1’b1	MC1 software reset (active low)

Bit Field	Name	Read/Write	Reset Value	Description
13	MC1_clken	RW	1'b1	Whether to enable MC1
26:24	HT0_freq_scale_ctrl	RW	3'b011	HT Controller 0 frequency division
27	HT0_clken	RW	1'b1	Whether to enable HT0
30:28	HT1_freq_scale_ctrl	RW	3'b011	HT Controller 1 frequency division
31	HT1_clken	RW	1'b1	Whether to enable HT1
42:40	Node_freq_ctrl	RW	3'b111	Node Frequency Division
43	-	RW	1'b1	
63:56	Cpu_version	R	2'h3D	CPU version

4.6. Pin Controller Driver Configuration Register (0x0188)

The offset address is 0x0188.

Table 20. Pin controller driver configuration register

Bit Field	Name	Read/Write	Reset Value	Description
15:0				
19:16	HT sideband	RW	2'b0	HT control signal driver configuration
23:20	I2C	RW	2'b0	I2C control signal driver configuration
27:24	UART	RW	2'b0	UART control signal driver configuration
31:28	SPI	RW	2'b0	SPI control signal driver configuration
35:32	GPIO	RW	2'b0	GPIO control signal driver configuration
39:36	SE UART	RW	2'b0	SE UART control signal driver configuration
43:40	SE SPI	RW	2'b0	SE SPI control signal driver configuration
47:44	SE I2C	RW	2'b0	SE I2C control signal driver configuration
51:48	SE SCI	RW	2'b0	SE SCI control signal driver configuration
55:52	SE RNG	RW	2'b0	SE RNG control signal driver configuration
59:56	SE GPIO	RW	2'b0	SE GPIO control signal driver configuration

4.7. Function Collection Register (0x0190)

The offset address is 0x0190.

Table 21. Function collection register

Bit Field	Name	Read/Write	Reset Value	Description
31:0		R		Reserved
37:32	Chip_config	R		Motherboard configuration control
47:38	Sys_clkseli	R		On-board frequency multiplier configuration
55:48	Bad_ip_core	R		core7-core0 are bad or not
57:56	Bad_ip_ddr	R		2 DDR controllers are bad or not
61:60	Bad_ip_ht	R		2 HT Controllers are bad or not

4.8. Temperature Collection Register (0x0198)

The offset address is 0x0198.

Table 22. Temperature collection register

Bit Field	Name	Read/Write	Reset Value	Description
15:0		R		Reserved
19:16		R		Reserved
20	dotest	R		Dotest pin status
21	iccc_en	R		Iccc_en pin status
23:22		R		Reserved
24	Thsens0_overflow	R		Temperature sensor 0 overflow
25	Thsens1_overflow	R		Temperature sensor 1 overflow
31:26				
47:32	Thsens0_out	R		Temperature sensor 0 centigrade temperature Node temperature=Thens0_out *731/0x4000-273 Temperature range: -40 degree - 125 degree
63:48	Thsens1_out	R		Temperature sensor 1 centigrade temperature Node temperature=Thens0_out *731/0x4000-273 Temperature range: -40 degree - 125 degree

4.9. Frequency Configuration Register (0x01B0)

The following sets of software multiplier setting registers are used to set the operating frequency of the chip master clock and the memory controller clock when CLKSEL is configured in software control mode (refer to the CLKSEL setting method in Descriptions of Pins).

Among other things, the MEM CLOCK configuration supports multiple modes. In 4x mode (mem div), MEM CLOCK should be 4x the memory controller clock; in 2x mode (mem div), MEM CLOCK should be 2x the memory controller clock; in 1x mode (mem div), MEM CLOCK should be the memory controller clock frequency .

The memory bus operates at 2 times the memory controller clock and the bus operates at 4 times the memory controller clock.

NODE CLOCK corresponds to the clock frequency of the processor core, on-chip network, and shared Cache.

Each clock configuration generally has three parameters, DIV_REFC, DIV_LOOPC, and DIV_OUT. The final clock frequency is (reference clock / DIV_REFC * DIV_LOOPC) / DIV_OUT.

In software control mode, the default corresponding clock frequency is the frequency of the external reference clock (100MHz or 25MHz), which needs to be set in software during the processor startup. The process of setting the individual clocks should be done as follows:

1. Set registers other than SEL_PLL_* and SOFT_SET_PLL, i.e., these two registers are written to 0 during the setting process.
2. Set registers other than SEL_PLL_* and SOFT_SET_PLL, i.e., these two registers are written to 0 during the setting process.

Wait for the lock signal LOCKED_* in the register to be 1.

1. Set SEL_PLL_* to 1, then the corresponding clock frequency will be switched to the software-set frequency.

The following register is the configuration register of Main CLOCK, Main Clock is used to generate the maximum operating frequency of node clock, core clock, etc. The base address is 0x1fe00000 and the offset address is 0x1b0:

Table 23. Node clock software multiplier configuration register

Bit Field	Name	Read/Write	Reset Value	Description
0	SEL_PLL_NODE	RW	0x0	Clock output selection 1: Node clock select PLL output 0: Node clock select SYS CLOCK
1		RW	0x0	Reserved
2	SOFT_SET_PLL	RW	0x0	Allow software to set the PLL
3	BYPASS_L1	RW	0x0	Bypass L1 PLL
7:4	-	RW	0x0	Reserved
8	VDDA_LDO_EN	RW	0x0	Enable VDDA LDO

Bit Field	Name	Read/Write	Reset Value	Description
9	VDDD_LDO_EN	RW	0x0	Enable VDDD LDO
11:10	-			
12	DACPD_L2	RW	0x0	L2 clock DACPD
13	DSMPD_L2	RW	0x0	L2 clock DSMPD
15:14		RW	0x0	Reserved
16	LOCKED_L1	R	0x0	L1 PLL is locked or not
18:17	-	R	0x0	Reserved
19	PD_L1	RW	0x0	Disable L1 PLL
21:20		RW	0x0	Reserved
22	L2_SEL	RW	0x0	Select L2 clock output
25:23		RW	0x0	Reserved
31:26	L1_DIV_REFC	RW	0x1	L1 PLL input parameters
40:32	L1_DIV_LOOPC	RW	0x1	L1 PLL input parameters
41				Reserved
47:42	L1_DIV_OUT	RW	0x1	L1 PLL input parameters
53:48	L2_DIV_REFC	RW		
63:54	L2_DIV_LOOPC	RW		
69:64	L2_DIV_OUT	RW		
95:70	-	RW		
119:96	L2_FRAC	RW		
122:120	VDDA_LDO_CTRL	RW		
123	VDDA_LDO_BYPASS	RW		
126:124	VDDD_LDO_CTRL	RW		
127:124	VDDD_LDO_BYPASS	RW		
Other	_	RW		Reserved

Note: `PLL ouput = (clk_ref /div_refc * div_loopc) / div_out`.

The result of `clk_ref/div_refc` for the PLL should be `25`/`50`/`100MHz`, with `50MHz` recommended. The VCO frequency (the part in parentheses in the above equation) must be in the range `4.8GHz-6.4GHz`. This requirement also applies to memory PLLs.

In addition, the recommended setting for div_loopc is less than `255`. The recommended setting for `div_out` is `1`/`2`/`4`/`6` and above `6`, and `3`/`5` is not recommended.

The following register is the `MEM CLOCK` configuration register, the `MEM CLOCK` clock frequency should be configured to `1`/`2` of the final DDR bus clock frequency. The base address is `0x1fe00000`, offset address is `0x1c0`:

Table 24. Memory clock software multiplier configuration register

Bit Field	Name	Read/Write	Reset Value	Description
[0]	SEL_MEM_PLL	RW	0x0	Clock output selection 1: Node clock select PLL output 0: Node clock select SYS CLOCK
[1]	SOFT_SET_MEM_PLL	RW	0x0	Allow software to set MEM PLL
[2]	BYPASS_MEM_PLL	RW	0x0	Bypass MEM_PLL
[3]	MEMDIV_RESETn	RW	0x1	Reset the internal frequency divider
[5:4]	MEMDIV_MODE	RW		00: 1X frequency multiplier mode 01: 2X frequency multiplier mode 10: 4X frequency multiplier mode 11: Reserved
[6]	LOCKED_MEM_PLL	R	0x0	MEM_PLL is locked or not
[7]	PD_MEM_PLL	RW	0x0	Disable MEM PLL
[13:8]	MEM_PLL_DIV_REFC	RW	0x1	MEM PLL input parameters When the NODE clock is selected (NODE_CLOCK_SEL is 1), it is used as a frequency divider input
[23:14]	MEM_PLL_DIV_LOOPC	RW	0x41	MEM PLL input parameters
[29:24]	MEM_PLL_DIV_OUT	RW	0x0	MEM PLL input parameters
[30]	NODE_CLOCK_SEL	RW	0x0	0: use MEM_PLL as MEM clock 1: use NODE_CLOCK as a frequency divider input
[31]	-			
[34:32]	VDDA_LDO_CTRL	RW		
[35]	VDDA_LDO_BYPASS	RW		
[38:36]	VDDD_LDO_CTRL	RW		
[39]	VDDD_LDO_BYPASS	RW		
[40]	VDDA_LDO_EN			
[41]	VDDD_LDO_EN	RW		
Other		RW		Reserved

4.10. Processor Core Frequency Division Configuration Register (0x01D0)

The following registers are used for dynamic frequency division of the processor core. Using this register to

set the frequency of the processor core, the frequency conversion operation can be done within 100ns with no additional overhead. The base address is `0x1fe00000` and the offset address is `0x01d0`.

Table 25. Processor core software frequency division configuration

Bit Field	Name	Read/Write	Reset Value	Description
2:0	core0_freqctrl	RW	0x7	Core 0 frequency division control value
3	core0_en	RW	0x1	Core 0 clock enable
6:4	core1_freqctrl	RW	0x7	Core 1 frequency division control value
7	core1_en	RW	0x1	Core 1 clock enable
10:8	core2_freqctrl	RW	0x7	Core 2 frequency division control value
11	core2_en	RW	0x1	Core 2 clock enable
14:12	core3_freqctrl	RW	0x7	Core 3 frequency division control value
15	core3_en	RW	0x1	Core 3 clock enable

Note: The clock frequency value after software dividing is equal to the original `(dividing control value + 1)/8`.

4.11. Processor Core Reset Control Register (0x01D8)

The following registers are used for software-controlled reset of the processor core. To reset, set resetn to `0`, resetn_pre to `0`, wait `500` microseconds, `resetn_pre` to `1`, and reset `n` to `1` to complete the reset process. The base address of this register is `0x1fe00000` and the offset address is `0x01d8`.

Table 26. Processor core software reset control register

Bit Field	Name	Read/Write	Reset Value	Description
0	Core0_resetn_pre	RW	0x1	Core 0 reset auxiliary control
1	Core0_resetn	RW	0x1	Core 0 reset
2	Core1_resetn_pre	RW	0x1	Core 1 reset auxiliary control
3	Core1_resetn	RW	0x1	Core 1 reset
4	Core2_resetn_pre	RW	0x1	Core 2 reset auxiliary control
5	Core2_resetn	RW	0x1	Core 2 reset
6	Core3_resetn_pre	RW	0x1	Core 3 reset auxiliary control
7	Core3_resetn	RW	0x1	Core 3 reset

4.12. Routing Configuration Register (0x0400)

The following registers are used to control some of the routing settings within the chip. The base address is `0x1fe00000` and the offset address is `0x0400`.

Table 27. Chip routing configuration register

Bit Field	Name	Read/Write	Reset Value	Description
3:0	scid_sel	RW	0x0	Shared Cache hash bit control

Bit Field	Name	Read/Write	Reset Value	Description
7:4	Node_mask	RW	0xF	Node mask to avoid no response when guessing the address of an unused node
8	xrouter_en	RW	0x0	HT1 inter-chip routing enable control
9	disable_0x3ff0	RW	0x0	Disable routing via base address 0x3ff0_0000 of configuration register space
10	Fast_path_36_en	RW	0x0	Enable 36 fast paths (8-way)
11	Fast_path_27_en	RW	0x0	Enable 27 fast paths (8-way)
12	mcc_en	RW	0x0	MCC mode enable
14	Scahe_1MB	RW		SCache capacity cut in half
19:16	ccsd_id	RW	0x0	
24	ccsd_en	RW	0x0	
31:30	mc_en	RW	0x3	Enable routing control for both MCs
37:32	interleave_bit	RW	0x0	Memory hash control
39	interleave_en	RW	0x0	Memory hash enable
43:40	ht_control	R		Ht-related configuration pins
47:44	ht_reg_disable	RW	0x0	Close ht space for consistency mode to avoid routing HT space addresses to HT
60:56	Line_ag_cfg	RW	0x0	Cross-chip bandwidth balancing configuration

4.13. Other Function Configuration Register (0x0420)

The following registers are used to control some of the functions enabled within the chip. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 28. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
0	disable_jtag	RW	0x0	Completely disable the JTAG interface
1	disable_jtag_LA464	RW	0x0	Completely disable the LA464JTAG debug interface
2	disable_LA132	RW	0x0	Completely disable LA132
3	disable_jtag_LA132	RW	0x0	Completely disable the LA132 JTAG debug interface
4	Disable_antifuse0	RW	0x0	Disable fuse
5	Disable_antifuse1	RW	0x0	Disable fuse
6	Disable_ID	RW	0x0	Disable ID modification
7				Reserved

Bit Field	Name	Read/Write	Reset Value	Description
8	resetn_LA132	RW	0x0	LA132 reset control
9	sleeping_LA132	R	0x0	LA132 go to sleep
10	soft_int_LA132	RW	0x0	LA132 inter-processor interrupt register
15:12	core_int_en_LA132	RW	0x0	LA132 I/O interrupt enable for each core
18:16	freqscale_LA132	RW	0x0	LA132 frequency division control
19	clken_LA132	RW	0x0	LA132 clock enable
20	stable_sel	RW	0x0	Stable clock selection 0: SYS CLOCK 1: NODE CLOCK
21	stable_resetn	RW	0x0	Stable clock reset control
22	freqscale_percore	RW	0x0	Enable private frequency adjustment registers for each core
23	clken_percore	RW	0x0	Enable private clock for each core
27:24	confbus_timeout	RW	0x8	Configure the bus timeout configuration. The actual time is the power of 2.
29:28	HT_softresetn	RW	0x3	HT Controller software reset control
35:32	freqscale_mode_core	RW	0x0	Frequency adjustment mode selection for each core 0: (n+1)/8 1: 1/(n+1)
36	freqscale_mode_node	RW	0x0	Frequency adjustment mode selection for nodes 0: (n+1)/8 1: 1/(n+1)
37	freqscale_mode_LA132	RW	0x0	Frequency adjustment mode selection for LA132 0: (n+1)/8 1: 1/(n+1)
39:38	freqscale_mode_HT	RW	0x0	Frequency adjustment mode selection for each HT 0: (n+1)/8 1: 1/(n+1)

Bit Field	Name	Read/Write	Reset Value	Description
40	freqscale_mode_stable	RW	0x0	Frequency adjustment mode selection for stable clock 0: (n+1)/8 1: 1/(n+1)
43:41				Reserved
46:44	freqscale_stable	RW	0x0	Stable clock frequency adjustment register
47	clken_stable	RW	0x0	Stable clock enable
48	EXT_INT_en	RW	0x0	Extended I/O interrupt enable
49	INT_encode	RW	0x0	Enable interrupt pin encode mode
53:52				
54				
57:56	thsensor_sel	RW	0x0	Temperature sensor selection
62:60	Auto_scale	R	0x0	Current value auto frequency adjustment
63	Auto_scale_doing	R	0x0	Flags in effect auto frequency adjustment

4.14. Centigrade Temperature Register (0x0428)

The following registers are used to observe the chip internal temperature sensor values. In degrees Celsius. The base address is 0x1fe00000 and the offset address is 0x0428. This register is available only when CSR[0x0008][0] is valid.

Table 29. Temperature observation register

Bit Field	Name	Read/Write	Reset Value	Description
7:0	Centigrade temperature	RO	0x0	Centigrade temperature
63:8		RW	0x0	

4.15. SRAM Adjustment Register (0x0430)

The following registers are used to adjust the operating frequency of Sram inside the processor core. The offset address is 0x0430.

Table 30. Processor core SRAM adjustment register

Bit Field	Name	Read/Write	Reset Value	Description
31:0	sram_ctrl	RW	0x0	Inter-processor sram configuration register
63:32		RW	0x0	

4.16. FUSE0 Observation Register (0x0460)

The following registers are used to observe the Fuse0 values visible to some software. The offset address is 0x0460.

Table 31. FUSE0 observation register

Bit Field	Name	Read/Write	Reset Value	Description
127:0	Fuse_0	RW	0x0	

4.17. FUSE1 Observation Register (0x0470)

The following registers are used to observe the Fuse1 values visible to some software. The offset address is 0x0470.

Table 32. FUSE1 observation register

Bit Field	Name	Read/Write	Reset Value	Description
127:0	Fuse_1	RW	0x0	

Chapter 5. Chip Clock Division and Enable Control

The Loongson 3A5000 can use a single external reference clock, SYS_CLOCK. The generation of each clock can depend on SYS_CLOCK, and the following sections describe each of these clocks.

The Loongson 3A5000 has separate frequency dividing mechanisms for the processor core, on-chip network and shared Cache, HT controller, and LA132 core. In line with the 3A4000, the 3A5000 also supports 1/n divider values,It can also be accessed using the configuration register instruction (IOCSR).

The base address of each chip configuration register in this chapter is 0x1fe00000.

5.1. Introduction to Chip Module clock

The chip reference clock SYS_CLOCK usually uses a 100MHz crystal input, but a 25MHz crystal input is also available. Different crystal frequencies need to be selected via CLKSEL[4].

The reference clock of the HT PHY can use the 200MHz differential reference input of each PHY in addition to the SYS CLOCK. Use CLKSEL[8] to make the selection. When SYS CLOCK is selected as the reference clock and a 25MHz crystal input is used, the HT PHY cannot operate at 3.2GHz.

The clocks used in the Loongson 3A5000 chip and their control methods are shown in the following table.

Table 33. Processor internal clock description

Clock	Clock Source	Frequency Multiplier Method	Frequency Division Control	Enable Control	Clock Description
Boot clock	SYS_CLOCK	*1	Not supported	Not supported	SPI, UART, I2C controller clock
Main clock	SYS PLL	PLL configuration	Not supported	Not supported	SYS PLL output Node clock, core clock, HTcore clock, LA132 clock source Optional mem clock, stable clock source
Node clock	Main clock	*1	Supported	Not supported	On-chip network, shared Cache, node clock, HT controller clock source
Core0 clock	Main clock	*1	Supported	Supported	Core0 clock
Core1 clock	Main clock	*1	Supported	Supported	Core1 clock
Core2 clock	Main clock	*1	Supported	Supported	Core2 clock
Core3 clock	Main clock	*1	Supported	Supported	Core3 clock
HTcore0 clock	Node clock	*1	Supported	Supported	HT0 controller clock, and software needs to be guaranteed to be below 1GHz after frequency division
HTcore1 clock	Node clock	*1	Supported	Supported	HT1 controller clock, and software needs to be guaranteed to be below 1GHz after frequency division

Clock	Clock Source	Frequency Multiplier Method	Frequency Division Control	Enable Control	Clock Description
LA132 clock	Main clock	*1	Supported	Supported	LA132 clock, and software needs to be guaranteed to be below 1GHz after frequency division
Stable clock	SYS_CLOCK	*1	Supported	Supported	Processor core constant counter clock
Mem clock	MEM PLL	PLL configuration	Not supported	Supported	Memory controller clock
	Main clock	/2, /4, /8	Not supported	Supported	Memory controller alternative clock

5.2. Processor Core Frequency Division and Enable Control

There are various modes of processor core frequency division, one is per-address access mode, and the other is processor configuration instruction access mode, which are described below. Each processor core can be controlled separately.

5.2.1. Accessing by Address

The per-address access mode is compatible with the 3A3000 processor and uses the processor core software frequency divider setup register, which uses the same address for setup.

Using this register to set the processor core for tuning allows the frequency conversion operation to be completed in 100ns with no other additional overhead. The base address is 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR),and the offset address is 0x01d0.

Table 34. Processor core software frequency division configuration

Bit Field	Name	Read/Write	Reset Value	Description
2:0	core0_freqctrl	RW	0x7	Core 0 frequency division control value
3	core0_en	RW	0x1	Core 0 clock enable
6:4	core1_freqctrl	RW	0x7	Core 1 frequency division control value
7	core1_en	RW	0x1	Core 1 clock enable
10:8	core2_freqctrl	RW	0x7	Core 2 frequency division control value
11	core2_en	RW	0x1	Core 2 clock enable
14:12	core3_freqctrl	RW	0x7	Core 3 frequency division control value
15	core3_en	RW	0x1	Core 3 clock enable

Note: The clock frequency value after software dividing is equal to the original (dividing control value + 1)/8.

In addition to the frequency division configuration compatible with the 3A3000 processor, the clock frequency after frequency division can be adjusted from (Frequency Division Control Value + 1)/8 to 1/(Frequency Division Control Value + 1)` by setting the register in the 3A5000. This register is located in other function configuration register. The base address is 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR),and the offset address is 0x0420.

Table 35. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
35:32	freqscale_mode_core	RW	0x0	Frequency adjustment mode selection for each core 0: (n+1)/8 1: 1/(n+1)

5.2.2. Accessing by Configuration Register Instructions

In addition to the legacy per-address access mode, the 3A5000 also supports access to private frequency division configuration registers using the configuration register instruction.

Note that the private frequency division configuration register control is mutually exclusive with the original processor core software frequency division setup register control, and only one of the two can be used. The choice is made by using the corresponding bit on the other function configuration register. This register has a base address of 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR), and an offset address of 0x0420.

Table 36. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
22	freqscale_percore	RW	0x0	Enable private frequency adjustment registers for each core
23	clken_percore	RW	0x0	Enable private clock for each core

When freqscale_percore is set to 1, the freqscale bit in the private divider configuration register is used to set the divider for its own clock (including freqscale_mode). When freqscale_percore is set to 1, the clken_mode bit in the private frequency division configuration register is used to set the clock enable. Bit in the private frequency division configuration register is used to control the clock enable when clken_percore is set to 1.

This configuration register is defined as follows. The offset address is 0x1050.

Table 37. Processor core private frequency division register

Bit Field	Name	Read/Write	Reset Value	Description
4	freqscale_mode	RW	0x0	Current processor core frequency division mode selection 0: (n+1)/8 1: 1/(n+1)
3	clken	RW	0x0	Current processor core clock enable
2:0	freqscale	RW	0x0	Current processor core frequency divider configuration

5.3. Node Clock Division and Enable Control

The node clock is the clock used by the on-chip network and shared Cache, and has two different control modes, a software setting mode and a hardware automatic frequency division setting.

The node clock does not support full shutdown, so there is no corresponding clken control bit.

5.3.1. Software Configuration

The software setup method is compatible with the 3A3000 processor and uses the same address to set the node frequency division bits in the Function Setup register.

This register has a base address of `0x1fe00000`,It can also be accessed using the configuration register instruction (IOCSR), and an offset address of `0x0180`.

Table 38. Function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
`42:40`	`Node0_freq_ctrl`	RW	`3'b111`	Node 0 frequency division

In line with the processor core's dividing control, the node clock can also be adjusted from `(dividing control value + 1)/8` to `1/(dividing control value + 1)` after dividing by setting the register. This register is located in other function configuration register. The base address is `0x1fe00000` and the offset address is `0x0420`.

Table 39. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description

`36`	`freqscale_mode_node`	RW	`0x0`	Frequency adjustment mode selection for nodes
				
				`0`: `(n+1)/8`
				
				`1`: `1/(n+1)`

5.3.2. Hardware Automatic Configuration

In addition to the active setting by the software, the node clock also supports the automatic frequency division setting triggered by the temperature sensor.

The auto-division setting is set by the software in advance for different temperatures, and the corresponding auto-division setting will be triggered when the temperature of the temperature sensor reaches the corresponding preset value.

In order to ensure the operation of the chip in a high-temperature environment, it can be set so that the high temperature automatic frequency reduction, so that the chip in excess of the preset range of active clock division, to achieve the effect of reducing the chip flip rate. See 12.3 for details on how to set it up.

5.4. HT Controller Frequency Division and Enable Control

The frequency division mechanism of the HT controller is similar to the others. The two HT controllers can be controlled separately. The settings are made using the corresponding bits in the function configuration register. Its base address is `0x1fe00000`,It can also be accessed using the configuration register instruction (IOCSR), and offset address `0x0180`.

Table 40. Function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
`26:24`	`HT0_freq_scale_ctrl`	RW	`3'b111`	HT Controller 0 frequency division
`27`	`HT0_clken`	RW	`1'b1`	Whether to enable HT0

Bit Field	Name	Read/Write	Reset Value	Description
30:28	HT1_freq_scale_ctrl	RW	3'b011	HT Controller 1 frequency division
31	HT1_clken	RW	1'b1	Whether to enable HT1

In line with other frequency division controls, the HT controller clock can be adjusted from `(frequency division control value + 1)/8` to `1/(frequency division control value + 1)` by setting the clock frequency after frequency division through a register. This register is located in other function configuration register. The base address is `0x1fe00000`,It can also be accessed using the configuration register instruction (IOCSR), and the offset address is `0x0420`.

Note that since the HT core clock is derived from the Node clock, it is also affected by the Node clock frequency division.

Table 41. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
39:38	freqscale_mode_HT	RW	0x0	Frequency adjustment mode selection for each HT 0: (n+1)/8 1: 1/(n+1)

5.5. Stable Counter Frequency Ddivision and Enable Control

Stable Counter's frequency division mechanism is similar to the others. It is set using the corresponding bits in the other function configuration register. Its base address is `0x1fe00000`,It can also be accessed using the configuration register instruction (IOCSR), and offset address `0x0420`.

Table 42. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
20	stable_sel	RW	0x0	Stable clock selection 0: SYS CLOCK 1: NODE CLOCK
21	stable_resetn	RW	0x0	Stable clock reset control 1: set reset state 0: unset software reset
40	freqscale_mode_stable	RW	0x0	Frequency adjustment mode selection for stable clock 0: (n+1)/8 1: 1/(n+1)
46:44	freqscale_stable	RW	0x0	Stable clock frequency adjustment register
47	clken_stable	RW	0x0	Stable clock enable

It should be noted that after stable_reset is set to `0`, only the software reset is released. At this time, if `GPIO_FUNC_en[13]` is `1`, the reset of the stable counter is still controlled by `GPIO[13]` (low valid).

The GPIO output enable register base address is 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR), offset address 0x0500.

Table 43. GPIO output enable register

Bit Field	Name	Read/Write	Reset Value	Description
`31:0`	`GPIO_OEn`	RW	`32'hffffffff`	GPIO output enable (active low)
`63:32`	`GPIO_FUNC_En`	RW	`32'hffff0000`	GPIO function enable (active low)

Chapter 6. Software Clock System

Several different levels of usage are defined in the Loongson 3A5000 processor for the clocks used by the system software. Inside the processor core are the legacy counter/compare registers, the stable counter registers, and the chip-level node counter registers.

The following is introductions to stable counter and node counter.

6.1. Stable Counter

The constant clock source in the 3A5000 is called the stable counter, which is a separate master clock from the processor core's own clock and from the node clock.

In the 3A5000, both the processor core clock and the node clock are derived from the master clock, but both can freely control the number of divisions (see the introduction in the previous chapter), while the clock of the stable counter is derived from the input reference clock and can also be independently divided and does not vary with the frequency of other clocks.

Based on this clock source, a timer and a timer are implemented. This chapter mainly introduces the registers related to the Stable couter.

6.1.1. Configuration Address for Stable Timer

Using the Stable counter clock source, a monotonically increasing timer counter and a timer timer that decrements down from the set value are implemented; each processor core has its own independent Stable counter and Stable timer. When the processor accesses the timer, it can only be accessed by `rdhwr`, `DRDTIME` and other specific instructions; when the processor accesses the timer, it can be accessed by address to load/store or by CSR configuration register instructions.

Table 44. Address access method

Name	Offset Address	Read/Write	Description
`Core0_timer_config`	`0x1060`	RW	Timer configuration register for processor core 0
`Core0_timer_ticks`	`0x1070`	R	Timer remaining value for processor core 0
`Core1_timer_config`	`0x1160`	RW	Timer configuration register for processor core 1
`Core1_timer_ticks`	`0x1170`	R	Timer remaining value for processor core 1
`Core2_timer_config`	`0x1260`	RW	Timer configuration register for processor core 2
`Core2_timer_ticks`	`0x1270`	R	Timer remaining value for processor core 2
`Core3_timer_config`	`0x1360`	RW	Timer configuration register for processor core 3
`Core3_timer_ticks`	`0x1370`	R	Timer remaining value for processor core 3

Table 45. Configuration register instruction access method

Name	Offset Address	Read/Write	Description
percore_timer_config	0x1060	RW	Timer configuration register of the current processor core
percore_timer_ticks	0x1070	R	Timer remaining value for the current processor core

Table 46. Register description

Bit Field	Name	Read/Write	Reset Value	Description
timer_config				
63	1	RW	0x1	Reset to 1, and should write 1
62	Periodic	RW	0x0	Cycle count enable. When this bit is 1, the timer is automatically reset to the value of the InitVal field in timer_config after decreasing to 0
61	Enable	RW	0x0	General enable. The timer is active when this bit is 1
47:0	InitVal	RW	0x0	The initial value for conducting the countdown
timer_ticks				
63:48	0	R	0x0	Value 0
47:0	Ticks	R	0x0	The remaining value of the countdown. When not in a cycle count, the value will stay at 48'hffff_ffff_ffff when the count is complete

6.1.2. Clock Control for Stable Counter

The Stable counter can optionally use either the reference clock input or the master clock and can be controlled by software dividing mechanism for dividing the frequency. In general, it is recommended to use the reference clock input, which is able to be completely free from dynamic frequency tuning interference compared to the master clock.

The following is the clock control register of the Stable counter. This register is located in the control chip other function configuration register. The base address is 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR), and the offset address 0x0420.

Table 47. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
20	stable_sel	RW	0x0	Stable clock selection 0: SYS CLOCK 1: NODE CLOCK

Bit Field	Name	Read/Write	Reset Value	Description
21	stable_resetn	RW	0x0	Stable clock reset control 1: set reset state 0: unset software reset
40	freqscale_mode_stable	RW	0x0	Frequency adjustment mode selection for stable clock 0: (n+1)/8 1: 1/(n+1)
46:44	freqscale_stable	RW	0x0	Stable clock frequency adjustment register
47	clken_stable	RW	0x0	Stable clock enable

After the BIOS has configured the Stable counter clock source, the MCSR section in each processor core needs to be updated to control the values of CPUCFG.0x4 and CPUCFG.0x5. Referring to the description in Instruction set features implemented in 3A5000, CPUCFG.0x4 should be filled with the crystal clock frequency in Hz; CPUCFG.0x5[31:16] should be filled with the dividing factor; CPUCFG.0x5[15:0] should be filled with the multiplication factor. The latter two should be filled in with the help of BIOS for calculation, so that the result of CCFreq*CFM/CFD is equal to the actual frequency of Stable Counter.

6.1.3. Calibration of Stable Counter

In the single-chip case, the Counter difference between each core is within 2 cycles, and no special calibration is needed. In the multi-chip case, there are large differences between different chips, and a special hardware and software calibration mechanism is needed to keep the counter difference of each core below 100ns.

First, to ensure that the master clock of each chip does not deviate during use, the same crystal is used to drive SYS_CLK for all chips.

Second, to ensure that the Stable counter of each chip starts timing at the same moment, the hardware needs to use the multiplexing function of two GPIO pins. Node 0 uses GPIO12 to output the reset signal, and all other nodes (including Node 0) use GPIO13 to input the reset signal (which needs to be configured for the Stable counter function). On the motherboard a buffer device need to be used to ensure the reset timing (mainly the signal slope), the better the reset timing, the less the difference in clocks between different chips.

The software must reset the global Stable counter via GPIO12 before using the Stable counter. Before resetting, need to ensure that the clock selection of each chip is consistent and that the reset of each chip has been lifted. This work is usually done by the BIOS. The connection scheme of the system is shown in the following figure.

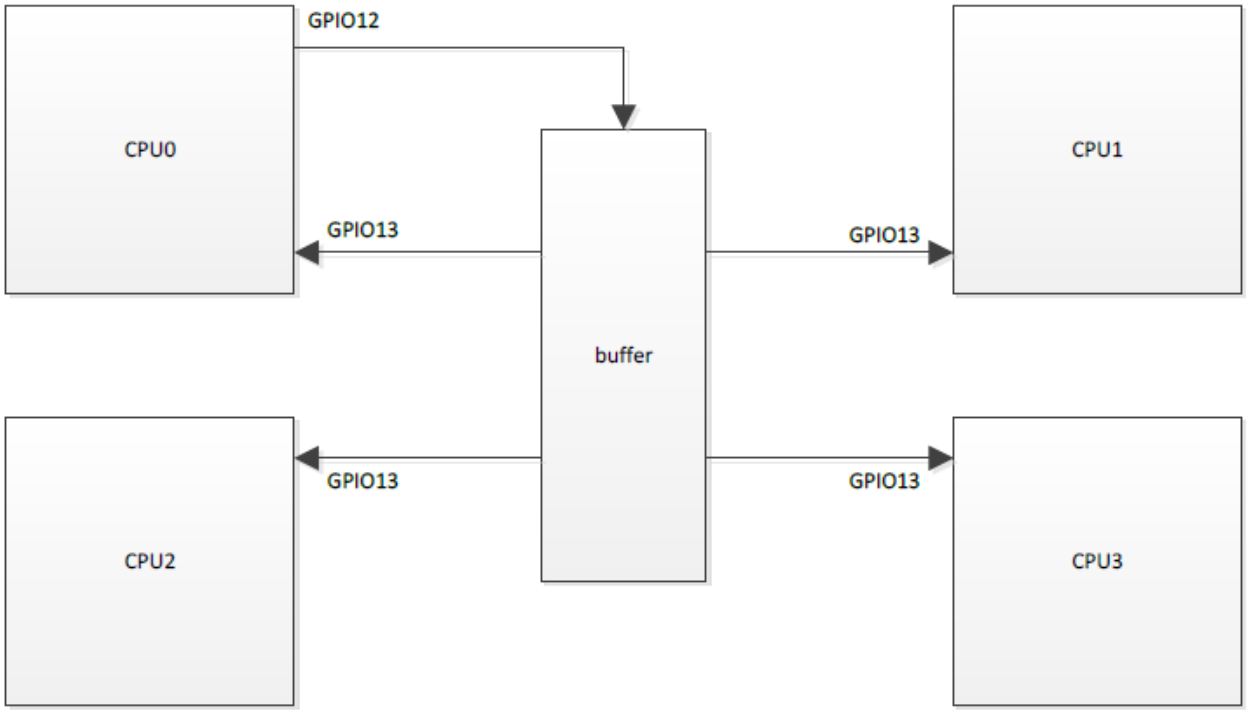


Figure 4. Stable reset control for multi-chip interconnection

6.2. Node Counter

The behavior of the Node counter in the Loongson 3A5000 is the same as that of the 3A4000.

It should be noted that the counting frequency of Node counter is exactly the same as Node clock. If you want to use Node counter as the basis for clock calculation, you should avoid inverting Node clock.

6.2.1. Accessing by Address

The per-address access mode is compatible with the 3A3000 processor and uses the same addresses for setup. The base address of the configuration register is `0x1fe00000`can also be accessed using the configuration register instruction (IOCSR), as shown in the table below.

Table 48. Node counter register

Name	Offset Address	Read/Write	Description
Node counter	0x0408	R	64-bit node clock count

6.3. Summary of Clock System

The new Stable counter in Loongson 3A5000 has an advantage over the node counter and CP0counter in terms of stability, as it does not change with other clocks (node clock and core clock).

In terms of ease of use, the Stable counter is also easier to access, using instruction for both user and Guest states. Stable counter is the preferred solution for software reference clock systems.

Node clock is more of a design for legacy compatibility and is a backup solution for the clock system. It will be phased out in future chip designs.

Chapter 7. GPIO Control

Up to 32 GPIOs are provided in the 3A5000 for system use, and most of them are multiplexed with other functions. The GPIOs can also be configured as interrupt inputs and their interrupt levels can be set through register settings.

The base address of each chip configuration register in this chapter is 0x1fe00000.

7.1. Output Enable Register (0x0500)

The base address is 0x1fe00000 and the offset address is 0x0500.

Table 49. Output enable register

Bit Field	Name	Read/Write	Reset Value	Description
31:0	GPIO_OEn	RW	32'hffffffff	GPIO output enable (active low)
63:32	GPIO_FUNC_En	RW	32'hffff0000	GPIO function enable (active low)

7.2. Input/Output Register (0x0508)

The base address is 0x1fe00000 and the offset address is 0x0508.

Table 50. Input/Output Register

Bit Field	Name	Read/Write	Reset Value	Description
31:0	GPIO_O	RW	32'h0	GPIO output configuration
63:32	GPIO_I	RO	32'h0	GPIO input status

7.3. Interrupt Control Register (0x0510)

The base address is 0x1fe00000 and the offset address is 0x0510.

Table 51. Interrupt control register

Bit Field	Name	Read/Write	Reset Value	Description
31:0	GPIO_INT_Pol	RW	32'h0	GPIO interrupt active level configuration 0 - active low 1 - active high
63:32	GPIO_INT_en	RW	32'h0	GPIO interrupt enable contrl (active high)

7.4. GPIO Pin Function Multiplexing Table

The 3A5000 has a large number of GPIO pins multiplexed with other functions, and the following list shows the pin function selection of the chip function pins.

It should be noted that GPIO00-GPIO15 are GPIO functions when the chip is reset, and the default state is input, not driving I/O.

In order to prevent the internal logic from driving the corresponding I/O, the corresponding HT0/1_Hi/Lo_Hostmode pins can be pulled down. At this point, although the default is still the HT function at reset, it will not drive the I/O pins and will not affect external devices. Only need to set the function to GPIO mode before using the GPIO function in software.

Table 52. Table of GPIO multiplexing function

GPIO Register	Pin Name	Multiplexing Function	Default Function
0	GPIO00	SPI_CSn1	GPIO
1	GPIO01	SPI_CSn2	GPIO
2	GPIO02	UART1_RXD	GPIO
3	GPIO03	UART1_TXD	GPIO
4	GPIO04	UART1_RTS	GPIO
5	GPIO05	UART1_CTS	GPIO
6	GPIO06	UART1_DTR	GPIO
7	GPIO07	UART1_DSR	GPIO
8	GPIO08	UART1_DCD	GPIO
9	GPIO09	UART1_RI	GPIO
10	GPIO10	-	GPIO
11	GPIO11	-	GPIO
12	GPIO12	-	GPIO
13	GPIO13	SCNT_RSTn	GPIO
14	GPIO14	PROCHOTn	GPIO
15	GPIO15	THERMTRIPn	GPIO
16	HT0_LO_POWEROK	GPIO16	HT0_LO_POWEROK
17	HT0_LO_RSTn	GPIO17	HT0_LO_RSTn
18	HT0_LO_LDT_REQn	GPIO18	HT0_LO_LDT_REQn
19	HT0_LO_LDT_STOPn	GPIO19	HT0_LO_LDT_STOPn
20	HT0_HI_POWEROK	GPIO20	HT0_HI_POWEROK
21	HT0_HI_RSTn	GPIO21	HT0_HI_RSTn
22	HT0_HI_LDT_REQn	GPIO22	HT0_HI_LDT_REQn
23	HT0_HI_LDT_STOPn	GPIO23	HT0_HI_LDT_STOPn
24	HT1_LO_POWEROK	GPIO24	HT1_LO_POWEROK
25	HT1_LO_RSTn	GPIO25	HT1_LO_RSTn
26	HT1_LO_LDT_REQn	GPIO26	HT1_LO_LDT_REQn
27	HT1_LO_LDT_STOPn	GPIO27	HT1_LO_LDT_STOPn

GPIO Register	Pin Name	Multiplexing Function	Default Function
28	HT1_HI_POWEROK	GPIO28	HT1_HI_POWEROK
29	HT1_HI_RSTn	GPIO29	HT1_HI_RSTn
30	HT1_HI_LDT_REQn	GPIO30	HT1_HI_LDT_REQn
31	HT1_HI_LDT_STOPn	GPIO31	HT1_HI_LDT_STOPn

7.5. GPIO Interrupt Control

The GPIO pins in the 3A5000 can all be used as interrupt inputs.

GPIO00, GPIO08, GPIO16, GPIO24 share the interrupt controller's interrupt line 0.

GPIO01, GPIO09, GPIO17, GPIO25 share the interrupt controller's interrupt line 1.

GPIO02, GPIO10, GPIO18, GPIO26 share the interrupt controller's interrupt line 2.

GPIO03, GPIO11, GPIO19, GPIO27 share the interrupt controller's interrupt line 3.

GPIO04, GPIO12, GPIO20, GPIO28 share the interrupt controller's interrupt line 4.

GPIO05, GPIO13, GPIO21, GPIO29 share the interrupt controller's interrupt line 5.

GPIO06, GPIO14, GPIO22, GPIO30 share the interrupt controller's interrupt line 6.

GPIO07, GPIO15, GPIO23, GPIO31 share the interrupt controller's interrupt line 7.

The interrupt enable of each GPIO is controlled by the configuration register GPIO_INT_en and the interrupt level is controlled by GPIO_INT_POL, the registers are as follows:

The base address is 0x1fe00000 and the offset address is 0x0510.

Table 53. Interrupt control register

Bit Field	Name	Read/Write	Reset Value	Description
31:0	GPIO_INT_Pol	RW	32'h0	GPIO interrupt active level configuration 0 - active low 1 - active high
63:32	GPIO_INT_en	RW	32'h0	GPIO interrupt enable contrl (active high)

When each interrupt line on the interrupt controller enables only one of the GPIOs, an interrupt can be triggered at a fixed edge (falling edge when POL is set to 0, rising edge when POL is set to 1) and logged in the interrupt controller using edge triggering.

Chapter 8. LA464 Processor Core

LA464 is a quad-launch 64-bit high-performance processor core. It can be used as a single core for high-end embedded and desktop applications, or as a basic processor core to form an on-chip multi-core system for server and high-performance machine applications. The multiple LA464 cores in the Loongson 3A5000 form a distributed multi-core architecture with a shared on-chip last-level Cache through the AXI interconnect network. The main features of LA464 are as follows:

- Support for the Loongson autonomous instruction set (LoongArch).
- Four-launch superscalar architecture with four fixed-point, two vector, and two access components.
- Each vector component is 256 bits wide, and each component supports up to eight double 32-bit floating-point multiplication and addition operations.
- Access components support 256-bit memory access, with 64-bit virtual addresses and 48-bit physical addresses.
- support for register renaming, dynamic scheduling, transfer prediction, and other chaotic execution techniques.
- 64 fully-associative items plus 2048 items connected in 8-way groups, for a total of 2112 TLBs, 64 instruction TLBs, and variable page size.
- First-level instruction Cache and data Cache size of 64KB each, 4-way group concatenation.
- Victim Cache as a private secondary Cache, 256KB in size, 16-way group concatenation.
- Supports access optimization techniques such as Non-blocking access and Load-Speculation.
- Supports Cache Consistency Protocol for on-chip multi-core processors.
- Supports parity check for first-level Cache and ECC check for second-level, on-chip last-level Cache.
- Supports standard JTAG debugging interface for easy hardware and software debugging.

The structure of LA464 is shown in the following figure.

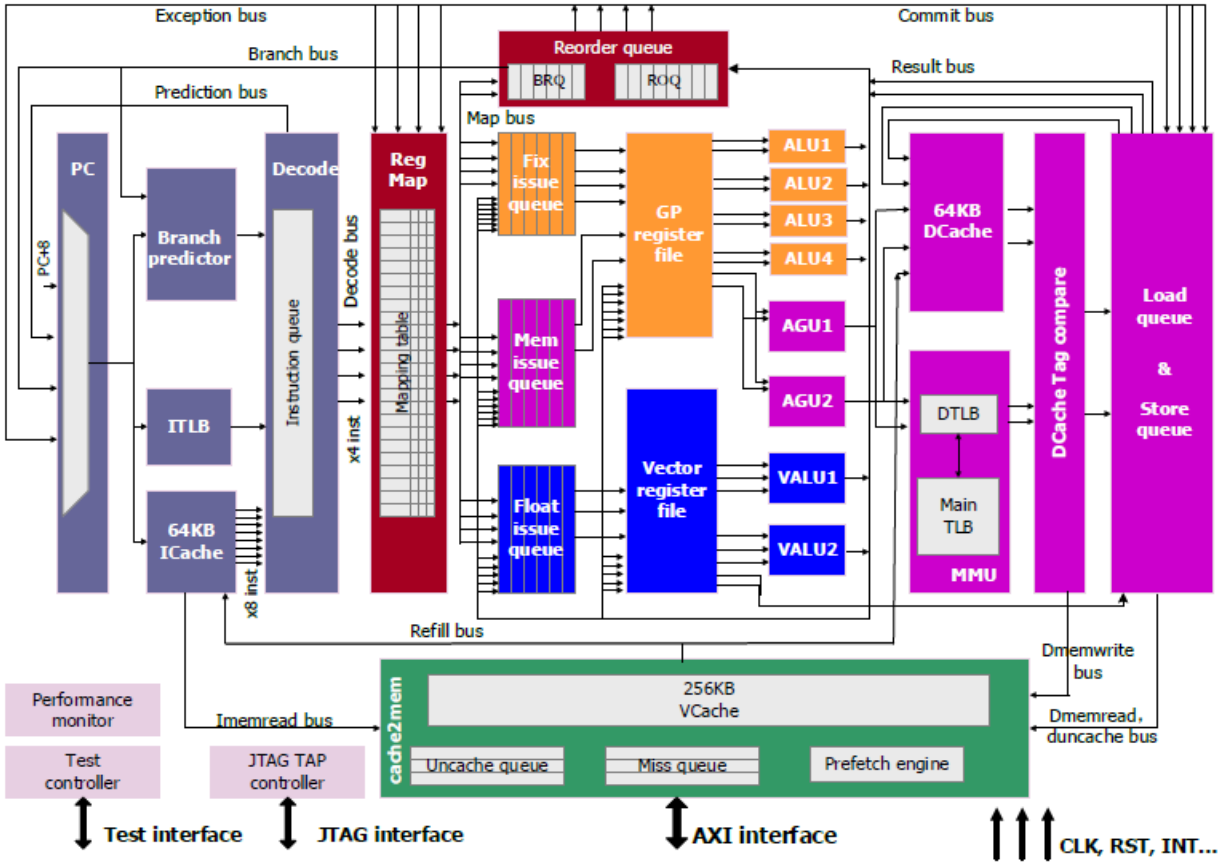


Figure 5. LA464 structure

8.1. Instruction set features implemented in 3A5000

The functional features of the Loongson instruction set implemented in the Loongson 3A5000 can be dynamically confirmed by the Loongson instruction set attribute identification mechanism.

The CPUCFG instruction is a user-state instruction, which is used as CPUCFG rd, rj, where the source operand rj register holds the register number of the configuration information word to be accessed, and the returned configuration word information is written to the rd register, each configuration information word contains up to 32 bits of configuration information. For example, bit 0 of configuration word 1 indicates whether the LA32 architecture is implemented, then this configuration information is expressed as CPUCFG.1.LA32[bit0], where 1 means that the font size of the configuration information word is 1, LA32 means that the helper name of this configuration information field is LA32, and bit 0 means that the field LA32 is located in bit 0 of the configuration word. If the configuration information needs to be expressed in multiple bits, then the location information will be recorded in the form of bitAA:BB, which means the consecutive (AA-BB+1) bits from the AAth to the BBth bit of the configuration information word.

The following table gives a list of configuration information for the instruction set functions implemented in the 3A5000. The last column, “Possible Value”, indicates a possible value to be read from this register, but does not imply that this is the value to be read from the 3A5000 processor. Please refer to the results of the actual hardware execution of the instruction, and make subsequent software judgments based on the actual read values.

Table 54. List of configuration information for the instruction set functions implemented in the 3A5000

Register Number	Bit Field	Name	Description	Possible Value
0x0	31:0	PRId	Processor Identity	32’h14_c010

0x1	1:0	ARCH	2'b00 indicates implementation of simplified LA32; 2'b01 indicates implementation of LA32; 2'b10 indicates implementation of LA64. 2'b11 is reserved	2'b10
	2	PGMMU	1 indicates that the MMU supports page mapping mode	1'b1
	3	IOCSR	1 indicates support for the IOCSR instruction	1'b1
	11:4	PALEN	The value of the supported physical address bits PALEN minus 1	8'd47
	19:12	VALEN	The value of the supported vitrual address bits VALEN minus 1	8'd47
	20	UAL	1 indicates support for non-aligned memory access	1'b1
	21	RI	1 indicates support for the "Read Inhibit" page attribute	1'b1
	22	EP	1 indicates support for "Execution Protection" page attribute	1'b1
	23	RPLV	1 indicates support for RPLV page attributes	1'b1
	24	HP	1 indicates support for huge page page attributes	1'b1
	25	IOCSR_BRD	1 indicates a string with processor product information recorded at address 0 of the IOCSR access space	1'b1
	26	MSG_INT	1 means that the external interrupt uses the message interrupt mode, otherwise it is the level interrupt line mode	1'b0

0x2	0	FP	1 means support for basic floating-point instructions	1'b1
	1	FP_SP	1 indicates support for single-precision floating-point numbers	1'b1
	2	FP_DP	1 indicates support for double-precision floating-point numbers	1'b1
	5:03	FP_ver	The version number of the floating-point arithmetic standard. 1 is the initial version number, indicating that it is compatible with the IEEE 754-2008 standard	3'h1
	6	LSX	1 indicates support for 128-bit vector extension	1'b1
	7	LASX	1 indicates support for 256-bit vector expansion	1'b1
	8	COMPLEX	1 indicates support for complex vector operation instructions	1'b1
	14	LLFTP	1 indicates support for constant frequency counter and timer	1'b1
	17:15	LLFTP_ver	Constant frequency counter and timer version number. 1 is the initial version	3'h1
	21	LSPW	1 indicates support for the software page table walking instruction	1'b1
	22	LAM	1 indicates support AM* atomic memory access instruction	1'b1

0x3	0	CCDMA	1 indicates support for hardware Cache coherent DMA	1'b1
	1	SFB	1 indicates support for Store Fill Buffer (SFB)	1'b1
	3	LLEXC	1 indicates support for LL instruction to fetch exclusive block function	1'b1
	4	SCDLY	1 indicates support random delay function after SC	1'b1
	5	LLDBAR	1 indicates support LL automatic with dbar function	1'b1
	6	ITLBT	1 indicates that the hardware maintains the consistency between ITLB and TLB	1'b1
	7	ICACHET	1 indicates that the hardware maintains the data consistency between ICache and DCache in one processor core	1'b1
	10:8	SPW_LVL	The maximum number of directory levels supported by the page walk instruction	3'h4
	11	SPW_HP_HF	1 indicates that the page walk instruction fills the TLB in half when it encounters a large page	1'b1
	12	RVA	1 indicates that the software configuration can be used to shorten the virtual address range	1'b1
	16:13	RVAMAX-1	The maximum configurable virtual address is shortened by -1	1'b1
0x5	15:00	CC_MUL	Constant frequency counter and timer and the corresponding multiplication factor of the clock used by the timer	N/A
	31:16	CC_DIV	Constant frequency counter and timer and the division coefficient corresponding to the clock used by the timer	N/A
0x6	0	PMP	1 indicates support for the performance counter	1'b1
	3:1	PMVER	In the performance monitor, the architecture defines the version number of the event, and 1 is the initial version	3'h1
	7:4	PMNUM	Number of performance monitors minus 1	4'h3
	13:08	PMBITS	Number of bits of a performance monitor minus 1	6'h3f
	14	UPM	1 indicates support for reading performance counter in user mode	1'b1

0x10	0	L1 IU_Present	1 indicates that there is a first-level instruction Cache or a first-level unified Cache	1'b1
	1	L1 IU Unify	1 indicates that the Cache shown by L1 IU_Present is the unified Cache	1'b0
	2	L1 D Prwsent	1 indicates there is a first-level data Cache	1'b1
	3	L2 IU Present	1 indicates there is a second-level instruction Cache or a second-level unified Cache	1'b1
	4	L2 IU Unitfy	1 indicates that the Cache shown by L2 IU_Present is the unified Cache	1'b1
	5	L2 IU Private	1 indicates that the Cache shown by L2 IU_Present is private to each core	1'b1
	6	L2 IU Inclusive	1 indicates that the Cache shown by L2 IU_Present has an inclusive relationship to the lower levels (L1)	1'b0
	7	L2 D Present	1 indicates there is a secondary data Cache	1'b0
	8	L2 D Private	1 indicates that the secondary data Cache is private to each core	1'b0
	9	L2 D Inclusive	1 indicates that the secondary data Cache has a containment relationship to the lower level (L1)	1'b0
	10	L3 IU Present	1 indicates there is a three-level instruction Cache or a three-level system Cache	1'b1
	11	L3 IU Unify	1 indicates that the Cache shown by L3 IU_Present is unified Cache	1'b1
	12	L3 IU Private	1 indicates that the Cache shown by L3 IU_Present is private to each core	1'b0
	13	L3 IU Inclusive	1 indicates that the Cache shown by L3 IU_Present has an inclusive relationship to the lower levels (L1 and L2)	1'b1
	14	L3 D Present	1 indicates there is a three-level data Cache	1'b0
	15	L3 F Inclusive	1 indicates that the three-level data Cache is private to each core	1'b0
	16	L3 D Inclusive	1 indicates that the three-level data Cache has an inclusive relationship to the lower levels (L1 and 12)	1'b0

0x11	15:0	Way-1	Number of channels minus 1 (Cache corresponding to L1 IU_Present in configuration word 10)	16'h3
	23:16	Index-log2	log2(number of Cache rows per channel) (Cache corresponding to L1 IU_Present in configuration word 10)	8'h8
	30:24	Linesize-log2	log2(Cache line bytes) (Cache corresponding to L1 IU_Present in configuration word 10)	8'h6
0x12	15:0	Way-1	Number of channels minus 1 (Cache corresponding to L1 D Present in Configuration Word 10)	16'h3
	23:16	Index-log2	log2(number of Cache rows per channel) (Cache corresponding to L1 D Present in Configuration Word 10)	8'h8
	30:24	Linesize-log2	log2(Cache row bytes) (Cache corresponding to L1 D Present in configuration word 10)	8'h6
0x13	15:0	Way-1	Number of channels minus 1 (Cache corresponding to L2 IU Present in configuration word 10)	16'hf
	23:16	Index-log2	log2(number of Cache rows per channel) (Cache corresponding to L2 IU Present in configuration word 10)	8'h8
	30:24	Linesize-log2	log2(Cache row bytes) (Cache corresponding to L2 IU Present in configuration word 10)	8'h6
0x14	15:00	Way-1	Number of channels minus 1 (Cache corresponding to L3 IU Present in configuration word 10)	16'hf
	23:16	Index-log2	log2(number of Cache rows per channel) (Cache corresponding to L3 IU Present in configuration word 10)	8'h8
	30:24	Linesize-log2	log2(Cache row bytes) (Cache corresponding to L3 IU Present in configuration word 10)	8'h6

8.2. Access to 3A5000 Control and Status Registers

The 3A5000 supports configuration status register space access. The CSR is accessed using a new, independent addressing space called the CSR space that does not overlap with existing register space, memory space, or JTAG space.

CSR read and write accesses are performed via the custom `IOCSRRD.B/H/W/D` and `IOCSRWR`. `IOCSRRD.B/H/W/D` is used as `IOCSRRD.B/H/W/D rd,rj`, where the source operand `rj` register holds the address of the CSR with access, and the CSR read back is written to the `rd` register. `IOCSRWR.B/H/W/D` is used as `IOCSRWR.B/H/W/D rd,rj`, where the source operand `rj` register holds the address of the CSR with access, and the source operand rd register holds the value of the CSR to be written. `IOCSRRD.B/H/W/D` and `IOCSRWR.B/H/W/D` are allowed to operate in the kernel state only.

`IOCSRRD.B/H/W/D` and `IOCSRWR.B/H/W/D` instructions can be used instead of the original address-mapped configuration registers, i.e., the `0x1fe00000` and `0x3ff00000` spaces, as described in the relevant sections for access details.

Chapter 9. Shared Cache (SCache)

The SCache module is a three-level Cache shared by all processor cores within the Loongson 3A5000 processor. The main features of the SCache module include:

- 16-item Cache access queue.
- Keyword priority.
- Supports Cache Consistency Protocol through directories.
- Can be used in on-chip multi-core architectures or directly interfaced with single-processor IP.
- 16-way group concatenation architecture.
- Supports ECC checksum.
- Supports DMA consistent read/write and prefetch reads.
- Supports 16 types of shared Cache hashing.
- Supports shared Cache by window lock.
- Guaranteed read data return atomicity.

The shared Cache module includes the shared Cache management module `scachemanage` and the shared Cache access module `scacheaccess`. `scachemanage` is responsible for processing access requests from processors and DMAs, while the shared Cache TAG, directory and data information is stored in the `scacheaccess` module. To reduce power consumption, the TAG, directory and data of the shared Cache can be accessed separately. The shared Cache status bits and `w` bits are stored together with TAG, TAG is stored in TAG RAM, directory is stored in DIR RAM and data is stored in DATA RAM. Failure request accesses the shared Cache and reads the TAG, directory of all roads at the same time and picks the directory according to TAG and reads the data according to the hit. Replacement requests, refill requests and write back requests operate only on the TAGs, directories and data of one way.

To improve the performance of some specific computing tasks, a locking mechanism is added to the shared Cache. Blocks that fall in the locked area of the Shared Cache are locked and therefore will not be replaced out of the Shared Cache (unless all 16 paths of the Shared Cache contain locked blocks).

The four sets of lock window registers inside the shared Cache module can be dynamically configured through the chip configuration register space, but it must be ensured that one of the 16 Shared Caches must not be locked. In addition, when the shared Cache receives a DMA write request, if the area to be written is hit and locked in the shared Cache, then the DMA write will be written directly to the shared Cache.

Table 55. Shared Cache lock window register configuration

Name	Address	Bit Field	Description
`Slock0_valid`	`0x00200`	`[63:63]`	Valid bits for lock window 0
`Slock0_addr`	`0x00200`	`[47:0]`	Lock address for lock window 0
`Slock0_mask`	`0x00240`	`[47:0]`	Mask for lock window 0
`Slock1_valid`	`0x00208`	`[63:63]`	Valid bits for lock window 1
`Slock1_addr`	`0x00208`	`[47:0]`	Lock address for lock window 1
`Slock1_mask`	`0x00248`	`[47:0]`	Mask for lock window 1
`Slock2_valid`	`0x00210`	`[63:63]`	Valid bits for lock window 2
`Slock2_addr`	`0x00210`	`[47:0]`	Lock address for lock window 2
`Slock2_mask`	`0x00250`	`[47:0]`	Mask for lock window 2

Name	Address	Bit Field	Description
Slock3_valid	0x00218	[63:63]	Valid bits for lock window 3
Slock3_addr	0x00218	[47:0]	Lock address for lock window 3
Slock3_mask	0x00258	[47:0]	Mask for lock window 3

For example, when an address addr causes slock0_valid && addr & slock0_mask) == (slock0_addr & slock0_mask to be 1, the address is locked by lock window 0.

The 4 SCache use the same configuration register with base address 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR), and offset address 0x0280.

Table 56. Shared Cache configuration register (SC_CONFIG)

Bit Field	Name	Read/Write	Reset Value	Description
0	LRU en	RW	1'b1	SCache LRU replacement algorithm enable
16	Prefetch En	RW	1'b1	SCache prefetch function enable
22:20	Prefetch config	RW	3'h1	Stop prefetching when SCache prefetching exceeds the address range of the configured size 0 - 4KB 1 - 16KB 2 - 64KB 3 - 1MB 7 - Unlimited (Note: Valid when SCID_SEL==0)

Bit Field	Name	Read/Write	Reset Value	Description
26:24	Prefetch lookahead	RW	3'h2	SCache prefetch size 0 - Reserved 1 - 0x100 2 - 0x200 3 - 0x300 4 - 0x400 5 - 0x500 6 - 0x600 7 - 0x700 (Note: Valid when SCID_SEL==0)
30:28	Sc stall dirq cycle	RW	3'h2	Number of clock cycles of SC instruction blocking dirq 0 - 1 cycle (nonstall) 1 - 16-31 cycle random 2 - 32-63 cycle random 3 - 64-127 cycle random 4 - 128-255 cycle random Other - Invalid
31	MCC storefill en	RW	1'b0	MCC storefill function enable
34:32				
35	MCC clean exclusive replace en	RW	1'b0	
36	MCC clean shared replace en	RW	1'b0	

Chapter 10. Inter-Processor Interrupts and Communication

The Loongson 3A5000 implements eight inter-processor interrupt registers (IPI) for each processor core to support interrupts and communication between processor cores during multi-core BIOS boot and OS runtime.

Two different access modes are supported in the Loongson 3A5000, one is an address access mode compatible with processors such as the 3A3000, and the other is designed to support direct private access to the processor register space. Separate descriptions are provided in later sections.

10.1. Accessing by Address

For the Loongson 3A5000, the following registers can be accessed using base address `0x1fe0_0000`,It can also be accessed using the configuration register instruction (IOCSR). Of these, base address `0x3ff0_0000` can be turned off by the `disable_0x3ff0` control bit in the Routing Setup Register. See the tables below for specific register descriptions and addresses.

Table 57. Registers related to inter-processor interrupt and their functional descriptions

Name	Read/Write	Description
`IPI_Status`	R	The processor core `INT4` interrupt line is set if any bit is set to `1` and the corresponding bit is enabled.
`IPI_Enable`	RW	32-bit enable register to control whether the corresponding interrupt bit is valid or not.
`IPI_Set`	W	32-bit set register. Write `1` to the corresponding bit, then the corresponding `STATUS` register bit is set to `1`.
`IPI_Clear`	W	32-bit clear register. Write `1` to the corresponding bit, then the corresponding `STATUS` register bit is cleared to `0`.
`MailBox0`	RW	Cache registers for passing parameters at boot time, accessible as 64-bit or 32-bit uncache.
`MailBox01`	RW	Cache registers for passing parameters at boot time, accessible as 64-bit or 32-bit uncache.
`MailBox02`	RW	Cache registers for passing parameters at boot time, accessible as 64-bit or 32-bit uncache.
`MailBox03`	RW	Cache registers for passing parameters at boot time, accessible as 64-bit or 32-bit uncache.

The registers related to interprocessor interrupts and their functions in the Loongson 3A5000 are described as follows:

Table 58. List of inter-processor interrupt and communication registers for processor core 0

Name	Offset Address	Read/Write	Description
`Core0_IPI_Status`	`0x1000`	R	`IPI_Status` register of processor core 0
`Core0_IPI_Enalbe`	`0x1004`	RW	`IPI_Enalbe` register of processor core 0

Name	Offset Address	Read/Write	Description
Core0_IPI_Set	0x1008	W	IPI_Set register of processor core 0
Core0_IPI_Clear	0x100c	W	IPI_Clear register of processor core 0
Core0_MailBox0	0x1020	RW	IPI_MailBox0 register of processor core 0
Core0_MailBox1	0x1028	RW	IPI_MailBox1 register of processor core 0
Core0_MailBox2	0x1030	RW	IPI_MailBox2 register of processor core 0
Core0_MailBox3	0x1038	RW	IPI_MailBox3 register of processor core 0

Table 59. List of inter-processor interrupt and communication registers for processor core 1

Name	Offset Address	Read/Write	Description
Core1_IPI_Status	0x1100	R	IPI_Status register of processor core 1
Core1_IPI_Enalbe	0x1104	RW	IPI_Enalbe register of processor core 1
Core1_IPI_Set	0x1108	W	IPI_Set register of processor core 1
Core1_IPI_Clear	0x110c	W	IPI_Clear register of processor core 1
Core1_MailBox0	0x1120	R	IPI_MailBox0 register of processor core 1
Core1_MailBox1	0x1128	RW	IPI_MailBox1 register of processor core 1
Core1_MailBox2	0x1130	W	IPI_MailBox2 register of processor core 1
Core1_MailBox3	0x1138	W	IPI_MailBox3 register of processor core 1

Table 60. List of inter-processor interrupt and communication registers for processor core 2

Name	Offset Address	Read/Write	Description
Core2_IPI_Status	0x1200	R	IPI_Status register of processor core 2
Core2_IPI_Enalbe	0x1204	RW	IPI_Enalbe register of processor core 2
Core2_IPI_Set	0x1208	W	IPI_Set register of processor core 2
Core2_IPI_Clear	0x120c	W	IPI_Clear register of processor core 2

Name	Offset Address	Read/Write	Description
Core2_MailBox0	0x1220	R	IPI_MailBox0 register of processor core 2
Core2_MailBox1	0x1228	RW	IPI_MailBox1 register of processor core 2
Core2_MailBox2	0x1230	W	IPI_MailBox2 register of processor core 2
Core2_MailBox3	0x1238	W	IPI_MailBox3 register of processor core 2

Table 61. List of inter-processor interrupt and communication registers for processor core 3

Name	Offset Address	Read/Write	Description
Core3_IPI_Status	0x1300	R	IPI_Status register of processor core 3
Core3_IPI_Enalbe	0x1304	RW	IPI_Enalbe register of processor core 3
Core3_IPI_Set	0x1308	W	IPI_Set register of processor core 3
Core3_IPI_Clear	0x130c	W	IPI_Clear register of processor core 3
Core3_MailBox0	0x1320	R	IPI_MailBox0 register of processor core 3
Core3_MailBox1	0x1328	RW	IPI_MailBox1 register of processor core 3
Core3_MailBox2	0x1330	W	IPI_MailBox2 register of processor core 3
Core3_MailBox3	0x1338	W	IPI_MailBox3 register of processor core 3

The above is a list of inter-processor interrupt-related registers for a single-node multiprocessor system composed of a single Loongson 3A5000 chip. When multiple Loongson 3A5000 chips are interconnected to form a multi-node CC-NUMA system, each node within a chip corresponds to a global system node number, and the IPI register addresses of the processor cores within the node are fixed at an offset from the base address of the node in the table above. For example, the IPI_Status address of processor core 0 of node 0 is 0x1fe01000, while the processor address of processor 0 of node 1 is 0x10001fe01000, and so on.

10.2. Accessing by Configuration Register Instructions

In the Loongson 3A5000, a new processor core direct register access instruction has been added to provide access to configuration registers through private space. In order to use inter-processor interrupt registers more easily, some adjustments are made to the inter-processor interrupt register definitions in this mode.

Table 62. List of inter-processor interrupt and communication registers for the current processor core

Name	Offset Address	Read/Write	Description
perCore_IPI_Status	0x1000	R	IPI_Status register of the current processor core
perCore_IPI_Enalbe	0x1004	RW	IPI_Enalbe register of the current processor core
perCore_IPI_Set	0x1008	W	IPI_Set register of the current processor core
perCore_IPI_Clear	0x100c	W	IPI_Clear register of the current processor core
perCore_MailBox0	0x1020	RW	IPI_MailBox0 register of the current processor core
perCore_MailBox1	0x1028	RW	IPI_MailBox1 register of the current processor core
perCore_MailBox2	0x1030	RW	IPI_MailBox2 register of the current processor core
perCore_MailBox3	0x1038	RW	IPI_MailBox3 register of the current processor core

In order to send inter-processor interrupt requests and MailBox communication to other cores, the following registers are accessed.

Table 63. Processor core inter-processor communication registers

Name	Offset Address	Read/Write	Description
IPI_Send	0x1040	WO	32-bit interrupt distribution register [31]: wait for completion flag; when set to 1 it will wait for the interrupt to take effect [30:26]: reserved [25:16]: processor core number [15:5]: reserved [4:0]: interrupt vector number, corresponding to the vector in IPI_Status

Name	Offset Address	Read/Write	Description
Mail_Send	0x1048	WO	64-bit MailBox Cache register [63:32]: MailBox data [31]: wait for completion flag; when set to 1 it will wait for the write to take effect [30:27]: write data mask; each bit indicates that the bytes corresponding to the 32-bit write data will not really be written to the target address, such as 1000b means write the 0-2 bytes, 0000b means write all 0-3 bytes [26]: reserved [25:16]: processor core number [15:5]: reserved [4:2]: MailBox number 0 - MailBox0 low 32-bit 1 - MailBox0 high 32-bit 2 - MailBox1 low 32-bit 3 - MailBox1 high 32-bit 4 - MailBox2 low 32-bit 5 - MailBox2 high 32-bit 6 - MailBox3 low 32-bit 7 - MailBox4 high 32-bit [1:0]: reserved

Name	Offset Address	Read/Write	Description
FREQ_Send	0x1058	WO	32-bit frequency enable register [31]: wait for completion flag; when set to 1 it will wait for the setting to take effect [30:27]: write data mask; each bit indicates that the bytes corresponding to the 32-bit write data will not really be written to the target address, such as 1000b means write the 0-2 bytes, 0000b means write all 0-3 bytes [26]: reserved [25:16]: processor core number [15:5]: reserved [4:0]: write to the corresponding processor core private frequency configuration register CSR[0x1050]

Note that since the Mail_Send register can only send 32 bits of data at a time, it must be split into two transmissions when sending 64 bits of data. Therefore, the target core needs to ensure the integrity of the transport by other software means while waiting for the contents of the Mail_Box. For example, after sending the Mail_Box data, an inter-processor interrupt is used to indicate that the transmission is complete.

10.3. Debug Support for Configuration Register Instructions

The configuration register instruction is in principle used without cross-chip access, but in order to meet the needs for debugging, etc., cross-chip access is supported here by using multiple register addresses. It is worth noting that such registers can only be written, not read.

In addition to IPI_Send, Mail Send, Freq Send mentioned in the previous section, there is also an Any Send register available with the following address.

Table 64. Processor core inter-processor communication registers

Name	Offset Address	Read/Write	Description
ANY_Send	0x1158	WO	64-bit register access register [63:32]: data being written [31]: wait for completion flag; when set to 1 it will wait for the interrupt to take effect [30:27]: write data mask; each bit indicates that the bytes corresponding to the 32-bit write data will not really be written to the target address, such as 1000b means write the 0-2 bytes, 0000b means write all 0-3 bytes [26]: reserved [25:16]: destination processor core number [15:0]: offset address of the register to be written

Chapter 11. I/O Interrupts

The Loongson 3A5000 chip supports two different interrupt methods. The first is the legacy interrupt method, which is compatible with processors such as the 3A3000, and the second is the new extended I/O interrupt method, which is used to support the interrupt cross-chip and dynamic distribution functions of the HT controller. The following describes each of the two interrupt methods.

11.1. Legacy I/O Interrupts

The legacy interrupts on the Loongson 3A5000 chip support 32 interrupt sources managed in a unified manner as shown in the figure below. Any of the I/O interrupt sources can be configured to enable or disable, how it is triggered, and the target processor core interrupt pin to be routed. Legacy interrupts do not support cross-chip distribution of interrupts, and can only interrupt processor cores within the same processor chip.

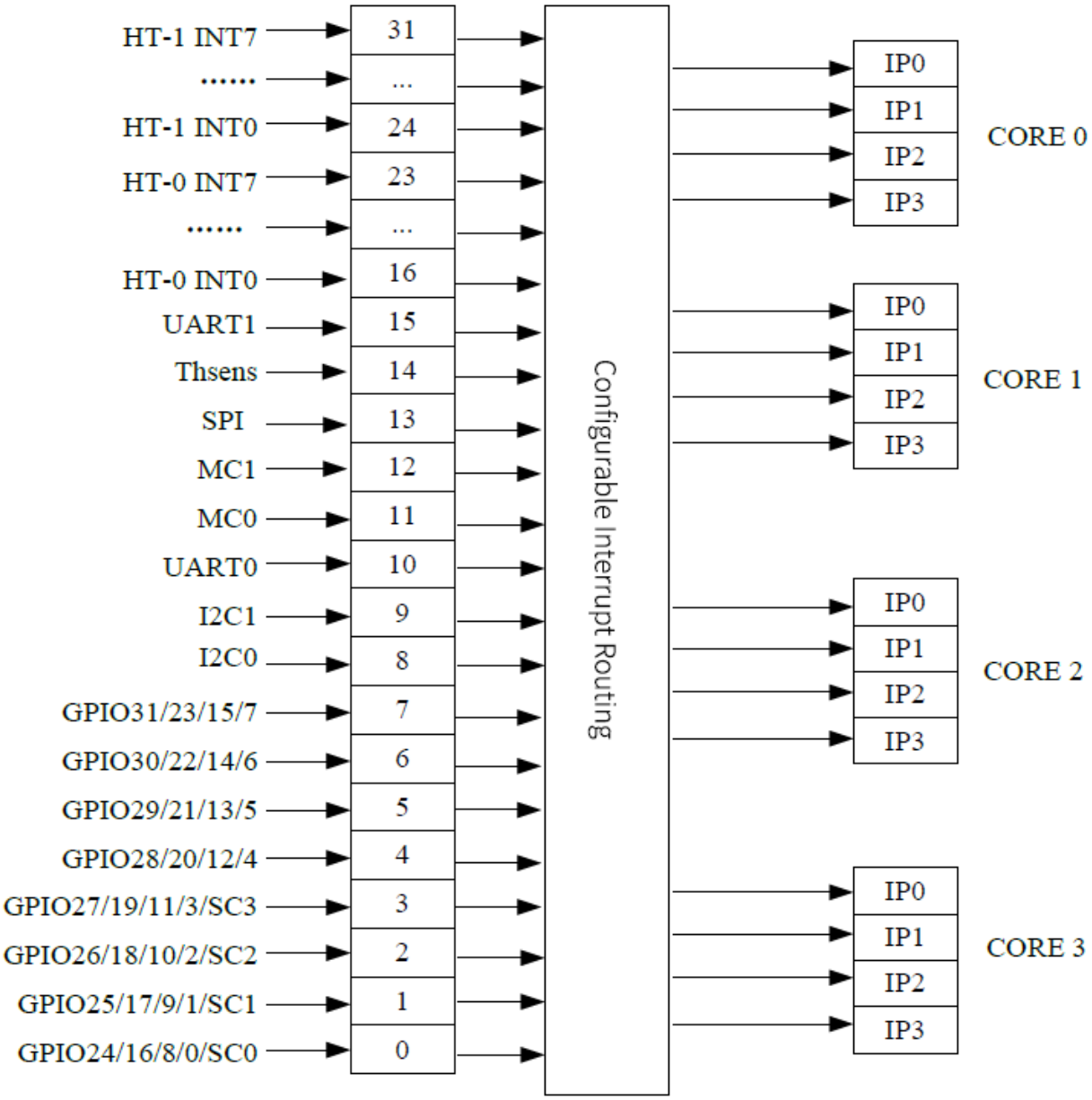


Figure 6. Interrupt routing of Loongson 3A5000 processor

The interrupt-related configuration registers are in the form of bits to control the corresponding interrupt lines, and the interrupt control bit connections and attributes are configured in the following table.

The configuration of interrupt enable (Enable) has three registers: Intenset, Intenclr and Inten. Intenset sets the interrupt enable, the interrupt corresponding to the bit written 1 in the Intenset register is enabled. Intenclr clears the interrupt enable, the interrupt corresponding to the bit written 1 in the Intenclr register is cleared. The Inten register reads the current status of each interrupt enable. The edge-triggered interrupt signal is selected by the Intedge configuration register, with a write of 1 for edge-triggered and a write of 0 for level-triggered. The interrupt handler can clear the interrupt record by using the corresponding bit of Intenclr. Clearing the interrupt will also clear the interrupt enable.

Table 65. Interrupt control register

Bit Field	Read/Write (Default Value)				
Intedge	Inten	Intenset	Intenclr	Interrupt Source	
0	RW (0)	R (0)	RW (0)	RW (0)	GPIO24/16/8/0/SC0
1	RW (0)	R (0)	RW (0)	RW (0)	GPIO25/17/9/1/SC1
2	RW (0)	R (0)	RW (0)	RW (0)	GPIO26/18/10/2/SC2
3	RW (0)	R (0)	RW (0)	RW (0)	GPIO27/19/11/3/SC3
4	RW (0)	R (0)	RW (0)	RW (0)	GPIO28/20/12/4
5	RW (0)	R (0)	RW (0)	RW (0)	GPIO29/21/13/5
6	RW (0)	R (0)	RW (0)	RW (0)	GPIO30/22/14/6
7	RW (0)	R (0)	RW (0)	RW (0)	GPIO31/23/15/7
8	RW (0)	R (0)	RW (0)	RW (0)	I2C0
9	RW (0)	R (0)	RW (0)	RW (0)	I2C1
10	RW (0)	R (0)	RW (0)	RW (0)	UART0
11	RW (0)	R (0)	RW (0)	RW (0)	MC0
12	RW (0)	R (0)	RW (0)	RW (0)	MC1
13	RW (0)	R (0)	RW (0)	RW (0)	SPI
14	RW (0)	R (0)	RW (0)	RW (0)	Thsens
15	RW (0)	R (0)	RW (0)	RW (0)	UART1
23:16	RW (0)	R (0)	RW (0)	RW (0)	HT0[7:0]
31:24	RW (0)	R (0)	RW (0)	RW (0)	HT1[7:0]

Similar to inter-processor interrupts, the base address of I/O interrupts can also be accessed using 0x1fe00000, or through the processor core's dedicated register configuration instructions.

11.1.1. Accessing by Address

This access is compatible with that of processors such as the 3A3000, where either 0x1fe00000 or 0x3ff00000 can be used for the base address. The 0x3ff00000 base address can be disabled via the disable_0x3ff0 control bit in the Routing Configuration Register.

Table 66. I/O control register address

Name	Offset Address	Description
Intisr	0x1420	32-bit interrupt status register
Inten	0x1424	32-bit interrupt enable status register
Intenset	0x1428	32-bit set enable register
Intenclr	0x142c	32-bit clear enable register
Intedge	0x1434	32-bit trigger mode register
CORE0_INTISR	0x1440	32-bit interrupt status routed to CORE0
CORE1_INTISR	0x1448	32-bit interrupt status routed to CORE1
CORE2_INTISR	0x1450	32-bit interrupt status routed to CORE2
CORE3_INTISR	0x1458	32-bit interrupt status routed to CORE3

Four processor cores are integrated in the Loongson 3A5000, and the 32-bit interrupt sources described above can be software configured to select the target processor core for the desired interrupt. Further, the interrupt sources can be routed to any of the processor cores INT0 to INT3. Each of the 32 I/O interrupt sources corresponds to an 8-bit routing controller with the format and addresses shown in Description of the interrupt destination processor core routing register and Interrupt destination processor core routing register address. The routing register uses a vector approach to routing, e.g., 0x48 indicates routing to INT2 of processor 3.

Starting with the 3A5000, the interrupt pin routing bits are added in a coded manner and are enabled by the CSR[0x420][49] bit control. When this bit is enabled, the [7:4] bits in the table below change from a bitmap representation to a numeric encoding method. Configurable values 0-7 indicate interrupt pins 0-7. For example, in this mode, 0x28 indicates routing to INT2 on processor 3.

Table 67. Description of the interrupt routing register

Bit Field	Description
3:0	Processor core vector number for routing
7:4	Processor core interrupt pin vector number for routing

Table 68. Interrupt routing register address

Name	Offset Address	Description	Name	Offset Address	Description
Entry0	0x1400	GPIO24/16/8/0	Entry16	0x1410	HT0-int0
Entry1	0x1401	GPIO25/17/9/1	Entry17	0x1411	HT0-int1
Entry2	0x1402	GPIO26/18/10/2	Entry18	0x1412	HT0-int2
Entry3	0x1403	GPIO27/19/11/3	Entry19	0x1413	HT0-int3
Entry4	0x1404	GPIO28/20/12/4	Entry20	0x1414	HT0-int4
Entry5	0x1405	GPIO29/21/13/5	Entry21	0x1415	HT0-int5
Entry6	0x1406	GPIO30/22/14/6	Entry22	0x1416	HT0-int6
Entry7	0x1407	GPIO31/23/15/7	Entry23	0x1417	HT0-int7
Entry8	0x1408	I2C0	Entry24	0x1418	HT1-int0

Name	Offset Address	Description	Name	Offset Address	Description
Entry9	0x1409	I2C1	Entry25	0x1419	HT1-int1
Entry10	0x140a	UART0	Entry26	0x141a	HT1-int2
Entry11	0x140b	MC0	Entry27	0x141b	HT1-int3
Entry12	0x140c	MC1	Entry28	0x141c	HT1-int4
Entry13	0x140d	SPI	Entry29	0x141d	HT1-int5
Entry14	0x140e	Thsens	Entry30	0x141e	HT1-int6
Entry15	0x140f	UART1	Entry31	0x141f	HT1-int7

11.1.2. Accessing by Configuration Register Instructions

In the Loongson 3A5000, the configuration registers can also be accessed through private space using the same access method as the configuration register instruction. The offset address used by the instruction is the same as that accessed through the address. In addition, for the convenience of users, a dedicated private interrupt status register is set for different current interrupt states of each core, as shown in the following table.

Table 69. Processor core private interrupt status register

Name	Offset Address	Description
perCore_INTISR	0x1010	32-bit interrupt status routing to the current processor core

11.2. Extended I/O Interrupts

In addition to being compatible with the legacy I/O interrupt method, the 3A5000 supports extended I/O interrupts, which are used to distribute 256-bit interrupts on the HT bus directly to each processor core instead of forwarding them through the HT interrupt line, increasing the flexibility of I/O interrupt usage.

Before the core can use the extended I/O interrupt, it needs to enable the corresponding bit in the “Other function configuration register”. This register has a base address of 0x1fe00000,It can also be accessed using the configuration register instruction (IOCSR), an offset address of 0x0420.

Table 70. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
48	EXT_INT_en	RW	0x0	Extended I/O interrupt enable

In Extended I/O interrupt mode, HT interrupts can be forwarded directly across slices and distributed in rotation. In the current version, up to s extended interrupt vectors can be supported.

11.2.1. Accessing by Address

The following are the associated extended I/O interrupt registers. As with the other configuration registers, the base address can be used as 0x1fe00000, or can be accessed via the processor core’s dedicated register configuration instructions.

Table 71. Extended I/O interrupt enable register

Name	Offset Address	Description
EXT_IOIen[63:0]	0x1600	Interrupt enable configuration for extended I/O interrupt [63:0]
EXT_IOIen[127:64]	0x1608	Interrupt enable configuration for extended I/O interrupt [127:64]
EXT_IOIen[191:128]	0x1610	Interrupt enable configuration for extended I/O interrupt [191:128]
EXT_IOIen[255:192]	0x1618	Interrupt enable configuration for extended I/O interrupt [255:192]

Table 72. Extended I/O interrupt auto-rotation enable register

Name	Offset Address	Description
EXT_IOIbounce[63:0]	0x1680	Auto-rotation enable register for extended I/O interrupt [63:0]
EXT_IOIbounce[127:64]	0x1688	Auto-rotation enable register for extended I/O interrupt [127:64]
EXT_IOIbounce[191:128]	0x1690	Auto-rotation enable register for extended I/O interrupt [191:128]
EXT_IOIbounce[255:192]	0x1698	Auto-rotation enable register for extended I/O interrupt [255:192]

Table 73. Extended I/O interrupt interrupt status register

Name	Offset Address	Description
EXT_IOIsr[63:0]	0x1700	Interrupt status for extended I/O interrupt [63:0]
EXT_IOIsr[127:64]	0x1708	Interrupt status for extended I/O interrupt [127:64]
EXT_IOIsr[191:128]	0x1710	Interrupt status for extended I/O interrupt [191:128]
EXT_IOIsr[255:192]	0x1718	Interrupt status for extended I/O interrupt [255:192]

Table 74. Extended I/O interrupt status register for each processor core

Name	Offset Address	Description
CORE0_EXT_IOIsr[63:0]	0x1800	Interrupt status of extended I/O interrupt [63:0] routed to processor core 0
CORE0_EXT_IOIsr[127:64]	0x1808	Interrupt status of extended I/O interrupt [127:64] routed to processor core 0
CORE0_EXT_IOIsr[191:128]	0x1810	Interrupt status of extended I/O interrupt [191:128] routed to processor core 0
CORE0_EXT_IOIsr[255:192]	0x1818	Interrupt status of extended I/O interrupt [255:192] routed to processor core 0

Name	Offset Address	Description
CORE1_EXT_IOIsr [63:0]	0x1900	Interrupt status of extended I/O interrupt [63:0] routed to processor core 1
CORE1_EXT_IOIsr [127:64]	0x1908	Interrupt status of extended I/O interrupt [127:64] routed to processor core 1
CORE1_EXT_IOIsr [191:128]	0x1910	Interrupt status of extended I/O interrupt [191:128] routed to processor core 1
CORE1_EXT_IOIsr [255:192]	0x1918	Interrupt status of extended I/O interrupt [255:192] routed to processor core 1
CORE2_EXT_IOIsr [63:0]	0x1A00	Interrupt status of extended I/O interrupt [63:0] routed to processor core 2
CORE2_EXT_IOIsr [127:64]	0x1A08	Interrupt status of extended I/O interrupt [127:64] routed to processor core 2
CORE2_EXT_IOIsr [191:128]	0x1A10	Interrupt status of extended I/O interrupt [191:128] routed to processor core 2
CORE2_EXT_IOIsr [255:192]	0x1A18	Interrupt status of extended I/O interrupt [255:192] routed to processor core 2
CORE3_EXT_IOIsr [63:0]	0x1B00	Interrupt status of extended I/O interrupt [63:0] routed to processor core 3
CORE3_EXT_IOIsr [127:64]	0x1B08	Interrupt status of extended I/O interrupt [127:64] routed to processor core 3
CORE3_EXT_IOIsr [191:128]	0x1B10	Interrupt status of extended I/O interrupt [191:128] routed to processor core 3
CORE3_EXT_IOIsr [255:192]	0x1B18	Interrupt status of extended I/O interrupt [255:192] routed to processor core 3

Similar to legacy I/O interrupts, the 256-bit interrupt source for Extended I/O interrupts can be software-configured to select the target processor core for the desired interrupt.

However, the interrupt sources are not individually selected to route to any of the processor core interrupts INT0 through INT3, but rather the routing of INT interrupts is done in groups. The following are the interrupt pin routing registers configured by group.

Starting with the 3A5000, the interrupt pin routing bits have been added in a coded manner and are enabled by the CSR[0x420][49] bit control. When this bit is enabled, the [3:0] bits in the table below changes from a bitmap representation to a numeric encoding method. Configurable values 0-7 indicate interrupt pins 0-7. For example, in this mode, 0x2 indicates routing to INT2.

Table 75. Description of the interrupt pin routing register

Bit Field	Description
3:0	Processor core interrupt pin vector number for routing
7:4	Reserved

Table 76. Interrupt routing register address

Name	Offset Address	Description
EXT_IOImap0	0x14C0	Pin routing method of EXT_IOI[31:0]
EXT_IOImap1	0x14C1	Pin routing method of EXT_IOI[63:32]
EXT_IOImap2	0x14C2	Pin routing method of EXT_IOI[95:64]
EXT_IOImap3	0x14C3	Pin routing method of EXT_IOI[127:96]
EXT_IOImap4	0x14C4	Pin routing method of EXT_IOI[159:128]
EXT_IOImap5	0x14C5	Pin routing method of EXT_IOI[191:160]
EXT_IOImap6	0x14C6	Pin routing method of EXT_IOI[223:192]
EXT_IOImap7	0x14C7	Pin routing method of EXT_IOI[255:224]

Each interrupt source additionally corresponds to an 8-bit routing controller with the format and address shown in Description of the interrupt destination processor core routing register and Interrupt destination processor core routing register address. The [7:4] bits are used to select the real node routing vector in Interrupt destination node mapping method configuration. The routing register uses a vector approach for routing, e.g., 0x48 indicates a route to processor core 3 of the node referred to by EXT_IOI_node_type4.

Table 77. Description of the interrupt destination processor core routing register

Bit Field	Description
3:0	Processor core vector number for routing
7:4	Selection of node mapping method of routing (as configured in Interrupt destination node mapping method configuration)

Note that when using the rotating distribution mode (corresponding to an EXT_IOIbounce of 1), rotate on the fully mapped mode of node number to processor core number. The setting of EXT_IOIbounce should follow the associated route mapping configuration.

For example, when the setting in the tables above is 0x27 and the setting of EXT_IOI_node_type2 in the the tables below is 0x0013, the interrupt will rotate in turn on node 0 core 0, node 0 core 1, node 0 core 2, node 1 core 0, node 1 core 1, node 1 core 2, node 4 core 0, node 4 core 1, and node 4 core 2.

When using fixed distribution mode (corresponding to an EXT_IOIbounce of 0), only one bit on the bitmap of the node number is allowed to be 1, or all 0 values, corresponding to local triggering.

Table 78. Interrupt destination processor core routing register address

Name	Offset Address	Description
EXT_IOImap_Core0	0x1C00	Processor core routing method of EXT_IOI[0]
EXT_IOImap_Core1	0x1C01	Processor core routing method of EXT_IOI[1]
EXT_IOImap_Core2	0x1C02	Processor core routing method of EXT_IOI[2]
……		
EXT_IOImap_Core254	0x1CFE	Processor core routing method of EXT_IOI[254]

Name	Offset Address	Description
EXT_IOImap_Core255	0x1CFF	Processor core routing method of EXT_IOI[255]

Table 79. Interrupt destination node mapping method configuration

Name	Offset Address	Description
EXT_IOI_node_type0	0x14A0	Mapping vector type 0 for 16 nodes (software configuration)
EXT_IOI_node_type1	0x14A2	Mapping vector type 1 for 16 nodes (software configuration)
EXT_IOI_node_type2	0x14A4	Mapping vector type 2 for 16 nodes (software configuration)
……		
EXT_IOI_node_type15	0x14BE	Mapping vector type 15 for 16 nodes (software configuration)

11.2.2. Accessing by Configuration Register Instructions

The biggest difference when accessing using the processor core's configuration register instructions is that access to the processor core's interrupt status registers becomes private, and each core only needs to issue a query request to the same address to get the current core's interrupt status.

Table 80. Extended I/O interrupt status register for the current processor core

Name	Offset Address	Description
perCore_EXT_IOIsr[63:0]	0x1800	Interrupt status of the extended I/O interrupt [63:0] routed to the current processor core
perCore_EXT_IOIsr[127:64]	0x1808	Interrupt status of the extended I/O interrupt [127:64] routed to the current processor core
perCore_EXT_IOIsr[191:128]	0x1810	Interrupt status of the extended I/O interrupt [191:128] routed to the current processor core
perCore_EXT_IOIsr[255:192]	0x1818	Interrupt status of the extended I/O interrupt [255:192] routed to the current processor core

11.2.3. Extended I/O Interrupt Trigger Register

To support the dynamic distribution of extended I/O interrupts, an extended I/O interrupt trigger register is added to the configuration register to set the corresponding I/O interrupts to be set. This register can be used for debugging or testing interrupts in normal times.

The description of this register is as follows:

Table 81. Extended I/O interrupt trigger register

Name	Offset Address	Read/Write	Description
EXT_IOI_send	0x1140	WO	Extended I/O interrupt setting register [7:0] is the interrupt vector expected to be set

11.2.4. Difference in Handling Between Extended I/O Interrupts and Legacy HT Interrupts

With legacy HT interrupt processing, HT interrupts are processed internally by the HT controller and mapped directly to the 256 interrupt vectors on the HT configuration registers, and then the 256 interrupt vectors are grouped to generate 4 or 8 interrupts that are routed to the various processor cores. Due to the legacy interrupt line connection, no cross-chip interrupts can be generated directly, so all HT I/O interrupts can only be handled directly by a single chip. On the other hand, the interrupts distributed by the hardware within the chip are only in units of the final 4 or 8 interrupts and cannot be handled on a bit-by-bit basis, which leads to the problem of poor hardware interrupt distribution.

With the extended I/O interrupt method, HT interrupts are sent directly from the HT controller to the chip's interrupt controller for processing, and the interrupt controller can directly get 256 Instead of the previous 4 or 8 interrupts, each of these 256-bit interrupts can be routed and distributed independently, and can be distributed and rotated across slices.

With Extended I/O interrupts, the software processing is slightly different than with legacy HT interrupts.

With legacy HT interrupts, the kernel looks directly at the interrupt vector of the HT controller (typically 0x90000efdfb000080) and then processes the interrupts by bit, regardless of how the routing mode is configured.

After using Extended I/O interrupts, the cores go directly to the Extended I/O status register (configuration space 0x1800) to read the interrupt status for processing. Each core will only read the interrupt's own interrupt status and process it, and there will be no interference between different cores.

Chapter 12. Temperature Sensor

12.1. Real-time Temperature Collection

Two temperature sensors are integrated inside the Loongson 3A5000, which can be observed through the sampling register starting at 0x1FE00198, and can be controlled using the flexible high and low temperature interrupt alarm or auto-tuning function. The corresponding bits of the temperature sensors in the sampling register are as follows (base address is 0x1FE00000, offset address is 0x0198):

Table 82. Description of temperature collection register

Bit Field	Name	Read/Write	Reset Value	Description
24	Thsens0_overflow	R		Temperature sensor 0 overflow
25	Thsens1_overflow	R		Temperature sensor 1 overflow
47:32	Thsens0_out	R		Temperature sensor 0 centigrade temperature Node temperature=Thens0_out *731/0x4000-273 Temperature range: -40 degree - 125 degree
65:48	Thsens1_out	R		Temperature sensor 1 centigrade temperature Node temperature=Thens0_out *731/0x4000-273 Temperature range: -40 degree - 125 degree

The control registers can be set to enable over preset temperature interrupt, under preset temperature interrupt and high temperature auto down function.

In addition, the current centigrade temperature can be read directly using the new centigrade temperature register. This register can also be accessed using a read operation with base address 0x1FE00000 or 0x3FF00000, or directly using a configuration register instruction with offset 0x0428. The register is described as follows:

Table 83. Extended I/O interrupt trigger register

Name	Offset Address	Read/Write	Description
Thsens_Temperature	0x0428	R	Temperature sensor centigrade temperature

12.2. High/Low Temperature Interrupt Trigger

For the high and low temperature interrupt alarm function, there are 4 groups of control registers to set the threshold value. Each group of registers contains the following three control bits:

GATE: Set the threshold value for high or low temperature. When the input temperature is higher than the high temperature threshold or lower than the low temperature threshold, an interrupt will be generated.

Note that the Gate value should be set to the 16-bit value corresponding to the `0x198` register, not the centigrade temperature.

`EN`: Interrupt enable control. The setting of this set of registers is valid only after setting `1`.

`SEL`: Input temperature selection. This register is used to configure which sensor's temperature is selected as input. Either `0` or `1` can be used.

The high temperature interrupt control register contains four sets of setting bits to control the triggering of high temperature interrupts; the low temperature interrupt control register contains four sets of setting bits to control the triggering of low temperature interrupts. There is another set of registers for displaying the interrupt status, corresponding to the high-temperature interrupt and low-temperature interrupt, respectively, and any write operation to this register will clear the interrupt status.

These registers are described below, and their base addresses are `0x1fe00000`,It can also be accessed using the configuration register instruction (IOCSR):

Table 84. Description of high/low temperature interrupt register

Register	Address	Read/Write	Description
High temperature interrupt control register `Thsens_int_ctrl_Hi`	`0x1460`	RW	`[7:0]`: `Hi_gate0`: high temperature threshold 0, above which an interrupt will be generated `[8:8]`: `Hi_en0`: high temperature interrupt enable 0 `[11:10]`: `Hi_Sel0`: select temperature sensor input source for high temperature interrupt 0 `[23:16]`: `Hi_gate1`: high temperature threshold 1, above which an interrupt will be generated `[24:24]`: `Hi_en1`: high temperature interrupt enable 1 `[27:26]`: `Hi_Sel1`: select temperature sensor input source for high temperature interrupt 1 `[39:32]`: `Hi_gate2`: high temperature threshold 2, above which an interrupt will be generated `[40:40]`: `Hi_en2`: high temperature interrupt enable 2 `[43:42]`: `Hi_Sel2`: select temperature sensor input source for high temperature interrupt 2 `[55:48]`: `Hi_gate3`: high temperature threshold 3, above which an interrupt will be generated `[56:56]`: `Hi_en3`: high temperature interrupt enable 3 `[59:58]`: `Hi_Sel3`: select temperature sensor input source for high temperature interrupt 3

Register	Address	Read/Write	Description
Low temperature interrupt control register `Thsens_int_ctrl_Lo`	`0x1468`	RW	`[7:0]`: `Lo_gate0`: low temperature threshold 0, below which an interrupt will be generated `[8:8]`: `Lo_en0`: low temperature interrupt enable 0 `[11:10]`: `Lo_Sel0`: select temperature sensor input source for low temperature interrupt 0 `[23:16]`: `Lo_gate1`: low temperature threshold 1, below which an interrupt will be generated `[24:24]`: `Lo_en1`: low temperature interrupt enable 1 `[27:26]`: `Lo_Sel1`: select temperature sensor input source for low temperature interrupt 1 `[39:32]`: `Lo_gate2`: low temperature threshold 2, below which an interrupt will be generated `[40:40]`: `Lo_en2`: low temperature interrupt enable 2 `[43:42]`: `Lo_Sel2`: select temperature sensor input source for low temperature interrupt 2 `[55:48]`: `Lo_gate3`: low temperature threshold 3, below which an interrupt will be generated `[56:56]`: `Lo_en3`: low temperature interrupt enable 3 `[59:58]`: `Lo_Sel3`: select temperature sensor input source for low temperature interrupt 3
Interrupt status register `Thsens_int_status/clr`	`0x1470`	RW	Interrupt status register; write `1` to clear the interrupt `[0]`: high temperature interrupt trigger `[1]`: low temperature interrupt trigger

Register	Address	Read/Write	Description
High order bits of high temperature interrupt control register `Thsens_int_up`	`0x1478`	RW	`[7:0]`: `Hi_gate0`: high 8-bit `[15:8]`: `Hi_gate1`: high 8-bit `[23:16]`: `Hi_gate2`: high 8-bit `[31:24]`: `Hi_gate3`: high 8-bit `[39:32]`: `Lo_gate0`: high 8-bit `[47:40]`: `Lo_gate1`: high 8-bit `[55:48]`: `Lo_gate2`: high 8-bit `[63:56]`: `Lo_gate3`: high 8-bit

12.3. High Temperature Automatic Underclock Configuration

In order to ensure the operation of the chip in a high temperature environment, it can be set to make the high temperature automatic frequency reduction, so that the chip is actively clocked when it exceeds the preset range to achieve the effect of reducing the chip flip rate.

For the high-temperature downconversion function, there are four sets of control registers to set its behavior. Each set of registers contains the following four control bits:

`GATE`: Set the threshold value for high or low temperature. When the input temperature is higher than the high temperature threshold or lower than the low temperature threshold, the frequency dividing operation will be triggered.

`EN`: Enable control. The setting of this group of registers is valid only after setting `1`.

`SEL`: Input temperature selection. This register is used to configure which sensor's temperature is selected as input.

`FREQ`: Frequency division number. When the dividing operation is triggered, the clock is divided using the preset `FREQ`. The dividing mode is controlled by `freqscale_mode_node`.

Its base address is `0x1fe00000`,It can also be accessed using the configuration register instruction (IOCSR).

Table 85. Description of high-temperature underclock control register

Register	Address	Read/Write	Description
High-temperature underclock control register `Thsens_freq_scale`	`0x1480`	RW	The four groups of configurations are prioritized from highest to lowest `[7:0]`: `Scale_gate0`: high temperature threshold 0, beyond which the frequency will be reduced `[8:8]`: `Scale_en0`: high temperature underclock enable 0 `[11:10]`: `Scale_Sel0`: select temperature sensor input source for high temperature underclock 0 `[14:12]`: `Scale_freq0`: frequency division value at underclock `[23:16]`: `Scale_gate1`: high temperature threshold 1, beyond which the frequency will be reduced `[24:24]`: `Scale_en1`: high temperature underclock enable 1 `[27:26]`: `Scale_Sel1`: select temperature sensor input source for high temperature underclock 1 `[30:28]`: `Scale_freq1`: frequency division value at underclock `[39:32]`: `Scale_gate2`: high temperature threshold 2, beyond which the frequency will be reduced `[40:40]`: `Scale_en2`: high temperature underclock enable 2 `[43:42]`: `Scale_Sel2`: select temperature sensor input source for high temperature underclock 2 `[46:44]`: `Scale_freq2`: frequency division value at underclock `[55:48]`: `Scale_gate3`: high temperature threshold 3, beyond which the frequency will be reduced `[56:56]`: `Scale_en3`: high temperature underclock enable 3 `[59:58]`: `Scale_Sel3`: select temperature sensor input source for high temperature underclock 3 `[62:60]`: `Scale_freq3`: frequency division value at underclock

Register	Address	Read/Write	Description
Thsens_freq_scale_up	0x1490	RW	High order bits of temperature sensor control register [7:0]: Scale_Hi_gate0: high 8-bit [15:8]: Scale_Hi_gate1: high 8-bit [23:16]: Scale_Hi_gate2: high 8-bit [31:24]: Scale_Hi_gate3: high 8-bit [39:32]: Scale_Lo_gate0: high 8-bit [47:40]: Scale_Lo_gate1: high 8-bit [55:48]: Scale_Lo_gate2: high 8-bit [63:56]: Scale_Lo_gate3: high 8-bit

12.4. Temperature Status Detection and Control

The pins PROCHOTn and THERMTRIPn are used for temperature status detection and control, which are multiplexed with GPIO14 and GPIO15 respectively. PROCHOTn can be used as both input and output, while THERMTRIPn has only output function. When PROCHOTn is used as an input, the chip is controlled by the external temperature detection circuit, and the external temperature detection circuit can set PROCHOTn to 0 when it needs to lower the chip temperature, and the chip will take down frequency measures after receiving this low level. When PROCHOTn is an output, the chip can output high-temperature interrupts, and select one of the four interrupts set by the high-temperature interrupt control register through the prochotn_o_sel register. Select one of the four interrupts set in the high-temperature interrupt control register as the external high-temperature interrupt.

THERMTRIPn as output is selected by the chip from the 4 interrupts set by the high-temperature interrupt control register through the thermtripn_o_sel register as the outgoing high-temperature interrupt.

Although both THERMTRIPn and PROCHOTn are external high temperature interrupts, THERMTRIPn has a higher degree of urgency than PROCHOTn. When PROCHOTn is set, the external temperature control circuit can also take certain measures, such as increasing the fan speed. In contrast, when THERMTRIPn is set, the external power control circuitry should take direct emergency power-off measures.

The specific control registers are as follows:

Table 86. Description of temperature status detection and control register

Register	Address	Read/Write	Description
Temperature status detection and control register Thsens_hi_ctrl	0x1498	RW	[0:0]: prochotn_oe: PROCHOTn pin output enable control, 0 for output, 1 for input [5:4]: prochotn_o_sel: PROCHOTn high temperature interrupt output selection [10:8]: prochotn_freq_scale: PROCHOTn frequency division value when input is valid [17:16]: thermtripn_o_sel: THERMTRIPn high temperature interrupt output selection

12.5. Control of temperature sensors

The 3A5000 has 4 internal temperature sensors, which can be configured via registers to adjust the temperature/voltage monitoring, monitoring point configuration and monitoring frequency, etc. The output of each temperature sensor can also be directly observed for debugging (base address is 0x1FE00000,It can also be accessed using the configuration register instruction (IOCSR), offset address of temperature sensor configuration register is 0x01580+vtsensor_id<<4, offset address of temperature sensor data register is 0x01588+vtsensor_id<<4).

Note that the voltage monitoring function is currently not available

Table 87. Description of temperature sensor configuration register

Bit Field	Name	Read/Write	Reset Value	Description
0	Thsens_trigger	RW	0	Enable temperature sensor configuration. If set, monitoring mode and monitoring point can be selected by thsens_mode and thsens_cluster; 0 is the default temperature monitoring mode and the monitoring point is configured by temp_cluster
2	Thsens_mode	RW	0	0: temperature mode; 1: voltage mode
3	Thsens_datarate	RW	0	Monitoring frequency: 0 - 10-20Hz 1 - 325-650Hz
6:4	Thsens_cluster	RW	0	Sensor monitoring point configuration: 0 is local monitoring point, 1-7 is remote monitoring point
8	Temp_valid	RW	0	Enable the temperature sensor output and replace the value of Thsens0_out and Thsens0_overflow in CSR[0x198] with the temperature monitoring value of this temperature sensor.

Bit Field	Name	Read/Write	Reset Value	Description
11:9	Temp_cluster	RW	0	Temperature sensor output monitoring point selection. It is disabled when Thsens_trigger is enabled

Table 88. Description of temperature sensor data register

Bit Field	Name	Read/Write	Reset Value	Description
3	Out_mode	R	0	Monitoring mode for sensor configuration 0: temperature mode; 1: voltage mode
6:4	Out_cluster	R	0	Monitoring points for sensor configuration
7	Overflow	R	0	Overflow of sensor monitoring values
29:16	Data	R	0	Sensor readout monitoring values

Calculation of the readout value:

`Node temperature = data*731/0x4000 - 273` (temperature range `-40` degrees - `125` degrees)

`Voltage = data*1.226/0x1000`

Chapter 13. DDR4 SDRAM Controller Configuration

The Loongson 3A5000 processor's internally integrated memory controller is designed to comply with the DDR4 SDRAM industry standard (JESD79-4).

13.1. Introduction to DDR4 SDRAM Controller Functions

The Loongson 3A5000 processor supports both DDP and 3DS packaging modes. The DDP supports up to 8 CSs (implemented by 8 DDR3/DDR4 SDRAM chip select signals, i.e., 4 double-sided memory sticks) and the 3DS supports up to 4 CSs (implemented by 8 DDR4 SDRAM chip select signals, i.e., 32 logical RANKS). A total of 22 bits of address bus (i.e., 18 bits of row address bus, 2 bits of logical Bank bus and 2 bits of logical Bank Group bus, where the row address bus is multiplexed with RASn, CASn, and Wen).

The Loongson 3A5000 processor can adjust the DDR4 controller parameter settings to support different memory chip types when they are specifically selected for use. The maximum supported chip selection (CS_n) is 8, the number of logical RANKS (Chip ID) is 8, the number of row addresses (ROW) is 18, the number of column addresses (COL) is 12, the number of logical body selections (BANK) is 2 (DDR4), and the number of logical body groups (BANK Group) is 2. The multiplexing relationship between CS_n and Chip ID can be matched, please see DDR4 SDRAM Parameter Configuration Format for details.

The physical address of the memory request sent by the CPU can be mapped in many different ways according to different configurations inside the controller.

The memory control circuitry integrated in the Loongson 3A5000 processor only accepts memory read/write requests from the processor or external devices, and is in the Slave State for all memory read/write operations.

The memory controller in the Loongson 3A5000 processor has the following features:

- Fully flowing operation of commands, read and write data on the interface.
- Memory command merging and sequencing to improve overall bandwidth.
- Configuration register read and write ports, which can modify the basic parameters of memory devices.
- Built-in dynamic delay compensation circuit (DCC) for reliable sending and receiving of data.
- ECC function can detect 1-bit and 2-bit errors on the data path and can automatically correct 1-bit errors.
- DDR3/4 SDRAM support and parameter configuration supports x4, x8, and x16 particles.
- Controller to PHY frequency ratio of 1/2.
- Support data transport rate range from 800Mbps to 3200Mbps.

13.2. DDR4 SDRAM Parameter Configuration Format

13.2.1. Parameter List of the Memory Controller

Table 89. Software-visible parameter list of the memory controller

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
PHY								
0x0000							version(RD)	
0x0008			x4_mode	ddr3_mode			capability(RD)	
0x0010							dram_init(RD)	init_start
0x0018								
0x0020							preamble2	rdfifo_valid
0x0028		rdfifo_empty(RD)				Overflow(RD)		
0x0030		dll_value(RD)	dll_init_done(RD)	dll_lock_mode	dll_bypass	dll_adjj_cnt	dll_increment	dll_start_point
0x0038				dll_dbl_fix			dll_close_disable	dll_ck
0x0040				dbl_ctrl_ckca				dll_dbl_ckca
0x0048	pll_ctrl_ckca				pll_lock_ckca(RD)	dll_lock_ckca(RD)	clken_ckca	clksel_ckca
0x0050				dbl_ctrl_ds_0				dll_dbl_ds_0
0x0058	pll_ctrl_ds_0				pll_lock_ds_0(RD)	dll_lock_ds_0(RD)	clken_ds_0	clksel_ds_0
0x0060				dbl_ctrl_ds_1				dll_dbl_ds_1
0x0068	pll_ctrl_ds_1				pll_lock_ds_1(RD)	dll_lock_ds_1(RD)	clken_ds_1	clksel_ds_1
0x0070				dbl_ctrl_ds_2				dll_dbl_ds_2
0x0078	pll_ctrl_ds_2				pll_lock_ds_2(RD)	dll_lock_ds_2(RD)	clken_ds_2	clksel_ds_2
0x0080				dbl_ctrl_ds_3				dll_dbl_ds_3

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x0088	pll_ctrl_ds_3				pll_lock_ds_3(RD)	dll_lock_ds_3(RD)	clken_ds_3	clksel_ds_3
0x0090				dbl_ctrl_ds_4				dll_dbl_ds_4
0x0098	pll_ctrl_ds_4				pll_lock_ds_4(RD)	dll_lock_ds_4(RD)	clken_ds_4	clksel_ds_4
0x00a0				dbl_ctrl_ds_5				dll_dbl_ds_5
0x00a8	pll_ctrl_ds_5				pll_lock_ds_5(RD)	dll_lock_ds_5(RD)	clken_ds_5	clksel_ds_5
0x00b0				dbl_ctrl_ds_6				dll_dbl_ds_6
0x00b8	pll_ctrl_ds_6				pll_lock_ds_6(RD)	dll_lock_ds_6(RD)	clken_ds_6	clksel_ds_6
0x00c0				dbl_ctrl_ds_7				dll_dbl_ds_7
0x00c8	pll_ctrl_ds_7				pll_lock_ds_7(RD)	dll_lock_ds_7(RD)	clken_ds_7	clksel_ds_7
0x00d0				dbl_ctrl_ds_8				dll_dbl_ds_8
0x00d8	pll_ctrl_ds_8				pll_lock_ds_8(RD)	dll_lock_ds_8(RD)	clken_ds_8	clksel_ds_8
0x00e0			vrefclk_inv	vref_sample		vref_num	vref_dly	dll_vref
……								
0x0100					dll_1xdly_0	dll_1xgen_0	dll_wrdqs_0	dll_wrdq_0
0x0108						dll_gate_0	dll_rddqs1_0	dll_rddqs0_0
0x0110	rdodt_ctrl_0	rdgate_len_0	rdgate_mode_0	rdgate_ctrl_0			dqs_oe_ctrl_0	dq_oe_ctrl_0
0x0118						dly_2x_0	redge_sel_0	rddqs_phase_0(RD)

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x0120	w_bdly0_0[31:28]	w_bdly0_0[27:24]	w_bdly0_0[23:20]	w_bdly0_0[19:16]	w_bdly0_0[15:12]	w_bdly0_0[11:8]	w_bdly0_0[7:4]	w_bdly0_0[3:0]
0x0128		w_bdly0_0[59:56]	w_bdly0_0[55:52]	w_bdly0_0[51:48]	w_bdly0_0[47:44]	w_bdly0_0[43:40]	w_bdly0_0[39:36]	w_bdly0_0[35:32]
0x0130	w_bdly1_0[24:21]	w_bdly1_0[20:18]	w_bdly1_0[17:15]	w_bdly1_0[14:12]	w_bdly1_0[11:9]	w_bdly1_0[8:6]	w_bdly1_0[5:3]	w_bdly1_0[2:0]
0x0138								w_bdly1_0[27:26]
0x0140							rg_bdly_0[7:4]	rg_bdly_0[3:0]
0x0148								
0x0150	rdqsp_bdly_0[31:28]	rdqsp_bdly_0[27:24]	rdqsp_bdly_0[23:20]	rdqsp_bdly_0[19:16]	rdqsp_bdly_0[15:12]	rdqsp_bdly_0[11:8]	rdqsp_bdly_0[7:4]	rdqsp_bdly_0[3:0]
0x0158								rdqsp_bdly_0[35:32]
0x0160	rdqsn_bdly_0[31:28]	rdqsn_bdly_0[27:24]	rdqsn_bdly_0[23:20]	rdqsn_bdly_0[19:16]	rdqsn_bdly_0[15:12]	rdqsn_bdly_0[11:8]	rdqsn_bdly_0[7:4]	rdqsn_bdly_0[3:0]
0x0168								rdqsn_bdly_0[35:32]
0x0170	rdq_bdly_0[24:21]	rdq_bdly_0[20:18]	rdq_bdly_0[17:15]	rdq_bdly_0[14:12]	rdq_bdly_0[11:9]	rdq_bdly_0[8:6]	rdq_bdly_0[5:3]	rdq_bdly_0[2:0]
0x0178								rdq_bdly_0[27:26]
0x0180					dll_1xdly_1	dll_1xgen_1	dll_wrdqs_1	dll_wrdq_1
0x0188						dll_gate_1	dll_rddqs1_1	dll_rddqs0_1
0x0190	rdodt_ctrl_1	rdgate_len_1	rdgate_mode_1	rdgate_ctrl_1			dqs_oe_ctrl_1	dq_oe_ctrl_1
0x0198						dly_2x_1	redge_sel_1	rddqs_phase_1(RD)
0x01a0	w_bdly0_1[31:28]	w_bdly0_1[27:24]	w_bdly0_1[23:20]	w_bdly0_1[19:16]	w_bdly0_1[15:12]	w_bdly0_1[11:8]	w_bdly0_1[7:4]	w_bdly0_1[3:0]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x01a8		w_bdly0_1[59:56]	w_bdly0_1[55:52]	w_bdly0_1[51:48]	w_bdly0_1[47:44]	w_bdly0_1[43:40]	w_bdly0_1[39:36]	w_bdly0_1[35:32]
0x01b0	w_bdly1_1[24:21]	w_bdly1_1[20:18]	w_bdly1_1[17:15]	w_bdly1_1[14:12]	w_bdly1_1[11:9]	w_bdly1_1[8:6]	w_bdly1_1[5:3]	w_bdly1_1[2:0]
0x01b8								w_bdly1_1[27:26]
0x01c0							rg_bdly_1[7:4]	rg_bdly_1[3:0]
0x01c8								
0x01d0	rdqsp_bdly_1[31:28]	rdqsp_bdly_1[27:24]	rdqsp_bdly_1[23:20]	rdqsp_bdly_1[19:16]	rdqsp_bdly_1[15:12]	rdqsp_bdly_1[11:8]	rdqsp_bdly_1[7:4]	rdqsp_bdly_1[3:0]
0x01d8								rdqsp_bdly_1[35:32]
0x01e0	rdqsn_bdly_1[31:28]	rdqsn_bdly_1[27:24]	rdqsn_bdly_1[23:20]	rdqsn_bdly_1[19:16]	rdqsn_bdly_1[15:12]	rdqsn_bdly_1[11:8]	rdqsn_bdly_1[7:4]	rdqsn_bdly_1[3:0]
0x01e8								rdqsn_bdly_1[35:32]
0x01f0	rdq_bdly_1[24:21]	rdq_bdly_1[20:18]	rdq_bdly_1[17:15]	rdq_bdly_1[14:12]	rdq_bdly_1[11:9]	rdq_bdly_1[8:6]	rdq_bdly_1[5:3]	rdq_bdly_1[2:0]
0x01f8								rdq_bdly_1[27:26]
0x0200					dll_1xdly_2	dll_1xgen_2	dll_wrdqs_2	dll_wrdq_2
0x0208						dll_gate_2	dll_rddqs1_2	dll_rddqs0_2
0x0210	rdodt_ctrl_2	rdgate_len_2	rdgate_mode_2	rdgate_ctrl_2			dqs_oe_ctrl_2	dq_oe_ctrl_2
0x0218						dly_2x_2	redge_sel_2	rddqs_phase_2(RD)
0x0220	w_bdly0_2[31:28]	w_bdly0_2[27:24]	w_bdly0_2[23:20]	w_bdly0_2[19:16]	w_bdly0_2[15:12]	w_bdly0_2[11:8]	w_bdly0_2[7:4]	w_bdly0_2[3:0]
0x0228		w_bdly0_2[59:56]	w_bdly0_2[55:52]	w_bdly0_2[51:48]	w_bdly0_2[47:44]	w_bdly0_2[43:40]	w_bdly0_2[39:36]	w_bdly0_2[35:32]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x0230	w_bdly1_2[24:21]	w_bdly1_2[20:18]	w_bdly1_2[17:15]	w_bdly1_2[14:12]	w_bdly1_2[11:9]	w_bdly1_2[8:6]	w_bdly1_2[5:3]	w_bdly1_2[2:0]
0x0238								w_bdly1_2[27:26]
0x0240							rg_bdly_2[7:4]	rg_bdly_2[3:0]
0x0248								
0x0250	rdqsp_bdly_2[31:28]	rdqsp_bdly_2[27:24]	rdqsp_bdly_2[23:20]	rdqsp_bdly_2[19:16]	rdqsp_bdly_2[15:12]	rdqsp_bdly_2[11:8]	rdqsp_bdly_2[7:4]	rdqsp_bdly_2[3:0]
0x0258								rdqsp_bdly_2[35:32]
0x0260	rdqsn_bdly_2[31:28]	rdqsn_bdly_2[27:24]	rdqsn_bdly_2[23:20]	rdqsn_bdly_2[19:16]	rdqsn_bdly_2[15:12]	rdqsn_bdly_2[11:8]	rdqsn_bdly_2[7:4]	rdqsn_bdly_2[3:0]
0x0268								rdqsn_bdly_2[35:32]
0x0270	rdq_bdly_2[24:21]	rdq_bdly_2[20:18]	rdq_bdly_2[17:15]	rdq_bdly_2[14:12]	rdq_bdly_2[11:9]	rdq_bdly_2[8:6]	rdq_bdly_2[5:3]	rdq_bdly_2[2:0]
0x0278								rdq_bdly_2[27:26]
0x0280					dll_1xdly_3	dll_1xgen_3	dll_wrdqs_3	dll_wrdq_3
0x0288						dll_gate_3	dll_rddqs1_3	dll_rddqs0_3
0x0290	rdodt_ctrl_3	rdgate_len_3	rdgate_mode_3	rdgate_ctrl_3			dqs_oe_ctrl_3	dq_oe_ctrl_3
0x0298						dly_2x_3	redge_sel_3	rddqs_phase_3(RD)
0x02a0	w_bdly0_3[31:28]	w_bdly0_3[27:24]	w_bdly0_3[23:20]	w_bdly0_3[19:16]	w_bdly0_3[15:12]	w_bdly0_3[11:8]	w_bdly0_3[7:4]	w_bdly0_3[3:0]
0x02a8		w_bdly0_3[59:56]	w_bdly0_3[55:52]	w_bdly0_3[51:48]	w_bdly0_3[47:44]	w_bdly0_3[43:40]	w_bdly0_3[39:36]	w_bdly0_3[35:32]
0x02b0	w_bdly1_3[24:21]	w_bdly1_3[20:18]	w_bdly1_3[17:15]	w_bdly1_3[14:12]	w_bdly1_3[11:9]	w_bdly1_3[8:6]	w_bdly1_3[5:3]	w_bdly1_3[2:0]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x02b8								w_bdly1_3[27:26]
0x02c0							rg_bdly_3[7:4]	rg_bdly_3[3:0]
0x02c8								
0x02d0	rdqsp_bdly_3[31:28]	rdqsp_bdly_3[27:24]	rdqsp_bdly_3[23:20]	rdqsp_bdly_3[19:16]	rdqsp_bdly_3[15:12]	rdqsp_bdly_3[11:8]	rdqsp_bdly_3[7:4]	rdqsp_bdly_3[3:0]
0x02d8								rdqsp_bdly_3[35:32]
0x02e0	rdqsn_bdly_3[31:28]	rdqsn_bdly_3[27:24]	rdqsn_bdly_3[23:20]	rdqsn_bdly_3[19:16]	rdqsn_bdly_3[15:12]	rdqsn_bdly_3[11:8]	rdqsn_bdly_3[7:4]	rdqsn_bdly_3[3:0]
0x02e8								rdqsn_bdly_3[35:32]
0x02f0	rdq_bdly_3[24:21]	rdq_bdly_3[20:18]	rdq_bdly_3[17:15]	rdq_bdly_3[14:12]	rdq_bdly_3[11:9]	rdq_bdly_3[8:6]	rdq_bdly_3[5:3]	rdq_bdly_3[2:0]
0x02f8								rdq_bdly_3[27:26]
0x0300					dll_1xdly_4	dll_1xgen_4	dll_wrdqs_4	dll_wrdq_4
0x0308						dll_gate_4	dll_rddqs1_4	dll_rddqs0_4
0x0310	rdodt_ctrl_4	rdgate_len_4	rdgate_mode_4	rdgate_ctrl_4			dqs_oe_ctrl_4	dq_oe_ctrl_4
0x0318						dly_2x_4	redge_sel_4	rddqs_phase_4(RD)
0x0320	w_bdly0_4[31:28]	w_bdly0_4[27:24]	w_bdly0_4[23:20]	w_bdly0_4[19:16]	w_bdly0_4[15:12]	w_bdly0_4[11:8]	w_bdly0_4[7:4]	w_bdly0_4[3:0]
0x0328		w_bdly0_4[59:56]	w_bdly0_4[55:52]	w_bdly0_4[51:48]	w_bdly0_4[47:44]	w_bdly0_4[43:40]	w_bdly0_4[39:36]	w_bdly0_4[35:32]
0x0330	w_bdly1_4[24:21]	w_bdly1_4[20:18]	w_bdly1_4[17:15]	w_bdly1_4[14:12]	w_bdly1_4[11:9]	w_bdly1_4[8:6]	w_bdly1_4[5:3]	w_bdly1_4[2:0]
0x0338								w_bdly1_4[27:26]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x0340							rg_bdly_4[7:4]	rg_bdly_4[3:0]
0x0348								
0x0350	rdqsp_bdly_4[31:28]	rdqsp_bdly_4[27:24]	rdqsp_bdly_4[23:20]	rdqsp_bdly_4[19:16]	rdqsp_bdly_4[15:12]	rdqsp_bdly_4[11:8]	rdqsp_bdly_4[7:4]	rdqsp_bdly_4[3:0]
0x0358								rdqsp_bdly_4[35:32]
0x0360	rdqsn_bdly_4[31:28]	rdqsn_bdly_4[27:24]	rdqsn_bdly_4[23:20]	rdqsn_bdly_4[19:16]	rdqsn_bdly_4[15:12]	rdqsn_bdly_4[11:8]	rdqsn_bdly_4[7:4]	rdqsn_bdly_4[3:0]
0x0368								rdqsn_bdly_4[35:32]
0x0370	rdq_bdly_4[24:21]	rdq_bdly_4[20:18]	rdq_bdly_4[17:15]	rdq_bdly_4[14:12]	rdq_bdly_4[11:9]	rdq_bdly_4[8:6]	rdq_bdly_4[5:3]	rdq_bdly_4[2:0]
0x0378								rdq_bdly_4[27:26]
0x0380					dll_1xdly_5	dll_1xgen_5	dll_wrdqs_5	dll_wrdq_5
0x0388						dll_gate_5	dll_rddqs1_5	dll_rddqs0_5
0x0390	rdodt_ctrl_5	rdgate_len_5	rdgate_mode_5	rdgate_ctrl_5			dqs_oe_ctrl_5	dq_oe_ctrl_5
0x0398						dly_2x_5	redge_sel_5	rddqs_phase_5(RD)
0x03a0	w_bdly0_5[31:28]	w_bdly0_5[27:24]	w_bdly0_5[23:20]	w_bdly0_5[19:16]	w_bdly0_5[15:12]	w_bdly0_5[11:8]	w_bdly0_5[7:4]	w_bdly0_5[3:0]
0x03a8		w_bdly0_5[59:56]	w_bdly0_5[55:52]	w_bdly0_5[51:48]	w_bdly0_5[47:44]	w_bdly0_5[43:40]	w_bdly0_5[39:36]	w_bdly0_5[35:32]
0x03b0	w_bdly1_5[24:21]	w_bdly1_5[20:18]	w_bdly1_5[17:15]	w_bdly1_5[14:12]	w_bdly1_5[11:9]	w_bdly1_5[8:6]	w_bdly1_5[5:3]	w_bdly1_5[2:0]
0x03b8								w_bdly1_5[27:26]
0x03c0							rg_bdly_5[7:4]	rg_bdly_5[3:0]
0x03c8								

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x03d0	rdqsp_bdly_5[31:28]	rdqsp_bdly_5[27:24]	rdqsp_bdly_5[23:20]	rdqsp_bdly_5[19:16]	rdqsp_bdly_5[15:12]	rdqsp_bdly_5[11:8]	rdqsp_bdly_5[7:4]	rdqsp_bdly_5[3:0]
0x03d8								rdqsp_bdly_5[35:32]
0x03e0	rdqsn_bdly_5[31:28]	rdqsn_bdly_5[27:24]	rdqsn_bdly_5[23:20]	rdqsn_bdly_5[19:16]	rdqsn_bdly_5[15:12]	rdqsn_bdly_5[11:8]	rdqsn_bdly_5[7:4]	rdqsn_bdly_5[3:0]
0x03e8								rdqsn_bdly_5[35:32]
0x03f0	rdq_bdly_5[24:21]	rdq_bdly_5[20:18]	rdq_bdly_5[17:15]	rdq_bdly_5[14:12]	rdq_bdly_5[11:9]	rdq_bdly_5[8:6]	rdq_bdly_5[5:3]	rdq_bdly_5[2:0]
0x03f8								rdq_bdly_5[27:26]
0x0400					dll_1xdly_6	dll_1xgen_6	dll_wrdqs_6	dll_wrdq_6
0x0408						dll_gate_6	dll_rddqs1_6	dll_rddqs0_6
0x0410	rdodt_ctrl_6	rdgate_len_6	rdgate_mode_6	rdgate_ctrl_6			dqs_oe_ctrl_6	dq_oe_ctrl_6
0x0418						dly_2x_6	redge_sel_6	rddqs_phase_6(RD)
0x0420	w_bdly0_6[31:28]	w_bdly0_6[27:24]	w_bdly0_6[23:20]	w_bdly0_6[19:16]	w_bdly0_6[15:12]	w_bdly0_6[11:8]	w_bdly0_6[7:4]	w_bdly0_6[3:0]
0x0428		w_bdly0_6[59:56]	w_bdly0_6[55:52]	w_bdly0_6[51:48]	w_bdly0_6[47:44]	w_bdly0_6[43:40]	w_bdly0_6[39:36]	w_bdly0_6[35:32]
0x0430	w_bdly1_6[24:21]	w_bdly1_6[20:18]	w_bdly1_6[17:15]	w_bdly1_6[14:12]	w_bdly1_6[11:9]	w_bdly1_6[8:6]	w_bdly1_6[5:3]	w_bdly1_6[2:0]
0x0438								w_bdly1_6[27:26]
0x0440							rg_bdly_6[7:4]	rg_bdly_6[3:0]
0x0448								
0x0450	rdqsp_bdly_6[31:28]	rdqsp_bdly_6[27:24]	rdqsp_bdly_6[23:20]	rdqsp_bdly_6[19:16]	rdqsp_bdly_6[15:12]	rdqsp_bdly_6[11:8]	rdqsp_bdly_6[7:4]	rdqsp_bdly_6[3:0]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x0458								rdqsp_bdly_6[35:32]
0x0460	rdqsn_bdly_6[31:28]	rdqsn_bdly_6[27:24]	rdqsn_bdly_6[23:20]	rdqsn_bdly_6[19:16]	rdqsn_bdly_6[15:12]	rdqsn_bdly_6[11:8]	rdqsn_bdly_6[7:4]	rdqsn_bdly_6[3:0]
0x0468								rdqsn_bdly_6[35:32]
0x0470	rdq_bdly_6[24:21]	rdq_bdly_6[20:18]	rdq_bdly_6[17:15]	rdq_bdly_6[14:12]	rdq_bdly_6[11:9]	rdq_bdly_6[8:6]	rdq_bdly_6[5:3]	rdq_bdly_6[2:0]
0x0478								rdq_bdly_6[27:26]
0x0480					dll_1xdly_7	dll_1xgen_7	dll_wrdqs_7	dll_wrdq_7
0x0488						dll_gate_7	dll_rddqs1_7	dll_rddqs0_7
0x0490	rdodt_ctrl_7	rdgate_len_7	rdgate_mode_7	rdgate_ctrl_7			dqs_oe_ctrl_7	dq_oe_ctrl_7
0x0498						dly_2x_7	redge_sel_7	rddqs_phase_7(RD)
0x04a0	w_bdly0_7[31:28]	w_bdly0_7[27:24]	w_bdly0_7[23:20]	w_bdly0_7[19:16]	w_bdly0_7[15:12]	w_bdly0_7[11:8]	w_bdly0_7[7:4]	w_bdly0_7[3:0]
0x04a8		w_bdly0_7[59:56]	w_bdly0_7[55:52]	w_bdly0_7[51:48]	w_bdly0_7[47:44]	w_bdly0_7[43:40]	w_bdly0_7[39:36]	w_bdly0_7[35:32]
0x04b0	w_bdly1_7[24:21]	w_bdly1_7[20:18]	w_bdly1_7[17:15]	w_bdly1_7[14:12]	w_bdly1_7[11:9]	w_bdly1_7[8:6]	w_bdly1_7[5:3]	w_bdly1_7[2:0]
0x04b8								w_bdly1_7[27:26]
0x04c0							rg_bdly_7[7:4]	rg_bdly_7[3:0]
0x04c8								
0x04d0	rdqsp_bdly_7[31:28]	rdqsp_bdly_7[27:24]	rdqsp_bdly_7[23:20]	rdqsp_bdly_7[19:16]	rdqsp_bdly_7[15:12]	rdqsp_bdly_7[11:8]	rdqsp_bdly_7[7:4]	rdqsp_bdly_7[3:0]
0x04d8								rdqsp_bdly_7[35:32]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x04e0	rdqsn_bdly_7[31:28]	rdqsn_bdly_7[27:24]	rdqsn_bdly_7[23:20]	rdqsn_bdly_7[19:16]	rdqsn_bdly_7[15:12]	rdqsn_bdly_7[11:8]	rdqsn_bdly_7[7:4]	rdqsn_bdly_7[3:0]
0x04e8								rdqsn_bdly_7[35:32]
0x04f0	rdq_bdly_7[24:21]	rdq_bdly_7[20:18]	rdq_bdly_7[17:15]	rdq_bdly_7[14:12]	rdq_bdly_7[11:9]	rdq_bdly_7[8:6]	rdq_bdly_7[5:3]	rdq_bdly_7[2:0]
0x04f8								rdq_bdly_7[27:26]
0x0500					dll_1xdly_8	dll_1xgen_8	dll_wrdqs_8	dll_wrdq_8
0x0508						dll_gate_8	dll_rddqs1_8	dll_rddqs0_8
0x0510	rdodt_ctrl_8	rdgate_len_8	rdgate_mode_8	rdgate_ctrl_8			dqs_oe_ctrl_8	dq_oe_ctrl_8
0x0518						dly_2x_8	redge_sel_8	rddqs_phase_8(RD)
0x0520	w_bdly0_8[31:28]	w_bdly0_8[27:24]	w_bdly0_8[23:20]	w_bdly0_8[19:16]	w_bdly0_8[15:12]	w_bdly0_8[11:8]	w_bdly0_8[7:4]	w_bdly0_8[3:0]
0x0528		w_bdly0_8[59:56]	w_bdly0_8[55:52]	w_bdly0_8[51:48]	w_bdly0_8[47:44]	w_bdly0_8[43:40]	w_bdly0_8[39:36]	w_bdly0_8[35:32]
0x0530	w_bdly1_8[24:21]	w_bdly1_8[20:18]	w_bdly1_8[17:15]	w_bdly1_8[14:12]	w_bdly1_8[11:9]	w_bdly1_8[8:6]	w_bdly1_8[5:3]	w_bdly1_8[2:0]
0x0538								w_bdly1_8[27:26]
0x0540							rg_bdly_8[7:4]	rg_bdly_8[3:0]
0x0548								
0x0550	rdqsp_bdly_8[31:28]	rdqsp_bdly_8[27:24]	rdqsp_bdly_8[23:20]	rdqsp_bdly_8[19:16]	rdqsp_bdly_8[15:12]	rdqsp_bdly_8[11:8]	rdqsp_bdly_8[7:4]	rdqsp_bdly_8[3:0]
0x0558								rdqsp_bdly_8[35:32]
0x0560	rdqsn_bdly_8[31:28]	rdqsn_bdly_8[27:24]	rdqsn_bdly_8[23:20]	rdqsn_bdly_8[19:16]	rdqsn_bdly_8[15:12]	rdqsn_bdly_8[11:8]	rdqsn_bdly_8[7:4]	rdqsn_bdly_8[3:0]

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x0568								rdqsn_bdly_8[35:32]
0x0570	rdq_bdly_8[24:21]	rdq_bdly_8[20:18]	rdq_bdly_8[17:15]	rdq_bdly_8[14:12]	rdq_bdly_8[11:9]	rdq_bdly_8[8:6]	rdq_bdly_8[5:3]	rdq_bdly_8[2:0]
0x0578								rdq_bdly_8[27:26]
……								
0x0700					leveling_cs	tLVL_DELAY	leveling_req(WR)	leveling_mode
0x0708							leveling_done(RD)	leveling_ready(RD)
0x0710	leveling_resp_7	leveling_resp_6	leveling_resp_5	leveling_resp_4	leveling_resp_3	leveling_resp_2	leveling_resp_1	leveling_resp_0
0x0718								leveling_resp_8
0x0720								
……								
0x0800	dfe_ctrl_ds	pad_ctrl_ds				pad_ctrl_ck		
0x0808		pad_reset_po	pad_opln_ca	pad_opdly_ca		pad_ctrl_ca		
0x0810	vref_ctrl_ds_3		vref_ctrl_ds_2		vref_ctrl_ds_1		vref_ctrl_ds_0	
0x0818	vref_ctrl_ds_7		vref_ctrl_ds_6		vref_ctrl_ds_5		vref_ctrl_ds_4	
0x0820							vref_ctrl_ds_8	
0x0828								
0x0830			pad_comp_o(RD)				pad_comp_i	
0x0838								
CTL								
0x1000		tRP	tWLDQSEN	tMOD	tXPR		tCKE	tRESET
0x1008								tODTL
0x1010	tREFretention				tRFC		tREF	
0x1018	tCKESR	tXSRD	tXS		tRFC_dlr			tREF_IDLE
0x1020					tRDPDEN	tCPDED	tXPDLL	tXP

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x1028					tZQperiod	tZQCL	tZQCS	tZQ_CMD
……								
0x1040	tRCD	tRRD_S_slr	tRRD_L_slr	tRRD_dlr				tRAS_min
0x1048				tRTP	tWR_CRC_DM	tWR	tFAW_slr	tFAW
0x1050	tWTR_S_CRC_DM	tWTR_L_CRC_DM	tWTR_S	tWTR		tCCD_dlr	tCCD_S_slr	tCCD_L_slr
0x1058								
0x1060			tPHY_WRLAT	tWL		tRDDATA	tPHY_RDLAT	tRL
0x1068				tCAL				tPL
0x1070			tW2P_sameba	tW2W_sameba	tW2R_sameba	tR2P_sameba	tR2W_sameba	tR2R_sameba
0x1078			tW2P_samebg	tW2W_samebg	tW2R_samebg	tR2P_samebg	tR2W_samebg	tR2R_samebg
0x1080			tW2P_samec	tW2W_samec	tW2R_samec	tR2P_samec	tR2W_samec	tR2R_samec
0x1088								
0x1090			tW2P_samecs	tW2W_samecs	tW2R_samecs	tR2P_samecs	tR2W_samecs	tR2R_samecs
0x1098				tW2W_diffcs	tW2R_diffcs		tR2W_diffcs	tR2R_diffcs
……								
0x1100			cs_ref	cs_resync	cs_zqcl	cs_zq	cs_mrs	cs_enable
0x1108	cke_map				cs_map			
0x1110				cs2cid				cid_map
0x1118								
0x1120	mrs_done(RD)	mrs_req(WR)	pre_all_done(RD)	pre_all_req(WR)	cmd_cmd	status_cmd(RD)	cmd_req(WR)	command_mode
0x1128	cmd_cke	cmd_a			cmd_ba	cmd_bg	cmd_c	cmd_cs
0x1130								cmd_pda
0x1138						cmd_dq0		
0x1140	mr_3_cs_0		mr_2_cs_0		mr_1_cs_0		mr_0_cs_0	
0x1148	mr_3_cs_1		mr_2_cs_1		mr_1_cs_1		mr_0_cs_1	
0x1150	mr_3_cs_2		mr_2_cs_2		mr_1_cs_2		mr_0_cs_2	

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x1158	mr_3_cs_3		mr_2_cs_3		mr_1_cs_3		mr_0_cs_3	
0x1160	mr_3_cs_4		mr_2_cs_4		mr_1_cs_4		mr_0_cs_4	
0x1168	mr_3_cs_5		mr_2_cs_5		mr_1_cs_5		mr_0_cs_5	
0x1170	mr_3_cs_6		mr_2_cs_6		mr_1_cs_6		mr_0_cs_6	
0x1178	mr_3_cs_7		mr_2_cs_7		mr_1_cs_7		mr_0_cs_7	
0x1180	mr_3_cs_0_ddr4		mr_2_cs_0_ddr4		mr_1_cs_0_ddr4		mr_0_cs_0_ddr4	
0x1188			mr_6_cs_0_ddr4		mr_5_cs_0_ddr4		mr_4_cs_0_ddr4	
0x1190	mr_3_cs_1_ddr4		mr_2_cs_1_ddr4		mr_1_cs_1_ddr4		mr_0_cs_1_ddr4	
0x1198			mr_6_cs_1_ddr4		mr_5_cs_1_ddr4		mr_4_cs_1_ddr4	
0x11a0	mr_3_cs_2_ddr4		mr_2_cs_2_ddr4		mr_1_cs_2_ddr4		mr_0_cs_2_ddr4	
0x11a8			mr_6_cs_2_ddr4		mr_5_cs_2_ddr4		mr_4_cs_2_ddr4	
0x11b0	mr_3_cs_3_ddr4		mr_2_cs_3_ddr4		mr_1_cs_3_ddr4		mr_0_cs_3_ddr4	
0x11b8			mr_6_cs_3_ddr4		mr_5_cs_3_ddr4		mr_4_cs_3_ddr4	
0x11c0	mr_3_cs_4_ddr4		mr_2_cs_4_ddr4		mr_1_cs_4_ddr4		mr_0_cs_4_ddr4	
0x11c8			mr_6_cs_4_ddr4		mr_5_cs_4_ddr4		mr_4_cs_4_ddr4	
0x11d0	mr_3_cs_5_ddr4		mr_2_cs_5_ddr4		mr_1_cs_5_ddr4		mr_0_cs_5_ddr4	
0x11d8			mr_6_cs_5_ddr4		mr_5_cs_5_ddr4		mr_4_cs_5_ddr4	
0x11e0	mr_3_cs_6_ddr4		mr_2_cs_6_ddr4		mr_1_cs_6_ddr4		mr_0_cs_6_ddr4	
0x11e8			mr_6_cs_6_ddr4		mr_5_cs_6_ddr4		mr_4_cs_6_ddr4	
0x11f0	mr_3_cs_7_ddr4		mr_2_cs_7_ddr4		mr_1_cs_7_ddr4		mr_0_cs_7_ddr4	
0x11f8			mr_6_cs_7_ddr4		mr_5_cs_7_ddr4		mr_4_cs_7_ddr4	
0x1200			nc16_map	nc	channel_width	ba_xor_row_offset	addr_new	cs_place
0x1208						bg_xor_row_offset		addr_mirror
0x1210	addr_base_1				addr_base_0			
0x1218								
0x1220	addr_mask_1				addr_mask_0			
0x1228								
0x1230			cs_diff	c_diff	bg_diff	ba_diff	row_diff	col_diff
0x1238				CF_confbus_timeout				

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x1240	WRQthreshold	tRDQidle	wr_pkc_num	rwq_rb	retry	no_dead_inorder	placement_en	stb_en/pbuf
0x1248								tRWGNTidle
0x1250							rfifo_age	
0x1258	prior_age3		prior_age2		prior_age1		prior_age0	
0x1260	retry_cnt(RD)					rbuffer_max(RD)	rdfifo_depth	stat_en
0x1268								
……								
0x1280	aw_512_align		rd_before_wr	ecc_enable		int_vector(RD)	int_trigger(RD)	int_enable
0x1288								
0x1290						int_cnt_fatal(RD)	int_cnt_err(RD)	int_cnt
0x1298	ecc_cnt_cs_7(RD)	ecc_cnt_cs_6(RD)	ecc_cnt_cs_5(RD)	ecc_cnt_cs_4(RD)	ecc_cnt_cs_3(RD)	ecc_cnt_cs_2(RD)	ecc_cnt_cs_1(RD)	ecc_cnt_cs_0(RD)
0x12a0	ecc_data_dir(RD)	ecc_code_dir(RD)	ecc_code_256(RD)					ecc_code_64(RD)
0x12a8	ecc_addr(RD)							
0x12b0	ecc_data[63:0](RD)							
0x12b8	ecc_data[127:64] (RD)							
0x12c0	ecc_data[191:128] (RD)							
0x12c8	ecc_data[255:192] (RD)							
……								
0x1300							ref_num	ref_sch_en
0x1308							Status_sref(RD)	srefresh_req
……								
0x1340	hardware_pd_7	hardware_pd_6	hardware_pd_5	hardware_pd_4	hardware_pd_3	hardware_pd_2	hardware_pd_1	hardware_pd_0
0x1348	power_sta_7(RD)	power_sta_6(RD)	power_sta_5(RD)	power_sta_4(RD)	power_sta_3(RD)	power_sta_2(RD)	power_sta_1(RD)	power_sta_0(RD)
0x1350	selfref_age		slowpd_age		fastpd_age		active_age	
0x1358				power_up				Age_step
0x1360	tCONF_IDLE				tLPMC_IDLE			

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
……								
0x1380								zq_overlap
0x1388								zq_stat_en
0x1390	zq_cnt_1(RD)				zq_cnt_0(RD)			
0x1398	zq_cnt_3(RD)				zq_cnt_2(RD)			
0x13a0	zq_cnt_5(RD)				zq_cnt_4(RD)			
0x13a8	zq_cnt_6(RD)				zq_cnt_6(RD)			
……								
0x13c0					odt_wr_cs_map			
0x13c8							odt_wr_length	odt_wr_delay
0x13d0					odt_rd_cs_map			
0x13d8							odt_rd_length	odt_rd_delay
……								
0x1400				tRESYNC_length	tRESYNC_delay	tRESYNC_shift	tRESYNC_max	tRESYNC_min
……								
0x1440					pre_predict		tm_cmdq_num	burst_length
0x1448								ca_timing
0x1450						wr/rd_dbi_en	ca_par_en	crc_en
0x1458							tCA_PAR	tWR_CRC
0x1460	bit_map_7	bit_map_6	bit_map_5	bit_map_6	bit_map_3	bit_map_2	bit_map_1	bit_map_0
0x1468	bit_map_15	bit_map_14	bit_map_13	bit_map_12	bit_map_11	bit_map_10	bit_map_9	bit_map_8
0x1470							bit_map_17	bit_map_16
0x1478								bitmap_mirror
0x1480				alertn_misc(RD)			alertn_cnt	alertn_clr

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x1488	alertn_addr(RD)							
……								
0x1500	win0_base							
0x1508	win1_base							
0x1510	win2_base							
0x1518	win3_base							
0x1520	win4_base							
0x1528	win5_base							
0x1530	win6_base							
0x1538	win7_base							
……								
0x1580	win0_mask							
0x1588	win1_mask							
0x1590	win2_mask							
0x1598	win3_mask							
0x15a0	win4_mask							
0x15a8	win5_mask							
0x15b0	win6_mask							
0x15b8	win7_mask							
……								
0x1600	win0_mmap							
0x1608	win1_mmap							
0x1610	win2_mmap							
0x1618	win3_mmap							
0x1620	win4_mmap							
0x1628	win5_mmap							
0x1630	win6_mmap							
0x1638	win7_mmap							
……								
0x1700							acc_hp	acc_en
0x1708	acc_fake_b				acc_fake_a			
0x1710								
0x1718								
0x1720	addr_base_acc_1				addr_base_acc_0			
0x1728								

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x1730	addr_mask_acc_1				addr_mask_acc_0			
0x1738								
MON								
0x2000								cmd_monitor
0x2008								
0x2010	cmd_fbck[63:0](RD)							
0x2018	cmd_fbck[127:64] (RD)							
0x2020					rw_switch_cnt(RD)			
……								
0x2100								scheduler_mon
0x2108								
0x2110	sch_cmd_num(RD)							
0x2118	ba_conflict_all(RD)							
0x2120	ba_conflict_last1(RD)							
0x2128	ba_conflict_last2(RD)							
0x2130	ba_conflict_last3(RD)							
0x2138	ba_conflict_last4(RD)							
0x2140	ba_conflict_last5(RD)							
0x2148	ba_conflict_last6(RD)							
0x2150	ba_conflict_last7(RD)							
0x2158	ba_conflict_last8(RD)							
0x2160	rd_conflict(RD)							
0x2168	wr_conflict(RD)							
0x2170	rtw_conflict(RD)							
0x2178	wtr_conflict(RD)							
0x2180	rd_conflict_last1(RD)							
0x2188	wr_conflict_last1(RD)							
0x2190	rtw_conflict_last1(RD)							
0x2198	wtr_conflict_last1(RD)							
0x21a0	wr_rd_turnaround(RD)							
0x21a8	cs_turnaround(RD)							
0x21b0	bg_conflict(RD)							
……								

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x2300						sm_leveling		sm_init
0x2308								
0x2310		sm_rank_03		sm_rank_02		sm_rank_01		sm_rank_00
0x2318		sm_rank_07		sm_rank_06		sm_rank_05		sm_rank_04
0x2320		sm_rank_11		sm_rank_10		sm_rank_09		sm_rank_08
0x2328		sm_rank_15		sm_rank_14		sm_rank_13		sm_rank_12
0x2330		sm_rank_19		sm_rank_18		sm_rank_17		sm_rank_16
0x2338		sm_rank_23		sm_rank_22		sm_rank_21		sm_rank_20
0x2340		sm_rank_27		sm_rank_26		sm_rank_25		sm_rank_24
0x2348		sm_rank_31		sm_rank_30		sm_rank_29		sm_rank_28
……								
TST								
0x3000						lpbk_mode	lpbk_start	lpbk_en
0x3008	lpbk_correct(RD)				lpbk_counter(RD)			lpbk_error(RD)
0x3010	lpbk_data_en[63:0]							
0x3018								lpbk_data_en[71:64]
0x3020							lpbk_data_mask_en	
0x3028								
0x3030	Lpbk_dat_w0[63:0]							
0x3038	Lpbk_dat_w0[127:64]							
0x3040	Lpbk_dat_w1[63:0]							
0x3048	Lpbk_dat_w1[127:64]							
0x3050		lpbk_ecc_mask_w0	lpbk_dat_mask_w0				lpbk_ecc_w0	

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x3058		lpbk_ecc_mask_w1	lpbk_dat_mask_w1				lpbk_ecc_w1	
0x3060								prbs_23
0x3068						prbs_init		
……								
0x3100					fix_data_pattern_index	bus_width	page_size	test_engine_en
0x3108			cs_diff_tst	c_diff_tst	bg_diff_tst	ba_diff_tst	row_diff_tst	col_diff_tst
0x3120	addr_base_tst							
0x3128								
0x3130	user_data_pattern							
0x3138								
0x3140	valid_bits[63:0]							
0x3148								valid_bits[71:64]
0x3150	ctrl[63:0]							
0x3158	ctrl[127:64]							
0x3160	obs[63:0] (RD)							
0x3168	obs[127:64] (RD)							
0x3170	obs[191:128] (RD)							
0x3178	obs[255:192] (RD)							
0x3180	obs[319:256] (RD)							
0x3188	obs[383:320] (RD)							
0x3190	obs[447:384] (RD)							
0x3198	obs[511:448] (RD)							
0x31a0	obs[575:512] (RD)							
0x31a8	obs[639:576] (RD)							
0x31b0					obs[671:640](RD)			
……								
0x3200								
0x3208								
0x3220	tud_i0							
0x3228	tud_i1							

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
0x3230	tud_o(RD)							
……								
0x3300	tst_300							
0x3308	tst_308							
0x3310	tst_310							
0x3318	tst_318							
0x3320	tst_320							
0x3328	tst_328							
0x3330	tst_330							
0x3338	tst_338							
0x3340	tst_340							
0x3348	tst_348							
0x3350	tst_350							
0x3358	tst_358							
0x3360	tst_360							
0x3368	tst_368							
0x3370	tst_370							
0x3378	tst_378							

13.3. Software Programming Guide

13.3.1. Initialization Operations

The initialization operation starts when the software writes 0x2 to register Init_start (0x010). Before setting the Init_start signal, all other registers must be set to the correct values. The DRAM initialization process in cooperation with hardware and software is as follows:

1. Set pm_clk_sel_ckca and pm_clk_sel_ds.
2. Set pm_phy_init_start to 1 to start initializing the PHY.

Wait for pm_dll_lock_* or pm_pll_lock_* of all clock generation modules to become 1.

1. Enabling all pm_clken_*.
2. Set pm_init_start to 1 to start initialization of the memory controller.

13.3.2. Control of Reset Pins

Wait for the memory controller initialization to complete, i.e., the value of pm_dram_init is the same as pm_cs_enable.

For simpler control of reset pins in states such as STR, special reset pin (DDR_RESETn) control can be performed via the pad_reset_po (0x808) register, and there are two main control modes:

1. General mode, `reset_ctrl[1:0] == 2'b00`. The behavior of the reset signal pins in this mode is compatible with the general control mode. `DDR_RESETn` is connected directly on the motherboard to the corresponding pin on the memory slot. The behavior of the pin is:
   - When not powered up: pin status is low.
   - At power-up: pin state is low.
   - When the controller starts initialization: pin state is high.
   - During normal operation: pin state is high.

   The timing is shown in the figure below:

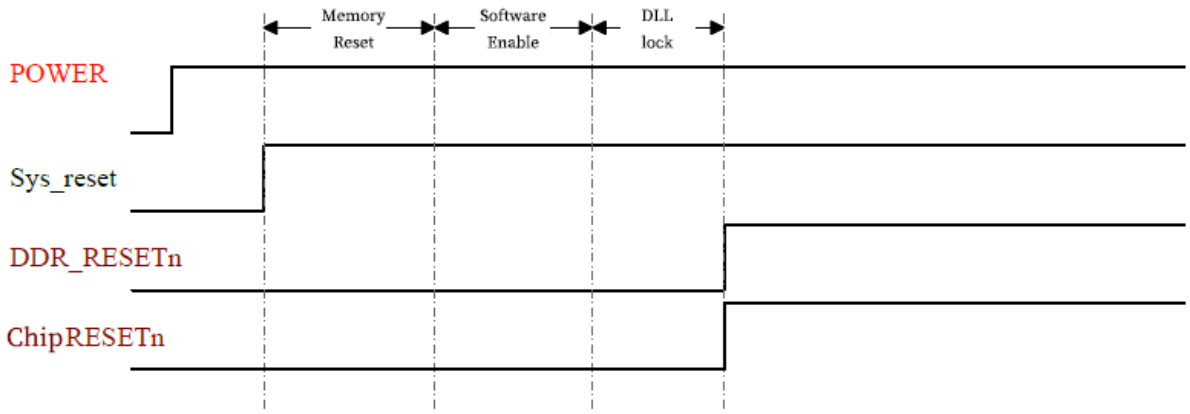


Figure 7. General mode timing

2. Reverse mode, `reset_ctrl[1:0] == 2'b10`. In this mode, the reset signal pins are at the opposite effective level from the normal control mode when doing the actual memory control. So the motherboard needs to connect DDR_RESETn to the corresponding pin on the memory slot through the inverter. The pin behavior is:
   - When not powered up: the pin state is low.
   - At power-up: pin state is low.
   - When the controller starts configuration: pin state is high.
   - When the controller starts initialization: pin state is low.
   - During normal operation: pin state is low.

   The timing sequence is shown in the following figure:

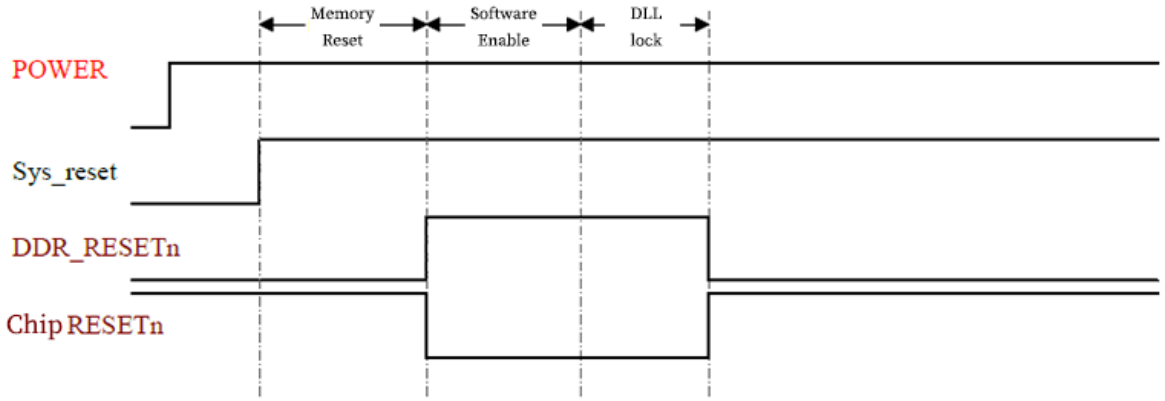


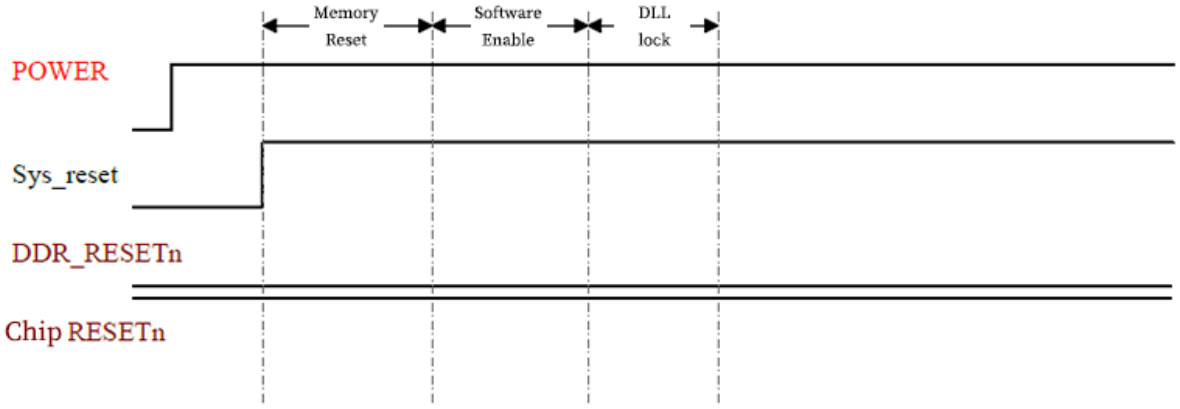
Figure 8. Reverse mode timing

3. Reset disable mode, `pm_pad_reset_o[1:0] == 2'b01`. In this mode, the reset signal pin remains low during the whole memory operation. So the motherboard needs to connect DDR_RESETn to the corresponding pin on the memory slot through the inverter. The behavior of the pins is:

- always low.

The timing is shown in the following diagram:

Reset disable mode timing



By the latter two reset modes in conjunction, STR control can be achieved directly with the use of the memory controller's reset signal. When the whole system is booted from the shutdown state, use the method 2 to use the memory stick to reset normally and start working. When the system resumes from STR, the method 3 is used to reconfigure the memory stick so that it starts working properly again without destroying the original state of the memory stick.

13.3.3. Leveling

The Leveling operation is an operation used in DDR3/4 to intelligently configure the phase relationship between various signals in the memory controller read and write operations. It usually includes Write Leveling, Read Leveling and GateLeveling. In this controller, only Write Leveling and Gate Leveling are implemented, and Read Leveling is not implemented. The software needs to judge the correctness of reading and writing to achieve the functions accomplished by Read Leveling. In addition to the DQS phase and GATE phase operated in the Leveling process, the configuration methods of Write DQ phase and Read DQ phase can be calculated based on these last confirmed phases. In addition, the design supports a bit-deskew function to compensate the delay difference between different bits within a dataslice.

13.3.3.1. Write Leveling

Write Leveling is used to configure the phase relationship between the write DQS and the clock, and the software programming needs to refer to the following steps.

1. Complete the controller initialization, see the previous subsection.
2. Set `Dll_wrdqs_x` (`x = 0…8`) to `0x20`.
3. Set `Dll_wrdq_x` (`x = 0..8`) to `0x0`.
4. Set `Lvl_mode` to `2'b01`.
5. Sample the `Lvl_ready` register, if it is `1`, it means that the Write Leveling request can start.
6. Set `Lvl_req` to `1`.
7. Sample the `Lvl_done` register, if it is `1`, a Write Leveling request is complete.
8. Sample the `Lvl_resp_x` register, if it is `0`, increments the corresponding Dll_wrdq_x[6:0] and `dll_1xdly[6:0]` by `1`, and repeats 5-7 until `Lvl_resp_x` is `1`, then moves to 9; if it is `1`, increments the corresponding `Dll_wrdq_x[6:0]` and `dll_1xdly[6:0]` by `1` and repeat 5-7 until `Lvl_resp_x` is `0`, then continue to increase the corresponding `Dll_wrdq_x[6:0]` and `dll_1xdly[6:0]` by `1` and repeat 5-7 until `Lvl_resp_x` is `1`, then turn to 9.

9. Subtract 0x40 from the values of Dll_wrdq_x and dll_1xdly, at which point the values of Dll_wrdq_x and dll_1xdly should be the correct values to set.
10. Set pm_dly_2x according to the DIMM type, for the particles to the right of the 0x0 bound the corresponding pm_dly_2x value is increased by 0x010101.
11. Set Lvl_mode (0x700) to 2'b00 to exit Write Leveling mode.

13.3.3.2. Gate Leveling

Gate Leveling is used to configure the timing of the enable sample read DQS window within the controller, refer to the following steps for software programming.

1. Complete controller initialization, see previous subsection.
2. Completing Write Leveling, see previous section.
3. Set Dll_gate_x (x = 0…8) to 0.
4. Set Lvl_mode to 2'b10.
5. Sample the Lvl_ready register; if it is 1, the Gate Leveling request can start.
6. Set Lvl_req to 1.
7. Sample the Lvl_done register, if it is 1, a Gate Leveling request is complete.
8. Sample the Lvl_resp_x[0] register, if the first sample finds Lvl_resp_x[0] to be 1, increase the corresponding Dll_gate_x[6:0] by 1 and repeat 6-8 until the sample result is 0; otherwise proceed to the next step.
9. If the sampling result is 0, increment the corresponding Dll_gate_x[6:0] by 1 and repeat 6-9; if it is 1, the Gate Leveling operation has succeeded.
10. Set pm_rdedge_sel (11) according to the value of pm_rddqs_phase to set Dll_gate_x (x = 0…8) minus 0x20.
11. If the value of Lvl_resp_x[7:5] and Lvl_resp_x[4:2] changes, if each increases to Burst_length/2, proceed to step 13; if it is not 4, it may be necessary to add or subtract Rd_oe_begin_x a plus or minus operation may be required for Rd_oe_begin_x, and if greater than Burst_length/2, some fine-tuning of the value of Dll_gate_x is likely to be required.
12. Set Lvl_mode (0x700) to 2'b00 to exit Gate Leveling mode.
13. This ends the Gate Leveling operation.

13.3.4. Power Control Configuration Flow

First set pm_pad_ctrl_ca[0] to 1, then set pm_pad_ctrl_ca[0] to 0 after memory initialization is complete. This function is only available when CAL Mode is enabled in DDR4 mode.

13.3.5. Initiate a Separate MRS Command

In DDR3 mode, the sequence of MRS commands issued by the memory controller to the memory are:

MR2_CS0, MR2_CS1, MR2_CS2, MR2_CS3, MR2_CS4, MR2_CS5, MR2_CS6, MR2_CS7, MR3_CS0, MR3_CS1, MR3_CS2, MR3_CS3, MR3_CS4, MR3_CS5, MR3_CS6, MR3_CS7, MR1_CS0, MR1_CS1, MR1_CS2, MR1_CS3, MR1_CS4, MR1_CS5, MR1_CS6, MR1_CS7, MR0_CS0, MR1_CS1, MR1_CS2, MR1_CS3, MR0_CS4, MR0_CS5, MR0_CS6, MR0_CS7.

In addition, for DDR4 mode, the sequence of MRS commands issued by the memory controller to the memory are:

MR3_CS0, MR3_CS1, MR3_CS2, MR3_CS3, MR3_CS4, MR3_CS5, MR3_CS6, MR3_CS7, MR6_CS0, MR6_CS1, MR6_CS2, MR6_CS3, MR6_CS4, MR6_CS5, MR6_CS6, MR6_CS7, MR5_CS0, MR5_CS1, MR5_CS2, MR5_CS3, MR5_CS4, MR5_CS5, MR5_CS6, MR5_CS7, MR4_CS0, MR1_CS1, MR1_CS2, MR1_CS3, MR4_CS4, MR4_CS5, MR4_CS6, MR4_CS7, MR2_CS0, MR2_CS1, MR2_CS2, MR2_CS3, MR2_CS4, MR2_CS5, MR2_CS6, MR2_CS7, MR1_CS0, MR1_CS1, MR1_CS2, MR1_CS3, MR1_CS4, MR1_CS5, MR1_CS6, MR1_CS7, MR0_CS0, MR1_CS1, MR1_CS2, MR1_CS3, MR0_CS4, MR0_CS5, MR0_CS6, MR0_CS7.

Among them, whether the MRS command corresponding to CS is valid or not is determined by Cs_mrs. Only when the bit corresponding to each chip select on Cs_mrs is valid, this MRS command is actually sent to DRAM. The value of each corresponding MR is determined by the register Mr*_cs*. These values are also used for the MRS command when initializing the memory.

The operation is as follows:

1. Set the registers Cs_mrs (0x1101), Mr*_cs* (0x1140-0x11f8) to the correct values.
2. Set Command_mode (0x1120) to 1 to put the controller into command sending mode.
3. Sample Status_cmd (0x1122), if it is 1, the controller is in command sending mode and can proceed to the next operation, if it is 0, it needs to continue to wait.

Write Mrs_req (0x1126) to 1 to send MRS command to DRAM.

1. sample Mrs_done (0x1127), if it is 1, it means the MRS command has been sent and can exit, if it is 0, it needs to continue to wait.
2. Set Command_mode (0x1120) to 0 to make the controller exit the command sending mode.

13.3.6. Arbitrary Operation Control Bus

The memory controller can send any combination of commands to DRAM via command send mode. The software can set Cmd_cs, Cmd_cmd, Cmd_ba, Cmd_a (0x1128) to be sent to DRAM in command send mode.

The specific operation is as follows:

1. Set registers Cmd_cs, Cmd_cmd, Cmd_ba, Cmd_a (0x1128) to the correct values.
2. Set Command_mode (0x1120) to 1 to put the controller into command sending mode;
3. Sample Status_cmd (0x1122), if it is 1, the controller is in command sending mode and can proceed to the next operation, if it is 0, it needs to continue to wait;

Write Cmd_req (0x1121) to 1 to send a command to DRAM.

1. Set Command_mode (0x1120) to 0 to make the controller exit the command sending mode.

13.3.7. Control of Self-cycling Test Mode

The self-cycling test mode can be used in test mode or normal function mode, respectively. For this purpose, this memory controller implements two separate sets of control interfaces, one for direct control by the test port in test mode and the other for configuration enable test by the register configuration module in normal function mode.

The multiplexing of these two sets of interfaces is controlled using port test_phy. When test_phy is

valid, the test_* port of the controller is used for control, and the self-test is fully controlled by hardware at this time; when test_phy is invalid, the parameters of pm_* programmed by software are used for control. The specific signal meaning of using the test port can be found in the same name section of the register parameters.

These two sets of interfaces are basically the same in terms of control parameters, only the access point is different, here to introduce the control method of software programming. The specific operation is as follows:

1. Set all the parameters of the memory controller correctly.

Wait for the clock reset to stabilize according to the initialization process.

1. Set register Lpbk_en to 1.
2. Set the register Lpbk_start to 1; then the self-loop test starts.
3. At this point the self-loop test has begun and the software needs to be checked frequently for errors, as follows:
4. Sample register Lpbk_error, if it is 1, it means that an error occurred, at this time the first error data and correct data through Lpbk_* and other observation registers can be observed; if this value is 0, it means that there has not been a data error.

13.3.8. Control of the Use of ECC Function

The ECC function is only available in 64-bit mode. ecc_enable (0x1280) consists of the following 2 control bits:

Ecc_enable[0] controls whether to enable the ECC function, which is enabled only if this valid bit is set. Ecc_enable[1] controls whether to report errors through the processor's internal read response path, so that a read access with an ECC two-bit error can immediately cause an exception to occur in the processor core.

In addition, ECC errors can be notified to the processor core via an interrupt. This interrupt is controlled through Int_enable. The interrupt consists of two vectors, Int_vector[0] for an ECC error (both 1-bit and 2-bit errors) and Int_vecotr[1] for an ECC two-bit error. Clearance of Int_vector is achieved by writing 1 to the corresponding bit.

13.3.9. Observation of Error States

After a memory controller error, the corresponding system configuration registers can be accessed to get the corresponding error information and perform simple debugging operations. The register base address is 0x1fe00000 or 0x3ff00000, which can also be accessed using the configuration register instruction, and the registers and their corresponding bits are as follows.

Table 90. Memory controller 0 error status observation register

Register	Offset Address	Read/Write	Description
Memory controller 0 ECC set register `Mc0_ecc_set`	`0x0600`	RW	Memory controller 0 ECC set register `[5:0]`: MC0 `int_enable`, interrupt enable `[8]`: MC0 `int_trigger`, interrupt trigger configuration `[21:16]`: MC0 `int_vector` (RO), interrupt vector (read only) `[33:32]`: MC0 `ecc_enable`, ECC-related function enable `[40]`: MC0 `rd_before_wr`, write after read function enable
	`0x0608`	RW	Reserved
Memory controller 0 ECC counter register `Mc0_ecc_cnt`	`0x0610`	RW	Memory controller 0 ECC counter register `[7:0]`: MC0 `int_cnt`, configure the ECC checking trigger interrupt count threshold `[15:8]`: MC0 `int_cnt_err` (RO), ECC check one bit error count (read only) `[23:16]`: MC0 `int_cnt_fatal` (RO), ECC checking error count (read only)
Memory controller 0 ECC error count register `Mc0_ecc_cs_cnt`	`0x0618`	RO	Memory controller 0 ECC error count register `[7:0]`: MC0 `ecc_cnt_cs_0`, statistics on the number of CS0 ECC checking errors `[15:8]`: MC0 `ecc_cnt_cs_1`, statistics on the number of CS1 ECC checking errors `[23:16]`: MC0 `ecc_cnt_cs_2`, statistics on the number of CS2 ECC checking errors `[31:24]`: MC0 `ecc_cnt_cs_3`, statistics on the number of CS3 ECC checking errors `[39:32]`: MC0 `ecc_cnt_cs_4`, statistics on the number of CS4 ECC checking errors `[47:40]`: MC0 `ecc_cnt_cs_5`, statistics on the number of CS5 ECC checking errors `[55:48]`: MC0 `ecc_cnt_cs_6`, statistics on the number of CS6 ECC checking errors `[63:56]`: MC0 `ecc_cnt_cs_7`, statistics on the number of CS7 ECC checking errors

Register	Offset Address	Read/Write	Description
Memory controller 0 ECC checking register Mc0_ecc_code	0x0620	RO	Memory controller 0 ECC checking register [7:0]: MC0 ecc_code_64, 64-bit ECC checking, invalid when memory directory function is enabled [41:32]: MC0 ecc_code_256, 256-bit ECC checking, valid when memory directory function is enabled [52:48]: MC0 ecc_code_dir, memory directory ECC checking, valid only when the memory directory function is enabled [60:56]: MC0 ecc_data_dir, memory directory ECC data, valid only when the memory directory function is enabled
Memory controller 0 ECC error address register Mc0_ecc_addr	0x0628	RO	Memory controller 0 ECC error address register [63:0]: MC0 ecc_addr, ECC checking error address information
Memory controller 0 ECC error data register 0 Mc0_ecc_data0	0x0630	RO	Memory controller 0 ECC error data register 0 [63:0]: Mc0_ecc_data0, data information in case of an ECC checking error. 64-bit ECC mode data, or 256-bit ECC mode data [63:0]
Memory controller 0 ECC error data register 1 Mc0_ecc_data1	0x0638	RO	Memory controller 0 ECC error data register 1 [63:0]: Mc0_ecc_data1, data information in case of an ECC checking error. 256-bit ECC mode data [127:64]
Memory controller 0 ECC error data register 2 Mc0_ecc_data2	0x0640	RO	Memory controller 0 ECC error data register 2 [63:0]: Mc0_ecc_data1, data information in case of an ECC checking error. 256-bit ECC mode data [191:128]
Memory controller 0 ECC error data register 3 Mc0_ecc_data3	0x0648	RO	Memory controller 0 ECC error data register 3 [63:0]: Mc0_ecc_data3, data information in case of an ECC checking error. 256-bit ECC mode data [255:192]

Table 91. Memory controller 1 error status observation register

Register	Offset Address	Read/Write	Description
Memory controller 1 ECC set register Mc1_ecc_set	0x0700	RW	Memory controller 1 ECC set register [5:0]: MC1 int_enable, interrupt enable [8]: MC1 int_trigger, interrupt trigger configuration [21:16]: MC1 int_vector (RO), interrupt vector (read only) [33:32]: MC1 ecc_enable, ECC-related function enable [40]: MC1 rd_before_wr, write after read function enable
	0x0708	RW	Reserved
Memory controller 1 ECC counter register Mc1_ecc_cnt	0x0710	RW	Memory controller 1 ECC counter register [7:0]: MC1 int_cnt, configure the ECC checking trigger interrupt count threshold [15:8]: MC1 int_cnt_err (RO), ECC check one bit error count (read only) [23:16]: MC1 int_cnt_fatal (RO), ECC checking error count (read only)
Memory controller 1 ECC error count register Mc1_ecc_cs_cnt	0x0718	RO	Memory controller 1 ECC error count register [7:0]: MC1 ecc_cnt_cs_0, statistics on the number of CS0 ECC checking errors [15:8]: MC1 ecc_cnt_cs_1, statistics on the number of CS1 ECC checking errors [23:16]: MC1 ecc_cnt_cs_2, statistics on the number of CS2 ECC checking errors [31:24]: MC1 ecc_cnt_cs_3, statistics on the number of CS3 ECC checking errors [39:32]: MC1 ecc_cnt_cs_4, statistics on the number of CS4 ECC checking errors [47:40]: MC1 ecc_cnt_cs_5, statistics on the number of CS5 ECC checking errors [55:48]: MC1 ecc_cnt_cs_6, statistics on the number of CS6 ECC checking errors [63:56]: MC1 ecc_cnt_cs_7, statistics on the number of CS7 ECC checking errors

Register	Offset Address	Read/Write	Description
Memory controller 1 ECC checking register Mc1_ecc_code	0x0720	RO	Memory controller 1 ECC checking register [7:0]: MC1 ecc_code_64, 64-bit ECC checking, invalid when memory directory function is enabled [41:32]: MC1 ecc_code_256, 256-bit ECC checking, valid when memory directory function is enabled [52:48]: MC1 ecc_code_dir, memory directory ECC checking, valid only when the memory directory function is enabled [60:56]: MC1 ecc_data_dir, memory directory ECC data, valid only when the memory directory function is enabled
Memory controller 1 ECC error address register Mc1_ecc_addr	0x0728	RO	Memory controller 1 ECC error address register [63:0]: MC1 ecc_addr, ECC checking error address information
Memory controller 1 ECC error data register 0 Mc1_ecc_data0	0x0730	RO	Memory controller 1 ECC error data register 0 [63:0]: Mc1_ecc_data0, data information in case of an ECC checking error. 64-bit ECC mode data, or 256-bit ECC mode data [63:0]
Memory controller 1 ECC error data register 1 Mc1_ecc_data1	0x0738	RO	Memory controller 1 ECC error data register 1 [63:0]: Mc1_ecc_data1, data information in case of an ECC checking error. 256-bit ECC mode data [127:64]
Memory controller 1 ECC error data register 2 Mc1_ecc_data2	0x0740	RO	Memory controller 1 ECC error data register 2 [63:0]: Mc1_ecc_data1, data information in case of an ECC checking error. 256-bit ECC mode data [191:128]
Memory controller 1 ECC error data register 3 Mc1_ecc_data3	0x0748	RO	Memory controller 1 ECC error data register 3 [63:0]: Mc1_ecc_data3, data information in case of an ECC checking error. 256-bit ECC mode data [255:192]

Chapter 14. HyperTransport Controller

In the Loongson 3A5000, the HyperTransport bus is used to enable external device connections and multi-chip interconnections. When used for peripheral connection, I/O Cache consistency can be freely selected by the user program (set through the address window Uncache, see Retry Count Register). When configured to support Cache consistency mode, the I/O device access to the internal DMA is transparent to the Cache hierarchy, i.e., the hardware automatically maintains its consistency without the need for software maintenance through program Cache instructions. When the HyperTransport bus is used for multi-chip interconnects, the HT0 controller (initial address 0x0C00_0000_0000-0x0DFF_FFFF_FFFF) can be pin-configured to support inter-chip Cache. The HT1 controller (initial address 0x0E00_0000_0000-0x0FFF_FFFF_FFFF) can be configured in software to support interchip Cache consistency maintenance, as described in HyperTransport Multi-processor Support. In an 8-chip interconnect fabric, the HT1_HI controller's conformance mode is configured via the pins in CHIP_CONFIG.

The HyperTransport controller supports up to 16-bit width in both directions and 2.4GHz operation. After the connection is established by automatic system initialization, the user program can change the width and operating frequency and re-initialize by modifying the corresponding configuration registers in the protocol as described in HyperTransport Hardware Configuration and Initialization.

The main features of the Loongson 3A5000 HyperTransport controller are as follows:

- Support for HT1.0/HT3.0 protocol
- Supports 200/400/800/1600/2000/2400/3200MHz operating frequency
- HT1.0 supports length of 8 bits
- HT3.0 supports length of 8/16 bits
- Each HT controller (HT0/HT1) can be configured as two 8-bit HT controllers
- Bus control signals (including PowerOK, Rstn, LDT_Stopn) direction can be configured
- Peripheral DMA space Cache/Uncache configurable
- Configurable for Cache Consistency Mode when used in multi-chip interconnects

14.1. HyperTransport Hardware Configuration and Initialization

The HyperTransport bus consists of a transmission signal bus and control signal pins, etc. The following table gives a description of the HyperTransport bus related pins and their functions.

Table 92. Pin signals related to HyperTransport bus

Pins	Name	Description
HT0_8x2	Bus width configuration	1: The 16-bit HyperTransport bus is configured as two separate 8-bit buses, controlled by two separate controllers, with the address space differentiated as HT0_Lo: address[40] = 0; HT0_Hi: address[40] = 1. 0: The 16-bit HyperTransport bus is used as one 16-bit bus, controlled by HT0_Lo, and the address space is the address of HT0_Lo, i.e., address[40] = 0; all signals of HT0_Hi are invalid.

Pins	Name	Description
HT0_Lo_mode	Master device mode	1: Set HT0_Lo to master device mode, in this mode, the bus control signals are driven by HT0_Lo, including HT0_Lo_Powerok, HT0_Lo_Rstn, HT0_Lo_Ldt_Stopn, etc. In this mode, these control signals can also be driven in both directions. Also this pin determines (inverts) the initial value of the register "Act as Slave", which is 0 if the Bridge bit in the packet on the HyperTransport bus is 1, otherwise it is 0. Also, this register is 0 if the If the request address on the HyperTransport bus does not hit in the controller's receive window, it will be resent back to the bus as a P2P request, and if this register is 1, it will respond as an error request if it does not hit. 0: Set HT0_Lo to slave device mode, in this mode, the bus control signals and so on are driven by the other device, these control signals include HT0_Lo_Powerok, HT0_Lo_Rstn, HT0_Lo_Ldt_Stopn. In this mode, these control signals are driven by the other device, if they are not driven correctly, the HT bus cannot work.
HT0_Lo_Powerok	Bus Powerok	HyperTransport bus Powerok signal. controlled by HT0_Lo when HT0_Lo_Mode is 1. When HT0_Lo_Mode is 0, it is controlled by the other device.
HT0_Lo_Rstn	Bus Rstn	HyperTransport bus Rstn signal. controlled by HT0_Lo when HT0_Lo_Mode is 1. When HT0_Lo_Mode is 0, it is controlled by the other device.
HT0_Lo_Ldt_Stopn	Bus Ldt_Stopn	HyperTransport bus Ldt_Stopn signal. controlled by HT0_Lo when HT0_Lo_Mode is 1. When HT0_Lo_Mode is 0, it is controlled by the other device.
HT0_Lo_Ldt_Reqn	Bus Ldt_Reqn	HyperTransport bus Ldt_Reqn signal.
HT0_Hi_mode	Master device mode	1: Set HT0_Hi to master device mode, in this mode, the bus control signals and so on are driven by HT0_Hi, these control signals include HT0_Hi_Powerok, HT0_Hi_Rstn, HT0_Hi_Ldt_Stopn. In this mode, these control signals can also be driven in both directions. Also this pin determines (inverts) the initial value of the register "Act as Slave", which is 0 if the Bridge bit in the packet on the HyperTransport bus is 1, otherwise it is 0. In addition, this register is 0 if the If the request address on the HyperTransport bus does not hit in the controller's receive window, it will be resent back to the bus as a P2P request, and if this register is 1, it will respond as an error request if it does not hit. 0: Set HT0_Hi to slave device mode, in this mode, the bus control signals and so on are driven by the other device, these control signals include HT0_Hi_Powerok, HT0_Hi_Rstn, HT0_Hi_Ldt_Stopn. In this mode, these control signals are driven by the other device, if they are not driven correctly, the HT bus cannot work.

Pins	Name	Description
HT0_Hi_Powerok	Bus Powerok	HyperTransport bus Powerok signal. controlled by HT0_Hi when HT0_Lo_Mode is 1. When HT0_Lo_Mode is 0, it is controlled by the other device. HT0_8x2 with a value of 1 to control the high 8-bit bus. When HT0_8x2 `is `0, it is invalid.
HT0_Hi_Rstn	Bus Rstn	HyperTransport bus Rstn signal. controlled by HT0_Hi when HT0_Lo_Mode is 1. When HT0_Lo_Mode is 0, it is controlled by the other device. HT0_8x2 with a value of 1, controlling the high 8-bit bus. When HT0_8x2 is 0, it is invalid.
HT0_Hi_Ldt_Stopn	Bus Ldt_Stopn	HyperTransport bus Ldt_Stopn signal. controlled by HT0_Hi when HT0_Lo_Mode is 1. When HT0_Lo_Mode is 0, it is controlled by the other device. HT0_8x2 is 1, controlling the high 8-bit bus. When HT0_8x2 is 0, it is invalid.
HT0_Hi_Ldt_Reqn	Bus Ldt_Reqn	HyperTransport bus Ldt_Reqn signal. HT0_8x2 is 1 to control the high 8-bit bus. When HT0_8x2 is 0, it is invalid.
HT0_Rx_CLKp[1:0] HT0_Rx_CLKn[1:0] HT0_Tx_CLKp[1:0] HT0_Tx_CLKp[1:0]	CLK[1:0]	HyperTransport bus CLK signal When HT0_8x2 is 1, CLK[1] is controlled by HT0_Hi CLK[0] is controlled by HT0_Lo When HT0_8x2 is 0, CLK[1:0] is controlled by HT0_Lo

Pins	Name	Description
HT0_Rx_CTLp[1:0] HT0_Rx_CTLn[1:0] HT0_Tx_CTLp[1:0] HT0_Tx_CTLn[1:0]	CTL[1:0]	HyperTransport bus CTL signal When HT0_8x2 is 1, CTL[1] is controlled by HT0_Hi CTL[0] is controlled by HT0_Lo When HT0_8x2 is 0, CTL[1] is invalid CTL[0] is controlled by HT0_Lo
HT0_Rx_CADp[15:0] HT0_Rx_CADn[15:0] HT0_Tx_CADp[15:0] HT0_Tx_CADn[15:0]	CAD[15:0]	HyperTransport bus CAD signal When HT0_8x2 is 1, CAD[15:8] is controlled by HT0_Hi CAD[7:0] is controlled by HT0_Lo When HT0_8x2 is 0, CAD[15:0] is controlled by HT0_Lo

The initialization of HyperTransport starts automatically after each reset. After a cold start the HyperTransport bus will automatically operate at the minimum frequency (200MHz) and minimum width (8-bit) and will try to perform a bus initialization handshake. Whether the initialization is completed or not can be read from the register InitComplete (see Capability Registers). After the initialization is complete, the width of the bus can be read from the Link Width Out and Link Width In registers (see Capability Registers). After initialization, the user can rewrite the Link Width Out, Link Width In and Link Freq registers. The corresponding registers of the other device can also be configured. After the configuration is completed, a hot reset of the bus or a re-initialization operation via the HT_Ldt_Stopn signal is required to make the rewritten register values effective. After the reinitialization, the HyperTransport bus will operate at the new frequency and width. It is important to note that the configuration of the devices on both sides of the HyperTransport needs to correspond to each other, otherwise the HyperTransport interface will not work properly.

14.2. HyperTransport Protocol Support

The Loongson 3A5000's HyperTransport bus supports most commands in the version 1.03/3.0 protocol and includes some extended commands in the extended conformance protocol that supports multi-chip interconnects. The commands that can be received by the HyperTransport receiver in both of these modes are shown in the table below. Note that atomic operation commands for the HyperTransport bus are not supported.

Table 93. Commands that can be received by the HyperTransport receiver

Encode	Channel	Commands	Standard mode	extension (consistency)
000000	-	NOP	Empty package or flow control	
000001	NPC	FLUSH	No operation	
x01xxx	NPC or PC	Write	bit 5:0 - Nonposted 1 - Posted bit 2:0 - Byte 1 - Doubleword bit 1: Don't Care bit 0: Don't Care	bit 5: Must be 1, POSTED bit 2:0 - Byte 1 - Doubleword bit 1: Don't Care bit 0: Must be 1
01xxxx	NPC	Read	bit 3: Don't Care bit 2:0 - Byte 1 - Doubleword bit 1: Don't Care bit 0: Don't Care	bit 3: Don't Care bit 2:0 - Byte 1 - Doubleword bit 1: Don't Care bit 0: Must be 1
110000	R	RdResponse	Read operation returns	
110011	R	TgtDone	The write operation returns	
110100	PC	WrCoherent	----	Write command extension
110101	PC	WrAddr	----	Write address extension
111000	R	RespCoherent	----	Read Response Extension
111001	NPC	RdCoherent	----	Read command extension
111010	PC	Broadcast	No operation	
111011	NPC	RdAddr	----	Read Address Extension
111100	PC	FENCE	Guaranteed sequential relationships	
111111	-	Sync/Error	Sync/Error	

For the sender, the commands that will be sent out in both modes are shown in the table below.

Table 94. Commands that will be transported in both modes

Code	Channel	Commands	Standard mode	extension (consistency)
000000	-	NOP	Blanket or flow control	
x01x0x	NPC or PC	Write	bit 5:0 - Nonposted 1 - Posted bit 2:0 - Byte 1 - Doubleword bit 0:Must be 0	bit 5:Must be 1, POSTED bit 2:0 - Byte 1 - Doubleword bit 0:Must be 1
010x0x	NPC	Read	bit 2:0 - Byte 1 - Doubleword bit 0:Don't Care	bit 2:0 - Byte 1 - Doubleword bit 0:Must be 1
110000	R	RdResponse	Read operation returns	
110011	R	TgtDone	The write operation returns	
110100	PC	WrCoherent	----	Write command extension
110101	PC	WrAddr	----	Write address extension
111000	R	RespCoherent	----	Read Response Extension
111001	NPC	RdCoherent	----	Read command extension
111011	NPC	RdAddr	----	Read Address Extension
111111	-	Sync/Error	Will only forward	

14.3. HyperTransport Interrupt Support

The HyperTransport controller provides 256 interrupt vectors to support Fix, Arbiter, and other types of interrupts, but does not provide support for hardware automatic EOI. For the above two supported types of interrupts, the controller automatically writes to the interrupt register after receiving them and performs interrupt notification to the system interrupt controller according to the setting of the interrupt mask register. See the description of the interrupt control registers in Interrupt Vector Register for specific interrupt control.

14.3.1. PIC Interrupts

The controller has made special support for PIC interrupts to speed up the processing of this type of interrupt.

A typical PIC interrupt is completed by the following steps:

1. The PIC controller sends a PIC interrupt request to the system.
2. The system sends an interrupt vector query to the PIC controller.
3. The PIC controller sends an interrupt vector number to the system.
4. The system clears the corresponding interrupt on the PIC controller.

Only after all the above 4 steps are completed, the PIC controller will send the next interrupt to the system. For the Loongson 3A5000 HyperTransport controller, the first 3 steps will be processed automatically and the PIC interrupt vector will be written to the corresponding location in the 256 interrupt vectors. And after the software system has processed the interrupt, it needs to process step 4, which is to issue a clear interrupt to the PIC controller. After that, the processing of the next interrupt starts.

14.3.2. Local Interrupts Handling

In the legacy interrupt handling model, all interrupts are stored by the interrupt vector inside the HT controller and then distributed through the interrupt line of the HT controller connected to the interrupt router on the chip. In this case, HT interrupts are only available to CPU cores through a limited number of connections, and cannot be distributed across chips, so the usage scenario is rather limited.

In this HT interrupt mode, when performing interrupt processing, the interrupt router on the chip is transparent to the software and the core goes directly to the interrupt vector of the HT controller (typically `0x90000efdfb000080`) for lookup and then per-bit processing. At this point, regardless of how the routing mode is configured, all interrupts on the HT controller are read directly.

14.3.3. Extended Interrupts Handling

The extended I/O interrupts implemented in the 3A5000 can greatly increase the flexibility of interrupt distribution and interrupt handling.

In HT's interrupt expansion mode, interrupts other than PIC interrupts are written directly to the new extended interrupt register on the chip interrupt router, and then routed or distributed according to the relevant configuration of the extended interrupt register.

After using the extended I/O interrupts, the HT controller is transparent to the software when processing the interrupts, and the cores read the interrupt status directly from the extended I/O status register (configuration space `0x1800`) for processing, and each core only reads the interrupt status of the interrupt itself and processes it without interference between different cores.

Interrupt forwarding is performed on the HT controller by enabling the external interrupt transition configuration register. As described in External Interrupt Conversion Configuration, the software needs to set `HT_int_trans` to the target address of the Extended I/O Interrupt Trigger Register. The address of this register in the 3A5000 is `0x1fe01140`, or `0x10000_00001140`.

Before the kernel can use the extended interrupt processing, it needs to enable the corresponding bit in the "Other function settings register". This register has a base address of `0x1fe00000`, It can also be accessed using the configuration register instruction (IOCSR), and an offset address of `0x0420`.

Table 95. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
51:48	EXT_INT_en	RW	0x0	Extended I/O interrupt enable

14.4. HyperTransport Address Windows

14.4.1. HyperTransport Space

The default address window distribution of the four HyperTransport interfaces in the Loongson 3A5000 processor is as follows.

Table 96. Default address window layout of the 4 HyperTransport interfaces

Base Address	End Address	Size	Definition
0x0A00_0000_0000	0x0AFF_FFFF_FFFF	1 Tbytes	HT0_LO Window
0x0B00_0000_0000	0x0BFF_FFFF_FFFF	1 Tbytes	HT0_HI Window
0x0E00_0000_0000	0x0EFF_FFFF_FFFF	1 Tbytes	HT1_LO Window
0x0F00_0000_0000	0x0FFF_FFFF_FFFF	1 Tbytes	HT1_HI Window

By default (no additional system address windows are configured), the software accesses each HyperTransport interface based on the address space described above, and can also access it from other address spaces by configuring the address windows on the cross-switch (see Address Routing Layout and Configuration). The internal 40-bit address space of each HyperTransport interface has the address windows distributed as shown in the table below.

Table 97. Address window distribution inside the HyperTransport interface of the Loongson 3 processor

Base Address	end address	Size	Definition
0x00_0000_0000	0xFC_FFFF_FFFF	1012 Gbytes	MEM Space
0xFD_0000_0000	0xFD_F7FF_FFFF	3968 Mbytes	Reserved
0xFD_F800_0000	0xFD_F8FF_FFFF	16 Mbytes	Interrupt
0xFD_F900_0000	0xFD_F90F_FFFF	1 Mbyte	PIC Interrupt Response
0xFD_F910_0000	0xFD_F91F_FFFF	1 Mbyte	System Information
0xFD_F920_0000	0xFD_FAFF_FFFF	30 Mbytes	Reserved
0xFD_FB00_0000	0xFD_FBFF_FFFF	16 Mbytes	HT Controller Configuration Space
0xFD_FC00_0000	0xFD_FDFF_FFFF	32 Mbytes	I/O space
0xFD_FE00_0000	0xFD_FFFF_FFFF	32 Mbytes	HT Controller Configuration Space
0xFE_0000_0000	0xFF_FFFF_FFFF	8 Gbytes	Reserved

14.4.2. HyperTransport Controller Internal Window Configuration

The HyperTransport interface of the Loongson 3A5000 processor provides a variety of rich address windows for users to use. The roles and functions of these address windows are described in the following table.

Table 98. Address window provided in the HyperTransport interface of the Loongson 3A5000 processor

Address Window	Number of windows	Acceptance bus	Role	Remarks
Receive window (See Receive Address Window Configuration Register for window configuration)	3	HyperTransport	Determines whether to receive accesses issued on the HyperTransport bus.	When in master bridge mode (i.e., act_as_slave is 0 in the configuration register), only accesses falling in these address windows will be responded to by the internal bus, and other accesses will be considered as P2P accesses and re-sent back to the HyperTransport bus When in device mode (i.e., act_as_slave is 1 in the configuration register), only accesses falling in these address windows will be received and processed by the internal bus, and other accesses will be returned with errors according to the protocol.
Post window (see POST Address Window Configuration Register for window configuration)	2	Internal Bus	Determine whether to treat internal bus write accesses to the HyperTransport bus as Post Write	External write accesses that fall in these address spaces will be treated as Post Write. Post Write: in HyperTransport protocol, this write access does not need to wait for a write completion response, i.e., the write access completion response to the processor will be made after the controller issues this write access to the bus.
Prefetchable window (see Prefetchable Address Window Configuration Register for window configuration)	2	Internal Bus	Determines whether to receive internal Cache accesses as Fetch instructions accesses.	When the processor core is executed in a chaotic order, some guess read accesses or fetch accesses are issued to the bus, which are wrong for some I/O spaces. By default, such accesses will be returned directly by the HT controller without accessing the HyperTransport bus. Such accesses to the HyperTransport bus can be enabled through these windows.

Address Window	Number of windows	Acceptance bus	Role	Remarks
Uncache window (see Uncache Address Window Configuration Register for window configuration)	2	HyperTransport	Determine whether to treat accesses on the HyperTransport bus as Uncache accesses to the internal	The I/O DMA accesses inside the Loongson 3A5000 processor will be accessed as Cache by default and will be determined by the SCache to be hit or miss, thus maintaining its I/O consistency information. The configuration of these windows allows accesses that hit in these windows to access memory directly as Uncache, without maintaining I/O consistency information through hardware.

14.5. Configuration Register

The configuration register module is mainly used to control the configuration register access requests arriving from the `AXI SLAVE` side or the `HT RECEIVER` side, to perform external interrupt processing, and to store a large number of software-visible configuration registers used to control the various operating modes of the system.

First of all, the configuration registers used to control the various actions of the HT controller are accessed and stored in this module, and the access offset address of this module is `0xFD_FB00_0000` to `0xFD_FBFF_FFFF` on the HT controller side. All software visible registers in the HT controller are shown in the following table:

Table 99. Software visible registers in the HT controller

Enable	0x00	Device ID		Vendor ID
0x04	Status		Command	
0x08	Class Code			Revision ID
0x0c	BIST	Header Type	Latency Timer	Cache Line Size
0x10				
0x14				
0x18				
0x1c				
0x20				
0x24				
0x28	Cardbus CIS Pointer			
0x2c	Subsystem ID		Subsystem Vendor ID	
0x30	Expansion ROM Enable Address			
0x34	Reserved			Capabilities Pointer
0x38	Reserved			

Enable	0x00	Device ID		Vendor ID	
0x3c	Bridge Control		Interrupt Pin	Interrupt Line	
Cap 0 PRI	0x40	Command		Capabilities Pointer	Capability ID
	0x44	Link Config 0		Link Control 0	
	0x48	Link Config 1		Link Control 1	
	0x4C	LinkFreqCap0		Link Error0/Link Freq 0	Revision ID
	0x50	LinkFreqCap1		Link Error1/Link Freq 1	Feature
	0x54	Error Handling		Enumeration Scratchpad	
	0x58	Reserved		Mem Limit Upper	Mem Enable Upper
Cap 1 Retry	0x60	Capability Type	Reserved	Capability Pointer	Capabiliter ID
	0x64	Status 1	Control 1	Status 0	Control 0
	0x68	Retry Count 1		Retry Count 0	
CAP 3	0x6C	Capability Type	Revision ID	Capability Pointer	Capabiliter ID
CAP 4 Interrupt	0x70	Capability Type	Index	Capability Pointer	Capabiliter ID
	0x74	Dataport			
	0x78	IntrInfo[31:0]			
	0x7C	IntrInfo[63:32]			
Int Vector	0x80	INT Vector[31:0]			
	0x84	INT Vector[63:32]			
	0x88	INT Vector[95:64]			
	0x8C	INT Vector[127:96]			
	0x90	INT Vector[159:128]			
	0x94	INT Vector[191:160]			
	0x98	INT Vector[223:192]			
	0x9C	INT Vector[255:224]			
	0xA0	INT Enable[31:0]			
	0xA4	INT Enable[63:32]			
	0xA8	INT Enable[95:64]			
	0xAC	INT Enable[127:96]			
	0xB0	INT Enable[159:128]			
	0xB4	INT Enable[191:160]			
	0xB8	INT Enable[223:192]			
	0xBC	INT Enable[255:224]			

Enable	0x00	Device ID		Vendor ID	
CAP 5 Gen3	0xC0	Capability Type	Cap Enum/Index	Capability Pointer	Capabiliter ID
	0xC4	Global Link Training			
	0xC8	Transmitter Configuration 0			
	0xCC	Receiver Configuration 0			
	0xD0	Link Training 0			
	0xD4	Frequency Extension			
	0xD8	Transmitter Configuration 1			
	0xDC	Receiver Configuration 1			
	0xE0	Link Training 1			
	0xE4	BIST Control			
Enable	0x100	Device ID		Vendor ID	
	0x104	Status		Command	
	0x108	Class Code			Revision ID
	0x10c	BIST	Header Type	Latency Timer	Cache Line Size
	0x110				
	0x114				
	0x118				
	0x11c				
	0x120				
	0x124				
	0x128	Cardbus CIS Pointer			
	0x12c	Subsystem ID	Subsystem Vendor ID		
	0x130	Expansion ROM Enable Address			
	0x134	Reserved			Capabilities Pointer
	0x138	Reserved			
	0x13c	Bridge Control		Interrupt Pin	Interrupt Line

Enable	0x00	Device ID	Vendor ID
Receive Windows	0x140	HT RX Enable 0	
	0x144	HT RX Mask 0	
	0x148	HT RX Enable 1	
	0x14C	HT RX Mask 1	
	0x150	HT RX Enable 2	
	0x154	HT RX Mask 2	
	0x158	HT RX Enable 3	
	0x15C	HT RX Mask 3	
	0x160	HT RX Enable 4	
	0x164	HT RX Mask 4	
Header Trans	0x168	HT RX Header Trans	
	0x16C	HT RX EXT Header Trans	
Post Windows	0x170	HT TX Post Enable 0	
	0x174	HT TX Post Mask 0	
	0x178	HT TX Post Enable 1	
	0x17C	HT TX Post Mask 1	
Prefetchable Windows	0x180	HT TX Prefetchable Enable 0	
	0x184	HT TX Prefetchable Mask 0	
	0x188	HT TX Prefetchable Enable 1	
	0x18C	HT TX Prefetchable Mask 1	
Uncache Windows	0x190	HT RX Uncache Enable 0	
	0x194	HT RX Uncache Mask 0	
	0x198	HT RX Uncache Enable 1	
	0x19C	HT RX Uncache Mask 1	
	0x1A0	HT RX Uncache Enable 2	
	0x1A4	HT RX Uncache Mask 2	
	0x1A8	HT RX Uncache Enable 3	
	0x1AC	HT RX Uncache Mask 3	
P2P Windows	0x1B0	HT RX P2P Enable 0	
	0x1B4	HT RX P2P Mask 0	
	0x1B8	HT RX P2P Enable 1	
	0x1BC	HT RX P2P Mask 1	

Enable	0x00	Device ID	Vendor ID
APP Config	0x1C0	APP Configuration 0	
	0x1C4	APP Configuration 1	
	0x1C8	RX Bus Value	
	0x1CC	PHY status	
Buffer	0x1D0	TX Buffer 0	
	0x1D4	TX Buffer 1 / Rx buffer hi	
	0x1D8	TX Buffer turning	
	0x1DC	RX Buffer lo	
Training	0x1E0	Training 0 Counter Short	
	0x1E4	Training 0 Counter Long	
	0x1E8	Training 1 Counter	
	0x1EC	Training 2 Counter	
	0x1F0	Training 3 Counter	
PLL	0x1F4	PLL Configuration	
PHY	0x1F8	I/O Configuration	
	0x1FC	PHY Configuration	
DEBUG	0x240	HT3 DEBUG 0	
	0x244	HT3 DEBUG 1	
	0x248	HT3 DEBUG 2	
	0x24C	HT3 DEBUG 3	
	0x250	HT3 DEBUG 4	
	0x254	HT3 DEBUG 5	
	0x258	HT3 DEBUG 6	
POST ID WINDOWS	0x260	HT TX POST ID WIN0	
	0x264	HT TX POST ID WIN1	
	0x268	HT TX POST ID WIN2	
	0x26C	HT TX POST ID WIN3	
POST ID WINDOWS	0x270	INT TRANS WIN lo	
	0x274	INT TRANS WIN hi	

The specific meaning of each register is shown in the following sections.

14.5.1. Bridge Control Register

Offset: 0x3C

Reset value: 0x00000000

Name: Bus reset control

Table 100. Bridge control register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:23	Reserved	9	0x0		Reserved
22	Reset	1	0x0	R/W	Bus reset control. 0 → 1: HT_RSTn set to 0, bus reset 1 → 0: HT_RSTn is set to 1, the bus is unreset
21:0	Reserved	22	0x0		Reserved

14.5.2. Capability Registers

Offset: 0x40

Reset value: 0x20010008

Name: Command, capabilities pointer, capability ID

Table 101. Definition of command, capabilities pointer, capability ID registers

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:29	Slave/Pri	3	0x0	R	Command format is HOST/Sec
28:26	Reserved	2	0x0	R	Reserved
25:21	Unit Count	5	0x0	R/W	Provided to the software for recording the current number of Units
20:16	Unit ID	5	0x0		HOST mode: can be used to record the number of IDs used SLAVE mode: record own Unit ID HOST/SLAVE mode is controlled by act_as_slave register
15:08	Capabilities Pointer	8	0x60	R	Next cap register offset address
7:0	Capability ID	8	0x08	R	HyperTransport capability ID

Offset: 0x44

Reset value: 0x00112000

Name: Link config, link control

Table 102. Definition of link config, link control registers

Bit Field	Name	Length	Reset Value	Read/Write	Description
30:28	Link Width Out	3	0x0	R/W	Sender Width The value after a cold reset is the maximum width of the current connection, the value written to this register will take effect after the next hot reset or HT Disconnect 000: 8-bit mode 001: 16-bit mode
27	Reserved	1	0x0		Reserved
26:24	Link Width In	3	0x0	R/W	Receiver width The value after a cold reset is the maximum width of the current connection, and the value written to this register will take effect after the next hot reset or HT Disconnect
23	Dw Fc out	1	0x0	R	Double-word flow control is not supported on the transmitter side
22:20	Max Link Width out	3	0x1	R	HT Maximum width of HT bus transmitter: 16 bits
19	Dw Fc In	1	0x0	R	Receive side does not support double-word flow control
18:16	Max Link Width In	3	0x1	R	Maximum width of HT bus receiver side: 16 bits
15:14	Reserved	2	0x0		Reserved
13	LDTSTOP#	1	0x1	R/W	Whether to turn off HT PHY when HT bus enters HT Disconnect state 1: Turn off 0: No shutdown
12:10	Reserved	3	0x0		Reserved
9	CRC Error (hi)	1	0x0	R/W	CRC error occurs in high 8 bits
8	CRC Error (lo)	1	0x0	R/W	CRC error occurs in low 8 bits
7	Trans off	1	0x0	R/W	HT PHY Off Control When in 16-bit bus operation mode 1: Turn off high/low 8-bit HT PHY 0: Enable Low 8-bit HT PHY, High 8-bit HT PHY is controlled by bit 0

Bit Field	Name	Length	Reset Value	Read/Write	Description
6	End of Chain	0	0x0	R	HT bus end
5	Init Complete	1	0x0	R	Whether HT bus initialization is complete
4	Link Fail	1	0x0	R	Indicates connection failure
3:2	Reserved	2	0x0		Reserved
1	CRC Flood Enable	1	0x0	R/W	Whether to flood HT bus when CRC error occurs
0	Trans off (hi)	1	0x0	R/W	High 8-bit PHY shutdown control when running 8-bit protocol with 16-bit HT bus 1: Turn off High 8-bit HT PHY 0: Enable High 8-bit HT PHY

Offset: 0x4C

Reset value: 0x80250023

Name: Revision ID, link freq, link error, link freq cap

Table 103. Definition of revision id, link freq, link error, link freq capregisters

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	Link Freq Cap	16	0x0000	R	HT bus frequency supported, with different values generated depending on the external PLL settings (this bit is meaningless when using the software configuration PLL (0x1F4)) {3.2G, 2.6G, 2.4G, 2.2G, 2.0G, 1.8G, 1.6G, 1.4G, 1.2G, 1.0G, 800M, 600M, 500M, 400M, 300M, 200M}
15:14	Reserved	2	0x0		Reserved
13	Over Flow Error	1	0x0	R	HT bus packet overflow
12	Protocol Error	1	0x0	R/W	Protocol error, meaning an unrecognizable command was received on the HT bus

Bit Field	Name	Length	Reset Value	Read/Write	Description
11:8	Link Freq	4	0x0	R/W	HT bus operating frequency, the value written to this register will take effect after the next thermal reset or HT Disconnect, the value set corresponds to the Link Freq Cap bit (when using software configuration PLL (0x1F4), this bit is meaningless)
7:0	Revision ID	8	0x60	R/W	Version number: 3.0

Offset: 0x50

Reset value: 0x00000002

Name: Feature capability

Table 104. Definition of feature registers

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:9	Reserved	23	0x0		Reserved
8	Extended Register	1	0x0	R	No
7:4	Reserved	3	0x0		Reserved
3	Extended CTL Time	1	0x0	R	Not needed
2	CRC Test Mode	1	0x0	R	Not supported
1	LDTSTOP#	1	0x1	R	Support LDTSTOP#
0	Isochronous Mode	1	0x0	R	Not supported

14.5.3. Error Retry Control Register

Used for HyerTransport 3.0 mode error retransmission enable, to configure the maximum number of Short Retry, to show whether the Retry counter is flipped or not.

Offset: 0x64

Reset value: 0x00000000

Name: Error retry control register

Table 105. Error Retry Control Register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:10	Reserved	22	0x0	R	Reserved
9	Retry Count Rollover	1	0x0	R	Retry Counter count rollover

Bit Field	Name	Length	Reset Value	Read/Write	Description
8	Reserved	1	0x0	R	Reserved
7:6	Short Retry Attempts	2	0x0	R/W	Maximum number of Short Retry allowed
5:1	Reserved	5	0x0	R	
0	Link Retry Enable	1	0x0	R/W	Error reconnect function enable control

14.5.4. Retry Count Register

Offset: 0x68

Reset value: 0x00000000

Name: Retry count register

Table 106. Retry Count Register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	Reserved	16	0x0	R	Reserved
15:0	Retry Count	16	0x0	R	Retry count

14.5.5. Revision ID Register

Used to configure the controller version, configure it to a new version number and take effect via Warm Reset.

Offset: 0x6C

Reset value: 0x00200000

Name: Revision ID register

Table 107. Revision ID Register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:24	Reserved	8	0x0	R	Reserved
23:16	Revision ID	8	0x20	R/W	Revision ID control register. 0x20: HyperTransport 1.00. 0x60: HyperTransport 3.00
15:0	Reserved	16	0x0	R	Reserved

14.5.6. Interrupt Discovery and Configuration

Offset: 0x70

Reset value: 0x80000008

Name: Interrupt capability

Table 108. Interrupt capability register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:24	Capabilities Pointer	8	0x80	R	Interrupt discovery and configuration block
23:16	Index	8	0x0	R/W	Read register offset address
15:8	Capabilities Pointer	8	0x0	R	Capabilities Pointer
7:0	Capability ID	8	0x08	R	Hypertransport Capablity ID

Offset: 0x74

Reset value: 0x00000000 Name: Dataport

Table 109. Interrupt dataport register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Dataport	32	0x0	R/W	When the previous register Index is 0x10, the result of reading and writing this register is 0xa8 register, otherwise it is 0xac

Offset: 0x78

Reset value: 0xF8000000

Name: IntrInfo[31:0]

Table 110. Interrupt intrInfo register 1

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:24	IntrInfo[31:24]	8	0xF8	R	Reserved
23:2	IntrInfo[23:2]	22	0x0	R/W	IntrInfo[23:2] When a PIC interrupt is issued, the value of IntrInfo is used to represent the interrupt vector
1:0	Reserved	2	0x0	R	Reserved

Offset: 0x7c

Reset value: 0x00000000

Name: IntrInfo[63:32]

Table 111. Interrupt intrInfo register 2

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	IntrInfo[63:32]	32	0x0	R	Reserved

14.5.7. Interrupt Vector Register

There are 256 interrupt vector registers, among which Fix, Arbiter and PIC interrupts on the HT bus are directly mapped to these 256 interrupt vectors, while other interrupts such as SMI, NMI, INIT, INTA, INTB, INTC, INTD can be mapped to any 8-bit interrupt vector through register 0x50[28:24]. The mapping order is {INTD, INTC, INTB, INTA, 1'b0, INIT, NMI, SMI}. The corresponding value of the interrupt vector is {Interrupt Index, internal vector [2:0]}.

By default, 256-bit interrupts can be distributed to 4-bit interrupt lines. When not using interrupts from the high 8-bit HT controller, 256-bit interrupts can also be distributed to 8-bit interrupt lines by setting ht_int_8bit.

The 256 interrupt vectors are mapped to different interrupt lines by selecting different register configurations according to the interrupt routing method as follows:

Table 112. Interrupt vectors mapping

Number of interruptions	Strip	0	1	2	3	4	5	6	7
4 X = [63:0]	1	[X]	[X+64]	[X+128]	[X+192]	-	-	-	-
	2	[2X]	[2X+1]	[2X+128]	[2X+129]	-	-	-	-
	4	[4X]	[4X+1]	[4X+2]	[4X+3]	-	-	-	-
8 X = [31:0] Y = [63:32]	1	[X]	[Y]	[X+64]	[Y+64]	[X+128]	[Y+128]	[X+192]	[Y+192]
	2	[2X]	[2Y]	[2X+1]	[2Y+1]	[2X+128]	[2Y+128]	[2X+129]	[2Y+129]
	4	[4X]	[4X+32]	[4X+1]	[4X+33]	[4X+2]	[4X+34]	[4X+3]	[4X+35]

As an example of using a 4-bit interrupt line, the different mappings are as follows.

ht_int_stripe_1:

[0,1,2,3……63] Corresponding to interrupted line 0 / HT HI Corresponding to interrupted line 4

[64,65,66,67……127] Corresponding interrupt line 1 / HT HI Corresponding interrupt line 5

[128,129,130,131 ……191] corresponds to interrupted line 2 / HT HI corresponds to interrupted line 6

[192,193,194,195 ……255] corresponds to the broken line 3 / HT HI corresponds to the broken line 7

ht_int_stripe_2:

[0,2,4,6 ……126] corresponds to interrupt line 0 / HT HI corresponds to interrupt line 4

[1,3,5,7 ……127] corresponds to interrupt line 1 / HT HI corresponds to interrupt line 5

[128,130,132,134 ……254] corresponds to interrupted line 2 / HT HI corresponds to interrupted line 6

[129,131,133,135 ……255] corresponds to the interrupt line 3 / HT HI corresponds to the interrupt line 7

ht_int_stripe_4:

[0,4,8,12 ……252] corresponds to interrupt line 0 / HT HI corresponds to interrupt line 4

[1,5,9,13 ……253] corresponds to interrupt line 1 / HT HI corresponds to interrupt line 5

[2,6,10,14 ……254] corresponds to interrupted line 2 / HT HI corresponds to interrupted line 6

[3,7,11,15 ……255] corresponds to interrupted line 3 / HT HI corresponds to interrupted line 7

The following description of the interrupt vector corresponds to ht_int_stripe_1, the other two ways can be obtained from the above description.

Offset: 0x80

Reset value: 0x00000000

Name: HT bus interrupt vector register [31:0]

Table 113. HT bus interrupt vector register definition 1

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[31:0]	32	0x0	R/W	HT bus interrupt vector register [31:0] corresponds to interrupt line 0. HT HI corresponds to interrupt line 4

Offset: 0x84

Reset value: 0x00000000

Name: HT bus interrupt vector register[63:32]

Table 114. HT bus interrupt vector register definition 2

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[63:32]	32	0x0	R/W	HT bus interrupt vector register [63:32] corresponds to interrupt line 0. HT HI corresponds to interrupt line 4

Offset: 0x88

Reset value: 0x00000000

Name: HT Bus Interrupt Vector Register [95:64]

Table 115. HT bus interrupt vector register definition 3

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[95:64]	32	0x0	R/W	HT bus interrupt vector register [95:64] corresponds to interrupt line 1. HT HI corresponds to interrupt line 5

Offset: 0x8c

Reset value: 0x00000000

Name: HT bus interrupt vector register [127:96]

Table 116. HT bus interrupt vector register definition 4

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[127:96]	32	0x0	R/W	HT bus interrupt vector register [127:96] corresponds to interrupt line 1. HT HI corresponds to interrupt line 5

Offset: 0x90

Reset value: 0x00000000

Name: HT bus interrupt vector register [159:128]

Table 117. HT bus interrupt vector register definition 5

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[159:128]	32	0x0	R/W	HT bus interrupt vector register [159:128] corresponds to interrupt line 2. HT HI corresponds to interrupt line 6

Offset: 0x94

Reset value: 0x00000000

Name: HT bus interrupt vector register [191:160]

Table 118. HT bus interrupt vector register definition 6

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[191:160]	32	0x0	R/W	HT bus interrupt vector register [191:160] corresponds to interrupt line 2. HT HI corresponds to interrupt line 6

Offset: 0x98

Reset value: 0x00000000

Name: HT bus interrupt vector register [223:192]

Table 119. HT bus interrupt vector register definition 7

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[223:192]	32	0x0	R/W	HT bus interrupt vector register [223:192] corresponds to interrupt line 3. HT HI corresponds to interrupt line 7

Offset: 0x9c

Reset value: 0x00000000

Name: HT bus interrupt vector register [255:224]

Table 120. HT bus interrupt vector register definition 8

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_case[255:224]	32	0x0	R/W	HT bus interrupt vector register [255:224] corresponds to interrupt line 3. HT HI corresponds to interrupt line 7

14.5.8. Interrupt Enable Register

There are 256 interrupt enable registers, which correspond to the interrupt vector registers. The interrupt enable register is set to 1 to turn on the corresponding interrupt, and set to 0 to mask the interrupt.

The 256 interrupt vectors are mapped to different interrupt lines according to the configuration of the interrupt routing registers, as follows:

ht_int_stripe_1:

[0,1,2,3……63] Corresponding interrupted line 0 / HT HI Corresponding interrupted line 4

[64,65,66,67……127] Corresponding interrupted line 1 / HT HI Corresponding interrupted line 5

[128,129,130,131……191] Corresponding interrupted line 2 / HT HI Corresponding interrupted line 6

[192,193,194,195……255] Corresponding interrupted line 3 / HT HI Corresponding interrupted line 7

ht_int_stripe_2:

[0,2,4,6……126] Corresponding interrupted line 0 / HT HI Corresponding interrupted line 4

[1,3,5,7……127] Corresponding interrupted line 1 / HT HI Corresponding interrupted line 5

[128,130,132,134……254] Corresponding interrupted line 2 / HT HI Corresponding interrupted line 6

[129,131,133,135……255] Corresponding interrupted line 3 / HT HI Corresponding interrupted line 7

ht_int_stripe_4:

[0,4,8,12……252] Corresponding interrupted line 0 / HT HI Corresponding interrupted line 4

[1,5,9,13……253] Corresponding interrupted line 1 / HT HI Corresponding interrupted line 5

[2,6,10,14……254] Corresponding interrupted line 2 / HT HI Corresponding interrupted line 6

[3,7,11,15……255] Corresponding interrupted line 3 / HT HI Corresponding interrupted line 7

The following description of the interrupt vector corresponds to ht_int_stripe_1, the other two ways can be obtained from the above description.

Offset: 0xa0

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [31:0]

Table 121. HT bus interrupt enable register definition 1

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[31:0]	32	0x0	R/W	HT bus interrupt enable register [31:0] corresponds to interrupt line 0. HT HI corresponds to interrupt line 4

Offset: 0xa4

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [63:32]

Table 122. HT bus interrupt enable register definition 2

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[63:32]	32	0x0	R/W	HT bus interrupt enable register [63:32] corresponds to interrupt line 0. HT HI corresponds to interrupt line 4

Offset: 0xa8

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [95:64]

Table 123. HT bus interrupt enable register definition 3

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[95:64]	32	0x0	R/W	HT bus interrupt enable register [95:64] corresponds to interrupt line 1. HT HI corresponds to interrupt line 5

Offset: 0xac

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [127:96]

Table 124. HT bus interrupt enable register definition 4

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[127:96]	32	0x0	R/W	HHT bus interrupt enable register [127:96] corresponds to interrupt line 1. HT HI corresponds to interrupt line 5

Offset: 0xb0

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [159:128]

Table 125. HT bus interrupt enable register definition 5

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[159:128]	32	0x0	R/W	HT bus interrupt enable register [159:128] corresponds to interrupt line 2. HT HI corresponds to interrupt line 6

Offset: 0xb4

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register[191:160]

Table 126. HT bus interrupt enable register definition 6

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[191:160]	32	0x0	R/W	HT bus interrupt enable register [191:160] corresponds to interrupt line 2. HT HI corresponds to interrupt line 6

Offset: 0xb8

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [223:192]

Table 127. HT bus interrupt enable register definition 7

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[223:192]	32	0x0	R/W	HT bus interrupt enable register [223:192] corresponds to interrupt line 3. HT HI corresponds to interrupt line 7

Offset: 0xbc

Reset value: 0x00000000

Name: HT Bus Interrupt Enable Register [255:224]

Table 128. HT bus interrupt enable register definition 8

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	Interrupt_mask[255:224]	32	0x0	R/W	HT bus interrupt enable register [255:224] corresponds to interrupt line 3. HT HI corresponds to interrupt line 7

14.5.9. Link Train Register

HyperTransport 3.0 Link initialization and link training control registers.

Offset: 0xD0

Reset value: 0x00000070

Name: Link train register

Table 129. Link Train Register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:23	Reserved	9	0x0	R	Reserved
22:21	Transmitter LS select	2	0x0	R/W	Link status of the sender in Disconnected or Inactive state. 2'b00 LS1 2'b01 LS0 2'b10 LS2 2'b11 LS3

Bit Field	Name	Length	Reset Value	Read/Write	Description
14	Dsiable Cmd Throttling	1	0x0	R/W	In HyperTransport 3.0 mode, only one Non-info CMD can appear in any four consecutive DWSs by default. 1'b0 - enable cmd throttling 1'b1 - disable cmd throttling
13:10	Reserved	4	0x0	R	Reserved
8:7	Receiver LS select	2	0x0	R/W	Link status of the receiving end in Disconnected or Inactive state. 2'b00 LS1 2'b01 LS0 2'b10 LS2 2'b11 LS3
6:4	Long Retry Count	3	0x7	R/W	Long Retry Maximum number of times
3	Scrambling Enable	1	0x0	R/W	Enable or disable Scramble 0: disable scramble 1: enable scramble
2	8B10B Enable	1	0x0	R/W	Whether to enable 8B10B 0: disabled 8B10B 1: enable 8B10B
1	AC	1	0x0	R	Whether AC mode is detected 0: AC mode is not detected 1: AC mode is detected
0	Reserved	1	0x0	R	Reserved

14.5.10. Receive Address Window Configuration Register

The formula for hitting the address window in the HT controller is as follows:

```
hit = ( BASE & MASK ) == ( ADDR & MASK )
addr_out_trans = TRANS_EN ? TRANS | ADDR & ~MASK : ADDR
addr_out = Multi_node_en ?
addr_out_trans[39:37], addr_out_trans[43:40], 3'b0,addr_out[36:0]:
addr_out_trans
```

It should be noted that when configuring the address window register, the high bits of `MASK` should be all 1s and the low bits should be all `0` values. The actual number of bits of `0` in `MASK` indicates the size of the address window.

The address of the receive address window is the address received on the HT bus. HT addresses that fall within the P2P window will be forwarded back to the HT bus as P2P commands, HT addresses that fall within the normal receive window and are not in the P2P window will be sent to the CPU, and commands for other addresses will be forwarded back to the HT bus as P2P commands.

Offset: `0x140`

Reset value: `0x00000000`

Name: HT bus receive address window 0 enable (external access)

Table 130. Definition of busreceive address window 0 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_image0_en	1	0x0	R/W	HT bus receives address window 0, enable signal
30	ht_rx_image0_trans_en	1	0x0	R/W	HT bus receive address window 0, map enable signal
29	ht_rx_image0_multi_node_en	1	0x0	R/W	HT bus receive address window 0, multi-node address mapping enable Convert [39:37] to [46:44] of the address
28	ht_rx_image0_conf_hit_en	1	0x0	R/W	HT bus receive address window 0, protocol address hit enable. Must be set to 0
25:0	ht_rx_image0_trans[49:24]	26	0x0	R/W	HT bus receive address window 0, [49:24] of the mapped address

Offset: `0x144`

Reset value: `0x00000000`

Name: HT bus receive address window 0 base address (external access)

Table 131. Definition of HT bus receive address window 0 base address (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_image0_base[39:24]	16	0x0	R/W	HT bus receive address window 0, address base address of [39:24]
15:0	ht_rx_image0_mask[39:24]	16	0x0	R/W	HT bus receive address window 0, address mask of [39:24]

Offset: 0x148

Reset value: 0x00000000

Name: HT bus receive address window 1 enable (external access)

Table 132. Definition of HT bus receive address window 1 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_image1_en	1	0x0	R/W	HT bus receives address window 1, enable signal
30	ht_rx_image1_trans_en	1	0x0	R/W	HT bus receive address window 1, mapping enable signal
29	ht_rx_image1_multi_node_en	1	0x0	R/W	HT bus receive address window 1, multi-node address mapping enable. Convert [39:37] to [46:44] of the address
28	ht_rx_image1_conf_hit_en	1	0x0	R/W	HT bus receive address window 1, protocol address hit enable. Must be set to 0
25:0	ht_rx_image1_trans[49:24]	26	0x0	R/W	HT bus receive address window 1, [49:24] of the mapped address

Offset: 0x14c

Reset value: 0x00000000

Name: HT bus receive address window 1 base address (external access)

Table 133. Definition of bus receive address window 1 base address (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_image1_base[39:24]	16	0x0	R/W	HT bus receive address window 1, [39:24] of the address base address
15:0	ht_rx_image1_mask[39:24]	16	0x0	R/W	HT bus receive address window 1, address mask of [39:24]

Offset: 0x150

Reset value: 0x00000000

Name: HT bus receive address window 2 enable (External Access)

Table 134. Definition of bus receive address window 2 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_image2_en	1	0x0	R/W	HT bus receives address window 2, enable signal
30	ht_rx_image2_trans_en	1	0x0	R/W	HT bus receive address window 2, mapping enable signal
29	ht_rx_image2_multi_node_en	1	0x0	R/W	HT bus receive address window 2, multi-node address mapping enable. Convert [39:37] to [46:44] of the address
28	ht_rx_image2_conf_hit_en	1	0x0	R/W	HT bus receive address window 2, protocol address hit enable. Must be set to 0
25:0	ht_rx_image2_trans[49:24]	26	0x0	R/W	HT bus receive address window 2, [49:24] of the mapped address

Offset: 0x154

Reset value: 0x00000000

Name: HT bus receive address window 2 base address (external access)

Table 135. Definition of HT bus receive address window 2 base address (external Access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_image2_base[39:24]	16	0x0	R/W	HT bus receive address window 2, address base address of [39:24]
15:0	ht_rx_image2_mask[39:24]	16	0x0	R/W	HT bus receive address window 2, address masked `[39:24]

Offset: 0x158

Reset value: 0x00000000

Name: HT bus receive address window 3 enable (external access)

Table 136. Definition of HT bus receive address window 3 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_image3_en	1	0x0	R/W	HT bus receives address window 3, enable signal

Bit Field	Name	Length	Reset Value	Read/Write	Description
30	ht_rx_image3_trans_en	1	0x0	R/W	HT bus receive address window 3, mapping enable signal
29	ht_rx_image3_multi_node_en	1	0x0	R/W	HT bus receive address window 3, multi-node address mapping enable. Convert [39:37] to [46:44] of the address
28	ht_rx_image3_conf_hit_en	1	0x0	R/W	HT bus receive address window 3, protocol address hit enable. Must be set to 0
25:0	ht_rx_image3_trans[49:24]	26	0x0	R/W	HT bus receive address window 3, [49:24] of the mapped address

Offset: 0x15C

Reset value: 0x00000000

Name: HT bus receive address window 3 base address (external access)

Table 137. Definition of HT bus receive address window 3 base address (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_image3_base[39:24]	16	0x0	R/W	HT bus receive address window 3, address base address of [39:24]
15:0	ht_rx_image3_mask[39:24]	16	0x0	R/W	HT bus receive address window 3, address masked [39:24]

Offset: 0x160

Reset value: 0x00000000

Name: HT bus receive address window 4 enable (external access)

Table 138. Definition of HT bus receive address window 4 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_image4_en	1	0x0	R/W	HT bus receives address window 4, enable signal
30	ht_rx_image4_trans_en	1	0x0	R/W	HT bus receive address window 4, map enable signal

Bit Field	Name	Length	Reset Value	Read/Write	Description
29	ht_rx_image4_multi_node_en	1	0x0	R/W	HT bus receive address window 4, multi-node address mapping enable. Convert [39:37] to [46:44] of the address
28	ht_rx_image4_conf_hit_en	1	0x0	R/W	HT bus receive address window 4, protocol address hit enable. Must be set to 0
25:0	ht_rx_image4_trans[49:24]	26	0x0	R/W	HT bus receive address window 4, [49:24] of the mapped address

Offset: 0x164

Reset value: 0x00000000

Name: HT bus receive address window 4 base address (external access)

Table 139. Definition of HT bus receive address window 4 base address (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_image4_base[39:24]	16	0x0	R/W	HT bus receive address window 4, address base address of [39:24]
15:0	ht_rx_image4_mask[39:24]	16	0x0	R/W	HT bus receive address window 4, address masked [39:24]

14.5.11. Space Conversion Configuration Register

Used to perform various transformations of the HT's configuration space.

Offset: 0x168

Reset value: 0x00000000

Name: Configuration space extension address translation

Table 140. Definition of configuration space extended address translation register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_header_trans_ext	1	0x1	R/W	Adjust the converted address type1 flag bit of the configuration space (0xFD_FE000000) from 24 bits to 28 bits for unification with the EXT HEADER space

Bit Field	Name	Length	Reset Value	Read/Write	Description
30	ht_rx_header_trans_en	1	0x1	R/W	Enable the high address ([39:24]) of the configuration space (0xFD_FE000000) to be converted
29:0	ht_rx_header_trans[53:24]	30	0xFE00	R/W	High address [53:24] after configuration space conversion (only [53:25] is actually available)

Offset: 0x16C

Reset value: 0x00000000

Name: Extended address translation

Table 141. Definition of extended address translation register

Bit Field	Name	Length	Reset Value	Read/Write	Description
30	ht_rx_ext_header_trans_en	1	0x0	R/W	Enables the high address ([39:28]) of the extended configuration space (0xFE_00000000) to be converted
29:0	ht_rx_ext_header_trans[53:24]	30	0x0	R/W	High address [53:24] after extended configuration space conversion (only [53:29] is actually available)

14.5.12. POST Address Window Configuration Register

The address window hit formula is detailed in Receive Address Window Configuration Register.

The address in this window is the address received on the AXI bus. All write accesses that land in this window will be returned immediately on the AXI B channel and sent to the HT bus in the POST WRITE command format. Write requests not in this window, on the other hand, are sent to the HT bus in NONPOST WRITE format and wait for the HT bus response before returning to the AXI bus.

Offset: 0x170

Reset value: 0x00000000

Name: HT bus POST address window 0 enable (internal access)

Table 142. HT bus POST address window 0 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_post0_en	1	0x0	R/W	HT bus POST address window 0, enable signal

Bit Field	Name	Length	Reset Value	Read/Write	Description
30	ht_split0_en	1	0x0	R/W	HT access unpacking enable (corresponds to the external uncache ACC operation window of the CPU core)
29:23	Reserved	14	0x0		Reserved
15:0	ht_post0_trans[39:24]	16	0x0	R/W	HT bus POST address window 0, [39:24] of the translated address

Offset: 0x174

Reset value: 0x00000000

Name: HT bus POST address window 0 base address (internal access)

Table 143. HT bus POST address window 0 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_post0_base[39:24]	16	0x0	R/W	HT bus POST address Window 0, address base address of [39:24]
15:0	ht_post0_mask[39:24]	16	0x0	R/W	HT bus POST address window 0, address masked [39:24]

Offset: 0x178

Reset value: 0x00000000

Name: HT bus POST address window 1 enable (internal access)

Table 144. HT bus POST address window 1 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_post1_en	1	0x0	R/W	HT bus POST address window 1, enable signal
30	ht_split1_en	1	0x0	R/W	HT access unpacking enable (corresponds to the external uncache ACC operation window of the CPU core)
29:16	Reserved	14	0x0		Reserved
15:0	ht_post1_trans[39:24]	16	0x0	R/W	HT bus POST address window 1, [39:24] of the translated address

Offset: 0x17c

Reset value: 0x00000000

Name: HT bus POST address window 1 base address (internal access)

Table 145. HT bus POST address window 1 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_post1_base[39:24]	16	0x0	R/W	HT bus POST address window 1, address base address of [39:24]
15:0	ht_post1_mask[39:24]	16	0x0	R/W	HT bus POST address window 1, [39:24] of address mask

14.5.13. Prefetchable Address Window Configuration Register

The address window hit formula is detailed in Receive Address Window Configuration Register.

The address in this window is the address received on the AXI bus. Fetch instructions and Cache accesses that land in this window are only sent to the HT bus. Other fetch or Cache accesses will not be sent to the HT bus, but will be returned immediately, or in the case of read instructions, the corresponding number of invalid reads.

Offset: 0x180

Reset value: 0x00000000

Name: HT bus prefetchable address window 0 enable (Internal Access)

Table 146. HT bus prefetchable address window 0 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_prefetch0_en	1	0x0	R/W	HT bus prefetchable address window 0, enable signal
30:16	Reserved	15	0x0	R/W	Reserved
15:0	ht_prefetch0_trans[39:24]	16	0x0	R/W	HT bus prefetchable address window 0, [39:24] of the translated address

Offset: 0x184

Reset value: 0x00000000

Name: HT bus prefetchable address window 0 base address (internal access)

Table 147. HT bus prefetchable address window 0 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_prefetch0_base[39:24]	16	0x0	R/W	HT bus prefetches address window 0, address base address of [39:24] bits
15:0	ht_prefetch0_mask[39:24]	16	0x0	R/W	HT bus prefetchable address window 0, [39:24] of address mask

Offset: 0x188

Reset value: 0x00000000

Name: HT bus prefetchable address window 1 enable (internal access)

Table 148. HT bus prefetchable address window 1 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_prefetch1_en	1	0x0	R/W	HT bus prefetchable address window 1, enable signal
30:16	Reserved	15	0x0		Reserved
15:0	ht_prefetch1_trans[39:24]	16	0x0	R/W	HT bus prefetchable address window 1, [39:24] of the translated address

Offset: 0x18c

Reset value: 0x00000000

Name: HT bus prefetchable address window 1 base address (internal access)

Table 149. HT bus prefetchable address window 1 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_prefetch1_base[39:24]	16	0x0	R/W	HT bus prefetchable address window 1, address base address of [39:24]
15:0	ht_prefetch1_mask[39:24]	16	0x0	R/W	HT bus prefetchable address window 1, address masked [39:24]

14.5.14. Uncache Address Window Configuration Register

The address window hit formula is detailed in Receive Address Window Configuration Register.

The address of this window is the address received on the HT bus. A read or write command that lands in this window will not be sent to SCache and will not invalidate the first-level Cache, but will be sent directly to memory or other address space, i.e., the read or write command in this address window will not maintain Cache consistency of the I/O. This window is mainly for some operations that will not hit in Cache and therefore can improve the efficiency of accessing memory, such as accessing video memory.

Offset: 0x190

Reset value: 0x00000000

Name: HT bus uncache address window 0 enable (internal access)

Table 150. HT bus uncache address window 0 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_uncache0_en	1	0x0	R/W	HT bus uncache address window 0, enable signal

Bit Field	Name	Length	Reset Value	Read/Write	Description
30	ht_uncache0_trans_en	1	0x0	R/W	HT bus uncache address window 0, mapping enable signal
29	ht_uncache0_multi_node_en	1	0x0	R/W	HT bus uncache receive address window 0, multi-node address mapping enable
28	ht_uncache0_conf_hit_en	1	0x0	R/W	HT bus uncache receive address window 0, protocol address hit enable
25:0	ht_uncache0_trans[49:24]	26	0x0	R/W	HT bus uncache address window 0, [49:24] of the translated address

Offset: 0x194

Reset value: 0x00000000

Name: HT bus uncache address window 0 base address (internal access)

Table 151. HT bus uncache address window 0 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_uncache0_base[39:24]	16	0x0	R/W	HT bus uncache address window 0, address base address of [39:24]
15:0	ht_uncache0_mask[39:24]	16	0x0	R/W	HT bus uncache address window 0, address masked [39:24]

Offset: 0x198

Reset value: 0x00000000

Name: HT bus uncache address window 1 enabled (internal access)

Table 152. HT bus uncache address window 1 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_uncache1_en	1	0x0	R/W	HT bus uncache address window 1, enable signal
30	ht_uncache1_trans_en	1	0x0	R/W	HT bus uncache address window 1, mapping enable signal
29	ht_uncache1_multi_node_en	1	0x0	R/W	HT bus uncache receive address window 1, multi-node address mapping enable
28	ht_uncache1_conf_hit_en	1	0x0	R/W	HT bus uncache receive address window 1, protocol address hit enable

Bit Field	Name	Length	Reset Value	Read/Write	Description
25:0	ht_uncache1_trans[49:24]	26	0x0	R/W	HT bus uncache address window 1, [49:24] of the translated address

Offset: 0x19c

Reset value: 0x00000000

Name: HT bus uncache address window 1 base address (internal access)

Table 153. HT bus uncache address window 1 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_uncache1_base[39:24]	16	0x0	R/W	HT bus uncache address window 1, address base address of [39:24]
15:0	ht_uncache1_mask[39:24]	16	0x0	R/W	HT bus uncache address window 1, [39:24] of address mask

Offset: 0x1A0

Reset value: 0x00000000

Name: HT bus uncache address window 2 enable (internal access)

Table 154. HT bus uncache address window 2 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_uncache2_en	1	0x0	R/W	HT bus uncache address window 2, enable signal
30	ht_uncache2_trans_en	1	0x0	R/W	HT bus uncache address window 2, mapping enable signal
29	ht_uncache2_multi_node_en	1	0x0	R/W	HT bus uncache receive address window 2, multi-node address mapping enable
28	ht_uncache2_conf_hit_en	1	0x0	R/W	HT bus uncache receive address window 2, protocol address hit enable
25:0	ht_uncache2_trans[49:24]	26	0x0	R/W	HT bus uncache address window 2, [49:24] of the translated address

Offset: 0x1A4

Reset value: 0x00000000

Name: HT bus uncache address window 2 base address (internal access)

Table 155. HT Bus uncache Address Window 2 Base Address (Internal Access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_uncache2_base[39:24]	16	0x0	R/W	HT bus uncache address window 2, address base address of [39:24]
15:0	ht_uncache2_mask[39:24]	16	0x0	R/W	HT bus uncache address window 2, address masked [39:24]

Offset: 0x1A8

Reset value: 0x00000000

Name: HT bus uncache address window 3 enabled (internal access)

Table 156. HT bus uncache address window 3 enable (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_uncache3_en	1	0x0	R/W	HT bus uncache address window 3, enable signal
30	ht_uncache3_trans_en	1	0x0	R/W	HT bus uncache address window 3, mapping enable signal
29	ht_uncache3_multi_node_en	1	0x0	R/W	HT bus uncache receive address window 3, multi-node address mapping enable
28	ht_uncache3_conf_hit_en	1	0x0	R/W	HT bus uncache receive address window 3, protocol address hit enable
25:0	ht_uncache3_trans[49:24]	26	0x0	R/W	HT bus uncache address window 3, [49:24] of the translated address

Offset: 0x1AC

Reset value: 0x00000000

Name: HT Bus uncache address window 3 base address (internal sccess)

Table 157. HT Bus uncache address window 3 base address (internal access)

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_uncache3_base[39:24]	16	0x0	R/W	HT bus uncache address window 3, address base address of [39:24]
15:0	ht_uncache3_mask[39:24]	16	0x0	R/W	HT bus uncache address window 3, address masked [39:24]

14.5.15. P2P Address Window Configuration Register

The address window hit formula is detailed in Receive Address Window Configuration Register.

The address of this window is the address received on the HT bus. The read and write commands that land on the address of this window are forwarded directly back to the bus as P2P commands, and this window has the highest priority compared to the normal receive window and the uncache window.

Offset: `0x1B0`

Reset value: `0x00000000`

Name: HT bus P2P address window 0 enable (external access)

Table 158. Definition of HT bus P2P address window 0 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_p2p0_en	1	0x0	R/W	HT bus P2P address window 0, enable signal
29:0	ht_rx_p2p0_trans[53:24]	30	0x0	R/W	HT bus P2P address window 0, [53:24] of the translated address

Offset: `0x1B4`

Reset value: `0x00000000`

Name: HT bus P2P address window 0 base address (external access)

Table 159. Definition of HT bus P2P address window 0 base address (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_p2p0_base[39:24]	16	0x0	R/W	HT bus P2P address window 1, address base address of [39:24]
15:0	ht_rx_p2p0_mask[39:24]	16	0x0	R/W	HT bus P2P address window 1, address masked [39:24]

Offset: `0x1B8`

Reset value: `0x00000000`

Name: HT bus P2P address window 1 enable (external access)

Table 160. Definition of HT bus P2P address window 1 enable (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	ht_rx_p2p1_en	1	0x0	R/W	HT bus P2P address window 1, enable signal
29:0	ht_rx_p2p1_trans[53:24]	30	0x0	R/W	HT bus P2P address window 1, [53:24] of the translated address

Offset: `0x1BC`

Reset value: `0x00000000`

Name: HT bus P2P address window 1 base address (external access)

Table 161. Definition of HT bus P2P address window 1 base address (external access) register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	ht_rx_p2p1_base[39:24]	16	0x0	R/W	HT bus P2P address window 1, address base address of [39:24]
15:0	ht_rx_p2p1_mask[39:24]	16	0x0	R/W	HT bus P2P address window 1, address masked [39:24]

14.5.16. Controller Parameter Configuration Register

Offset: 0x1C0

Reset value: 0x00904321

Name: APP CONFIG 0

Table 162. Definition of controller parameter configuration register 0

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:30	Reserved	1	0x0		Reserved
29	Ldt Stop Gen	1	0x0	R/W	Putting the bus into LDT DISCONNECT mode The correct way is: 0 → 1
28	Ldt Req Gen	1	0x0	R/W	Wake up HT bus from LDT DISCONNECT, set LDT_REQ_n The correct way is to set 0 and then 1: 0 → 1 In addition, a direct read/write request to the bus can also wake up the bus automatically
27	rx sample en	1	0x0	R/W	Enable cad and ctl for sample input, displayed in the 0x1c8 register for debugging
26	Dword Write	1	0x1	R/W	For 32/64/128/256-bit MEM write access, whether to use the Dword Write command format (Byte Write method of writing will be converted to 128-bit write with MASK on reception)
25	Dword Write cfg	1	0x1	R/W	For configuration space write access, whether to use the Dword Write command format (Byte Write method of writing will be converted to 128-bit write with MASK when received)

Bit Field	Name	Length	Reset Value	Read/Write	Description
24	Dword Write I/O	1	0x1	R/W	For I/O space write access, whether to use the Dword Write command format (Byte Write method of writing will be converted to 128-bit write with MASK when received)
23	axi aw resize	1	0x0	RW	Whether to reset the size of the 128-bit write with MASK by Mask
22	Coherent Mode	1	0x0	RW	Whether it is processor consistency mode, the initial value is determined by the ICCC_EN pin and takes effect after reset
21	Coherent_split	1	0x0	RW	Consistency mode, split all packets into 32 byte processing
20	Not Care Seqid	1	0x0	R/W	Whether the receiver does not care about HT sequential relations
19:16	Fixed Seqid	4	0x0	R/W	Configure Seqid issued by HT bus when Not Axi2Seqid is valid
15:12	Priority Nop	4	0x4	R/W	HT bus Nop flow control packet priority
11:8	Priority NPC	4	0x3	R/W	Non Post channel read/write priority
7:4	Priority RC	4	0x2	R/W	Response channel read/write priority
3:0	Priority PC	4	0x1	R/W	Post channel read/write priority 0x0: Highest priority 0xF: Lowest priority For each channel priority are used according to the time change to increase the priority policy, the group memory is used to configure the initial priority of each channel

Offset: 0x1C4

Reset value: 0x00904321

Name: APP CONFIG1

Table 163. Definition of controller parameter configuration register 1

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	tx post split en	1	0x0	R/W	Enable write unwrapping when the tx post ID window hits (all write requests that cross a 32-byte bound are split into two consecutive write requests (byte write))
30	tx wr passPW pc	1	0x0	R/W	Set the passPW bit to 1 for all outgoing Post channel write requests
29	tx wr passPW npc	1	0x0	R/W	Set the passPW bit to 1 for all outgoing Nonpost channel write requests
28	tx rd passPW	1	0x0	R/W	Set the passPW bit to 1 for all outgoing read requests
27	stop same id wr	1	0x0	R/W	When the sender encounters a write request with the same AXI ID, it stops sending until the previous request with the same ID returns.
26	stop same id rd	1	0x0	R/W	When the sender encounters a read request with the same AXI ID, it stops sending until the previous request with the same ID returns.
25	Not axi2seqid wr	1	0x0	R/W	Disable the conversion of write request AXI ID to seqid, use fixed seqid directly
24	Not axi2seqid rd	1	0x0	R/W	Disable the conversion of AXI ID to seqid for read requests, use fixed seqid directly
23:22	Reserved	2	0x0	R/W	Reserved
21	act as slave	1	0x1	R/W	Set SLAVE mode
20	Host hide	1	0x0	R/W	Disable access to the configuration register space at the receiving end

Bit Field	Name	Length	Reset Value	Read/Write	Description
19:16	Rrequest delay	4	0x3	R/W	Used to control the random delay range of Rrequest transmission in consistency mode 000: 0 delay 001: Random delay 0-8 010: Random delay 8-15 011: Random delay 16-31 100: Random delay 32-63 101: Random Delay 64-127 110: Random Delay 128-255 111: 0 delay
15	Crc Int en	1	0x0	R/W	Enable interrupt sending on CRC error
14:12	Crc Int route	3	0x0	R/W	Interrupt pin selection in case of CRC interrupt
11	Reserved				
10	ht int 8 bit	1	0x0	R/W	Use of 8 interrupted wires
9:8	ht_int_stripe	2	0x0	R/W	Corresponding to the 3 interrupt routing methods, described in the interrupt vector register 0x0: ht_int_stripe_1 0x1: ht_int_stripe_2 0x2: ht_int_stripe_4

Bit Field	Name	Length	Reset Value	Read/Write	Description
4:0	Interrupt Index	5	0x0	R/W	Which interrupt vector to redirect interrupts other than standard interrupts (including SMI, NMI, INIT, INTA, INTB, INTC, INTD). There are 256 interrupt vectors in total, this register represents the high 5 bits of the interrupt vector, and the internal interrupt vector is as follows. 000: SMI 001: NMI 010: INIT 011: Reservered 100: INTA 101: INTB 110: INTC 111: INTD

14.5.17. Receive Diagnostic Register

Offset: 0x1C8

Reset value: 0x00000000

Name: Receive diagnostic register

Table 164. Receive diagnostic register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	rx_cad_phase_0	16	0x0	R/W	Save the sampled value of input CAD[15:0]
15:8	rx_ctl_catch	8	0x0	R/W	Save the input ctl obtained by sampling. (0, 2, 4, 6) corresponds to the four phases sampled by CTL0. (1, 3, 5, 7) corresponds to the four phases sampled by CTL1
7:0					

14.5.18. PHY Status Register

Used to observe the PHY-related status and debug.

Offset: 0x1CC

Reset value: 0x83308000

Name: PHY status register

Table 165. PHY status register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:29	Reserved	3	0x0	R	Reserved
28	dll locked hi	1	0x0	R	High 8 bits - bit DLL locking
27	dll locked lo	1	0x0	R	Low 8 bits - bit DLL locking
26	cdr locked hi	1	0x0	R	High 8 - bit CDR locking
25	cdr locked lo	1	0x0	R	Low 8 bits - bit CDR lock
24	phase locked	1	0x0	R	Phase Lock
23:20	phy state	4	0x0	R	PHY Status
19:17	tx training status	3	0x0	R	TX training state
16:14	rx training status	3	0x0	R	RX training status
13:8	Init done	6	0x0	R	Initialization complete
7:0	Reserved	8		R	Reserved

14.5.19. Transport Command Cache Size Register

The Command Send Cache Size register is used to observe the number of caches available on the transmitter for each command channel.

Offset: 0x1D0

Reset value: 0x00000000

Name: Command send Cache size register

Table 166. Transport command Cache size register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:24	B_CMD_txbuffer	8	0x0	R	Number of B channel command Cache at the transmitter
23:16	R_CMD_txbuffer	8	0x0	R	Number of R channel command Cache at the sender
15:8	NPC_CMD_txbuffer	8	0x0	R	Number of sender NPC channel command Cache

Bit Field	Name	Length	Reset Value	Read/Write	Description
7:0	PC_CMD_txbuffer	8	0x0	R	Number of sender PC channel command Cache

14.5.20. Transport Data Cache Size Register

The data send Cache size register is used to observe the number of caches available on the transmitter for each data channel.

Offset: 0x1D4

Reset value: 0x00000000

Name: Data send buffer size register

Table 167. Transport data Cache size register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	Reserved	1	0x0	R	Reserved
30	rx_buffer_r_data[4]	1	0x0	R/W	Bit [4] of the read data buffer initialization information of the receive buffer
29	rx_buffer_npc_data[4]	1	0x0	R/W	Bit [4] of the npc data buffer initialization information of the receive buffer
28	rx_buffer_pc_data[4]	1	0x0	R/W	Bit [4] of the pc data buffer initialization information of the receive buffer
27	rx_buffer_b_cmd[4]	1	0x0	R/W	Bit [4] of the bresponse command buffer initialization information of the receive buffer
26	rx_buffer_r_cmd[4]	1	0x0	R/W	Bit [4] of the read command buffer initialization information of the receive buffer
25	rx_buffer_npc_cmd[4]	1	0x0	R/W	Bit [4] of the npc command buffer initialization information of the receive buffer
24	rx_buffer_pc_cmd[4]	1	0x0	R/W	Bit [4] of the initialization information of the pc command buffer of the receive buffer
23:16	R_DATA_txbuffer	8	0x0	R	Number of R channel data buffers at the transmitter
15:8	NPC_DATA_txbuffer	8	0x0	R	Number of NPC channel data buffers on the transmitter side
7:0	PC_DATA_txbuffer	8	0x0	R	Number of PC channel data buffer at the sender

14.5.21. Transport Cache Debug Register

The transmit buffer debug register is used to manually set the number of buffers on the transmit side of the HT controller and to adjust the number of different transmit buffers by increasing or decreasing.

Offset: 0x1D8

Reset value: 0x00000000

Name: Send Cache debug register

Table 168. Transport Cache debug register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	b_interleave	1	0x0	R/W	Consistent mode to enable interleaved transmission between channel B and other channels
30	nop_interleave	1	0x0	R/W	Enables interleaved transmission of flow control packets with other virtual channels
29	Tx_neg	1	0x0	R/W	Sender Cache debug symbols 0: increase the number of registers 1: decrease (the number of registers plus 1)
28	Tx_buff_adj_en	1	0x0	R/W	Sender Cache debug enable register 0 → 1: the value of this register will be incremented or decremented once
27:24	R_DATA_txadj	4	0x0	R/W	Increase or decrease the number of R channel data Cache at the transmitter. When tx_neg is 0, increase the number of R_DATA_txadj. When tx_neg is 1, decrease R_DATA_txadj+1

Bit Field	Name	Length	Reset Value	Read/Write	Description
23:20	NPC_DATA_txadj	4	0x0	R/W	Increase or decrease the number of R channel data Cache at the transmitter. When tx_neg is 0, increase the number of R_DATA_txadj. When tx_neg is 1, decrease R_DATA_txadj + 1 number of NPC channel data Cache increment/decrement on the transmitter side
19:16	PC_DATA_txadj	4	0x0	R/W	Increase or decrease the number of PC channel data Cache at the transmitter. When tx_neg is 0, increase the number of PC_DATA_txadj. When tx_neg is 1, decrease PC_DATA_txadj+1
15:12	B_CMD_txadj	4	0x0	R/W	Increase or decrease the number of B channel command Cache at the transmitter. When tx_neg is 0, increase the number of B_CMD_txadj. When tx_neg is 1, decrease B_CMD_txadj+1
11:8	R_CMD_txadj	4	0x0	R/W	Increase or decrease the number of R channel command Cache on the transmitter side. When tx_neg is 0, increase the number of R_CMD_txadj. When tx_neg is 1, decrease R_CMD_txadj+1
7:4	NPC_CMD_txadj	4	0x0	R/W	Increase or decrease the number of NPC channel command/data Cache on the transmitter side. When tx_neg is 0, increase the number of NPC_CMD_txadj. When tx_neg is 1, decrease NPC_CMD_txadj+1

Bit Field	Name	Length	Reset Value	Read/Write	Description
3:0	PC_CMD_txadj	4	0x0	R/W	Increase or decrease the number of PC channel command Cache on the transmitter side. When tx_neg is 0, increase the number of PC_CMD_txadj. When tx_neg is 1, decrease PC_CMD_txadj+1

14.5.22. Receive Buffer Initialization Configuration Register

Offset: 0x1DC

Reset value: 0x07778888

Name: Receive buffer initialization configuration register

Table 169. Receive buffer initialization configuration register

Bit Field	Name	Length	Reset Value	Read/Write	Description
27:24	rx_buffer_r_data	4	0x0	R/W	Read data buffer initialization information of the receive buffer
23:20	rx_buffer_npc_data	4	0x0	R/W	Npc data buffer initialization information of the receive buffer
19:16	rx_buffer_pc_data	4	0x0	R/W	Receive buffer initialization information for pc data buffer
15:12	rx_buffer_b_cmd	4	0x0	R/W	Receive the bresponse command buffer initialization information of the buffer
11:8	rx_buffer_r_cmd	4	0x0	R/W	Receive the read command buffer initialization information of the buffer
7:4	rx_buffer_npc_cmd	4	0x0	R/W	Npc command buffer initialization information of the receive buffer
3:0	rx_buffer_pc_cmd	4	0x0	R/W	Receive the pc command buffer initialization information of the buffer

14.5.23. Training 0 Timeout Short Counter Register

Used to configure the Training 0 short timing timeout threshold in HyerTransport 3.0 mode, with a counter clock frequency of 1/4 of the HyperTransport 3.0 link bus clock frequency.

Offset: 0x1E0

Reset value: 0x00000080

Name: Training 0 timeout short counter register

Table 170. Training 0 timeout short counter register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	Gen3_timing_soft	1	0x0	R/W	
30:23	Retry_nop_num	8	0x0	R/W	
22:0	T0 time	23	0x80	R/W	Training 0 timeout short counter register

14.5.24. Training 0 Timeout Long Counter Register

Used in HyerTransport 3.0 mode Training 0 long count timeout threshold with counter clock frequency 1/4 of HyperTransport 3.0 link bus clock frequency.

Offset: 0x1E4

Reset value: 0x000fffff

Name: Training 0 timeout count register

Table 171. Training 0 timeout long counter register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	T0 time	32	0xfffff	R/W	Training 0 timeout long counter register

14.5.25. Training 1 Counter Register

For HyerTransport 3.0 mode Training 1 count threshold, the counter clock frequency is 1/4 of the HyperTransport 3.0 link bus clock frequency.

Offset: 0x1E8

Reset value: 0x0004fffff

Name: Training 1 counter register

Table 172. Training 1 counter register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	T1 time	32	0x4fffff	R/W	Training 1 counter register

14.5.26. Training 2 Counter Register

For the Training 2 count threshold in HyerTransport 3.0 mode, the counter clock frequency is 1/4 of the HyperTransport 3.0 link bus clock frequency.

Offset: 0x1EC

Reset value: 0x0007fffff

Name: Training 2 counter register

Table 173. Training 2 counter register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	T2 time	32	0x7fffff	R/W	Training 2 counter register

14.5.27. Training 3 Counter Register

For the Training 3 count threshold in HyerTransport 3.0 mode, the counter clock frequency is 1/4 of the HyperTransport 3.0 link bus clock frequency.

Offset: 0x1F0

Name: Training 3 counter register

Table 174. Training 3 counter register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	T3 time	32	0x7fffff	R/W	Training 3 counter register

14.5.28. Software Frequency Configuration Register

The controller is used to switch to the link frequency and controller frequency supported by any protocol and PLL during operation; the specific switching method is: under the premise of enabling software configuration mode, set bit 1 of the software frequency configuration register and write the new clock-related parameters, including div_refc and div_loop, which determine the PLL output frequency, the link div_hi_div and phy_lo_div, and the controller's divide factor core_div. The controller will automatically reset the PLL and configure the new clock parameters by entering warm reset or LDT disconnect.

PHY_LINK_CLK is the HT bus frequency. The clock frequency is calculated by the formula:

When using SYS_CLOCK as the reference clock input and SYS_CLOCK is 25MHz (CLKSEL[8] is 1 and CLKSEL[5] is 1), the frequency is calculated as follows:

HyperTransport 1.0:

```
PHY_LINK_CLK = 12.5MHz × div_loop / div_refc / phy_div
```

HyperTransport 3.0:

```
PHY_LINK_CLK = 25MHz × div_loop / div_refc / phy_div
```

In other cases, the frequency is calculated as:

HyperTransport 1.0:

```
PHY_LINK_CLK = 50MHz × div_loop / div_refc / phy_div
```

HyperTransport 3.0:

```
PHY_LINK_CLK = 100MHz × div_loop / div_refc / phy_div
```

The wait time for PLL re-lock is approximately 30us by default with system clk at 33M; a custom wait count limit can also be written into the register.

Note that in the 3A5000, HT_CORE_CLK is no longer controlled by this configuration, but by the NODE clock divider.

Offset: 0x1F4

Reset value: 0x00000000

Name: Software frequency configuration register

Table 175. Software frequency configuration register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:27	PLL relock counter	5	0x0	R/W	Counter limit configuration register, when set counter select, the counter limit is {PLL_relock_counter ,5'h1f}, otherwise the count limit is 10'3ff
26	Counter select	1	0x0	R/W	Lock Timer Custom Enable. 1'b0 - using the default count limit 1'b1 - calculated by PLL_relock_counter
25	Reserved	1	0x0	R	Reserved
24:16	Soft_div_loop	7	0x0	R/W	PLL internal frequency multiplication factor
15:12	Soft_div_refc	4	0x0	R/W	PLL internal dividing factor
11:8	Soft_phy_hi_div	4	0x0	R/W	High PHY frequency division factor
7:4	Soft_phy_lo_div	4	0x0	R/W	Low PHY frequency division factor
3	Locked	1	0x0	R	Lock Flag
2	Reserved	1	0x0	R	Reserved
1	Soft cofig enable	1	0x0	R/W	Software configuration enable bit. 1'b0 - disable software frequency configuration 1'b1 - enables software frequency configuration
0	Reserved	1	0x0	R	Reserved

14.5.29. PHY Impedance Matching Control Register

Used to control the impedance matching enable of the PHY and the impedance matching parameter setting of the transmitter and receiver.

Offset: 0x1F8

Reset value: 0x00000000

Name: PHY impedance matching control register

Table 176. PHY impedance matching control register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	Tx_scanin_en	1	0x0	R/W	TX impedance matching enable
30	Rx_scanin_en	1	0x0	R/W	RX impedance matching enable
27:24	Tx_scanin_ncode	4	0x0	R/W	TX impedance matching scan input ncode
23:20	Tx_scanin_pcode	4	0x0	R/W	TX impedance matching scan input pcode
19:12	Rx_scanin_code	8	0x0	R/W	RX impedance matching scan input

14.5.30. PHY Configuration Register

Used to configure PHY related physical parameters, when the controller is two independent 8-bit controllers, the high PHY and low PHY are controlled independently by the two controllers. When the controller is one 16-bit controller, the configuration parameters of the high and low PHY are unified by the low controller.

Offset: 0x1FC

Reset value: 0x83308000

Name: PHY configuration register

Table 177. PHY configuration register

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	Rx_ckpll_term	1	0x1	R/W	PLL to RX side on-Chip transmission line termination impedance
30	Tx_ckpll_term	1	0x0	R/W	PLL to TX side on-Chip transmission line termination impedance
29	Rx_clk_in_sel_	1	0x0	R/W	Clock PAD supply the clock selection for the data PAD, automatically selected as CLKPAD in HT1 mode: 1'b0 - external clock source 1'b1 - PLL clock

Bit Field	Name	Length	Reset Value	Read/Write	Description
28	Rx_ckdll_sell	1	0x0	R/W	Clock selection used to lock the DLL. 1'b0 - PLL clock 1'b1 - external clock source
27:26	Rx_ctle_bitc	2	0x0	R/W	PAD EQD high frequency gain
25:24	Rx_ctle_bitr	2	0x3	R/W	PAD EQD low frequency gain
23:22	Rx_ctle_bitlim	2	0x0	R/W	PAD EQD compensation limit
21	Rx_en_ldo	1	0x1	R/W	LDO Control 1'b0 - LDO disable 1'b1 - LDO enable
20	Rx_en_by	1	0x1	R/W	BandGap control 1'b0 - bandGap disable 1'b1 - bandGap enable
19:17	Reserved	3	0x0	R	Reserved
16:12	Tx_preenmp	5	0x08	R/W	PAD pre-emphasis control signal
11:0	Reserved	12	0x0	R	Reserved

14.5.31. Link Initialization Debug Register

Used to configure whether to use the CDRlock signal provided by the PHY as the link CDR completion flag during link initialization in HyperTransport 3.0 mode. If the lock signal is ignored, the controller is required to count and wait for a certain amount of time before the default CDR is completed.

Offset: 0x240

Reset value: 0x00000000

Name: Link initialization debug register

Table 178. Link initialization debug register

Bit Field	Name	Length	Reset Value	Read/Write	Description
15	Cdr_ignore_enable	1	0x0	R/W	Whether to ignore CRC lock during link initialization and wait for completion by counter count. 1'b0 - wait for CDR lock 1'b1 - ignore CDR lock signal and wait by counter accumulation

Bit Field	Name	Length	Reset Value	Read/Write	Description
14:0	Cdr_wait_counter	15	0x0	R/W	Wait for the counter count limit and finish counting based on the controller clock

14.5.32. LDT Debug Register

Software changes to the controller frequency will result in inaccurate timing of the LDT reconnect phase. The counter needs to be configured as the time between the invalidation of the LDT signal and the start of link initialization of the controller after the software configuration of the frequency, which is based on the controller clock.

Offset: 0x244

Reset value: 0x00000000

Name: LDT Debug register 1

Table 179. LDT debug register 1

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	Rx_wait_time	16	0x0	R/W	The RX side waits for the initial value of the counter
15:0	Tx_wait_time	16	0x0	R/W	The TX side waits for the initial value of the counter

Offset: 0x248

Reset value: 0x00000000

Name: LDT Debug register 2

Table 180. LDT debug register 2

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:30	Reserved	16	0x0	R/W	
29:0	rx lane ts 0	16	0x0	R/W	

Offset: 0x24C

Reset value: 0x00000000

Name: LDT Debug register 3

Table 181. LDT debug register 3

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:30	Reserved	16	0x0	R/W	
29:0	rx lane ts 1	16	0x0	R/W	

Offset: 0x250

Reset value: 0x00000000

Name: LDT Debug register 4

Table 182. LDT debug register 4

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:30	Reserved	16	0x0	R/W	
29:0	rx lane ts 2	16	0x0	R/W	

Offset: 0x254

Reset value: 0x00000000

Name: LDT Debug register 5

Table 183. LDT debug register 5

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:22	Reserved	10	0x0	R/W	
21:18	wait ctl	4	0x0	R/W	
17:0	phase lock	18	0x0	R/W	

Offset: 0x258

Reset value: 0x00000000

Name: LDT Debug register 6

Table 184. LDT debug register 6

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:0	wait cad	32	0x0	R/W	

14.5.33. HT TX POST ID Window Configuration Register

This window sends hit requests outbound through the HT POST channel by comparing the ID of the internal write request to a pre-defined window.

Offset: 0x260

Reset value: 0x00000000

Name: HT TX POST ID WIN0

Table 185. HT TX POST ID WIN0

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	HT TX POST ID0 MASK	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the MASK bit of the ID
15:0	HT TX POST ID0 BASE	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the BASE bit of the ID

Offset: 0x264

Reset value: 0x00000000

Name: HT TX POST ID WIN1

Table 186. HT TX POST ID WIN1

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	HT TX POST ID1 MASK	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the MASK bit of the ID
15:0	HT TX POST ID1 BASE	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the BASE bit of the ID

Offset: 0x268

Reset value: 0x00000000

Name: HT TX POST ID WIN2

Table 187. HT TX POST ID WIN2

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	HT TX POST ID2 MASK	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the MASK bit of the ID
15:0	HT TX POST ID2 BASE	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the BASE bit of the ID

Offset: 0x26C

Reset value: 0x00000000

Name: HT TX POST ID WIN3

Table 188. HT TX POST ID WIN3

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:16	HT TX POST ID3 MASK	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the MASK bit of the ID
15:0	HT TX POST ID3 BASE	16	0x0	R/W	AXI ID hit requests are transmitted using the POST window, the BASE bit of the ID

14.5.34. External Interrupt Conversion Configuration

This setting converts an interrupt received by the HT into a write operation to a specific address that is written directly to the extended I/O interrupt vector inside the chip, rather than generating an interrupt inside the HT controller. With this approach, advanced features such as direct cross-chip distribution of I/O interrupts can be used.

Offset: 0x270

Reset value: 0x00000000

Name: HT RX INT TRANS Lo

Table 189. HT RX INT TRANS Lo

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:4	INT_trans_addr[31:4]	28	0x0	R/W	Low order bits of interrupt translation address
3:0	Reserved	4	0x0	R	Reserved

Offset: 0x274

Reset value: 0x00000000

Name: HT RX INT TRANS Hi

Table 190. HT RX INT TRANS Hi

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	INT_trans_en	1	0x0	R/W	Interrupt transition enable
30	INT_trans_allow	1	0x0	R/W	Whether to allow interrupt transition This bit is set for INT_trans_en or EXT_INT_en of the chip to take effect
29:26	INT_trans_cache	4	0x0	R/W	Interrupt Transition Cache Field
25:0	INT_trans_addr[58:32]	26	0x0	R/W	High order bits of interrupt translation address

14.6. Access to HyperTransport Bus Configuration Space

The protocol for the HyperTransport interface software layer is basically the same as the PCI protocol, with slightly different specific access details as the access to the configuration space is directly related to the underlying protocol. As listed in Address window distribution inside the HyperTransport interface of the Loongson 3 processor, the address range of the HT bus configuration space is `0xFD_FE00_0000` to `0xFD_FFFF_FFFF`. For the configuration access in the HT protocol, the following format is used in the Loongson 3A5000:

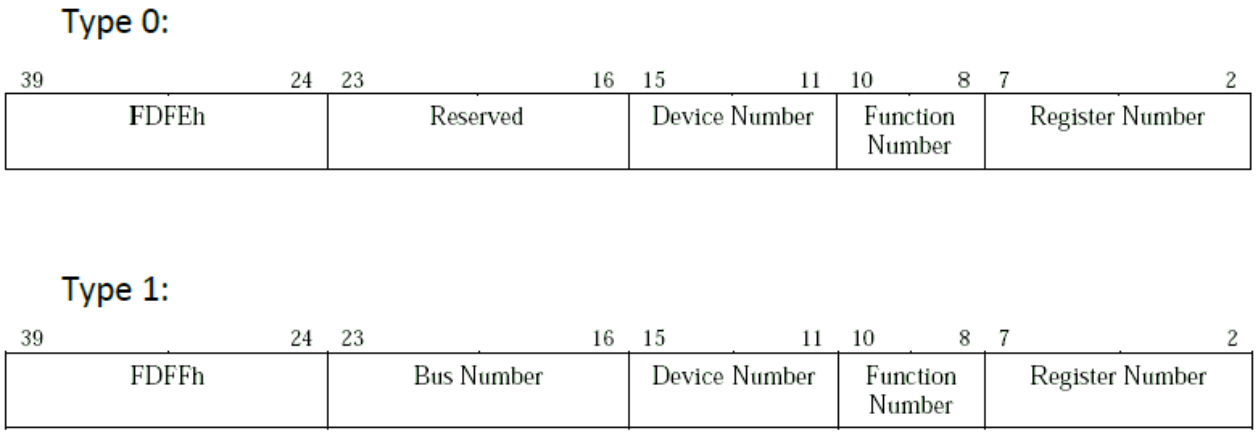


Figure 9. Access to the HT protocol configuration in the Loongson 3A5000

14.7. HyperTransport Multi-processor Support

The loongson3 processor uses the HyperTransport interface for multiprocessor interconnects and can automatically maintain consistency requests between 2-8 chips in hardware.

Loongson 3 Interconnect Routing

Loongson 3 interconnect routing has two methods, one is to use the simple X-Y routing method. If a request is sent from `11` to `00`, it is routed from `11` to `00`, first in the `X` direction, from `11` to `10`, and then in the `Y` direction, from `10` to `00`. When its response returns from `00` to `11`, it is routed first in the `X` direction, from `00` to `01`, and then in the `Y` direction, from `01` to `11`. from `00` to `01`, and then the `Y` direction, from `01` to `11`. The other is diagonal direct access, which is achieved by connecting two diagonal chips in hardware to greatly reduce access latency, and this access requires separate enablement through software. Due to the characteristics of this algorithm, a variety of different approaches can be used when building multi-chip interconnects.

Structure of Four Loongson 3 Chips Interconnected

The four CPUs are connected in a two-by-two ring structure. Each CPU is connected to two adjacent chips using the two 8-bit controllers of `HT0` and to the diagonal chip using `HT1  HI`, resulting in the interconnection structure shown below:

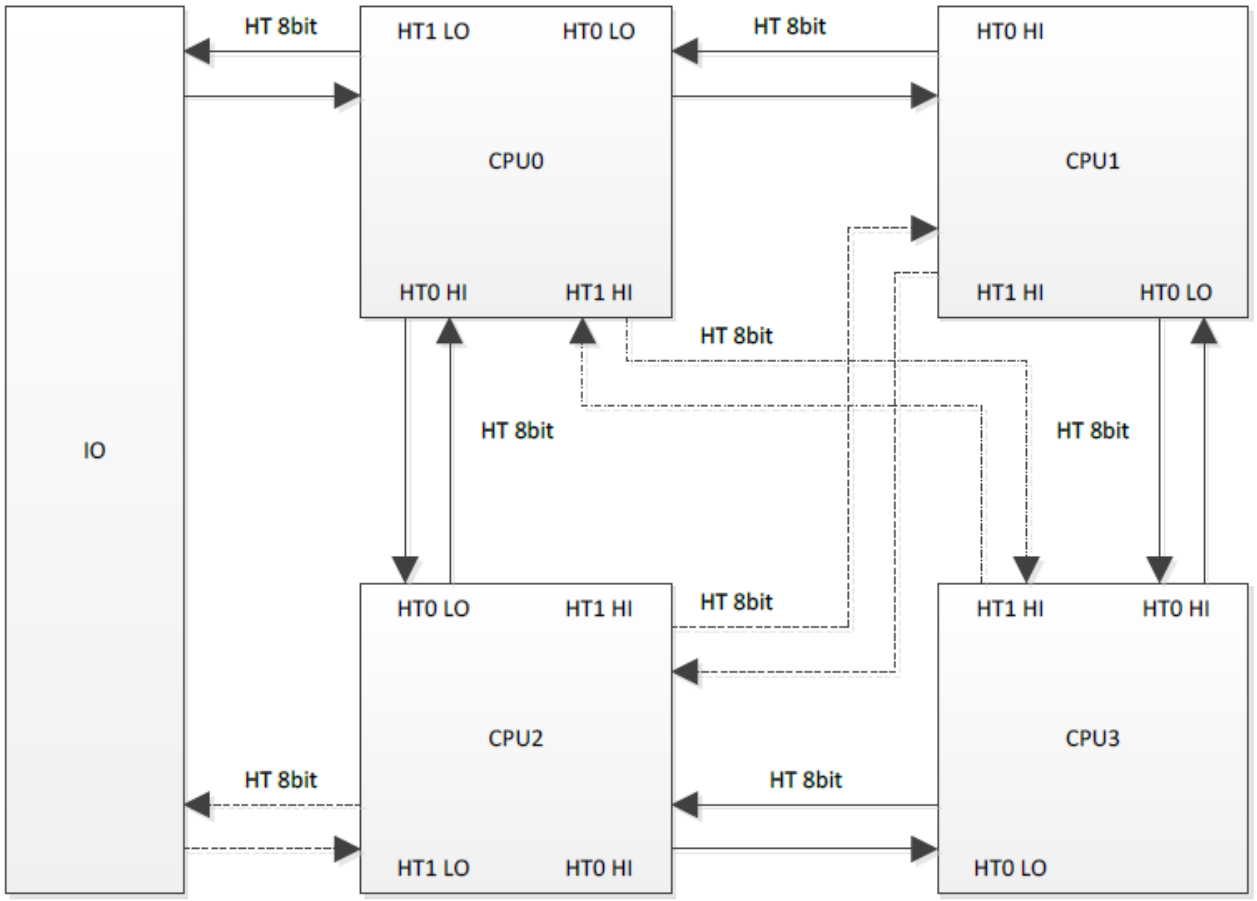


Figure 10. Structure of four Loongson 3 chips interconnected

Structure of Sixteen Loongson 3 Chips Interconnected

The sixteen interconnects use the remaining HT1_LO after the above four interconnects (called Clusters) for interconnecting between Clusters. The structure is as follows:

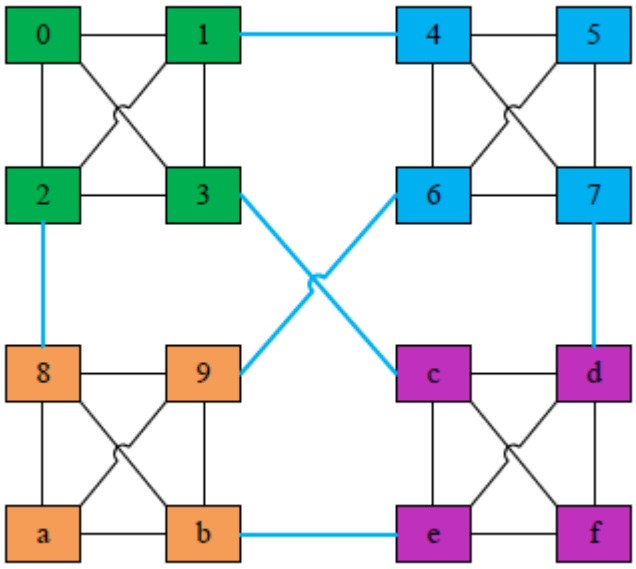


Figure 11. Structure of sixteen Loongson 3 chips interconnected

Structure of Two Loongson 3 Chips with 8-bit Interconnection

8-bit HT bus interconnect. In this interconnect, only 8-bit HT interconnects can be used between the two processors. The two chip numbers are `00` and `01` respectively, and from the routing algorithm, it can be known that both chips access each other through the same 8-bit HT bus as in the four-chip interconnect. This is shown below:

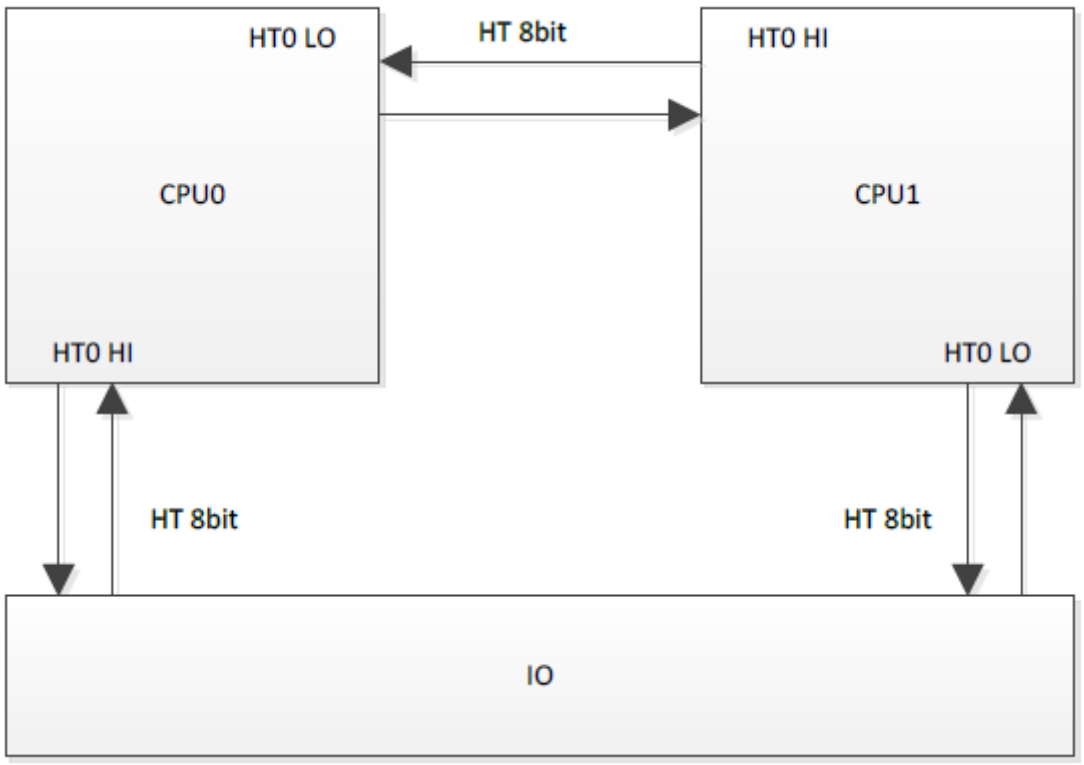


Figure 12. Structure of two Loongson 3 chips with 8-bit interconnection

However, the HT bus can be used in 16-bit mode at the widest, and the resulting connection method to maximize bandwidth should be to use a 16-bit interconnect structure. In Loongson 3, as long as the `HT0` controller is set to 16-bit mode, all commands sent to the HT0 controller will be sent to `HT0_LO` instead of to `HT0_HI` or `HT0_LO` respectively according to the routing table as before, so that the 16-bit bus for interconnection can be used. So, only need to configure the 16-bit mode of `CPU0` and `CPU1` correctly and connect the high and low buses correctly to use the 16-bit HT bus interconnect. This interconnection structure can also be accessed using the 8-bit HT bus protocol. The resulting interconnection structure is as follows:

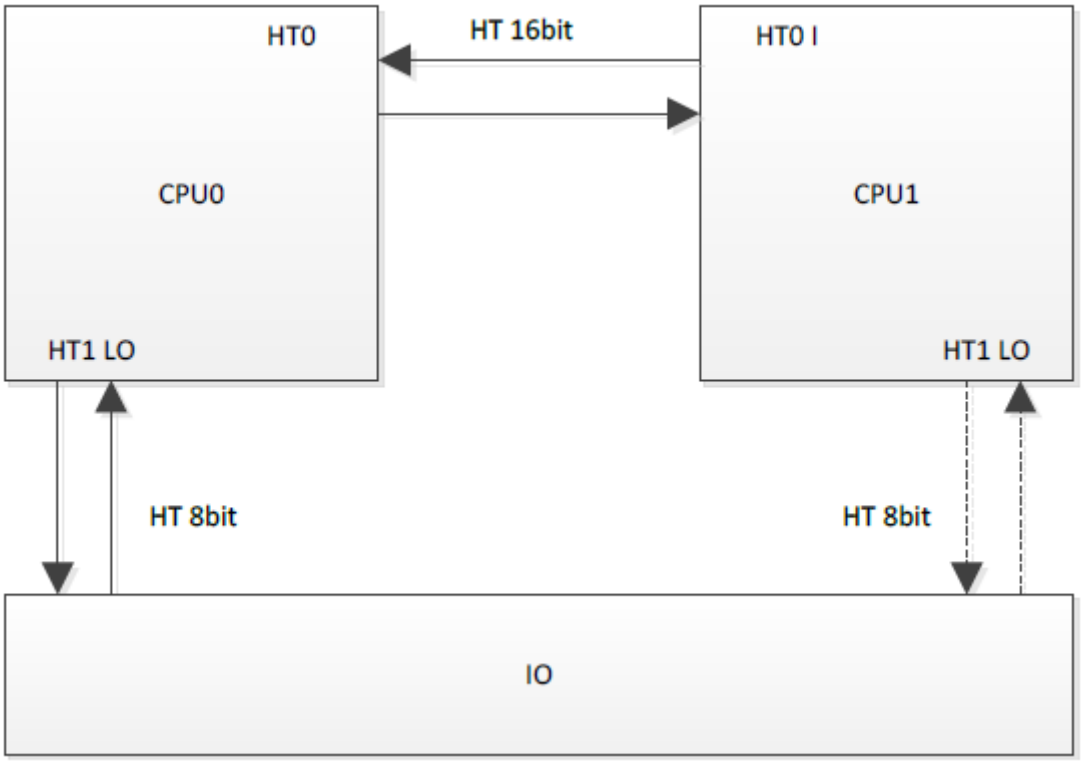


Figure 13. Structure of two Loongson 3 chips with 16-bit interconnection

Chapter 15. Low-speed I/O Controller Configuration

The Loongson 3 I/O controllers include a UART controller, SPI controller, I2C and GPIO registers. These I/O controllers share an AXI port, and CPU requests are sent to the corresponding devices after address decoding.

15.1. UART Controller

The UART controller has the following features

- Full duplex asynchronous data receive/send
- Programmable data format
- 16-bit programmable clock counter
- Support for receive timeout detection
- Multi-interrupt system with arbitration
- FIFO-only operation
- Compatible with NS16550A in terms of registers and functions

Two UART interfaces are integrated inside the chip, and the functional registers are exactly the same, only the access base address is different. UART0 register physical address base address is `0x1FE001E0`.

UART1 register physical address base address is `0x1FE001E8`.

A physical address is also provided for each of the two UARTs, `0x1FE00100` (`UART0`) and `0x1FE00110` (`UART1`) respectively. The two additional registers `RFC` and `TFC` can be accessed through this set of addresses.

15.1.1. Data Transport Register (DAT)

Name: Data transport register

Length: `[7:0]`

Offset: `0x00`

Reset value: `0x00`

Table 191. Data transport register

Bit Field	Name	Length	Read/Write	Description
`7:0`	`Tx FIFO`	`8`	W	Data transport register

15.1.2. Interrupt Enable Register (IER)

Name: Interrupt enable register

Length: `[7:0]`

Offset: `0x01`

Reset value: 0x00

Table 192. Interrupt enable register

Bit Field	Name	Length	Read/Write	Description
7:4	Reserved	4	RW	Reserved
3	IME	1	RW	Modem status interrupt enable 0 - off, 1 - on
2	ILE	1	RW	Receiver line status interrupt enable 0 - off, 1 - on
1	ITxE	1	RW	Transport save register empty interrupt enable 0 - off, 1 - on
0	IRxE	1	RW	Receive valid data interrupt enable 0 - off, 1 - on

15.1.3. Interrupt Identity Register (IIR)

Name: Interrupt source register

Length: [7:0]

Offset: 0x02

Reset value: 0xc1

Table 193. Interrupt identity register

Bit Field	Name	Length	Read/Write	Description
7:4	Reserved	4	R	Reserved
3:1	II	3	R	Bits to indicate the interrupt source, as detailed in the following table
0	INTp	1	R	Bits to indicate the interrupt

Table 194. Table of interrupt control function

Bit 3	Bit 2	Bit 1	Priority	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	1st	Receive line status	Odd or even, overflow, frame errors, or interrupt interruptions	Read LSR
0	1	0	2nd	Valid data received	The number of characters in the FIFO reaches the level of a trigger	The number of characters in the FIFO is lower than the trigger value

Bit 3	Bit 2	Bit 1	Priority	Interrupt Type	Interrupt Source	Interrupt Reset Control
1	1	0	2nd	Receive timeout	At least one character in the FIFO, but no operations, including read and write operations, during the 4-character transport time	Read receive FIFO
0	0	1	3rd	Transport save register is empty	Transport save register is empty	Write data to THR or read IIR
0	0	0	4th	Modem status	CTS, DSR, RI or DCD	Read MSR

15.1.4. FIFO Control Register (FCR)

Name: FIFO control register

Length: [7:0]

Offset: 0x02

Reset value: 0xc0

Table 195. FIFO control register

Bit Field	Name	Length	Read/Write	Description
7:6	TL	2	W	Receive trigger value for interrupt request from FIFO 00 - 1 byte 01 - 4 byte 10 - 8 byte 11 - 14 byte
5:3	Reserved	3	W	Reserved
2	Txset	1	W	1 indicates that clearing the contents of the transport FIFO and resetting its logic
1	Rxset	1	W	1 indicates that clearing the contents of the receiving FIFO and resetting its logic
0	Reserved	1	W	Reserved

15.1.5. Line Control Register (LCR)

Name: Line control register

Length: [7:0]

Offset: 0x03

Reset value: 0x03

Table 196. Line control register

Bit Field	Name	Length	Read/Write	Description
7	dlab	1	RW	Frequency division latch access bit 1 - access to the operating frequency divider latch 0 - access to the operation normal register
6	bcb	1	RW	Interrupt control bit 1 - at this time the output of the serial port is set to 0 (interrupted state) 0 - normal operation
5	spb	1	RW	Specify parity bit 0 - No need to specify parity bit 1 - If the LCR[4] bit is 1 then the transport and parity bit is 0. If the LCR[4] bit is 0 then the transport and parity bit is 1
4	eps	1	RW	Parity bit selection 0 - Odd number of 1s in each character (including data and the parity bit) 1 - Even number of 1s in each character
3	pe	1	RW	Parity bits enable 0 - no parity bits 1 - generate parity bits on output, and determine parity bits on input
2	sb	1	RW	Define the number of bits to generate the stop bit 0 - 1 stop bit 1 - 1.5 stop bits at 5-bit character length, 2 stop bits at other lengths

Bit Field	Name	Length	Read/Write	Description
1:0	bec	2	RW	Set the number of bits per character 00 - 5 bits 01 - 6 bits 10 - 7 bits 11 - 8 bits

15.1.6. MODEM Control Register (MCR)

Name: Modem control register

Length: [7:0]

Offset: 0x04

Reset value: 0x00

Table 197. MODEM Control Register

Bit Field	Name	Length	Read/Write	Description
7:5	Reserved	3	W	Reserved
4	Loop	1	W	Loopback mode control bit 0 - normal operation 1 - Loopback mode. In the loopback mode, the TXD output is always 1 and the output shift register is directly connected to the input shift register. Other connections are listed below. DTR → DSR RTS → CTS Out1 → RI Out2 → DCD
3	OUT2	1	W	Connect to DCD input in loopback mode
2	OUT1	1	W	Connect to RI input in loopback mode
1	RTSC	1	W	RTS signal control bit
0	DTRC	1	W	DTR signal control bit

15.1.7. Line State Register (LSR)

Name: Line status register

Length: [7:0]

Offset: 0x05

Reset value: 0x00

Bit Field	Name	Length	Read/Write	Description
7	ERROR	1	R	Error indication bit 1 - at least one of parity bit error, frame error or interrupted interrupt 0 - no errors
6	TE	1	R	Transport empty indication bit 1 - both the transport FIFO and the transport shift register are empty. Clear when writing data to the transport FIFO 0 - data exists
5	TFE	1	R	Transport FIFO bit empty indication bit 1 - the current transport FIFO is empty and is cleared when writing data to the transport FIFO 0 - data exists
4	BI	1	R	interrupted interrupt indication bit 1 - when received start bit + data + parity bit + stop bit are all 0, there is an interrupted interrupt 0 - the interrupt is not interrupted
3	FE	1	R	Frame error indication bit 1 - the received data has no stop bit 0 - no errors
2	PE	1	R	Parity bit error indication bit 1 - current received data has parity error 0 - no parity errors

Bit Field	Name	Length	Read/Write	Description
1	OE	1	R	Data overflow indication bit 1 - data overflow exists 0 - no overflow
0	DR	1	R	Receive data valid indication bit 0 - no data in the FIFO 1 - data exists in the FIFO

When reading this register, LSR[4:1] and LSR[7] are cleared to zero, LSR[6:5] is cleared when writing data to the transmit FIFO, and LSR[0] is judged for the receive FIFO.

15.1.8. MODEM State Register (MSR)

Name: Modem status register

Length: [7:0]

Offset: 0x06

Reset value: 0x00

Table 198. MODEM state register

Bit Field	Name	Length	Read/Write	Description
7	CDCD	1	R	The inverse of the DCD input, or connected to Out2 in loopback mode
6	CRI	1	R	The inverse of the RI input, or connected to OUT1 in loopback mode
5	CDSR	1	R	The inverse of the DSR input, or connected to DTR in loopback mode
4	CCTS	1	R	The inverse of the CTS input, or connected to the RTS in loopback mode
3	DDCD	1	R	DDCD indicator bit
2	TERI	1	R	RI edge detection, RI state changes from low to high
1	DDSR	1	R	DDSR indicator bit
0	DCTS	1	R	DCTS indicator bit

15.1.9. Receive FIFO Counter (RFC)

Name: Receive FIFO count value

Length: [7:0]

Offset: 0x08

Reset value: 0x00

Table 199. Receive FIFO counter

Bit Field	Name	Length	Read/Write	Description
7:0	RFC	8	R	Reflects the number of valid data in the current received FIFO

15.1.10. Transport FIFO Counter (TFC)

Name: Transport FIFO Count value

Length: [7:0]

Offset: 0x09

Reset value: 0x00

Table 200. Transport FIFO counter

Bit Field	Name	Length	Read/Write	Description
7:0	RFC	8	R	Reflects the number of valid data in the current transport FIFO

15.1.11. Frequency Division Latchs

Name: Frequency Divider Latch 1

Length: [7:0]

Offset: 0x00

Reset value: 0x00

Table 201. Frequency division latchs 1

Bit Field	Name	Length	Read/Write	Description
7:0	LSB	8	RW	Store the lower 8 bits of the division latch

Name: Frequency Divider Latch 2

Length: [7:0]

Offset: 0x01

Reset value: 0x00

Table 202. Frequency division latchs 2

Bit Field	Name	Length	Read/Write	Description
7:0	MSB	8	RW	Store the higher 8 bits of the division latch

Name: frequency divider latch 3

Length: [7:0]

Offset: 0x02

Reset value: 0x00

Table 203. Frequency division latchs 3

Bit Field	Name	Length	Read/Write	Description
7:0	D_DIV	8	RW	Store the decimal division value of the division latch

15.1.12. Use of New Registers

The new receive FIFO counter (RFC) allows the CPU to detect the number of valid data in the Receive FIFO, so that the CPU can read multiple data continuously after receiving an interrupt, improving the CPU's ability to process UART received data.

Transport FIFO Counter (TFC) for the CPU to detect the number of valid data in the transport FIFO, whereby the CPU can continuously send multiple data while ensuring that the transport FIFO does not overflow, improving the CPU's ability to process UART transport data.

Frequency divider latch 3 (i.e. fractional divider register) is used to solve the problem that the required baud rate cannot be obtained accurately by dividing by integers only. The integer part of the quotient is assigned to the MSB and LSB by the divider latch, and the fractional part is assigned to the divider latch D_DIV by multiplying by 256.

15.2. SPI Controller

The SPI controller has the following features:

- Full-duplex synchronous serial data transmission
- Variable length byte transport support up to 4
- Master mode support
- Mode failure generates an error flag and issues an interrupt request
- Dual buffered receivers
- Polarity and phase programmable serial clock
- SPI can be controlled in wait mode
- Boot from SPI support
- Dual/Quad mode SPI FLASH support

The SPI controller register physical address base address is 0x1FE001F0.

Address Name	Address Range	Size
SPI Boot	0X1FC0_0000-0X1FD0_0000	1MByte
SPI Memory	0X1D00_0000-0X1E00_0000	16MByte
SPI Register	0X1FE0_01F0-0X1FE0_01FF	16Byte

The SPI Boot address space is the first address space accessed by the processor when the system boots up, and the 0xBFC00000 address is automatically routed to the SPI.

The SPI Memory space can also be accessed directly through a read request from the CPU, and its minimum 1M bytes overlap with the SPI BOOT space.

15.2.1. Control Register (SPCR)

Name: SPI FLASH control register

Length: [7:0]

Offset: 0x00

Reset value: 0x10

Table 204. Control register

Bit Field	Name	Length	Read/Write	Description
7	Spie	1	RW	Interrupt output enable signal (active high)
6	spe	1	RW	System operation enable signal (active high)
5	Reserved	1	RW	Reserved
4	mstr	1	RW	Master mode select bit, this bit is always 1
3	cpol	1	RW	Clock polarity bits
2	cpha	1	RW	Clock phase bit of 1 indicates opposite phase, 0 indicates same phase
1:0	spr	2	RW	sclk_o frequency division setting, need to be used with sper's `spre

15.2.2. State Register (SPSR)

Name: Status register

Length: [7:0]

Offset: 0x01

Reset value: 0x05

Table 205. State register

Bit Field	Name	Length	Read/Write	Description
7	spif	1	RW	Interrupt flag bit. 1 indicates an interrupt request and write 1 to clear
6	wcol	1	RW	Write register overflow flag bit. 1 indicates overflow, and write 1 to clear
5:4	Reserved	2	RW	Reserved
3	wffull	1	RW	Write register full flag. 1 indicates full
2	wfempty	1	RW	Write register empty flag. 1 indicates empty
1	rffull	1	RW	Read register full flag. 1 indicates full
0	rfempty	1	RW	Read register empty flag. 1 indicates empty

15.2.3. Transport Data Register (TxFIFO)

Name: Transport data register

Length: [7:0]

Offset: 0x02

Reset value: 0x00

Table 206. Transport data register

Bit Field	Name	Length	Read/Write	Description
7:0	Tx FIFO	8	W	Transport data register

15.2.4. External Register (SPER)

Name: External register

Length: [7:0]

Offset: 0x03

Reset value: 0x00

Table 207. External register

Bit Field	Name	Length	Read/Write	Description
7:6	icnt	2	RW	Send an interrupt request signal after how many bytes have been transported 00 - 1 byte 01 - 2 bytes 10 - 3 bytes 11 - 3 bytes
5:2	Reserved	4	RW	Reserved
1:0	spre	2	RW	Set the ratio of the frequency division together with spr

Table 208. Frequency division factor

spre	00	00	00	00	01	01	01	01	10	10	10	10
spr	00	01	10	11	00	01	10	11	00	01	10	11
Frequency division factor	2	4	16	32	8	64	128	256	512	1024	2048	4096

15.2.5. Parameter Control Register (SFC_PARAM)

Name: SPI FLASH parameter control register

Length: [7:0]

Offset: 0x04

Reset value: 0x21

Table 209. Parameter control register

Bit Field	Name	Length	Read/Write	Description
7:4	clk_div	4	RW	Clock division number selection (the division factor is the same as the {spre, spr} combination)
3	dual_io	1	RW	Use dual I/O mode with higher priority than fast read mode
2	fast_read	1	RW	Use fast read mode
1	burst_en	1	RW	SPI FLASH supports continuous address read mode
0	memory_en	1	RW	SPI FLASH read enable, when invalid csn[0] can be controlled by software

15.2.6. Chip Select Control Register (SFC_SOFTCS)

Name: SPI FLASH chip select control register

Length: [7:0]

Offset: 0x05

Reset value: 0x00

Table 210. Chip select control register

Bit Field	Name	Length	Read/Write	Description
7:4	csn	4	RW	csn pin output
3:0	csen	4	RW	The cs line corresponding to the bit with 1 is controlled by the [7:4] bits

15.2.7. Timing Control Register (SFC_TIMING)

Name: SPI FLASH timing control register

Length: [7:0]

Offset: 0x06

Reset value: 0x03

Table 211. Timing control register

Bit Field	Name	Length	Read/Write	Description
7:4	Reserved	4	RW	Reserved
3	quad_io	1	RW	4-wire mode enable. 1 indicates valid
2	tFast	1	RW	
1:0	tCSH	2	RW	The minimum invalidation time of the SPI FLASH chip select signal, calculated as the clock period T after frequency division 00: 1T 01: 2T 10: 4T 11: 8T

15.2.8. Custom Controller Register (CTRL)

Name: SPI FLASH custom control register

Length: [7:0]

Offset: 0x08

Reset value: 0x00

Table 212. Custom controller register

Bit Field	Name	Length	Read/Write	Description
7:4	nbyte	4	RW	Number of bytes in one transport
3:2	reserve	2	RW	Reserved
1	nbmode	1	RW	Multi-byte transport mode
0	start	1	RW	Start multi-byte transport, auto-zero when finished

15.2.9. Custom Command Register (CMD)

Name: SPI FLASH custom command register

Length: [7:0]

Offset: 0x09

Reset value: 0x00

Table 213. Custom command register

Bit Field	Name	Length	Read/Write	Description
7:0	cmd	8	RW	Set the command to transport to SPI FLASH

15.2.10. Custom Data Register 0 (BUF0)

Name: SPI FLASH custom data register 0

Length: [7:0]

Offset: 0x0a

Reset value: 0x00

Table 214. Custom data register 0

Bit Field	Name	Length	Read/Write	Description
7:0	buf0	8	RW	When transporting a write command to the SPI, this register configures the first byte of data sent; when transporting a read command to the SPI, this register stores the first data read back.

15.2.11. Custom Data Register 1 (BUF1)

Name: SPI FLASH custom data register 1

Length: [7:0]

Offset: 0x0b

Reset value: 0x00

Table 215. Custom data register 1

Bit Field	Name	Length	Read/Write	Description
7:0	buf1	8	RW	When transporting a write command to the SPI, this register configures the second byte of data sent; when transporting a read command to the SPI, this register stores the second data read back.

15.2.12. Custom Timing Register 0 (TIMER0)

Name: SPI FLASH custom timing register 0

Length: [7:0]

Offset: 0x0c

Reset value: 0x00

Table 216. Custom timing register 0

Bit Field	Name	Length	Read/Write	Description
7:0	time0	8	RW	Lower 8 bits of the time value required for the custom command

15.2.13. Custom Timing Register 1 (TIMER1)

Name: SPI FLASH custom timing register 1

Length: [7:0]

Offset: 0x0d

Reset value: 0x00

Table 217. Custom timing register 1

Bit Field	Name	Length	Read/Write	Description
7:0	time1	8	RW	Middle 8 bits of the time value required for the custom command

15.2.14. Custom Timing Register 2 (TIMER2)

Name: SPI FLASH custom timing register 2

Length: [7:0]

Offset: 0x0e

Reset value: 0x00

Table 218. Custom timing register 2

Bit Field	Name	Length	Read/Write	Description
7:0	time2	8	RW	Higher 8 bits of the time value required for the custom command

15.2.15. Guide to the Use of SPI Dual/Quad Mode

In addition to the legacy single-wire mode, the SPI controller also supports two operating modes, dual mode and quad mode, for booting from the SPI FLASH. The SPI controller can be put into dual mode by setting the dual_io register, and quad mode by setting the quad_io register. The configuration code for these two registers can be added to the first few instructions of the BIOS code, and then the controller will be pointed to the corresponding operating mode after the configuration is completed, which can improve the boot-up speed.

Note that some SPI FLASHs do not enable quad mode by default, or need to configure timing related parameters in quad mode (e.g.Dummy clocks). In order to increase the applicability of SPI controller to various FLASHs, this controller adds custom registers (0x8-0xe). The specific usage is:

1. Setting the Custom command register (CMD) (0x9), which is the command sent to the SPI FLASH.
2. Configuring the wait time into the custom timing registers TIMER0-TIMER2 (0xc-0xe) if the SPI FLASH requires that the command sent this time takes a while to complete, otherwise these registers remain at the default value of 0.
3. If writing configuration information to the SPI FLASH, the configuration information needs to be written to the custom data registers BUF0-BUF1 (0xa-0xb); if reading configuration information to the SPI FLASH, these two registers store the read back values.
4. Configuration custom control register CTRL[7:1] where CTRL[1] (nbmode) represents that the multi-byte transport mode will be performed, and the number of bytes to be transported this time is given by CTRL[7:4] (nbyte).
5. Configure the custom control register CTRL[0] to start this transport.

Generally, the registers to be configured are located in the non-volatile memory of FLASH, so the above configuration is only needed once.

15.3. I2C Controller

This chapter gives a detailed description of the I2C and its configuration for use. The system chip has an integrated I2C interface, which is mainly used to implement the exchange of data between two devices. The I2C bus is a serial bus consisting of a data line SDA and a clock SCL to send and receive data. Bi-directional transmission is performed between devices with a maximum transmission rate of 400kbps.

The I2C controller integrated in the Loongson 3A5000 can act as either a master or a slave device, and the two modes are switched between by configuring internal registers. As a slave device, it is only used to read the internal temperature of the chip, and the address of the slave device is specified by register SLV_CTRL[6:0].

The physical address base address of the I2C0 controller register is 0x1FE00120. The I2C1 controller register physical address base address is 0x1FE00130. The specific internal registers are described below.

15.3.1. Frequency Division Latch Low-order Byte Register (PRERlo)

Name: Frequency division latch low-order byte register

Length: [7:0]

Offset: 0x00

Reset value: 0xff

Table 219. Frequency division latch low-order byte register

Bit Field	Name	Length	Read/Write	Description
7:0	PRERlo	8	RW	Store the lower 8 bits of the division latch

15.3.2. Frequency Division Latch High-order Byte Register (PRERhi)

Name: Frequency division latch high-order byte register

Length: [7:0]

Offset: 0x01

Reset value: 0xff

Table 220. Frequency division latch high-order byte register

Bit Field	Name	Length	Read/Write	Description
7:0	PRERhi	8	RW	Store the high 8 bits of the division latch

Assuming that the value of the divider latch is prescale, the frequency of the PCLK clock input from the LPB bus is clock_a, and the output frequency of the SCL bus is clock_s, the following relationship should be satisfied:

```
Prcescale = clock_a/(4*clock_s)-1
```

15.3.3. Control Register (CTR)

Name: Control register

Length: [7:0]

Offset: 0x02

Reset value: 0x20

Table 221. Control register

Bit Field	Name	Length	Read/Write	Description
7	EN	1	RW	Module operating enable bit 1 - normal operation mode 0 - operation of the division register
6	IEN	1	RW	Interrupt enable bit. 1 indicates enable interrupt
5	MST_EN	1	RW	Module master-slave selection 0: slave mode 1: master mode
4:0	Reserved	5	RW	Reserved

15.3.4. Transport Data Register (TXR)

Name: Transport data register

Length: [7:0]

Offset: 0x03

Reset value: 0x00

Table 222. Transport data register

Bit Field	Name	Length	Read/Write	Description
7:1	DATA	7	W	Store the next byte to be transported
0	DRW	1	W	When data is transported, this bit stores the lowest bit of the data. When the address is transported, this bit indicates the read and write status

15.3.5. Receive Data Register (RXR)

Name: Receive data register

Length: [7:0]

Offset: 0x03

Reset value: 0x00

Table 223. Receive data register

Bit Field	Name	Length	Read/Write	Description
7:0	RXR	8	R	Store the last received byte

15.3.6. Command Control Register (CR)

Name: Command register

Length: [7:0]

Offset: 0x04

Reset value: 0x00

Table 224. Command control register

Bit Field	Name	Length	Read/Write	Description
7	STA	1	W	Generate the START signal
6	STO	1	W	Generate the STOP signal
5	RD	1	W	Generate the read signal
4	WR	1	W	Generate the write signal
3	ACK	1	W	Generate the response signal
2:1	Reserved	2	W	Reserved
0	IACK	1	W	Generate interrupt response signal

Both are automatically cleared by the hardware after the I2C sends data. Read operation of these bits always reads back 0. A bit 3 of 1 means that the controller does not send ack at the end of this transmission, and vice versa at the end.

15.3.7. State Register (SR)

Name: Status register

Length: [7:0]

Offset: 0x04

Reset value: 0x00

Table 225. State register

Bit Field	Name	Length	Read/Write	Description
7	RxACK	1	R	Receive response bit 1 - no receive the response bit 0 - receive the response bit
6	Busy	1	R	I2c bus busy flag bit 1 - bus is busy 0 - bus is free

Bit Field	Name	Length	Read/Write	Description
5	AL	1	R	When the I2C core loses control of the I2C bus, this bit is 1
4:2	Reserved	3	R	Reserved
1	TIP	1	R	Indicate the process of transport 1 - indicate that data is being transported 0 - indicate that data transport is complete
0	IF	1	R	Interrupt flag bit. When one data transport is finished, or another device initiates data transport, this bit is 1

15.3.8. Slave Device Controller Register (SLV_CTRL)

Name: Slave device control register

Length: [7:0]

Offset: 0x07

Reset value: 0x00

Table 226. Slave device controller register

Bit Field	Name	Length	Read/Write	Description
7	SLV_EN	1	WR	Slave mode enable, active when MST_EN is 0. It can be used to reset the internal logic of the slave
6:0	SLV_ADDR	7	WR	Slave mode I2C address. It can be configured via software

Chapter 16. Kernel Support

16.1. New Feature Support

In order to use the new features provided by the 3A5000 processor in the kernel, they can be identified or enabled according to the following methods. Only the parts that can improve system performance are described here.

16.1.1. Extended Interrupt Mode

In order to enable the extended interrupt mode in the kernel, set it up in the following order.

1. Extended interrupt mode support is identified by `CSR[0x8][3]`.
2. The external interrupt translation register of the HT controller that is expected to support extended interrupt mode needs to be configured to the correct value in PMON. The registers are defined as follows and set to the following values:

   `INT_trans_en = 0 //Use CSR register for enable control`, `CSR[0x420][48]` and this register can both enable extended interrupt mode, in PMON the default does not enable this mode, by the kernel configuration `CSR[0x420][48]` to turn on

   `INT_trans_allow = 1 // Allow external interrupt transition function`

   `INT_trans_addr = 0x1000000001140 // Extended interrupt register address`, see 14.3.3.

   `INT_trans_cache = 0 //Uncache mode`

   Offset: `0x270`

   Reset value: `0x00000000`

   Name: HT RX INT TRANS Lo

   Table 227. HT RX INT TRANS Lo

Bit Field	Name	Length	Reset Value	Read/Write	Description
31:4	INT_trans_addr[31:4]	28	0x0	R/W	Low order bits of interrupt translation address
3:0	Reserved	4	0x0	R	Reserved

   Offset: `0x274`

   Reset value: `0x00000000`

   Name: HT RX INT TRANS Hi

   Table 228. HT RX INT TRANS Hi

Bit Field	Name	Length	Reset Value	Read/Write	Description
31	INT_trans_en	1	0x0	R/W	Interrupt transition enable

Bit Field	Name	Length	Reset Value	Read/Write	Description
30	INT_trans_allow	1	0x0	R/W	Whether to allow interrupt transition
29:26	INT_trans_cache	4	0x0	R/W	Interrupt Transition Cache Field
25:0	INT_trans_addr[58:32]	26	0x0	R/W	High order bits of interrupt translation address

3. The kernel first identifies the extended interrupt mode support by CSR[0x8][3], and then enables the extended interrupt mode by register CSR[0x420][48].

   The base address is 0x1fe00000, It can also be accessed using the configuration register instruction (IOCSR), and the offset address is 0x0420.

   Table 229. Other function configuration register

Bit Field	Name	Read/Write	Reset Value	Description
48	EXT_INT_en	RW	0x0	Extended I/O interrupt enable

4. Set the corresponding routing and internal control for the extended interrupt mode.

16.2. Configuration Register Instruction Debug Support

The configuration register instruction is in principle used without cross-chip access, but in order to meet the need for debugging and other functions, cross-chip access is supported here using multiple register addresses. It is worth noting that such registers can only be written, not read.

Together with the existing inter-processor interrupts and other registers that can be accessed across slices, all such registers and their addresses are listed below.

Table 230. Processor core inter-processor communication registers

Name	Offset Address	Read/Write	Description
IPI_Send	0x1040	WO	32-bit interrupt distribution register [31]: wait for completion flag; when set to 1 it will wait for the interrupt to take effect [30:26]: reserved [25:16]: processor core number [15:5]: reserved [4:0]: interrupt vector number, corresponding to the vector in IPI_Status

Name	Offset Address	Read/Write	Description
Mail_Send	0x1048	WO	64-bit MailBox Cache register [63:32]: MailBox data [31]: wait for completion flag; when set to 1 it will wait for the write to take effect [30:27]: write data mask; each bit indicates that the bytes corresponding to the 32-bit write data will not really be written to the target address, such as 1000b means write the 0-2 bytes, 0000b means write all 0-3 bytes [26]: reserved [25:16]: processor core number [15:5]: reserved [4:2]: MailBox number 0 - MailBox0 low 32-bit 1 - MailBox0 high 32-bit 2 - MailBox1 low 32-bit 3 - MailBox1 high 32-bit 4 - MailBox2 low 32-bit 5 - MailBox2 high 32-bit 6 - MailBox3 low 32-bit 7 - MailBox4 high 32-bit [1:0]: reserved

Name	Offset Address	Read/Write	Description
FREQ_Send	0x1058	WO	32-bit frequency enable register [31]: wait for completion flag; when set to 1 it will wait for the setting to take effect [30:27]: write data mask; each bit indicates that the bytes corresponding to the 32-bit write data will not really be written to the target address, such as 1000b means write the 0-2 bytes, 0000b means write all 0-3 bytes [26]: reserved [25:16]: processor core number [15:5]: reserved [4:0]: write to the corresponding processor core private frequency configuration register CSR[0x1050]
ANY_Send	0x1158	WO	64-bit register access register [63:32]: write data [31]: wait for completion flag; when set to 1 it will wait for the interrupt to take effect [30:27]: write data mask; each bit indicates that the bytes corresponding to the 32-bit write data will not really be written to the target address, such as 1000b means write the 0-2 bytes, 0000b means write all 0-3 bytes [26]: reserved [25:16]: destination processor core number [15:0]: offset address of the register to be written