龙芯中科
LOONGSON TECHNOLOGY

LOONGSON

# Loongson 3A1000 processor user's manual

## Part i

**Multicore processor architecture, register description and system software programming guide**

**V1.15**

***Sep 2015***

***Loongson technology co. LTD***



北京市海淀区温泉镇中关村环保科技示范园龙芯产业园2号楼 100095
Loongson Industrial Park, building 2，Zhongguancun environmental protection park
Haidian District，Beijing
www.loongson.cn

## Copyright statement



## disclaimer

This document only provides periodic information, and the content can be updated at any time according to the actual situation of the product without prior notice. The company shall not be liable for any direct or indirect loss caused by improper use of the documents.

## Loongson technology co. LTD

Loongson Technology Corporation Limited
Address: Loongson Industrial Park, Building No. 2, loongson Industrial Park, zhongguancun environmental science and technology demonstration Park, haidian district, Beijing
Zhongguancun Environmental Protection Park, Haidian District, Beijing
Tel: 010-62546668 Fax: 010-62600826

# Reading guide

Loongson 3A1000 processor user manual is divided into volume 1 and volume 2.

Loongson 3A1000 processor user's manual is divided into two parts. The first part (chapter 1 ~ chapter 10) introduces loongson 3A1000 multi-core processor architecture and register description, and gives a detailed description of chip system architecture, functions and configuration of main modules, register list and bit domain.The second part (chapter 11 ~ chapter 16) is the system software programming guide, which introduces the common problems in the development of BIOS and operating system.

The second volume of loongson 3A1000 processor user manual introduces in detail the GS464 high-performance processor core adopted by loongson 3A1000 from the perspective of system software developers.

# Revision history

| Document update record | The document name: | Loongson 3A1000 processor user's manual - the days |
|---|---|---|
| | The version number | V1.15 |
| | The founders: | Research and development center |
| | Date of creation: | 2015-09-11 |

Update history

| The serial number | Updated date | The version number | Update the content |
|---|---|---|---|
| 1 | 2009-10-30 | V1.0 | Add the definition of DDR parameters;Fixed UART, SPI base address |
| 2 | 2009-11-13 | V1.1 | Add the definition of PCI_CONFIG to the configuration pin |
| 3 | 2010-06-25 | V1.2 | The second part of the manual is added, including the configuration and use of interrupt, configuration and use of serial port, EJTAG debugging instructions, address window configuration conversion, system memory space distribution design and memory allocation of X system |
| 4 | 2010-06-29 | V1.3 | Modified the first chapter overview, the second chapter address distribution |
| 5 | 2010-07-20 | V1.4 | Fixed some text errors in the HT configuration register |
| 6 | 2010-07-28 | V1.5 | In section 10.5, Chip Config and Chip Sample register definitions are added |
| 7 | 2010-12-17 | V1.6 | Revise the definition of DDR parameters |
| 8 | 2011-11-24 | V1.7 | Modify the cover |
| 9 | 2012-02-14 | V1.8 | Add CLKSEL setting restrictions |
| 10 | 2012-02-23 | V1.9 | Added DDR configuration register interrupt vector description |
| 11 | 2012-04-25 | V1.10 | Added details of the chip configuration register<br>Added details of HT diagnostic register |
| 12 | 2012-08-23 | V1.11 | Revise the DDR parameter definition<br>Added HT register definition supported by LS3A1000E |
| 13 | 2012-10-30 | V1.12 | Added matrix handling register supported by LS3A1000E |
| 14 | 2014-04-02 | V1.13 | According to the chip naming rules, the loongson 3A processor was renamed loongson 3A1000 processor |
| 15 | 2014-07-24 | V1.14 | Industrial level chip content added |

| | | | |
|---|---|---|---|
| 16 | 2015-09-11 | V1.15 | Fixed GPIO configuration register description |

# 目录

# The first part

## Multi-core processor architecture, register description

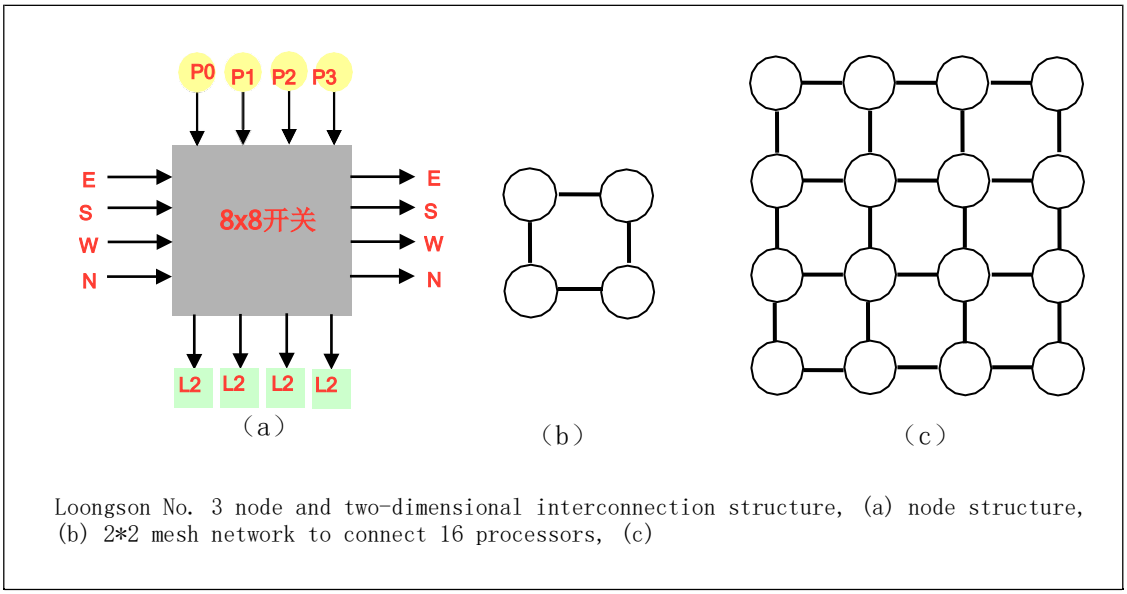
# 1 overview

## 1. 1 Introduction to loongson series processors

The loongson processor mainly includes three series. Loongson 1 processor and its IP series are mainly for embedded applications, loongson 2 superstandard processor and its IP series are mainly for desktop applications, and loongson 3 multi-core processor series are mainly for server and high-performance machine applications.According to the application needs, part of the loongson 2 can also be oriented to part of the high-end embedded should With, part of low - end loongson 3 can also face part of the desktop application.The three series will develop in parallel.

Based on the scalable multi-core interconnect architecture, the loong chip 3 multi-core series integrates multiple high-performance processor cores and a large number of level-2 caches on a single chip, and interconnects multiple chips through high-speed I/O interfaces to form a larger system.

The retractable interconnection structure adopted by loongson 3 is shown in figure 1-1below.Loongson 3 and multi - chip system are adopted 2D mesh interconnection structure, in which each node is composed of 8*8 cross-switches, each cross-switch connects four processor cores and four second-level caches, and interconnects with other nodes in the four directions of east (E), south (N), west (W) and north (N).Therefore, a 2*2 mesh can connect 16 processor cores, and a 4*4 mesh can connect 64 processor cores.




Loongson No. 3 node and two-dimensional interconnection structure, (a) node structure, (b) 2*2 mesh network to connect 16 processors, (c)

Figure 1-1 loongson 3 system structure

The node structure of loongson 3 is shown in figure 1-2 below.Each node has two levels of AXI cross-switches connected to the processor, a secondary Cache, a memory controller,

and an IO controller.The first level AXI cross Switch (called the X1 Switch) connects the processor to the secondary Cache.The second level cross Switch (called the X2 Switch) connects the second level Cache to the memory controller.

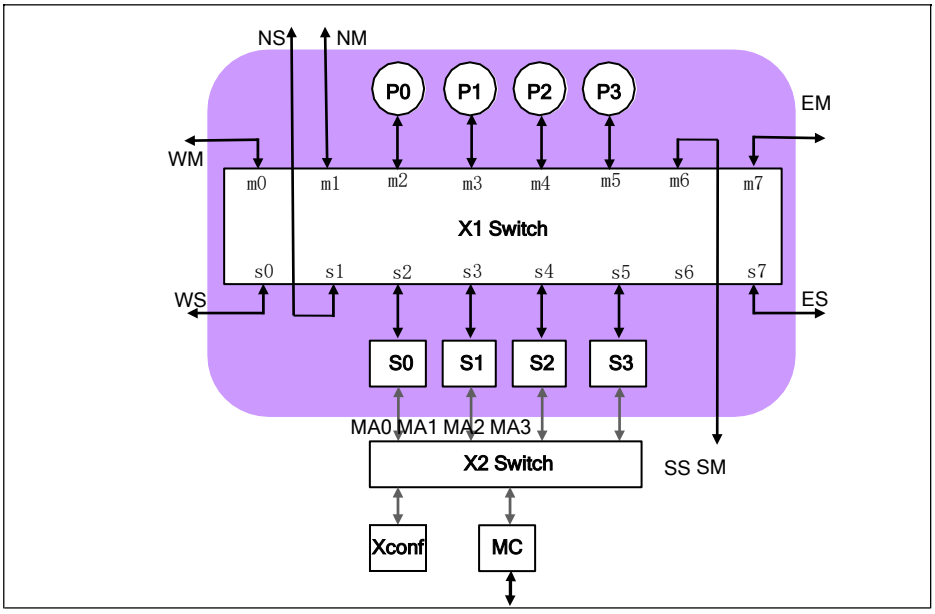



Figure 1-2 structure of loongson node3

At most 8*8 X1 crossovers are connected to four GS464 processor cores via four Master ports at each node

(P0, P1, P2, P3 in the figure), connect the four interleave second-level Cache blocks (S0, S1, S2, S3 in the figure) with four Slave ports, and connect other nodes or IO nodes (EM/ES, SM/SS, WM/WS, NM/NS in the figure) with four Master/Slave ports in the east, south, west, and north directions.

The X2 cross-switch connects four secondary caches through four Master ports, at least one Slave port to a memory controller, and at least one Slave port to a cross-switch configuration module (Xconf) to configure the local node's X1 and X2 address Windows.You can also connect more memory controllers and IO ports as needed.

The interconnection system in loongson 3 only defines the upper layer protocol, which does not specify the implementation of the transport protocol. Therefore, the interconnection between nodes can be realized either through the on-chip network or through the I/O control link to realize multi-chip interconnection.In a 4-node 16-core system, for example, it can be composed of either 4 4-core chips or 2 8-core chips

A nuclear chip, or a 16-core chip composed of four nodes based on a single chip.Because the physical implementation of the interconnected system is transparent to the software,

The above three configurations of the system can run the same operating system.

## 1.2 Introduction to loongson 3A1000

The loongson 3A1000 is the first product of the loongson no. 3 multi-core processor series. It is a single-node processor with 4 cores.

Four 64-bit quad-emission superscalar GS464 high-performance processor cores are integrated on the chip.

4 MB split Shared second-level Cache(composed of 4 individual modules, each with a capacity of 1MB);

Maintain Cache consistency for multi-core and I/O DMA access through directory protocols;

Two 64 bit 400MHz ddr2/3 controllers are integrated on the chip.

Two 16-bit 800MHz HyperTransport controllers are integrated on the chip;

Each 16-bit HT port is split into two 8-way HT ports for use.

32-bit 100MHz PCIX/66MHz PCI;

1 LPC, 2 UART, 1 SPI and 16 GPIO interfaces are integrated on the chip.

The overall architecture of loongson 3A1000 chip is based on two-level interconnection, as shown in figure 1-3 below.

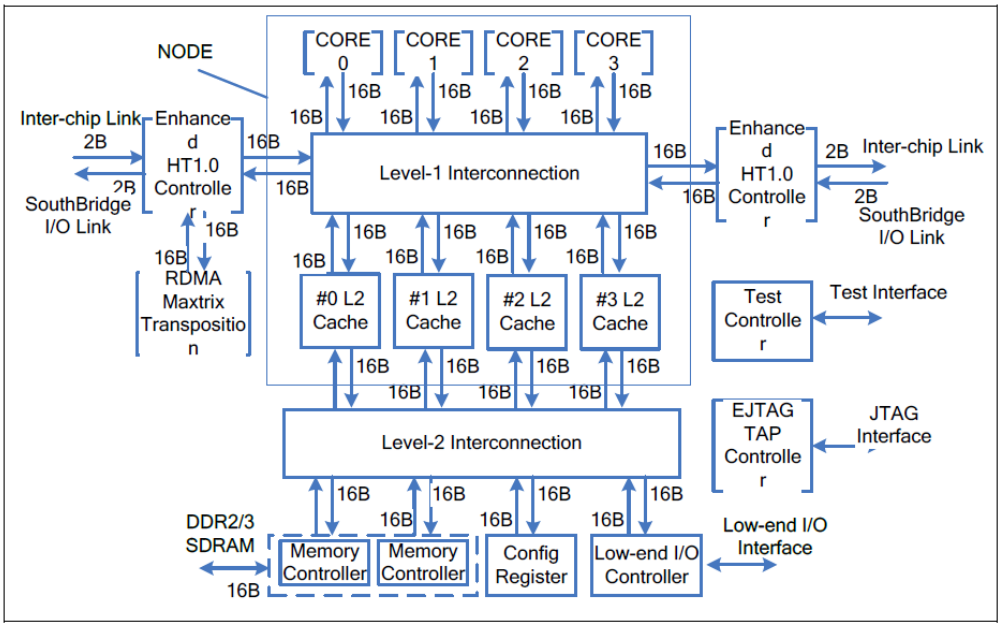



Figure 1-3 chip structure of Godson 3a10 1

The first level of interconnect USES 6x6 cross-switches to connect four cpus (as Master devices), four secondary Cache modules (as Slave devices), and two IO ports (one Master and one Slave for each port).Each IO port connected by the level 1 interconnect switch is connected to a 16-bit HT controller, and each 16-bit HT port can also be used as two 8-bit HT ports.The HT controller is connected to a level 1 interconnect switch by a DMA controller, which is responsible for the DMA control of IO and the maintenance of inter-chip consistency.The loongson 3's DMA controller can also be configured to prefetch and matrix transpose or shift.

The second level interconnection USES a 5x4 cross-switch to connect 4 secondary Cache modules (as the main device) and 2 DDR2

Memory controller, low speed and high speed I/O (including PCI, LPC, SPI, etc.) and the control register module inside the chip.The above two level interconnect switches all adopt the data channel separated by read and write. The data channel width is 128bit and

works at the same place

The same frequency is used to provide high-speed on-chip data transmission.

Based on the loongson no. 3 extensible interconnection architecture, 4 pieces of four-core loongson 3A1000 can be connected via HT port to form 4 chips

16 - core SMP structure.

## 1. 3 Loongson 3A1000 commercial grade and industrial grade chips

Loongson 3A1000 chip has two kinds of industrial grade and commercial grade. Its main features are as follows:

| configuration | Commercial grade | industrial-grade |
|---|---|---|
| Working temperature | 0 ℃ ~ 70 ℃ | - 40 ℃ ~ 85 ℃ |
| Whether screening | - | Square root |
| Quality consistency test | - | Square root |
| Quality consistency test standard | - | GB 4937-1995 |

As with most semiconductor devices, the failure rate of the loongson 3A chip conforms to the bathtub curve model.In order to ensure a more long-term, stable and reliable operation, and to adapt to the more stringent environmental temperature requirements, loongson 3A industrial-grade chip was selected for reliability, so as to eliminate the early failure of the chip.This reliability screening is 100% trial, through which the chips are screened for industrial-grade applications

The main contents of loongson 3A screening test are as follows:

| Screening program | Methods and conditions (summary) | requirements |
|---|---|---|
| 1, visual inspection | Clear identification, no contamination, no oxidation of welding ball, chip intact | 100% |
| 2. Stable baking | 125 ℃ for 24 h | 100% |
| 3. The temperature changes rapidly | 10 cycles at maximum and minimum storage temperatures | 100% |
| 4. Serial number | | 100% |
| 5, intermediate (before the old) electrical test | The normal temperature | 100% |
| 6, aging | TC = 85 ℃, 160 h | 100% |
| 7. Intermediate (after aging) electrical test | The normal temperature | 100% |
| 8. Percentage of defective products allowed (PDA)<br>To calculate | PDA≤5%, normal temperature, when 5% < PDA≤10%, can be heavy<br>New submissions, but only once | All batch |
| 9. Terminal electrical test | Three temperature, record all test data | 100% |
| 10. External visual inspection | Clear identification, no contamination, no oxidation of welding ball, chip intact | 100% |

# 2 System configuration and control

## 2.1 Chip working mode

According to the structure of the composing system, loongson 3A1000 has two working modes:

Single chip mode. The system only integrates one loongson 3A1000, which is a symmetric multiprocessor system (SMP).

Multi-chip interconnection model. The system consists of 2 or 4 loongson 3A1000, which is interlinked through the HT port of loongson 3A1000. It is a non-uniform memory multi-processor system (cc-numa)

.

Control pin instructions

The control pins of loongson 3A1000 include DO_TEST, ICCC_EN, NODE_ID[1:0], CLKSEL[15:0], and PCI_CONFIG. Table 2-1 control pin description

| signal | Pull up and down | role |
|---|---|---|
| DO_TEST | On the pull | 1 'b1 represents the functional mode<br>1 'b0 represents the test mode |
| ICCC_EN | The drop-down | 1 'b1 represents the consistent interconnection mode of multiple chips<br>1 'b0 stands for single chip mode |
| NODE_ID [1:0] | | Represents the processor number in the multichip consistent interconnection mode |
| | | Power on clock control<br>HT clock control |

| | signal | role |
|---|---|---|
| CLKSEL [15:0] | CLKSEL [15] | 1 'b1 indicates the use of internal reference voltage<br>1 'b0 indicates external reference voltage |
| | CLKSEL [14] | 1 'b1 means HT PLL USES differential clock input<br>1 'b0 means HT PLL USES normal clock input |
| | CLKSEL [12] | 2 'b00 represents the PHY clock of 1.6GHZ/1<br>2 'b01 represents the PHY clock of 3.2GHZ/2<br>2 'b10 represents the PHY clock. It is an ordinary input clock<br>2 'b11 represents PHY clock and is a differential input clock |
| | CLKSEL [10] | 2 'b00 represents HT controller clock 200MHz<br>2 'b01 represents HT controller clock 400MHz |

| | |
|---|---|
| | 2 'b1x means that HT controller clock is an ordinary |

input clock MEM clock control

| signal | role |
|---|---|
| CLKSEL [will] | 5 'b11111 means that MEM clock directly adopts memclk. In other cases, MEM clock is memclk*(clksel[8:5]+30)/(clksel[9]+3)<br>Note:<br>Memclk *(clksel[8:5]+30) must be 600MHz~ 1.36ghz memclk must be 10~40MHz |

CORE clock control

| signal | role |
|---|---|
| CLKSEL [Wednesday] | 5 'b11111 means that the CORE clock directly adopts sysclk. In other cases, the CORE clock is sysclk*(clksel[3:0]+30)/(clksel[4]+1).<br>Note:<br>Sysclk *(clksel[3:0]+30) must be 600MHz~ 1.36ghz |

IO configuration control

PCI_CONFIG [away]

7HT control signal pin voltage control bit 1* 6:5 PCIX bus speed selection *

4PCIX bus mode selection

3PCI main device mode

2 the system is started from the PCI space

1 use external PCI mediation

0HT control signal pin voltage control bit 0*

Note * :

| | | | |
|---|---|---|---|
| 6 | - | 4 | PCIX bus pattern |
| 0 | - | 0 | PCI 33/66 |
| 0 | 1 | 1 | PCI - 66 X |
| 1 | 0 | 1 | PCI - 10 - X |
| 1 | 1 | 1 | PCI - 133 X |

| | | |
|---|---|---|
| 7 | 0 | HT controls the pin voltage of the signal. These signals include HT_8x2, HT_Mode, HT_Powerok, HT_Rstn, HT_Ldt_Stopn, HT_Ldt_Reqn. |
| 0 | 0 | 1.8 v |
| 0 | 1 | Reserved |
| 1 | 0 | 2.5 v |
| 1 | 1 | 3.3 v |

## 2.2 The Cache consistency

The loongson 3A1000 is maintained by the hardware for the Cache consistency between the processor and the I/O accessed through the HT port, but the hardware does not maintain the Cache consistency of the I/O devices accessed through PCI into the system.At the time of driver development, the software is required to maintain Cache consistency for DMA (Direct Memory Access) transfers to devices accessed through PCI.

## 2.3 Physical address space distribution at the system node level

The system physical address distribution of the loongson 3 series processor adopts globally accessible hierarchical addressing design to ensure

System development extension compatibility. The overall system physical address width is 48 bits.According to the height of the address, the whole address is empty

It is evenly distributed among 16 nodes, that is, each node is allocated with a 44-bit address space.

The godson 3A1000 adopts a single-node 4-core configuration, so the corresponding addresses of the DDR memory controller, HT bus and PCI bus integrated by the godson 3A1000 chip are all contained in the 44-bit address space from 0x0 (inclusive) to 0x1000_0000_0000 (not inclusive). Please refer to the following relevant sections for the address distribution of each device.

Table 2-2 node-level system global address distribution

| Node number | Address [47:44] | The starting address | End address |
|---|---|---|---|
| | | 0 x0000_0000_0000 | 0 x1000_0000_0000 |
| 1 | 1 | 0 x1000_0000_0000 | 0 x2000_0000_0000 |
| 2 | 2 | 0 x2000_0000_0000 | 0 x3000_0000_0000 |
| 3 | 3 | 0 x3000_0000_0000 | 0 x4000_0000_0000 |
| 4 | 4 | 0 x4000_0000_0000 | 0 x5000_0000_0000 |
| 5 | 5 | 0 x5000_0000_0000 | 0 x6000_0000_0000 |
| 6 | 6 | 0 x6000_0000_0000 | 0 x7000_0000_0000 |
| 7 | 7 | 0 x7000_0000_0000 | 0 x8000_0000_0000 |
| 8 | 8 | 0 x8000_0000_0000 | 0 x9000_0000_0000 |

| | | | |
|---|---|---|---|
| 9 | 9 | 0 x9000_0000_0000 | 0 xa000_0000_0000 |
| 10 | 0 xa | 0 xa000_0000_0000 | 0 xb000_0000_0000 |
| 11 | 0 xb | 0 xb000_0000_0000 | 0 xc000_0000_0000 |
| 12 | 0 xc | 0 xc000_0000_0000 | 0 xd000_0000_0000 |
| 13 | 0 xd | 0 xd000_0000_0000 | 0 xe000_0000_0000 |
| 14 | 0 xe | 0 xe000_0000_0000 | 0 xf000_0000_0000 |
| 15 | 0 xf | 0 xf000_0000_0000 | 0 x1_0000_0000_0000 |

Within each node, the 44-bit address space is further evenly distributed among up to eight possible devices connected within the node.The low 43-bit address is owned by four 2-level Cache modules, and the high 43-bit address is further determined by the [43:42] of the address.

Bits are distributed to devices connected to four directional ports.Depending on the chip and system configuration, if there is no slave device connected to a port, the corresponding address space is reserved address space and access is not allowed.

Table 2-3 address distribution within nodes

| equip ment | Address [43:41] | The initial address within the node | End of node address |
|---|---|---|---|
| Level 2 Cache | 0,1,2,3 | 0 x000_0000_0000 | 0 x800_0000_0000 |
| In the east | 4 | 0 x800_0000_0000 | 0 xa00_0000_0000 |
| south | 5 | 0 xa00_0000_0000 | 0 xc00_0000_0000 |
| In the west | 6 | 0 xc00_0000_0000 | 0 xe00_0000_0000 |
| north | 7 | 0 xe00_0000_0000 | 0 x1000_0000_0000 |

For example, the base address of the east end port device of node 0 is 0x0800_0000_0000, the base address of the east end port device of node 1 is 0x1800_0000_0000, and so on.

Unlike the directional port mapping, the loongson 3A1000 can determine the cross-addressing mode of the secondary Cache based on the actual access behavior of the application.The four level-2 Cache modules in the node correspond to a total of 43 address Spaces, and the address space corresponding to each level-2 module is determined according to a certain two selection bits of the address bit, which can be dynamically configured and modified by the software.A configuration register named SCID_SEL is set in the system to determine the address selection bit, as shown in the following table.By default, it is distributed in a [6:5] status hash, where the address [6:5] determines the corresponding level 2 Cache number.The register address is 0x3FF00400.

Table 2-4 address distribution within nodes

| SCID_SEL | Address bit selection | SCID_SEL | Address bit selection |
|---|---|---|---|
| 4'h0 | 6: 5 | 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 |

## 2.4 Address routing distribution and configuration

The routing of loongson 3A1000 is mainly realized through the two-stage crossover switch of the system.A level 1 cross switch can configure the routing of requests received by each Master port. Each Master port has 8 address Windows, which can complete the target routing of 8 address Windows.Each address window consists of three 64-bit registers, BASE, MASK and MMAP. BASE is aligned with K bytes.The MASK used a format similar to the network MASK with a high position of 1.The lower three bits of MMAP represent the number of the target Slave port, MMAP[4] means the allowed fetch-pointing, MMAP[5] means the allowed block reading, and MMAP[7] means the enabled window.

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

Because the loongson 3 adopts fixed route by default, the configuration window is closed when it is powered on and started, so the system software is required to enable it to be configured when it is in use.

The address window conversion register is shown in the following table.

Table 2-5 register table of level 1 crossover switch address window

| address | register | address | register |
|---|---|---|---|
| 0 x3ff0_2000 | CORE0_WIN0_BASE | 0 x3ff0_2100 | CORE1_WIN0_BASE |
| 0 x3ff0_2008 | CORE0_WIN1_BASE | 0 x3ff0_2108 | CORE1_WIN1_BASE |
| 0 x3ff0_2010 | CORE0_WIN2_BASE | 0 x3ff0_2110 | CORE1_WIN2_BASE |
| 0 x3ff0_2018 | CORE0_WIN3_BASE | 0 x3ff0_2118 | CORE1_WIN3_BASE |
| 0 x3ff0_2020 | CORE0_WIN4_BASE | 0 x3ff0_2120 | CORE1_WIN4_BASE |
| 0 x3ff0_2028 | CORE0_WIN5_BASE | 0 x3ff0_2128 | CORE1_WIN5_BASE |
| 0 x3ff0_2030 | CORE0_WIN6_BASE | 0 x3ff0_2130 | CORE1_WIN6_BASE |
| 0 x3ff0_2038 | CORE0_WIN7_BASE | 0 x3ff0_2138 | CORE1_WIN7_BASE |
| 0 x3ff0_2040 | CORE0_WIN0_MASK | 0 x3ff0_2140 | CORE1_WIN0_MASK |
| 0 x3ff0_2048 | CORE0_WIN1_MASK | 0 x3ff0_2148 | CORE1_WIN1_MASK |
| 0 x3ff0_2050 | CORE0_WIN2_MASK | 0 x3ff0_2150 | CORE1_WIN2_MASK |
| 0 x3ff0_2058 | CORE0_WIN3_MASK | 0 x3ff0_2158 | CORE1_WIN3_MASK |
| 0 x3ff0_2060 | CORE0_WIN4_MASK | 0 x3ff0_2160 | CORE1_WIN4_MASK |
| 0 x3ff0_2068 | CORE0_WIN5_MASK | 0 x3ff0_2168 | CORE1_WIN5_MASK |
| 0 x3ff0_2070 | CORE0_WIN6_MASK | 0 x3ff0_2170 | CORE1_WIN6_MASK |
| 0 x3ff0_2078 | CORE0_WIN7_MASK | 0 x3ff0_2178 | CORE1_WIN7_MASK |
| 0 x3ff0_2080 | CORE0_WIN0_MMAP | 0 x3ff0_2180 | CORE1_WIN0_MMAP |
| 0 x3ff0_2088 | CORE0_WIN1_MMAP | 0 x3ff0_2188 | CORE1_WIN1_MMAP |
| 0 x3ff0_2090 | CORE0_WIN2_MMAP | 0 x3ff0_2190 | CORE1_WIN2_MMAP |

| | | | |
|---|---|---|---|
| 0 x3ff0_2098 | CORE0_WIN3_MMAP | 0 x3ff0_2198 | CORE1_WIN3_MMAP |
| 0 x3ff0_20a0 | CORE0_WIN4_MMAP | 0 x3ff0_21a0 | CORE1_WIN4_MMAP |
| 0 x3ff0_20a8 | CORE0_WIN5_MMAP | 0 x3ff0_21a8 | CORE1_WIN5_MMAP |
| 0 x3ff0_20b0 | CORE0_WIN6_MMAP | 0 x3ff0_21b0 | CORE1_WIN6_MMAP |
| 0 x3ff0_20b8 | CORE0_WIN7_MMAP | 0 x3ff0_21b8 | CORE1_WIN7_MMAP |
| 0 x3ff0_2200 | CORE2_WIN0_BASE | 0 x3ff0_2300 | CORE3_WIN0_BASE |
| 0 x3ff0_2208 | CORE2_WIN1_BASE | 0 x3ff0_2308 | CORE3_WIN1_BASE |
| 0 x3ff0_2210 | CORE2_WIN2_BASE | 0 x3ff0_2310 | CORE3_WIN2_BASE |
| 0 x3ff0_2218 | CORE2_WIN3_BASE | 0 x3ff0_2318 | CORE3_WIN3_BASE |
| 0 x3ff0_2220 | CORE2_WIN4_BASE | 0 x3ff0_2320 | CORE3_WIN4_BASE |
| 0 x3ff0_2228 | CORE2_WIN5_BASE | 0 x3ff0_2328 | CORE3_WIN5_BASE |
| 0 x3ff0_2230 | CORE2_WIN6_BASE | 0 x3ff0_2330 | CORE3_WIN6_BASE |
| 0 x3ff0_2238 | CORE2_WIN7_BASE | 0 x3ff0_2338 | CORE3_WIN7_BASE |
| 0 x3ff0_2240 | CORE2_WIN0_MASK | 0 x3ff0_2340 | CORE3_WIN0_MASK |
| 0 x3ff0_2248 | CORE2_WIN1_MASK | 0 x3ff0_2348 | CORE3_WIN1_MASK |
| 0 x3ff0_2250 | CORE2_WIN2_MASK | 0 x3ff0_2350 | CORE3_WIN2_MASK |
| 0 x3ff0_2258 | CORE2_WIN3_MASK | 0 x3ff0_2358 | CORE3_WIN3_MASK |
| 0 x3ff0_2260 | CORE2_WIN4_MASK | 0 x3ff0_2360 | CORE3_WIN4_MASK |
| 0 x3ff0_2268 | CORE2_WIN5_MASK | 0 x3ff0_2368 | CORE3_WIN5_MASK |
| 0 x3ff0_2270 | CORE2_WIN6_MASK | 0 x3ff0_2370 | CORE3_WIN6_MASK |
| 0 x3ff0_2278 | CORE2_WIN7_MASK | 0 x3ff0_2378 | CORE3_WIN7_MASK |
| 0 x3ff0_2280 | CORE2_WIN0_MMAP | 0 x3ff0_2380 | CORE3_WIN0_MMAP |
| 0 x3ff0_2288 | CORE2_WIN1_MMAP | 0 x3ff0_2388 | CORE3_WIN1_MMAP |
| 0 x3ff0_2290 | CORE2_WIN2_MMAP | 0 x3ff0_2390 | CORE3_WIN2_MMAP |
| 0 x3ff0_2298 | CORE2_WIN3_MMAP | 0 x3ff0_2398 | CORE3_WIN3_MMAP |
| 0 x3ff0_22a0 | CORE2_WIN4_MMAP | 0 x3ff0_23a0 | CORE3_WIN4_MMAP |
| 0 x3ff0_22a8 | CORE2_WIN5_MMAP | 0 x3ff0_23a8 | CORE3_WIN5_MMAP |
| 0 x3ff0_22b0 | CORE2_WIN6_MMAP | 0 x3ff0_23b0 | CORE3_WIN6_MMAP |
| 0 x3ff0_22b8 | CORE2_WIN7_MMAP | 0 x3ff0_23b8 | CORE3_WIN7_MMAP |
| 0 x3ff0_2400 | EAST_WIN0_BASE | 0 x3ff0_2500 | SOUTH_WIN0_BASE |
| 0 x3ff0_2408 | EAST_WIN1_BASE | 0 x3ff0_2508 | SOUTH_WIN1_BASE |
| 0 x3ff0_2410 | EAST_WIN2_BASE | 0 x3ff0_2510 | SOUTH_WIN2_BASE |
| 0 x3ff0_2418 | EAST_WIN3_BASE | 0 x3ff0_2518 | SOUTH_WIN3_BASE |

| | | | |
|---|---|---|---|
| 0 x3ff0_2420 | EAST_WIN4_BASE | 0 x3ff0_2520 | SOUTH_WIN4_BASE |
| 0 x3ff0_2428 | EAST_WIN5_BASE | 0 x3ff0_2528 | SOUTH_WIN5_BASE |
| 0 x3ff0_2430 | EAST_WIN6_BASE | 0 x3ff0_2530 | SOUTH_WIN6_BASE |
| 0 x3ff0_2438 | EAST_WIN7_BASE | 0 x3ff0_2538 | SOUTH_WIN7_BASE |
| 0 x3ff0_2440 | EAST_WIN0_MASK | 0 x3ff0_2540 | SOUTH_WIN0_MASK |
| 0 x3ff0_2448 | EAST_WIN1_MASK | 0 x3ff0_2548 | SOUTH_WIN1_MASK |
| 0 x3ff0_2450 | EAST_WIN2_MASK | 0 x3ff0_2550 | SOUTH_WIN2_MASK |
| 0 x3ff0_2458 | EAST_WIN3_MASK | 0 x3ff0_2558 | SOUTH_WIN3_MASK |
| 0 x3ff0_2460 | EAST_WIN4_MASK | 0 x3ff0_2560 | SOUTH_WIN4_MASK |
| 0 x3ff0_2468 | EAST_WIN5_MASK | 0 x3ff0_2568 | SOUTH_WIN5_MASK |
| 0 x3ff0_2470 | EAST_WIN6_MASK | 0 x3ff0_2570 | SOUTH_WIN6_MASK |
| 0 x3ff0_2478 | EAST_WIN7_MASK | 0 x3ff0_2578 | SOUTH_WIN7_MASK |
| 0 x3ff0_2480 | EAST_WIN0_MMAP | 0 x3ff0_2580 | SOUTH_WIN0_MMAP |
| 0 x3ff0_2488 | EAST_WIN1_MMAP | 0 x3ff0_2588 | SOUTH_WIN1_MMAP |
| 0 x3ff0_2490 | EAST_WIN2_MMAP | 0 x3ff0_2590 | SOUTH_WIN2_MMAP |
| 0 x3ff0_2498 | EAST_WIN3_MMAP | 0 x3ff0_2598 | SOUTH_WIN3_MMAP |
| 0 x3ff0_24a0 | EAST_WIN4_MMAP | 0 x3ff0_25a0 | SOUTH_WIN4_MMAP |
| 0 x3ff0_24a8 | EAST_WIN5_MMAP | 0 x3ff0_25a8 | SOUTH_WIN5_MMAP |
| 0 x3ff0_24b0 | EAST_WIN6_MMAP | 0 x3ff0_25b0 | SOUTH_WIN6_MMAP |
| 0 x3ff0_24b8 | EAST_WIN7_MMAP | 0 x3ff0_25b8 | SOUTH_WIN7_MMAP |
| 0 x3ff0_2600 | WEST_WIN0_BASE | 0 x3ff0_2700 | NORTH_WIN0_BASE |
| 0 x3ff0_2608 | WEST_WIN1_BASE | 0 x3ff0_2708 | NORTH_WIN1_BASE |
| 0 x3ff0_2610 | WEST_WIN2_BASE | 0 x3ff0_2710 | NORTH_WIN2_BASE |
| 0 x3ff0_2618 | WEST_WIN3_BASE | 0 x3ff0_2718 | NORTH_WIN3_BASE |
| 0 x3ff0_2620 | WEST_WIN4_BASE | 0 x3ff0_2720 | NORTH_WIN4_BASE |
| 0 x3ff0_2628 | WEST_WIN5_BASE | 0 x3ff0_2728 | NORTH_WIN5_BASE |
| 0 x3ff0_2630 | WEST_WIN6_BASE | 0 x3ff0_2730 | NORTH_WIN6_BASE |
| 0 x3ff0_2638 | WEST_WIN7_BASE | 0 x3ff0_2738 | NORTH_WIN7_BASE |
| 0 x3ff0_2640 | WEST_WIN0_MASK | 0 x3ff0_2740 | NORTH_WIN0_MASK |
| 0 x3ff0_2648 | WEST_WIN1_MASK | 0 x3ff0_2748 | NORTH_WIN1_MASK |
| 0 x3ff0_2650 | WEST_WIN2_MASK | 0 x3ff0_2750 | NORTH_WIN2_MASK |
| 0 x3ff0_2658 | WEST_WIN3_MASK | 0 x3ff0_2758 | NORTH_WIN3_MASK |
| 0 x3ff0_2660 | WEST_WIN4_MASK | 0 x3ff0_2760 | NORTH_WIN4_MASK |

| | | | |
|---|---|---|---|
| 0 x3ff0_2668 | WEST_WIN5_MASK | 0 x3ff0_2768 | NORTH_WIN5_MASK |
| 0 x3ff0_2670 | WEST_WIN6_MASK | 0 x3ff0_2770 | NORTH_WIN6_MASK |
| 0 x3ff0_2678 | WEST_WIN7_MASK | 0 x3ff0_2778 | NORTH_WIN7_MASK |
| 0 x3ff0_2680 | WEST_WIN0_MMAP | 0 x3ff0_2780 | NORTH_WIN0_MMAP |
| 0 x3ff0_2688 | WEST_WIN1_MMAP | 0 x3ff0_2788 | NORTH_WIN1_MMAP |
| 0 x3ff0_2690 | WEST_WIN2_MMAP | 0 x3ff0_2790 | NORTH_WIN2_MMAP |
| 0 x3ff0_2698 | WEST_WIN3_MMAP | 0 x3ff0_2798 | NORTH_WIN3_MMAP |
| 0 x3ff0_26a0 | WEST_WIN4_MMAP | 0 x3ff0_27a0 | NORTH_WIN4_MMAP |
| 0 x3ff0_26a8 | WEST_WIN5_MMAP | 0 x3ff0_27a8 | NORTH_WIN5_MMAP |
| 0 x3ff0_26b0 | WEST_WIN6_MMAP | 0 x3ff0_27b0 | NORTH_WIN6_MMAP |
| 0 x3ff0_26b8 | WEST_WIN7_MMAP | 0 x3ff0_27b8 | NORTH_WIN7_MMAP |

In the level 2 XBAR of loongson 3, there are three ip-related address Spaces (including HT space), DDR2 address space, and PCI address space.The address window is set for routing and address translation between CPU and pci-dma, two IP with Master functions.Both the CPU and pci-dma have eight address Windows that enable the selection of the target address space and the conversion from the source address space to the target address space.Each address window consists of three 64-bit registers, BASE, MASK, and MMAP. BASE is k-byte aligned, MASK is in a format similar to the network MASK with a high digit of 1, and the lower digit of MMAP is routing.

At level 2 XBAR, the corresponding relationship between the label and the module is as follows, indicating the number corresponding to the new address space (where two DDR2 labels are 0 and 1, PCI/Local IO Numbers are 2, and the configuration register module is connected on port 3).

Table 2-6 at level 2 XBAR, the corresponding relationship between the label and the module

| label | The default value |
|---|---|
| 0 | No. 0 ddr2/3 controller |
| 1 | No. 1 ddr2/3 controller |
| 2 | Low speed I/O (PCI, LPC, etc.) |
| 3 | Configuration register |

This is shown in the following table.MMAP[4] is allowed to fetch, MMAP[5] is allowed to read blocks, and MMAP[7] is allowed to make Windows

Can.

Table 2-7 the MMAP field corresponds to the spatial access properties

| [4] | [5] | [7] |
|---|---|---|
| Allowed to take to | Allow the block read | The window can make |

The address configuration of level 2 XBAR adds the capability of address translation compared with the address configuration of level 1 XBAR.In contrast, the window configuration of level 1 XBAR cannot address Cache consistent requests, otherwise the address in the level 2 Cache will not match the address in the processor's level 1 Cache, resulting in Cache consistency maintenance errors.

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

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

Address window conversion registers are shown in the following table.

Table 2-8 the register table is converted from the level 2 XBAR address window

| address | register | describe | The default value |
|---|---|---|---|
| 3 ff0 0000 | CPU_WIN0_BASE | Base address of CPU window 0 | 0 x0 |
| 3 ff0 0008 | CPU_WIN1_BASE | Base address of CPU window 1 | 0 x1000_0000 |
| 3 ff0 0010 | CPU_WIN2_BASE | Base address of CPU window 2 | 0 x0 |
| 3 ff0 0018 | CPU_WIN3_BASE | Base address of CPU window 3 | 0 x0 |
| 3 ff0 0020 | CPU_WIN4_BASE | Base address of CPU window 4 | 0 x0 |
| 3 ff0 0028 | CPU_WIN5_BASE | Base address of CPU window 5 | 0 x0 |
| 3 ff0 0030 | CPU_WIN6_BASE | Base address of CPU window 6 | 0 x0 |
| 3 ff0 0038 | CPU_WIN7_BASE | The base address of CPU window 7 | 0 x0 |
| 3 ff0 0040 | CPU_WIN0_MASK | The mask for CPU window 0 | 0 xffff_ffff_f000_0000 |
| 3 ff0 0048 | CPU_WIN1_MASK | Mask for CPU window 1 | 0 xffff_ffff_f000_0000 |
| 3 ff0 0050 | CPU_WIN2_MASK | CPU window 2 mask | 0 x0 |
| 3 ff0 0058 | CPU_WIN3_MASK | Mask for CPU window 3 | 0 x0 |
| 3 ff0 0060 | CPU_WIN4_MASK | Mask for CPU window 4 | 0 x0 |
| 3 ff0 0068 | CPU_WIN5_MASK | Mask for CPU window 5 | 0 x0 |
| 3 ff0 0070 | CPU_WIN6_MASK | Mask for CPU window 6 | 0 x0 |
| 3 ff0 0078 | CPU_WIN7_MASK | CPU window 7 mask | 0 x0 |
| 3 ff0 0080 | CPU_WIN0_MMAP | New base address for CPU window 0 | 0 xf0 |
| 3 ff0 0088 | CPU_WIN1_MMAP | New base address for CPU window 1 | 0 x1000_00f2 |

| | | | |
|---|---|---|---|
| 3 ff0 0090 | CPU_WIN2_MMAP | New base address for CPU window 2 | 0 |
| 3 ff0 0098 | CPU_WIN3_MMAP | New base address for CPU window 3 | 0 |
| 3 ff0 00 a nought | CPU_WIN4_MMAP | New base address for CPU window 4 | 0 x0 |
| 3 ff0 00 a8 | CPU_WIN5_MMAP | New base address for CPU window 5 | 0 x0 |
| 3 ff0 00 b0 | CPU_WIN6_MMAP | New base address for CPU window 6 | 0 |
| 3 ff0 00 b8 | CPU_WIN7_MMAP | New base address for CPU window 7 | 0 |
| 3 ff0 0100 | PCI_WIN0_BASE | Base address for PCI window 0 | 0 x8000_0000 |
| 3 ff0 0108 | PCI_WIN1_BASE | Base address for PCI window 1 | 0 x0 |
| 3 ff0 0110 | PCI_WIN2_BASE | PCI window 2 base address | 0 x0 |
| 3 ff0 0118 | PCI_WIN3_BASE | Base address for PCI window 3 | 0 x0 |
| 3 ff0 0120 | PCI_WIN4_BASE | Base address for PCI window 4 | 0 x0 |
| 3 ff0 0128 | PCI_WIN5_BASE | Base address for PCI window 5 | 0 x0 |
| 3 ff0 0130 | PCI_WIN6_BASE | Base address for PCI window 6 | 0 x0 |
| 3 ff0 0138 | PCI_WIN7_BASE | Base address for PCI window 7 | 0 x0 |
| 3 ff0 0140 | PCI_WIN0_MASK | Mask for PCI window 0 | 0 xffff_ffff_8000_0000 |
| 3 ff0 0148 | PCI_WIN1_MASK | Mask for PCI window 1 | 0 x0 |
| 3 ff0 0150 | PCI_WIN2_MASK | Mask for PCI window 2 | 0 x0 |
| 3 ff0 0158 | PCI_WIN3_MASK | Mask for PCI window 3 | 0 x0 |
| 3 ff0 0160 | PCI_WIN4_MASK | Mask for PCI window 4 | 0 x0 |
| 3 ff0 0168 | PCI_WIN5_MASK | Mask for PCI window 5 | 0 x0 |
| 3 ff0 0170 | PCI_WIN6_MASK | Mask for PCI window 6 | 0 x0 |
| 3 ff0 0178 | PCI_WIN7_MASK | Mask for PCI window 7 | 0 x0 |
| 3 ff0 0180 | PCI_WIN0_MMAP | New base address for PCI window 0 | 0 xf0 |
| 3 ff0 0188 | PCI_WIN1_MMAP | New base address for PCI window 1 | 0 x0 |
| 3 ff0 0190 | PCI_WIN2_MMAP | New base address for PCI window 2 | 0 |
| 3 ff0 0198 | PCI_WIN3_MMAP | New base address for PCI window 3 | 0 |
| 3 ff0 a0 01 | PCI_WIN4_MMAP | New base address for PCI window 4 | 0 x0 |

| | | | |
|---|---|---|---|
| 3 ff0 01 a8 | PCI_WIN5_MMAP | New base address for PCI window 5 | 0 x0 |
| 3 ff0 b0 01 | PCI_WIN6_MMAP | New base address for PCI window 6 | 0 |
| 3 ff0 b8 01 | PCI_WIN7_MMAP | New base address for PCI window 7 | 0 |

According to the default register configuration, CPU 0x00000000-0x0fffffff address range after chip startup

(256M) is mapped to the address range of 0x00000000-0x0fffffff of DDR2, 0x10000000-0x1fffffff of CPU (256M) is mapped to 0x10000000-0x1fffffff of PCI, and 0x80000000 of PCIDMA

The address interval of -0x8fffffff (256M) maps to the address interval of 0x00000000-0x0fffffff of DDR2. The software can realize the new address space routing and transformation by modifying the corresponding configuration registers.

In addition, when read access to illegal addresses occurs due to CPU guess execution, all 8 address Windows fail to hit,

The configuration register module returns all zeros to the CPU to prevent the CPU from waiting.

Table 2-9 XBAR level 2 default address configuration

| Base address | high | The owner of the |
|---|---|---|
| 0 x0000_0000_0000_0000 | 0 x0000_0000_1000_0000 | No. 0 DDR controller |
| 0 x0000_0000_1000_0000 | 0 x0000_0000_2000_0000 | Low speed I/O (PCI, etc.) |

## 2.5 Chip configuration and sampling register

The chip configuration register (Chip_config) and the chip sampling register (chip_sample) in the loong chip 3 provide the mechanism for reading and writing the chip configuration.

Table 2-10 chip configuration register (physical address 0x1fe00180)

| A domain | The field name | access | Reset value | describe |
|---|---|---|---|---|
| The 2-0 | Freq_scale_ctrl | RW | 3 'b111 | Processor core frequency division<br>The actual frequency of the processor core is zero<br>PLL frequency * (Freq_scal_ctrl + 1)/8 |
| 3 | DDR_Clksel_en | RW | 1 'b0 | Whether to use software to configure DDR octave<br>1: use software configuration<br>0: use pin CLKSEL configuration |
| 8 | Disable_ddr2_confspace | RW | 1 'b0 | Whether to disable DDR configuration space<br>1: disable<br>0: I can't help it |
| 9 | DDR_buffer_cpu | RW | 1 'b0 | Whether to turn on the DDR read access buffer<br>1: open the<br>0: disable |
| 12 | Core0_en | RW | 1 'b1 | Whether to enable<br>processor core 0 1: on<br>0: disable |
| 13 | Core1_en | RW | 1 'b1 | Whether to enable<br>processor core 1 1:<br>open<br>0: disable |
| 14 | Core2_en | RW | 1 'b1 | Whether to enable<br>processor core 2 1: on<br>0: disable |

| | | | | |
|---|---|---|---|---|
| 15 | Core3_en | RW | 1 'b1 | Whether to enable processor core 3 1: on 0: disable |
| 16 | Mc0_en | RW | 1 'b1 | Whether to enable DDR controller 0 1: open the |
| | | | | 0: disable |
| 17 | Mc1_en | RW | 1 'b1 | Whether to enable DDR controller 1 1: on 0: disable |
| 18 | DDR_reset0 | RW | 1 'b1 | Software reset DDR controller 0 1: reset 0: unreset |
| 19 | DDR_reset1 | RW | 1 'b1 | Software reset DDR controller 1 1: reset 0: unreset |
| 22 | HT0_en | RW | 1 'b1 | Whether to enable HT controller 0 1: on 0: disable |
| 23 | HT1_en | RW | 1 'b1 | Whether to enable HT controller 1 1: on 0: disable |
| throug hout | DDR_Clksel | RW | 5 'b11111 | Software configuration DDR clock frequency doubling relationship (when DDR_Clksel_en is 1) |
| take | HT_freq_scale_ctrl0 | RW | 3 'b111 | The actual frequency of the HT controller 0 frequency divider is the HT controller frequency * (HTFreq_scal_ctrl + 1) / 8 |
| That which | HT_freq_scale_ctrl0 | RW | 3 'b111 | HT controller 1 the actual frequency of the frequency division controller is HT controller frequency * (HTFreq_scal_ctrl + 1) / 8 |
| 35 | Mc0_prefetch_disable | RW | 1 'b0 | Whether or not to disable MC0 prefetch (which has different performance effects on different program behaviors) 1: disable 0: I can't help it |

| | | | | |
|---|---|---|---|---|
| 36 | Mc1_prefetch_disable | RW | 1 'b0 | Whether to disable MC1 prefetch (which has different performance effects on different program behaviors)<br>1: disable<br>0: I can't help it |
| other | | R | | reserve |

Table 2-11 chip sampling register (physical address 0x1fe00190)

| A domain | The field name | access | Reset value | describe |
|---|---|---|---|---|
| 15:0 | Pad2v5_ctrl | RW | 16 'h780 | 2 v5pad control |
| Caused the | Pad3v3_ctrl | RW | 16 'h780 | 3 v3pad control |
| 47:32 | Sys_clksel | R | | On board frequency doubling setting |
| spoilers | Bad_ip_core | R | | Represents whether the processor core is unavailable, and each corresponds to the processor core 3 to the processor core 0<br>0 - available<br>1 - do not use |
| 53:52 | Bad_ip_ddr | R | | Whether 2 DDR controllers are bad |
| 57:56 | Bad_ip_ht | R | | Is the 2 HT controllers bad |
| A 102-96 | Thsens0_out | R | | Temperature sensor 0 temperature, used to monitor the temperature near the level 2 cache, accuracy is +/-6 ℃ (note: this sensor sometimes has abnormal Temperature) |
| 103 | Thsens0_overflow | R | | Temperature sensor 0 temperature overflow (over 128<br>C) |
| A 110-104 | Thsens1_out | R | | Temperature sensor 1 temperature, used to monitor the temperature near the processor core, precision is +/-6 degrees Celsius |
| 111 | Thsens1_overflow | R | | Temperature sensor 1 temperature overflow (over 128<br>C) |
| other | | R | | reserve |

# 3 GS464 processor core

The GS464 is a four-shot 64-bit high-performance processor core.The processor core can be used as a single core for high-end embedded applications and desktop applications, and can also be used as a basic processor core for in-chip multi-core systems for server and high-performance machine applications.The GS464 cores in loongson 3A1000 and the secondary Cache modules form a distributed Shared secondary Cache multicore structure via a AXI interconnection network.The main features of GS464 are as follows:

MIPS64 compatible, support loongson extended instruction set;

Four emission superscalar structure, two fixed points, two floating point, one memory access parts;

Each floating point component supports full-flow 64-bit/double-32-bit floating point multiplication and addition;

Memory access parts support 128 bit storage access, virtual address and physical address are 48 bits;

Support register renaming, dynamic scheduling, transfer prediction and other out-of-order execution techniques;

64 fully linked TLB, 16 independent instruction TLB, variable page size;

The size of the first-level instruction Cache and the data Cache is 64KB respectively, and the four-way groups are connected.

Support the optimization technology of non-blocking visit and load-speculation;

Support Cache consistency protocol, which can be used in chip multi-core processors;

The instruction Cache implements parity check, and the data Cache implements ECC check.

Standard EJTAG debugging standard is supported to facilitate debugging of software and hardware.

Standard 128 - bit AXI interface.

The structure of GS464 is shown in the following figure.  Please refer to the GS464 user manual and MIPS64 for more details

User's manual.

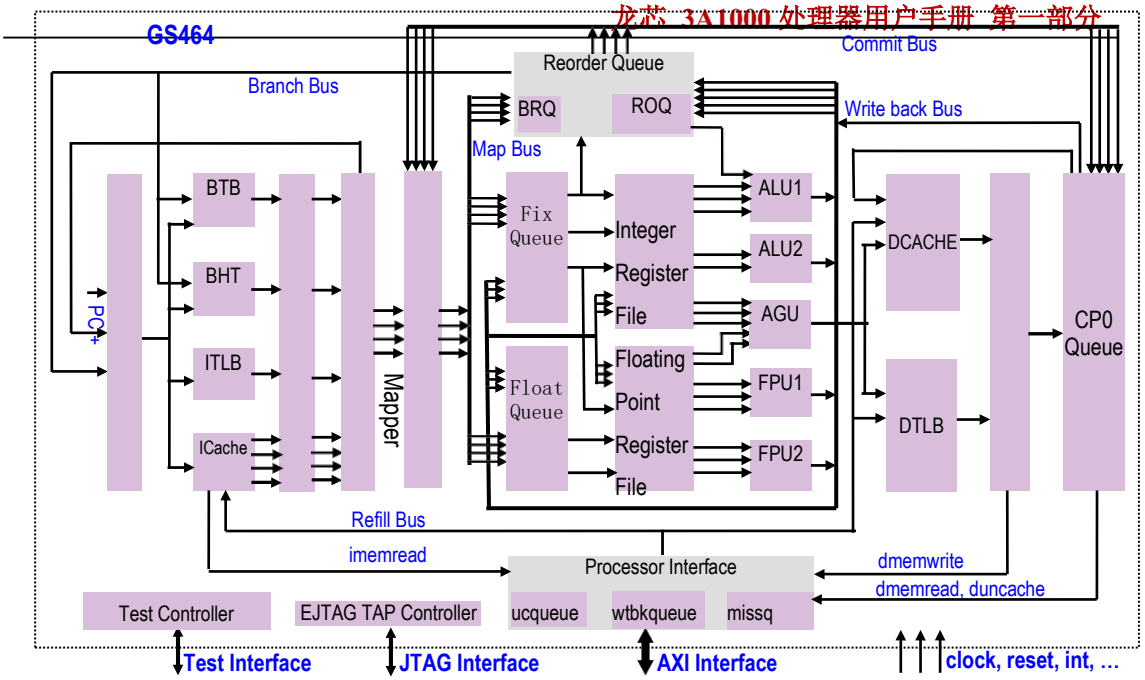



Figure 3-1 GS464 structure diagram

# 4 Level 2 Cache

The secondary Cache module is designed with GS464 processor IP.This module can not only connect with GS464, but also make GS464 the IP of the processor including the second-level Cache.It is also possible to connect multiple GS464 and multiple secondary Cache modules through a AXI network to form a CMP structure of multiple processors on the chip.The main features of the second-level Cache module include:

Use 128 - bit AXI interface.

Eight Cache items access the queue.

Keyword first.

Receive read invalidation requests to return data at the fastest 8 beats.

Cache consistency protocol is supported through directories.

It can be used for multi-core structure on chip, and also can be directly connected with IP of single processor.

The soft IP level can configure the size of the secondary Cache (512KB/1MB).

Adopt four - way group - linked structure.

The runtime can be closed dynamically.

Support ECC validation.

Support DMA consistent read-write and prefetch reads.

Supports 16 secondary Cache hashes.

Supports locking the second level of Cache by window.

Ensure that the read data returns atomicity.

The second level Cache module includes the second level Cache management module scachemanage and the second level Cache access module scacheaccess.The Scachemanage module is responsible for the processor's access requests from the processor and DMA, while the TAG, directory, and data of the secondary Cache are stored in the scacheaccess module.To reduce power consumption, the TAG, directory and DATA of the second-level Cache can be accessed separately. The second-level Cache status bit and the w bit are stored together with the TAG, which is stored in TAG RAM, the directory in DIR RAM, and the DATA in DATA RAM.The invalidation request accesses the second level Cache, reads the tags, directories, and data of all the paths, and selects the data and directories according to the TAG.Replace requests, refill requests, and write back requests operate only on tags, directories, and data

along the way.

To improve the performance of certain computing tasks, a locking mechanism was added to the second level Cache.A secondary Cache block that falls in a locked area is locked and will not be replaced with a secondary Cache (unless all four secondary Cache blocks are locked).Confbus can dynamically configure the four groups of lock window registers inside the second-level Cache module, but it must ensure that all of the four-way second-level caches are locked.The size of each group of Windows can be adjusted according to the mask, but not more than 3/4 of the overall second-level Cache size.In addition, when the second level Cache receives a DMA write request, if the region is written

When a hit is made and locked in a second-level Cache, the DMA write is written directly to the second-level Cache instead of memory.

Table 4-1 configuration of the second level Cache lock window register

| The name of the | address | A domain | describe |
|---|---|---|---|
| Slock0_valid | 0 x3ff00200 | [63-63] | Zero lock window valid bit |
| Slock0_addr | 0 x3ff00200 | [47:0] | Lock window 0 lock address |
| Slock0_mask | 0 x3ff00240 | [47:0] | Lock window mask no. 0 |
| Slock1_valid | 0 x3ff00208 | [63-63] | No. 1 lock window valid bit |
| Slock1_addr | 0 x3ff00208 | [47:0] | No. 1 lock window lock address |
| Slock1_mask | 0 x3ff00248 | [47:0] | No. 1 lock window mask |
| Slock2_valid | 0 x3ff00210 | [63-63] | No. 2 lock window valid bit |
| Slock2_addr | 0 x3ff00210 | [47:0] | No. 2 lock window lock address |
| Slock2_mask | 0 x3ff00250 | [47:0] | No. 2 lock window mask |
| Slock3_valid | 0 x3ff00218 | [63-63] | No. 3 lock window valid bit |
| Slock3_addr | 0 x3ff00218 | [47:0] | No. 3 lock window lock address |
| Slock3_mask | 0 x3ff00258 | [47:0] | No. 3 lock window mask |

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

# 5 Matrix processing accelerator

Logodson 3A1000 is built with two matrix processing accelerators independent of the processor core. Its basic function is to realize the function of transposing or moving the matrix stored in memory from the source matrix to the target matrix through the configuration of software

(previous versions of LS3A1000E only supported transpose).Two accelerators are respectively integrated in two of the loongson 3A1000

Inside the HyperTransport controller, level 1 cross switches are used to read and write the level 2 Cache and memory.

Before due to transpose the same Cache line element order after the transposed matrix is distributed, in order to improve the efficiency of reading and writing, need read many rows of data, makes the data can be in after the transposed matrix unit to write to the Cache behavior, thus set up a size 32 in the module the buffer zone, realize transverse writing (matrix into the buffer from the source), longitudinal read (matrix) by the buffer is written to the target.

The working process of matrix processing is to read in the 32 rows of source matrix data, and then write the 32 rows of data to the target matrix, and so on, until the entire matrix is transposed or moved.The matrix processing accelerator can also prefetch the data to a level 2 Cache by prefetching the source matrix instead of writing the target matrix as needed.

The source matrix involved in transpose or shift may be a small matrix located in a large matrix, so its matrix address may not be completely contiguous, and the addresses between adjacent rows will be spaced, requiring more programming control interfaces to be implemented. The following tables 5-1 through 5-4 illustrate the programming interfaces involved in matrix processing.

Table 5-1 matrix processing programming interface description

| addre ss | The name of the | attribu te | instr uctio ns |
|---|---|---|---|
| 0 x3ff00600 | src_start_addr | RW | The starting address of the source matrix |
| 0 x3ff00608 | dst_start_addr | RW | The starting address of the destination matrix |
| 0 x3ff00610 | The row | RW | The number of elements in a row in the source matrix |
| 0 x3ff00618 | Col. | RW | The number of elements in a column of the source matrix |
| 0 x3ff00620 | length | RW | The row span (bytes) of the large matrix in which the source matrix is located |

| | | | |
|---|---|---|---|
| 0 x3ff00628 | width | RW | The row span (bytes) of the large matrix in which the target matrix is located |
| 0 x3ff00630 | trans_ctrl | RW | Transpose the control register |
| 0 x3ff00638 | trans_status | RO | Transpose the status register |

Table 5-2 matrix processing register address description

| addre ss | The name of the |
|---|---|
| 0 x3ff00600 | Src_start_addr of the zero transpose module |
| 0 x3ff00608 | Dst_start_addr of the zero transpose module |
| 0 x3ff00610 | Row of the zero transpose module |
| 0 x3ff00618 | Col of the 0 transposed module |
| 0 x3ff00620 | Length of the zero transpose module |
| 0 x3ff00628 | Width of the zero transpose module |
| 0 x3ff00630 | Trans_ctrl for the zero transpose module |
| 0 x3ff00638 | Trans_status of the zero transpose module |
| 0 x3ff00700 | Src_start_addr of the # 1 transpose module |
| 0 x3ff00708 | Dst_start_addr of the # 1 transpose module |
| 0 x3ff00710 | Src_row_stride of no. 1 transpose module |
| 0 x3ff00718 | Src_last_row_addr of the # 1 transpose module |
| 0 x3ff00720 | Length of the number 1 transpose module |
| 0 x3ff00728 | Width of no. 1 transpose module |
| 0 x3ff00730 | Trans_ctrl for the # 1 transpose module |
| 0 x3ff00738 | Trans_status for the # 1 transpose module |

Table 5-3 interpretation of the trans_ctrl register

| field | instr uctio ns |
|---|---|
| 0 | Can make a |
| 1 | Whether writing the target matrix is allowed. When the value is 0, the transpose process only prefetches the source matrix, but does not write the target matrix. |
| 2 | After the source matrix is read, whether it is valid or not is interrupted. |
| 3 | After the completion of writing the target matrix, whether it is valid or not, |
| 7.. 4 | Arcmd, read command internal control bit.When arcache is 4 'hf, it must be set to 4' hc.When arcache is another value, it is meaningless. |

| | |
|---|---|
| 11.. 8 | Arcache, read command internal control bit.The cache path is used when 4 'hf, and the uncache path is used when 4' h0.its<br><br>It has no value. |
| 15.. 12 | Awcmd, write command internal control bit.When awcache is 4 'hf, it must be set to 4' hb.It is meaningless when awcache is other values. |
| 19.. 16 | Awcache, write command internal control bit.The cache path is used when 4 'hf, and the uncache path is used when 4' h0.its<br><br>It has no value. |
| 21.. 20 | The element size of the matrix, 00 for 1 byte, 01 for 2 bytes, 10 for 4 bytes, and 11 for 8 bytes |
| 22 | Trans_yes, 1 means transpose;A value of 0 means no transpose (LS3A1000E was previously read-only and supported only)<br><br>Transpose function) |

Table 5-4 definitions of the trans_status register

| field | instructions |
|---|---|
| 0 | The source matrix is read |
| 1 | The target matrix is written |

# 6 Interrupt and communication between processor cores

Loongson 3A1000 implements eight intercore interrupt registers (IPI) for each processor core to support interrupt and communication between processor cores during multi-core BIOS startup and operating system runtime, as shown in tables 6-1 through 6-5 for instructions and addresses.

Table 6-1 registers associated with interrupts between processor cores and their functional descriptions

| The name of the | Read and write access | describe |
|---|---|---|
| IPI_Status | R | The 32-bit status register, where any bit is set to 1 and the corresponding bit is enabled, is set to the processor core INT4 interrupts. |
| IPI_Enable | RW | The 32-bit enable register controls whether the corresponding interrupt bit is valid |
| IPI_Set | W. | 32 position bit register, write 1 to the corresponding bit, then the corresponding STATUS register Bit is set 1 |
| IPI_Clear | W. | 32 bit clear register, write 1 to the corresponding bit, the corresponding STATUS register bit is clear 0 |
| MailBox0 | RW | Cache register, used to pass parameters at startup, 64 or 32 bits Uncache method for access. |
| MailBox01 | RW | Cache register, used to pass parameters at startup, 64 or 32 bits Uncache method for access. |
| MailBox02 | RW | Cache register, used to pass parameters at startup, 64 or 32 bits Uncache method for access. |
| MailBox03 | RW | Cache register, used to pass parameters at startup, 64 or 32 bits Uncache method for access. |

The registers and functions related to interrupts between loongson 3A1000 and processor cores are described as follows:

Table 6- list of intercore interrupt and communication registers for processor no. 2

| The name of the | address | permissions | describe |
|---|---|---|---|
| Core0_IPI_Status | 0 x3ff01000 | R | The IPI_Status register for the number 0 processor core |
| Core0_IPI_Enalbe | 0 x3ff01004 | RW | The IPI_Enalbe register of the number 0 processor core |
| Core0_IPI_Set | 0 x3ff01008 | W. | The IPI_Set register of the number 0 processor core |
| Core0 _IPI_Clear | 0 x3ff0100c | W. | The IPI_Clear register for the number 0 processor core |
| Core0_MailBox0 | 0 x3ff01020 | RW | Register IPI_MailBox0 of the number 0 processor core |
| Core0_ MailBox1 | 0 x3ff01028 | RW | Register IPI_MailBox1 of processor core 0 |
| Core0_ MailBox2 | 0 x3ff01030 | RW | Register IPI_MailBox2 for the number 0 processor core |
| Core0_ MailBox3 | 0 x3ff01038 | RW | Register IPI_MailBox3 of processor core 0 |

Table 6-3 intercore interrupt and communication registers list for processor core no. 1

| The name of the | address | permissions | describe |
|---|---|---|---|
| Core1_IPI_Status | 0 x3ff01100 | R | The IPI_Status register for the number 1 processor core |
| Core1_IPI_Enalbe | 0 x3ff01104 | RW | The IPI_Enalbe register of the number 1 processor core |
| Core1_IPI_Set | 0 x3ff01108 | W. | The IPI_Set register of the number 1 processor core |
| Core1 _IPI_Clear | 0 x3ff0110c | W. | The IPI_Clear register of the number 1 processor core |
| Core1_MailBox0 | 0 x3ff01120 | R | Register IPI_MailBox0 for the number 1 processor core |
| Core1_ MailBox1 | 0 x3ff01128 | RW | Register IPI_MailBox1 of the number 1 processor core |
| Core1_ MailBox2 | 0 x3ff01130 | W. | The IPI_MailBox2 register of the number 1 processor core |
| Core1_ MailBox3 | 0 x3ff01138 | W. | Register IPI_MailBox3 for the number 1 processor core |

Table 6-4 intercore interrupt and communication registers list for no. 2 processor core

| The name of the | address | permissions | describe |
|---|---|---|---|
| Core2_IPI_Status | 0 x3ff01200 | R | The IPI_Status register of the number 2 processor core |
| Core2_IPI_Enalbe | 0 x3ff01204 | RW | The IPI_Enalbe register of the number 2 processor core |
| Core2_IPI_Set | 0 x3ff01208 | W. | The IPI_Set register of the number 2 processor core |
| Core2 _IPI_Clear | 0 x3ff0120c | W. | The IPI_Clear register of the number 2 processor core |
| Core2_MailBox0 | 0 x3ff01220 | R | Register IPI_MailBox0 of the number 2 processor core |
| Core2_ MailBox1 | 0 x3ff01228 | RW | Register IPI_MailBox1 in the number 2 processor core |
| Core2_ MailBox2 | 0 x3ff01230 | W. | The IPI_MailBox2 register of the number 2 processor core |
| Core2_ MailBox3 | 0 x3ff01238 | W. | The IPI_MailBox3 register of the number 2 processor core |

Table 6-5 intercore interrupt and communication registers list for no. 3 processor core

| The name of the | address | permissions | describe |
|---|---|---|---|
| Core3_IPI_Status | 0 x3ff01300 | R | The IPI_Status register of the number 3 processor core |
| Core3_IPI_Enalbe | 0 x3ff01304 | RW | The IPI_Enalbe register of the number 3 processor core |
| Core3_IPI_Set | 0 x3ff01308 | W. | The IPI_Set register of the number 3 processor core |
| Core3 _IPI_Clear | 0 x3ff0130c | W. | The IPI_Clear register of the number 3 processor core |
| Core3_MailBox0 | 0 x3ff01320 | R | Register IPI_MailBox0 for the number 3 processor core |
| Core3_ MailBox1 | 0 x3ff01328 | RW | Register IPI_MailBox1 for the number 3 processor core |
| Core3_ MailBox2 | 0 x3ff01330 | W. | The IPI_MailBox2 register of the number 3 processor core |
| Core3_ MailBox3 | 0 x3ff01338 | W. | The IPI_MailBox3 register of the number 3 processor core |

The above is a list of intercore interrupt related registers for a single-node multiprocessor system consisting of a single loongson 3A1000 chip.When a multi-node cc-numa system is constructed by multi-chip loongson 3A1000 interconnection, each node in the chip corresponds to a global node number of the system, and the IPI register address of the processor core in the node is in a fixed offset relationship according to the base address of the node in

the above table.For example, the IPI_Status address of the processor core of node 0 is 0x3ff01000, and the address of processor 0 of node 1 is 0x10003ff01000, and so on.

# 7 I/O interrupt

The logodson 3A1000 chip supports up to 32 interrupt sources, which are managed in a unified manner, as shown in figure 7-1 below. Any IO interrupt source can be configured to enable or disable, trigger mode, and the target processor core interrupt pin to be routed.

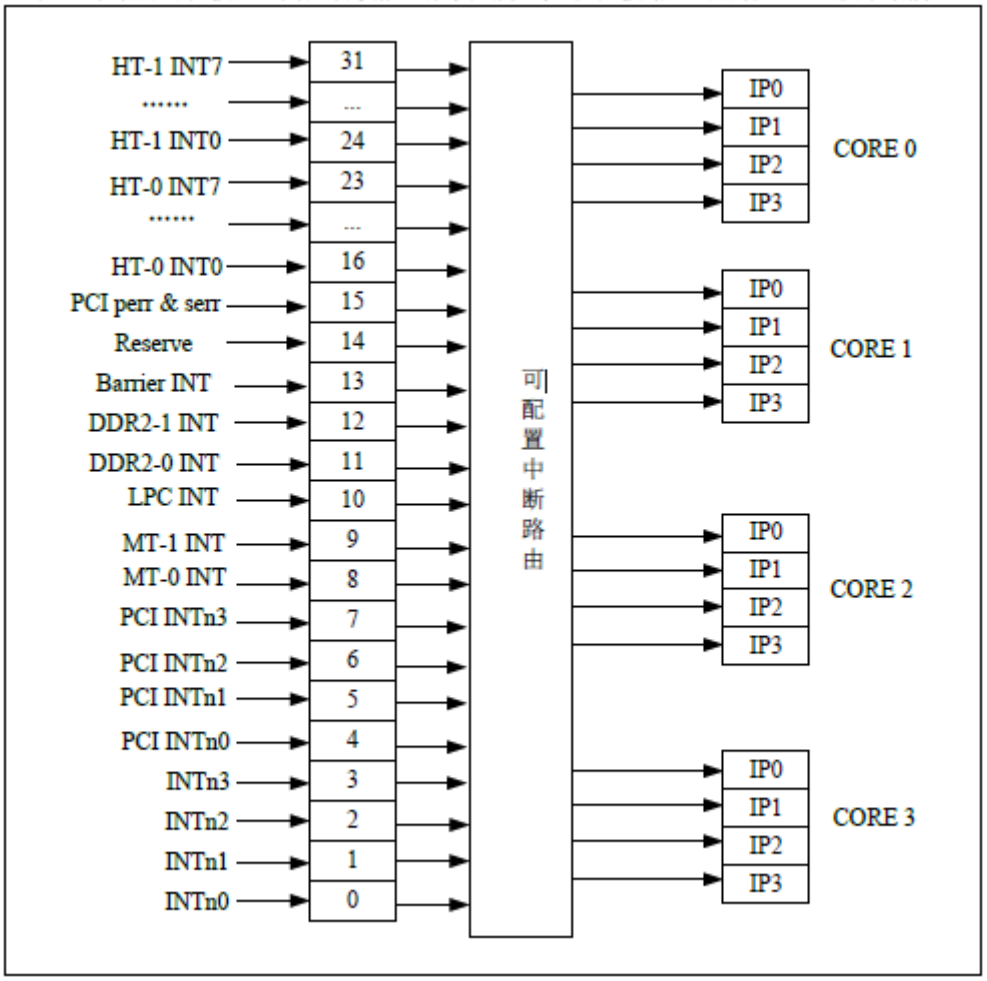



FIG. 7-1 schematic diagram of interrupt 1

The interruption-related configuration registers control the corresponding interrupts in the form of bits. See table 7-1 below for the connection and property configuration of interrupt control bits.The interrupt Enable configuration has three registers: Intenset, Intenclr, and Inten.The Intenset sets the interrupt enable, and the interrupt corresponding to the bit write 1 in the Intenset register is enabled. The Intenclr clears interrupts to enable, and the interrupt corresponding to the Intenclr register write 1 is cleared. The Inten register reads the current status of each interrupt enable.Interrupt signals in the form of pulses (such as PCI_SERR) are selected by the Intedge configuration register, with write 1 for pulse trigger and write 0 for level trigger.The interrupt handler can clear the pulse record by the corresponding bit of the Intenclr.

Table 7-1 interrupt control registers

| A domain | Access properties/default values | | | | |
|---|---|---|---|---|---|
| | Intedge | Inten | Intenset | Intenclr | The interrupt source |
| 3:0 | RW / 0 | R / 0 | W / 0 | W / 0 | Sys_int0-3 |
| 7:4 | RO / 0 | R / 0 | RW / 0 | RW / 0 | PCI_INTn |
| 8 | RO / 0 | R / 0 | RW / 0 | RW / 0 | Matrix_int0 |
| 9 | RO / 1 | R / 0 | RW / 0 | RW / 0 | Matrix_int1 |
| 10 | RO / 1 | R / 0 | RW / 0 | RW / 0 | Lpc |
| 12:11 | RW / 0 | reserve | reserve | reserve | Mc0-1 |
| 13 | RW / 0 | R / 0 | RW / 0 | RW / 0 | The Barrier |
| 14 | RW / 0 | R / 0 | RW / 0 | RW / 0 | reserve |
| 15 | RW / 0 | R / 0 | RW / 0 | RW / 0 | Pci_perr |
| 23:16 | RW / 0 | R / 0 | RW / 0 | RW / 0 | HT0 int0-7 |
| 31:24 | RW / 0 | R / 0 | RW / 0 | RW / 0 | HT1 int0-7 |

Table 7-2 IO control register addresses

| The name of the | Address offset | Describe |
|---|---|---|
| Intisr | 0 x3ff01420 | 32-bit interrupt status register |
| Inten | 0 x3ff01424 | 32-bit interrupt enabled status register |
| Intenset | 0 x3ff01428 | The 32-bit setting enables the register |
| Intenclr | 0 x3ff0142c | 32-bit clear enable register |
| Intedge | 0 x3ff01438 | 32-bit trigger mode register |
| CORE0_INTISR | 0 x3ff01440 | Routing to the 32-bit interrupt state of CORE0 |
| CORE1_INTISR | 0 x3ff01448 | Routing a 32-bit interrupt state to CORE1 |
| CORE2_INTISR | 0 x3ff01450 | Routing a 32-bit interrupt state to CORE2 |
| CORE3_INTISR | 0 x3ff01458 | Routing a 32-bit interrupt state to CORE3 |

With four processor cores integrated into the loson 3A1000, the 32-bit interrupt source described above can be configured to select the target processor core to interrupt.Further, the interrupt source can choose to route to any of the processor core interrupts INT0 through INT3, IP2 through IP5 corresponding to CP0_Status.Each of the 32 I/O interrupt sources corresponds to an 8-bit routing controller, whose format and address are shown in tables 7-3 and 7-4 below.The routing register USES a vector approach for routing, such as 0x48 to indicate routing to INT2 of processor no. 3.

Table 7-3 description of interrupt routing registers

| A domain | Said Ming |
|---|---|
| 3-0 | Routing the processor core vector number |
| The log | Routing the processor core interrupt pin vector number |

Table 7-4 interrupt routing register addresses

| The name of the | Address offset | describe | The name of the | Address offset | describe |
|---|---|---|---|---|---|
| Entry0 | 0 x3ff01400 | Sys_int0 | Entry16 | 0 x3ff01410 | HT0 - int0 |
| Entry1 | 0 x3ff01402 | Sys_int1 | Entry17 | 0 x3ff01411 | HT0 - int1 |
| Entry2 | 0 x3ff01403 | Sys_int2 | Entry18 | 0 x3ff01412 | HT0 - int2 |
| Entry3 | 0 x3ff01404 | Sys_int3 | Entry19 | 0 x3ff01413 | HT0 - int3 |
| Entry4 | 0 x3ff01405 | Pci_int0 | Entry20 | 0 x3ff01414 | HT0 - int4 |
| Entry5 | 0 x3ff01406 | Pci_int1 | Entry21 | 0 x3ff01415 | HT0 - int5 |
| Entry6 | 0 x3ff01407 | Pci_int2 | Entry22 | 0 x3ff01416 | HT0 - int6 |
| Entry7 | 0 x3ff01408 | Pci_int3 | Entry23 | 0 x3ff01417 | HT0 - int7 |
| Entry8 | 0 x3ff01409 | Matrix int0 | Entry24 | 0 x3ff01418 | HT1 - int0 |
| Entry9 | 0 x3ff0140a | Matrix int1 | Entry25 | 0 x3ff01419 | HT1 - int1 |
| Entry10 | 0 x3ff0140b | Lpc int | Entry26 | 0 x3ff0141a | HT1 - int2 |
| Entry11 | 0 x3ff0140c | Mc0 | Entry27 | 0 x3ff0141b | HT1 - int3 |
| Entry12 | 0 x3ff0140d | Mc1 | Entry28 | 0 x3ff0141c | HT1 - int4 |
| Entry13 | 0 x3ff0140e | The Barrier | Entry29 | 0 x3ff0141d | HT1 - int5 |
| Entry14 | 0 x3ff0140f | reserve | Entry30 | 0 x3ff0141e | HT1 - int6 |
| Entry15 | 0 x3ff0140f | Pci_perr/serr | Entry31 | 0 x3ff0141f | HT1 - int7 |

# 8 Ddr2/3 SDRAM controller configuration

The integrated memory controller inside the loongson 3 processor is designed to comply with the ddr2/3 SDRAM industry standard (jesd79-2 and jesd79-3).In the loongson 3 processor, all memory read/write operations are implemented in accordance with jesd79-2b and jesd79-3.

## 8.1 Ddr2/3 SDRAM controller features overview

The loongson 3 processor supports up to 4 CS (implemented by 4 DDR2 SDRAM chip selectors, namely two double-sided memory strips) and contains a total of 18 bit address bus (i.e., 15 bit line and column address bus and 3 bit logical Bank bus).

Loongson 3 processor can adjust the parameter setting of ddr2/3 controller to support the use of different memory chip types.Among them, the maximum number of supported block selections (CS_n) is 4, the number of row addresses (RAS_n) is 15, and the number of column addresses

The number (CAS_n) is 14, and the logical body choice (BANK_n) is 3.The maximum supported address space is 128GB (237).The physical address of the memory request sent by the CPU will be converted as shown in the following figure:

Take the 4GB address space as an example and configure it as follows: slice = 4Bank = 8

Number of row addresses = 12 number of column addresses = 12


Figure 8-1 DDR2 SDRAM column and column address and CPU physical address conversion

The memory control circuit integrated by the loongson 3 processor only accepts memory read/write requests from the processor or external devices. In all memory read/write operations, the memory control circuit is in Slave State.

The memory controller in loongson 3 processor has the following characteristics:

Interface command, read and write data full flow operation

Memory command merge, sort to improve the overall bandwidth

Configure the register read-write port to modify the basic parameters of the memory device

Built - in dynamic delay compensation circuit (DCC) for reliable data sending and receiving

The ECC function can detect 1 - and 2-bit errors in the data path and automatically correct 1 - bit errors

Support 133-400mhz operating frequency

## 8.2 Ddr2/3 SDRAM read operation protocol

The protocol for ddr2/3 SDRAM read operations is shown in figure 11-2. In the figure, the Command (CMD) is given by

RAS_n, CAS_n and WE_n are composed of three signals.For read operations, RAS_n=1, CAS_n = 0, and WE_n =1.

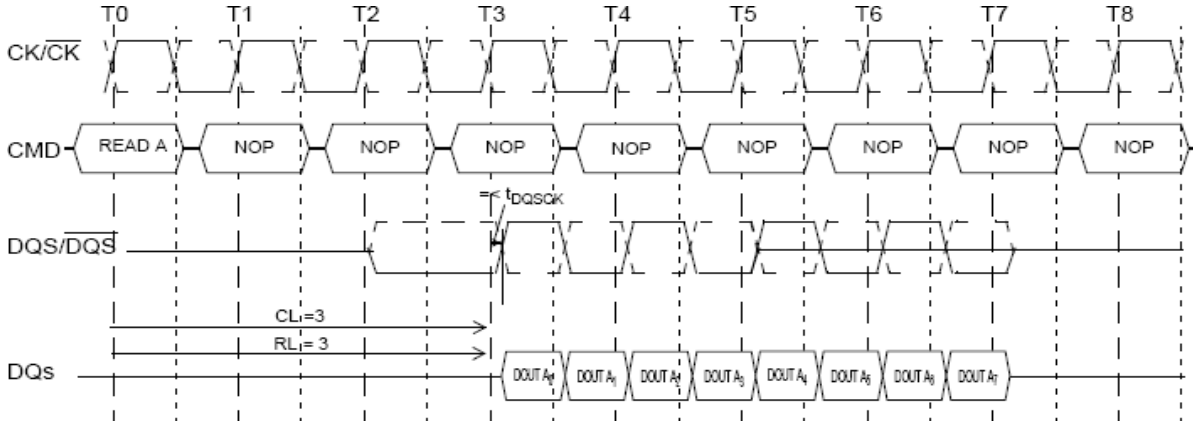
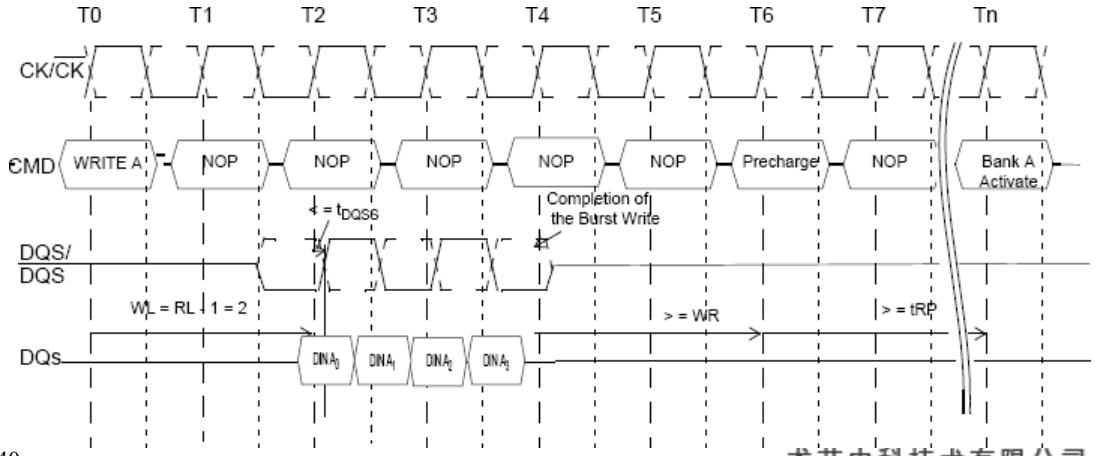



Figure 8-2 DDR2 SDRAM read operation protocol

In the figure above, Cas Latency (CL) = 3, Read Latency (RL) = 3, and Burst Length = 8.

## 8.3 Ddr2/3 SDRAM write operation protocol

The protocol for ddr2/3 SDRAM write operations is shown in figure 11-3.In the figure, the command CMD is composed of RAS_n, CAS_n, and WE_n.For write operations,



RAS_n=1, CAS_n = 0, and WE_n = 0.Also, unlike read operations, write operations require DQM to identify the mask of the write operation, that is, the number of bytes to write.DQM synchronizes with DQs signal in the figure.

Figure 8-3 DDR2 SDRAM write operation protocol

In the figure above, Cas Latency (CL) = 3, Write Latency (WL) = Read Latency (RL) -- 1 = 2, and Burst Length = 4.

## 8.4 Ddr2/3 SDRAM parameter configuration format

Since different types of ddr2/3 SDRAM may be used in the system, ddr2/3 SDRAM needs to be configured after the system is power-on reset.Detailed configuration operations and configuration procedures are specified in jesd79-2b and jesd79-3,

Ddr2/3 is not available until the memory initialization operation for ddr2/3 has been completed.Memory initialization operations are performed in the following order:

System reset, at which point all registers inside the controller are cleared to their original values.

System reset.

Issue 64-bit write instructions to the configuration register address to configure all 180 configuration registers.Now if I write

CTRL_03, where the parameter START should be set to 0.All registers must be configured correctly to work properly.

Sends 64-bit write instructions to the configuration register CTRL_03.The parameter START should be set to 1.After completion, the memory controller will automatically initiate the initialization instruction to the memory.

In the design of loongson 3 processor, the configuration of ddr2/3 SDRAM is carried out after the initialization of the system motherboard and before the use of memory.The specific configuration operation is to the physical address 0x0000 0000 0FF0 0000

The corresponding 180 64-bit registers write the corresponding configuration parameters.A register may contain data for more than one, one, or part of a parameter.The meanings of these configuration registers and the parameters they contain are shown in the following table (all unused bits in registers are reserved bits). A register configuration method based on DDR2 667 is also shown in the table. The specific configuration can be determined according to the actual situation:

Table 8-1 DDR2 SDRAM configuration parameter register format

| The parameter name | position | The default value | The scope of | descr ibe |
|---|---|---|---|---|
| CONF_CTL_00 [63:0] | Offset: 0 x00 | | DDR2 667:0 x0000010000000101 | |
| CONCURRENTAP | 48:48 | 0 x0 | 0 x0-0 x1 | Whether to allow the controller to do auto precharge on one bank, issue a command to the other bank.Note: Some memory chips are not supported |
| BANK_SPLIT_EN | 40:40 | 0 x0 | 0 x0-0 x1 | Whether to allow the command queue reorder logic to split the bank (split) |
| AUTO_REFRESH_MODE | " | 0 x0 | 0 x0-0 x1 | Set auto-refresh to the next burst or the next command boundary |
| AREFRESH | " | 0 x0 | 0 x0-0 x1 | Issue an automatic refresh command into memory (write only) based on the auto_refresh_mode parameter |
| AP | The archite | 0 x0 | 0 x0-0 x1 | Enable memory controller to automatically refresh function, set 1, represents Memory access is in the CLOSE PAGE mode. |
| ADDR_CMP_EN | Hand, | 0 x0 | 0 x0-0 x1 | Whether to allow command queue reordering logic on address conflict detection |
| CONF_CTL_01 [63:0] | Offset: 0 x10 | | DDR2 667:0 x0000010100010000 | |
| FWC | 56:56 | 0 x0 | 0 x0-0 x1 | When this parameter is set, the memory controller will specify the number and data with the xor_check_bits parameter |
| | | | | Xor write to memory (write only) |
| FAST_WRITE | 48:48 | 0 x0 | 0 x0-0 x1 | Whether to allow the controller to turn on the quick write function. After the quick write function is turned on, the controller issues a write command to the memory module without receiving all the write data. |
| ENABLE_QUICK_SREFRESH | 40:40 | 0 x0 | 0 x0-0 x1 | Whether to enable fast self-refresh. When this parameter is enabled, the memory initialization is not completed before it enters a self-refresh state |
| EIGHT_BANK_MODE | " | 0 x0 | 0 x0-0 x1 | Indicates whether the memory module has 8 Banks |
| ECC_DISBALE_W_UC_ERR | " | 0 x0 | 0 x0-0 x1 | Disables setting the memory check code to error when the write operation detects an unrecoverable error |
| DQS_N_EN | The archite | 0 x0 | 0 x0-0 x1 | Set the DQS signal to be single-ended or differential signal. |

| | | | | |
|---|---|---|---|---|
| DLLLOCKREG | 0-0 draw | 0 x0 | 0 x0-0 x1 | Indicates whether the DLL is locked (read only), and only after the DLL is locked can read and write operations initiated to memory reach memory effectively, so you can use the standard to determine the timing of the first write to memory. |
| CONF_CTL_02 [63:0] | Offset: 0 x20 | | DDR2 667:0 x0100010100000000 | |
| PRIORITY_EN | 56:56 | 0 x0 | 0 x0-0 x1 | Whether to enable command queue reorder logic to use priority |
| POWER_DOWN | 48:48 | 0 x0 | 0 x0-0 x1 | When enabling this parameter, the memory controller will close all pages of the memory module with the pre-charge command, making the clock enable signal low and not sending all commands received until this parameter is reloaded<br>The new setting is 0 |
| PLACEMENT_EN | 40:40 | 0 x0 | 0 x0-0 x1 | Enable command reorder logic |
| ODT_ADD_TURN_CLK_EN | " | 0 x0 | 0 x0-0 x1 | Whether to insert a turn-around clock in the middle of a quick back-to-back read or write command for different movie selections.Usually, inserting one such cycle is required for memory. |
| NO_CMD_INIT | " | 0 x0 | 0 x0-0 x1 | During the memory initialization process, whether the memory module is disabled<br>Other commands are issued during tDLL time |
| INTRPTWRITENA | The archite | 0 x0 | 0 x0-0 x1 | Whether to allow the use of autoprecharge commands on the same<br>The other write command of the bank interrupts the previous write command |
| INTRPTREADA | Hand, | 0 x0 | 0 x0-0 x1 | Whether to allow the use of autoprecharge commands on the same<br>The other read command of the bank interrupts the previous read command |
| INTRPTAPBURST | 0-0 draw | 0 x0 | 0 x0-0 x1 | Whether other commands to another bank are allowed to interrupt the current one<br>Auto - a precharge command |
| CONF_CTL_03 [63:0] | Offset: 0 x30 | | DDR2 667:0 x0101010001000000 | |
| SWAP_PORT_RW_SAME_EN | 56:56 | 0 x0 | 0 x0-0 x1 | When swap_en is enabled, this parameter determines whether similar commands on the same port will be swapped |
| SWAP_EN | 48:48 | 0 x0 | 0 x0-0 x1 | When enabling command queue reorder logic, when a high priority command |
| | | | | Whether to exchange the command being executed with the new command when it arrives |

| | | | | |
|---|---|---|---|---|
| START | 40:40 | 0 x0 | 0 x0-0 x1 | Whether to initiate memory initialization. This bit needs to be set after all the parameter configurations have been completed to allow memory to enter the initialization configuration. Configuring a bit without completing the configuration of the other bit is likely to result in memory access errors. |
| SREFRESH | " | 0 x0 | 0 x0-0 x1 | Whether the memory module enters the self-refresh working mode |
| RW_SAME_EN | " | 0 x0 | 0 x0-0 x1 | Whether to consider the same bank in the command queue reorder logic<br>Reorganization of read and write commands |
| REG_DIMM_EN | The archite | 0 x0 | 0 x0-0 x1 | Whether to enable DIMM memory module |
| REDUC | Hand, | 0 x0 | 0 x0-0 x1 | Whether to use only 32-bit wide memory data channels, usually<br>Case, the bit should not be set |
| PWRUP_SREFRESH<br>Associated with | 0-0 draw | 0 x0 | 0 x0-0 x1 | Use the self-refresh command instead of the normal memory initialization command to get out of power mode |
| CONF_CTL_04 [63:0] | Offset: 0 x40 | | DDR2 667:0 x0102010100010101 | |
| RTT_0 | 57:56 | 0 x0 | 0 x0-0 x3 | Enable the on-chip terminal resistance of a memory module.<br>00 - disable<br>Other -- enable, the resistance size is determined by the value in mrs_data |
| CTRL_RAW | 49:48 | 0 x0 | 0 x0-0 x3 | Set the error detection and error correction modes of ECC<br>2 'b00 - ECC is not used<br>2 'b01 -- error reporting, no error correction<br>2 'b10 -- no ECC devices are used<br>2 'b11 -- error correction using ECC |
| AXI0_W_PRIORITY | 41:40 | 0 x0 | 0 x0-0 x3 | Set the AXI0 port write command priority |
| AXI0_R_PRIORITY | 33:32 | 0 x0 | 0 x0-0 x3 | Set the priority of the AXI0 port read command |
| WRITE_MODEREG | " | 0 x0 | 0 x0-0 x1 | Whether or not to write the memory module's EMRS register (write only), the controller will configure the parameters emrs_data and<br>Mrs_data is sent to memory. |
| WRITEINTERP | The archite | 0 x0 | 0 x0-0 x1 | Defines whether a write burst can be interrupted by a read command |
| TREF_ENABLE | Hand, | 0 x0 | 0 x0-0 x1 | Whether to enable the automatic refresh function inside the controller, normally the position should be 1 |

| | | | | |
|---|---|---|---|---|
| TRAS_LOCKOUT | 0-0 draw | 0 x0 | 0 x0-0 x1 | Whether to issue auto-prechareg before tRAS time expires<br>The command |
| CONF_CTL_05 [63:0] | Offset: 0 x50 | | DDR2 667:0 x0700000404050100 | |
| Q_FULLNESS | 58:56 | 0 x0 | 0 x0 0 x7 | Defines how many commands are in the memory controller command queue |
| | | | | Command queue full |
| PORT_DATA_ERROR_TYPE | 50:48 | 0 x0 | 0 x0 0 x7 | Define the data error type (read only) bit 0 - the number of burst data greater than 16 on the memory controller port<br>Bit 1 - write data interleaving<br>Bit 2 - ECC 2 dislocation |
| OUT_OF_RANGE_TYPE | 42:40 | 0 x0 | 0 x0 0 x7 | Define the error type when an out-of-bounds access occurs (read only) |
| MAX_CS_REG | That which | 0 x4 | 0 x0 0 x4 | Define the number of chips used for the controller (read only) |
| COLUMN_SIZE | they | 0 x0 | 0 x0 0 x7 | Setting the difference between the actual number of column addresses and the maximum number of column addresses (14) should be configured according to the specific memory particle.<br>Number of column addresses used in memory = 14-column_size |
| CASLAT | thou | 0 x0 | 0 x0 0 x7 | Set CAS latency value.It should be configured at different operating frequencies depending on the specific memory particles. |
| ADDR_PINS | 10:8 | 0 x0 | 0 x0 0 x7 | Sets the difference between the actual address pin number and the maximum address number (15)<br>Number of address lines used in memory = 15 -- ADDR_PINS |
| CONF_CTL_06[63:0]Offset: 0x60DDR2 667:0x0a04040603040003 | | | | |
| APREBIT | 59:56 | 0 x0 | 0 x0-0 xf | Define which address line to use to issue autoprecharge commands to memory, typically bit 10. |
| WRLAT | 50:48 | 0 x0 | 0 x0 0 x7 | The time (in clock cycles) between the time the write command is sent and the time the first data is received during a write operation, and the time when the corresponding ODT signal is made valid.<br>Note: when WRLAT = (CASLAT_LIN /2), the Add a beat extra delay between CS read and write. |

| | | | | |
|---|---|---|---|---|
| TWTR | 42:40 | 0 x0 | 0 x0 0 x7 | Defines the number of clock cycles required to switch from write to read, which needs to be configured according to the specific memory size and running frequency. |
| TWR_INT | That which | 0 x0 | 0 x0 0 x7 | Define the write recovery time of the memory module, which needs to be configured according to the specific memory particles and running frequency. |
| TRTP | they | 0 x0 | 0 x0 0 x7 | Define the number of read to precharge cycles for the memory module, which need to be configured according to the specific memory size and running frequency. |
| TRRD | thou | 0 x0 | 0 x0 0 x7 | Define active command time intervals to different Banks, which need to be configured according to the specific memory granularity and running frequency. |
| TCKE | The 2-0 | 0 x0 | 0 x0 0 x7 | Define the minimum pulse width of CKE signal |
| CONF_CTL_07[63:0]Offset: 0x70DDR2 667:0x0f0e0200000f0a0a | | | | |
| MAX_ROW_REG | 59:56 | 0 xf | 0 x0-0 xf | System maximum number of line addresses (read only) |
| MAX_COL_REG | spoilers | 0 xe | 0 x0 0 xe | System maximum number of column addresses (read only) |
| INITAREF | 43:40 | 0 x0 | 0 x0-0 xf | Define the number of autorefresh commands that need to be executed when the system is initialized.DDR2 is set to 2, DDR3 is set to 0. |
| CS_MAP | He hath | 0 x0 | 0 x0-0 xf | Define the available slice signal. This parameter should be correctly configured according to the actual number of slices used. Incorrect configuration will lead to incorrect memory access. The four bits of this parameter correspond to CS0- CS3 |
| CASLAT_LIN | 3-0 | 0 x0 | 0 x0-0 xf | When the line running delay on the board is 0.5~1.5 times of the clock cycle of DDR2: CASLAT_LIN = CASLAT×2 is less than 0.5 times: CASLAT_LIN = CASLAT×2-1<br>More than 1.5 times: CASLAT_LIN = CASLAT×2+1<br>(in half a clock cycle) |
| CONF_CTL_08[63:0]Offset: 0x80DDR2 667:0x0804020108040201 | | | | |

| | | | | |
|---|---|---|---|---|
| ODT_WR_MAP_CS3 | 59:56 | 0 x0 | 0 x0-0 xf | When CS3 has a write command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.The four bits of this parameter correspond to CS0- CS3 |
| ODT_WR_MAP_CS2 | spoilers | 0 x0 | 0 x0-0 xf | When CS2 has a write command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.The four bits of this parameter correspond to CS0- CS3 |
| ODT_WR_MAP_CS1 | 43:40 | 0 x0 | 0 x0-0 xf | When CS1 has a write command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.The four bits of this parameter correspond to CS0- CS3 |
| ODT_WR_MAP_CS0 | has | 0 x0 | 0 x0-0 xf | When CS0 is defined to have a write command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.Four bits of the parameter<br>Don't correspond to CS0 minus CS3 |
| ODT_RD_MAP_CS3 | he | 0 x0 | 0 x0-0 xf | When CS3 has a read command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.Four bits of the parameter<br>Don't correspond to CS0 minus CS3 |
| ODT_RD_MAP_CS2 | He hath | 0 x0 | 0 x0-0 xf | When CS2 has a read command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.The four bits of this parameter correspond to CS0- CS3 |

| | | | | |
|---|---|---|---|---|
| ODT_RD_MAP_CS1 | and | 0 x0 | 0 x0-0 xf | When CS1 has a read command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.The four bits of this parameter correspond to CS0- CS3 |
| ODT_RD_MAP_CS0 | 3-0 | 0 x0 | 0 x0-0 xf | When CS0 is defined to have a read command, the ODT terminal resistance of the specified CS will be valid. The specific configuration should refer to the requirements of the corresponding memory particle manual for the ODT configuration.The four bits of this parameter correspond to CS0- CS3 |
| CONF_CTL_09[63:0]Offset: 0x90DDR2 667:0x0000070d00000000 | | | | |
| OCD_ADJUST_PUP_CS0 | 60:56 | 0 x0 | 0 x0 x1f 0 | Set memory module slice to select 0 OCD pull-up adjustment value.The memory controller will issue the OCD tuning command to the memory module based on the value of this parameter at initialization time |
| OCD_ADJUST_PDN_CS0 | 52:48 | 0 x0 | 0 x0 x1f 0 | Set memory module slice to select 0 OCD drop-down adjustment value.The memory controller will issue the OCD tuning command to the memory module based on the value of this parameter at initialization time |
| TRP | 43:40 | 0 x0 | 0 x0-0 xf | Define the number of clock cycles required by the memory module for pre-charge execution, which shall be matched according to the specific memory particles and running frequency<br>Buy. |
| TDAL | has | 0 x0 | 0 x0-0 xf | When the auto-precharge parameter is set, the parameter is defined<br>Auto-precharge and write recovery clock cycles.<br>TDAL = auto-precharge + write recovery<br>This parameter takes effect only after the AP is set. |
| PORT_CMD_ERROR_TYPE | He hath | 0 x0 | 0 x0-0 xf | Type (read only) bit 0 where a<br>command error occurred on the port -<br>data bit too wide<br>Bit 1 - keyword priority operation<br>address not aligned bit 2 - keyword<br>priority operation word count is not 2 |

| | | | | |
|---|---|---|---|---|
| | | | | power<br>Bit 3-narrow transform was wrong |
| CONF_CTL_10[63:0]Offset: 0xa0DDR2 667:0x0000003f3f140612 | | | | |
| COMMAND_AGE_CO<br>without | Go<br>forth | 0 x0 | 0 x0 x3f 0 | Define the command queue reordering logic using the aging algorithm when each<br>The initial aging value of the command |
| AGE_COUNT | A<br>partner | 0 x0 | 0 x0 x3f 0 | Define the command queue reordering logic using the aging algorithm when each |
| | | | | The initial aging value of the command |
| TRC | " | 0 x0 | 0 x0 x1f 0 | Defines the number of clock cycles between active commands on the same bank of the memory module, which needs to be configured according to the specific memory grain and running frequency. |
| TMRD | " | 0 x0 | 0 x0 x1f 0 | Defines the number of clock cycles required to configure the memory module mode register, usually 2 cycles |
| TFAW | 4:0! | 0 x0 | 0 x0 x1f 0 | Define the memory module tFAW parameter used for 8 logical Banks |
| | | | | |
| CONF_CTL_12[63:0]Offset: 0xc0DDR2 667:0x00002c0511000000 | | | | |
| TRFC | 47:40 | 0 x0 | 0 x0-0<br>XFF | Define the number of clock cycles required for the memory module refresh operation, which needs to be configured according to the specific memory particle and running frequency. |
| TRCD_INT | 39:32 | 0 x0 | 0 x0-0<br>XFF | Define the number of clock cycles between the memory module RAS and CAS, which need to be configured according to the specific memory granularity and running frequency. |
| TRAS_MIN | came | 0 x0 | 0 x0-0<br>XFF | Defines the minimum number of clock cycles for the memory module line address valid command |
| OUT_OF_RANGE_LE<br>NGTH | Ephron; | 0 x0 | 0 x0-0<br>XFF | Command length when an out-of-bounds access occurs (read only) |
| ECC_U_SYND | " | 0 x0 | 0 x0-0<br>XFF | Cause of 2bit uncorrectable error (read only) |
| ECC_C_SYND | away | 0 x0 | 0 x0-0<br>XFF | Cause of 1bit correctable error (read only) |

| | | | | |
|---|---|---|---|---|
| | | | | |
| CONF_CTL_17[63:0]Offset: 0x110DDR2 667:0x0000000000000c2d | | | | |
| TREF | 13:0 | 0 x0 | 0 x0 x3ff 0 | Define the clock interval between the memory module's two refresh commands, which need to be configured according to the specific memory particle and running frequency. |
| CONF_CTL_18[63:0]Offset: 0x120DDR2 667:0x001c000000000000 | | | | |
| AXI0_EN_LT_WIDTH_INSTR | 63:48 | 0 x0000 | 0 x0 0 XFFFF | Defines whether AXI0 port receives memory access less than 64 bits wide |
| CONF_CTL_19[63:0]Offset: 0x130DDR2 667:0x6d56000302000000 | | | | |
| TRAS_MAX | 63:48 | 0 x0000 | 0 x0 0 XFFFF | Defines the maximum number of clock cycles for a valid command on a memory module line, which needs to be configured according to the specific memory grain and running frequency. |
| TPDEX | 47:32 | 0 x0000 | 0 x0 0 XFFFF | Defines the number of clock cycles for the memory module power out exit command |
| TDLL | Caused the | 0 x0000 | 0 x0 0 XFFFF | Defines the number of clock cycles required for memory module DLL locking |
| TCPD | 15:0 | 0 x0000 | 0 x0 0 XFFFF | Define the number of clock cycles between the memory module clock and the precharge, depending on the specific memory particle and running frequency Configuration. |
| CONF_CTL_20[63:0]Offset: 0x140DDR2 667:0x0000204002000030 | | | | |
| XOR_CHECK_BITS | 63:48 | 0 x0000 | 0 x0 0 XFFFF | When the FWC parameter is set, the check bit of the next write operation will be written to the memory after the same or the same parameter |
| The VERSION | 47:32 | 0 x2040 | 0 x2040 | Define memory controller version number (read only) |
| TXSR | Caused the | 0 x0000 | 0 x0 0 XFFFF | Defines the number of clock cycles that the memory module needs to self-refresh to exit |
| TXSNR | 15:0 | 0 x0000 | 0 x0 0 XFFFF | Defines the memory module tXSNR parameter |
| CONF_CTL_21[63:0]Offset: 0x150DDR2 667:0x0000000000000000 | | | | |
| ECC_C_ADDR [man] | 60:32 | 0 x0 | 0 x0 x1fffffff 0 f | Record the address information when a 1bit ECC error occurs (read only) |
| ECC_C_ADDR [away] | came | 0 x0000 | 0 x0 x1fffffff 0 f | Record the address information when a 1bit ECC error occurs (read only) |
| TINIT | 23:0 | 0 x0000 | 0 x0 XFFFFF 0 | Define the number of clock cycles required for memory module initialization, which needs to be configured according to the specific memory particle |

| | | | | |
|---|---|---|---|---|
| | | | | and running frequency. It's usually 200us. |
| CONF_CTL_22[63:0]Offset: 0x160DDR2 667:0x0000000000000000 | | | | |
| ECC_U_ADDR took Jeremiah] | Took Jeremiah | 0 x0 | 0 x0 x1fffffff 0 f | Record address information when a 2bit ECC error occurs (read only) |
| ECC_U_ADDR [31:0] | 31:0 | 0 x0 | 0 x0 x1fffffff 0 f | Record address information when a 2bit ECC error occurs (read only) |
| CONF_CTL_23[63:0]Offset: 0x170DDR2 667:0x0000000000000000 | | | | |
| OUT_OF_RANGE_ADDR took Jeremiah] | Took Jeremiah | 0 x0 | 0 x0 x1fffffff 0 f | Record address information when an out-of-bounds access occurs (read only) |
| OUT_OF_RANGE_ADDR [31:0] | 31:0 | 0 x0 | 0 x0 x1fffffff 0 f | Record address information when an out-of-bounds access occurs (read only) |
| CONF_CTL_24[63:0]Offset: 0x180DDR2 667:0x0000000000000000 | | | | |
| PORT_CMD_ERROR_ADDR took Jeremiah] | Took Jeremiah | 0 x0 | 0 x0 x1fffffff 0 f | Record the address information when a command error occurs on the port (read only) |
| PORT_CMD_ERROR_ADDR [31:0] | 31:0 | 0 x0 | 0 x0 x1fffffff 0 f | Record the address information when a command error occurs on the port (read only) |
| CONF_CTL_25[63:0]Offset: 0x190DDR2 667:0x0000000000000000 | | | | |
| ECC_C_DATA [63:32] | 63:32 | 0 x0 | 0 x0 x1fffffff 0 f | Record the data information when a 1bit ECC error occurs (read only) |
| ECC_C_DATA [31:0] | 31:0 | 0 x0 | 0 x0 x1fffffff 0 f | Record the data information when a 1bit ECC error occurs (read only) |
| CONF_CTL_26[63:0]Offset: 0x1a0DDR2 667:0x0000000000000000 | | | | |
| ECC_U_DATA [63:32] | 63:32 | 0 x0 | 0 x0 x1fffffff 0 f | Record the data information when a 2bit ECC error occurs (read only) |
| ECC_U_DATA [31:0] | 31:0 | 0 x0 | 0 x0 x1fffffff 0 f | Record the data information when a 2bit ECC error occurs (read only) |
| CONF_CTL_27[63:0]Offset: 0x1b0DDR2 667:0x0000000000000000 | | | | |
| CKE_DELAY | The 2-0 | 0 x0 | 0 x0 0 x7 | CKE effective delay.<br>Note: used to control the response time of the internal srefresh_enter command, invalid for loongson 3. |
| | | | | |
| CONF_CTL_29[63:0]Offset: 0x1d0DDR2 667:0x0103070400000101 | | | | |

| | | | | |
|---|---|---|---|---|
| TDFI_PHY_WRLAT_BASE | 59:56 | 0 x0 | 0 x0-0 xf | Set the delay to be added for writing data in DDR PHY.For loongson 3 this value should be 2 |
| TDFI_PHY_WRLAT | spoilers | 0 x0 | 0 x0-0 xf | Used to show the number of cycles between the actual write command issue and the write data issue (read only) |
| TDFI_PHY_RDLAT | 44:40 | 0 x0 | 0 x0-0 xf | Sets the number of cycles between the read command and the read data return interval |
| TDFI_CTRLUPD_MIN | has | 0 x4 | 0 x0-0 xf | Save DFI Tctrlup_min time parameter (read only) |
| DRAM_CLK_DISABLE | He hath | 0 x0 | 0 x0-0 xf | Sets whether to output DRAM clock signal, with one chip for each.0: output clock signal;1: do not output clock signal. |
| ODT_ALT_EN | Hand, | 0 x0 | 0 x0-0 x1 | Does it support ODT signal when CAS = 3?<br>Note: invalid for loongson 3 |
| DRIVE_DQ_DQS | 0-0 draw | 0 x0 | 0 x0-0 x1 | Sets whether to drive the data bus when the controller is idle |
| CONF_CTL_30[63:0]Offset: 0x1e0DDR2 667:0x0c2d0c2d0c2d0205 | | | | |
| TDFI_PHYUPD_TYPE0 | 61:48 | 0 x0000 | 0 x0 x3fff 0 | This value is equal to TREF (read only) |
| TDFI_PHYUPD_RESP | 45:32 | 0 x0000 | 0 x0 x3fff 0 | This value is equal to TREF (read only) |
| TDFI_CTRLUPD_MAX | Was a | 0 x0000 | 0 x0 x3fff 0 | This value is equal to TREF (read only) |
| TDFI_RDDATA_EN_BASE | " | 0 x00 | 0 x0 x1f 0 | The basic time between the issuance of the DDR PHY internal read command and the return of the read.For loongson 3 this value is 2 |
| TDFI_RDDATA_EN | 4:0! | 0 x00 | 0 x0 x1f 0 | Used to display the actual number of cycles from the read command to the read data return |
| CONF_CTL_31[63:0]Offset: 0x1f0DDR2 667:0x0020008000000000 | | | | |
| DLL_CTRL_REG_0_0 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The 0th data group (dq7-dq0) DLL control signal |

| | | | | |
|---|---|---|---|---|
| | | | | 24: control the enable signal of the internal DLL, the DLL is valid when it is 0<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360.In loongson 3, this value is usually 1/4, or 8 'h20<br>7:0: controls the precision of the internal DLL, in loongson 3, this<br>The values are usually 8 'h80 |
| DFT_CTRL_REG | away | 0 x00 | 0 x0-0 XFF | Test enabled signal, 0x0 in normal working mode |
| CONF_CTL_32 [63:0] | Offset: 0 x200 | | DDR2 667:0 x0020008000200080 | |
| DLL_CTRL_REG_0_2 | 63:32 | 0 x000000 | 0 x0 0 XFFFFFFFF | The second data group (dq23-dq16) DLL control signal<br>24: control the enable signal of the internal DLL, the DLL is valid when it is 0<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360.In loongson 3, this value is usually 1/4, or 8 'h20<br>7:0: controls the precision of the internal DLL, in loongson 3, this<br>The values are usually 8 'h80 |
| DLL_CTRL_REG_0 _1 | 31:0 | 0 x0000 | 0 x0-0 XFFF FFF ff | The first data group (dq15-dq8) DLL control signal<br>24: control the enable signal of the internal DLL, when it is 0<br>DLL effectively<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360.In loongson 3, this value is usually 1/4, or 8 'h20<br>7:0: controls the precision of the internal DLL in loongson 3<br>In, the value is generally 8 'h80 |
| CONF_CTL_33 [63:0] | Offset: 0 x210 | | DDR2 667:0 x0020008000200080 | |
| | | | | The fourth data group (dq39-dq32) DLL control signal<br>24: control the enable signal of the internal DLL, when it is 0 |

| | | | | |
|---|---|---|---|---|
| | | | | 7:0: controls the precision of the internal DLL. In loongson, this value is usually 8 'h80 3 No. |
| DLL_CTRL_REG_0_3 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The third data group (dq31-dq24) DLL control signal<br>24: control the enable signal of the internal DLL, the DLL is valid when it is 0<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360. In loongson 3, this value is usually 1/4, or 8'h20<br>7:0: controls the precision of the internal DLL, in loongson 3, this<br>The values are usually 8 'h80 |
| CONF_CTL_34 [63:0] | Offset: 0 x220 | | DDR2 667:0 x0020008000200080 | |
| DLL_CTRL_REG_0_6 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The sixth data group (dq55-dq48) DLL control signal<br>24: control the enable signal of the internal DLL, the DLL is valid when it is 0<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360. In loongson 3, this value is usually 1/4, or 8'h20<br>7:0: controls the precision of the internal DLL, in loongson 3, this<br>The values are usually 8 'h80 |
| DLL_CTRL_REG_0_5 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The fifth data group (dq47-dq40) DLL control signal<br>24: control the enable signal of the internal DLL, the DLL is valid when it is 0<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360. In loongson 3, this value is usually 1/4, or 8'h20<br>7:0: controls the precision of the internal DLL, in loongson 3, this<br>The values are usually 8 'h80 |
| CONF_CTL_35 [63:0] | Offset: 0 x230 | | DDR2 667:0 x0020008000200080 | |

| | | | | |
|---|---|---|---|---|
| DLL_CTRL_REG_0_8 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The 8th data group (dq71-dq64) DLL control signal<br>24: control the enable signal of the internal DLL, the DLL is valid when it is 0<br>23:16: controls the phase relationship between write data (DQ) and DQS, and each value is expressed as (1/ precision) * 360. In loongson 3, this value is usually 1/4, or 8'h20<br>7:0: controls the precision of the internal DLL, in loongson 3, this |

:

| | | | | |
|---|---|---|---|---|
| | | | | DLL test control signal, normally is 8 'h0. |
| DLL_CTRL_REG_1_6 | 31:0 | 0 x0000 | 0 x0 0 XFFFFFFFF | The sixth data group DLL control signal<br>15:8: phase delay of DQSn when read data is returned.<br>5:0: DLL test control signal, normally 8 'h0 |
| CONF_CTL_40 [63:0] | Offset: 0 x280 | | DDR2 667:0 x0000000000001e00 | |
| DLL_OBS_REG_0_0 | 33:32 | 0 x0 | 0 x0-0 x3 | The 0th data group DLL output in test mode (read only) |
| DLL_CTRL_REG_1_8 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The eighth data group DLL control signal<br>15:8: phase delay of DQSn when read data is returned.<br>5:0: DLL test control signal, normally 8 'h0 |
| CONF_CTL_41 [63:0] | Offset: 0 x290 | | DDR2 667:0 x0000000000000000 | |
| DLL_OBS_REG_0_2 | 33:32 | 0 x0 | 0 x0-0 x3 | 2 data group DLL output in test mode (read only) |
| DLL_OBS_REG_0_1 | 1-0 | 0 x0 | 0 x0-0 x3 | Data group 1 DLL output in test mode (read only) |
| CONF_CTL_42 [63:0] | Offset: 0 x2a0 | | DDR2 667:0 x0x0000000000000000 | |
| DLL_OBS_REG_0_4 | 33:32 | 0 x0 | 0 x0-0 x3 | The fourth data group DLL output in test mode (read only) |
| DLL_OBS_REG_0_3 | 1-0 | 0 x0 | 0 x0-0 x3 | Third data group DLL output in test mode (read only) |
| CONF_CTL_43 [63:0] | Offset: 0 x2b0 | | DDR2 667:0 x0x0000000000000000 | |
| DLL_OBS_REG_0_6 | 33:32 | 0 x0 | 0 x0-0 x3 | The 6th data group DLL output in test mode (read only) |
| DLL_OBS_REG_0_5 | 1-0 | 0 x0 | 0 x0-0 x3 | The 5th data group DLL output in test mode (read only) |
| CONF_CTL_44 [63:0] | Offset: 0 x2c0 | | DDR2 667:0 x0000000000000000 | |
| DLL_OBS_REG_0_8 | 33:32 | 0 x0 | 0 x0-0 x3 | 8 data group DLL output in test mode (read only) |
| DLL_OBS_REG_0_7 | 1-0 | 0 x0 | 0 x0-0 x3 | 7 data group DLL output in test mode (read only) |
| CONF_CTL_45 [63:0] | Offset: 0 x2d0 | | DDR2 667:0 xf30029470000019d | |
| PHY_CTRL_REG_0_0 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | Delay control for the 0th data set.<br>28: whether to use deburring circuit for reading DQS, refer to gate<br>Whether the signal is delayed through PAD_feedback<br>27: using read FIFO valid signal to automatically control read data back to sampling (1), or using fixed time sampling (0) in |

| | | | | |
|---|---|---|---|---|
| | | | | 18: read whether DQS samples are sampled 1/4 cycle ahead (with clk_wr)<br>Synchronization)<br>17: whether the write data /DQS delay increases the half-cycle delay<br>16: whether the CAS delay is half a cycle<br>15:12: write DQS valid start time, DDR3 should be turned on one cycle earlier than DDR2, and provide particle requirements for Preamble DQS<br>11:8: write DQS valid end time<br>6:4: start time for writing data<br>2:0: the end time of writing data that is valid<br>Pin control signal<br>25:22: corresponding to COMPZCP_dig<br>21:18: corresponding to COMPZCN_dig |
| PAD_CTRL_REG_0 | 25:0 | 0 x0000 | 0 x0 x3fffffff | 17: TQ1v8 of the corresponding pin<br>16: enable signal corresponding to internal feedback pin, low effective<br>15: output enable signal of internal feedback pin, low efficiency<br>14: the output enable signal of the corresponding data gate pin is low effective<br>13: output enable signal of corresponding data shielding pin, low efficiency<br>12: the output enable signal of the corresponding data pin is low effective<br>11: USEPAD corresponding to pin<br>0: use internal reference voltage;<br>1: use external reference voltage.<br>8: enable signal of clock pin {1,3,5}, highly effective<br>7: enable signal of clock pin {0,2,4}, highly effective<br>6: enable signal of corresponding address pin, low effective<br>5: PROGB1v8 of the corresponding pin<br>4: PROGA1v8 corresponding to the pin<br>Used to control pin drive capability<br>3: ODTB of the corresponding pin<br>2: ODTA of the corresponding pin<br>Used to control the size of pin ODT resistance<br>ODTA ODTB DDRII DDRIII<br>1 0 150 120 |

| | | | | |
|---|---|---|---|---|
| | | | | 117560<br>00DisableDisable 1: MODEZI1v8 corresponding to the pin<br>For loongson 3 it should be set to 0.<br>0: DDR1v8 of the corresponding pin<br>0:1.8v mode corresponding to DDRII<br>1:1.5v mode corresponding to DDRIII |
| CONF_CTL_46[63:0]Offset: 0x2e0DDR2 667:0xf3002947f3002947 | | | | |
| PHY_CTRL_REG_0_2 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The second data set delay control.<br>28: whether to use deburring circuit for reading DQS, refer to gate<br>Whether the signal is delayed through PAD_feedback<br>27: using read FIFO valid signal to automatically control read data back to sampling (1), or using fixed time sampling (0) in 26:24<br>26:24: read data returns the sampling completion time, the delay of sampling from the internal clock domain.<br>21: in Read ever, the level of the sampled data bus<br>20: the data effectively controls the level of the signal, which is 0 in loongson 3<br>19: whether to add another cycle to the write data delay<br>18: read whether DQS samples are sampled 1/4 cycle ahead (with clk_wr)<br>Synchronization) |

| | | | | |
|---|---|---|---|---|
| | | | | 26:24: read data returns the sampling completion time, the delay of sampling from the internal clock domain.<br>21: in Read ever, the level of the sampled data bus<br>20: the data effectively controls the level of the signal, which is 0 in loongson 3<br>19: whether to add another cycle to the write data delay<br>18: read whether DQS samples are sampled 1/4 cycle ahead (with clk_wr)<br>Synchronization)<br>17: whether the write data /DQS delay increases the half-cycle delay<br>16: whether the CAS delay is half a cycle<br>15:12: write DQS valid start time, DDR3 should be turned on one cycle earlier than DDR2, and provide particle requirements for Preamble DQS<br>11:8: write DQS valid end time<br>6:4: start time for writing data<br>2:0: the end time of writing data that is valid |
| CONF_CTL_47[63:0]Offset: 0x2f0DDR2 667:0xf3002947f3002947 | | | | |
| | | | | The fourth data group delay control.<br>28: whether to use deburring circuit for reading DQS, refer to gate<br>Whether the signal is delayed through PAD_feedback |

| | | | | |
|---|---|---|---|---|
| | | | | The Preamble of DQS<br><br>11:8: write DQS valid end time<br><br>6:4: start time for writing data<br><br>2:0: the end time of writing data that is valid |
| PHY_CTRL_REG_0_3 | 31:0 | 0 x0000 | 0 x0 0 XFFFFFFFF | The third data set delay control.<br><br>28: whether to use deburring circuit for reading DQS, refer to gate<br><br>Whether the signal is delayed through PAD_feedback<br><br>27: using read FIFO valid signal to automatically control read data back to sampling (1), or using fixed time sampling (0) in 26:24<br><br>26:24: read data returns the sampling completion time, the delay of sampling from the internal clock domain.<br><br>21: in Read ever, the level of the sampled data bus<br><br>20: the data effectively controls the level of the signal, which is 0 in loongson 3<br><br>19: whether to add another cycle to the write data delay<br><br>18: read whether DQS samples are sampled 1/4 cycle ahead (with clk_wr)<br><br>Synchronization)<br><br>17: whether the write data /DQS delay increases the half-cycle delay<br><br>16: whether the CAS delay is half a cycle |

| | | | | |
|---|---|---|---|---|
| | | | | Level level<br><br>20: the data effectively controls the level of the signal, which is 0 in loongson 3<br><br>19: whether to add another cycle to the write data delay<br><br>18: read whether DQS samples are sampled 1/4 cycle ahead (with clk_wr)<br>Synchronization)<br><br>17: whether the write data /DQS delay increases the half-cycle delay<br><br>16: whether the CAS delay is half a cycle<br><br>)<br>15:12: write DQS valid start time, DDR3 should be turned on one cycle earlier than DDR2, and provide particle requirements for Preamble DQS<br>11:8: write DQS valid end time<br><br>6:4: start time for writing data<br><br>2:0: the end time of writing data that is valid |
| | | | | The fifth data group delay control.<br><br>28: whether to use deburring circuit for reading DQS, refer to gate<br>Whether the signal is delayed through PAD_feedback<br><br>27: using read FIFO valid signal to automatically control read data back to sampling (1), or using fixed time sampling (0) in 26:24<br>26:24: read data returns the sampling completion |

| CONF_CTL_49[63:0]Offset: 0x310DDR2 667:0xf3002947f3002947 | | | | |
|---|---|---|---|---|
| PHY_CTRL_REG_0_8 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The 8th data group delay control.<br>28: whether to use deburring circuit for reading DQS, refer to gate<br>Whether the signal is delayed through PAD_feedback<br>27: using read FIFO valid signal to automatically control read data back to sampling (1), or using fixed time sampling (0) in 26:24<br>26:24: read data returns the sampling completion time, the delay of sampling from the internal clock domain.<br>21: in Read ever, the level of the sampled data bus<br>20: the data effectively controls the level of the signal, which is 0 in loongson 3<br>19: whether to add another cycle to the write data delay<br>18: read whether DQS samples are sampled 1/4 cycle ahead (with clk_wr)<br>Synchronization)<br>17: whether the write data /DQS delay increases the half-cycle delay<br>16: whether the CAS delay is half a cycle<br>15:12: write DQS valid start time, DDR3 should be turned on one cycle earlier than DDR2, and provide particle requirements for Preamble DQS<br>11:8: write DQS valid end time<br>6:4: start time for writing data |

| | | | | |
|---|---|---|---|---|
| | | | | Synchronization)<br><br>17: whether the write data /DQS delay increases the half-cycle delay<br><br>16: whether the CAS delay is half a cycle<br><br>15:12: write DQS valid start time, DDR3 should be turned on one cycle earlier than DDR2, and provide particle requirements for Preamble DQS<br>11:8: write DQS valid end time<br><br>6:4: start time for writing data<br><br>2:0: the end time of writing data that is valid |
| CONF_CTL_50[63:0]Offset: 0x320DDR2 667:0x07c0000007c00000 | | | | |
| PHY_CTRL_REG_1_1 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in data group 1 is only enabled when a read is initiated<br>31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: timing control of terminal resistance shutdown<br><br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br><br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br><br>20:16: test signal, normal should be 0 |

| | | | | |
|---|---|---|---|---|
| | | | | 14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| PHY_CTRL_REG_1_0 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in data group 0 is only enabled when a read is initiated<br>31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3 |

| | | | | |
|---|---|---|---|---|
| | | | | 1<br><br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br><br>20:16: test signal, normal should be 0<br><br>14:12: test signal, normal should be 0<br><br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| CONF_CTL_51[63:0]Offset: 0x330DDR2 667:0x07c0000007c00000 | | | | |

| | | | | |
|---|---|---|---|---|
| PHY_CTRL_REG_1_3 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in the third data group is only enabled when a read is initiated<br>31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0<br>14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| PHY_CTRL_REG_1_2 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in the second data set is only enabled when a read is initiated<br>31:28: terminal resistance start time control, from send to read command<br>Each value represents half a period |

| | | | | |
|---|---|---|---|---|
| | | | | 27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0<br>14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| CONF_CTL_52[63:0]Offset: 0x340DDR2 667:0x07c0000007c00000 | | | | |
| PHY_CTRL_REG_1_5 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in data group 5 is only enabled when a read is initiated<br>31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: start time control of terminal resistance<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0 |

| | | | | |
|---|---|---|---|---|
| | | | | 14:12: test signal, normal should be 0<br><br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| PHY_CTRL_REG_1_4 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of PAD in the fourth data group initiates the read operation<br>When done, it will be enabled |

| | | | | |
|---|---|---|---|---|
| | | | | 31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0<br>14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| CONF_CTL_53[63:0]Offset: 0x350DDR2 667:0x07c0000007c00000 | | | | |

| | | | | |
|---|---|---|---|---|
| PHY_CTRL_REG_1_7 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in data group 7 is only enabled when a read is initiated<br>31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0<br>14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |

| | | | | |
|---|---|---|---|---|
| PHY_CTRL_REG_1_6 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | The terminal resistance control of the PAD in data group 6 is only enabled when a read is initiated<br>31:28: start time control of terminal resistance, starting from 4 beats after sending read command, each value represents half period<br>27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0<br>14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| CONF_CTL_54[63:0]Offset: 0x360DDR2 667:0x0800c00507c00000 | | | | |
| PHY_CTRL_REG_2 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | Read and write data delay control<br>27: select the read data buffer type, which defaults to 0<br>26: used to clear the read return buffer, normally 0<br>25: high-speed pin enable, when 1, all signals pass through the pin<br>The outgoing delay is reduced by 1 cycle<br>16:13: set the effective time of reading data and the delay of collecting data from FIFO to return to the controller.If the delay from pin to FIFO increases, this value must also increase<br>8: set whether DQS signal output is in DDR3 mode. In DDR3 mode, Preamble writing DQS will contain a pulse<br>5: test mode signal, normal is 0<br>4: test mode signal, normal is 0 |

| | | | | |
|---|---|---|---|---|
| PHY_CTRL_REG_1_8 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | Terminal resistance control in data group 8<br>The PAD's terminal resistance control is only enabled when a read is initiated<br>31:28: terminal resistance start time control, from send to read command |
| | | | | Each value represents half a period<br>27:24: timing control of terminal resistance shutdown<br>23: effective level control of terminal resistance, for loongson no. 3<br>1<br>22: when the enable signal of the terminal resistance is 1, the dynamic side is used<br>Enable the resistance of the control terminal; At 0, the terminal resistance on the 23rd bit PAD can be either always valid (set 0) or never valid (set 1)<br>21: test signal, normal should be 0<br>20:16: test signal, normal should be 0<br>14:12: test signal, normal should be 0<br>11:8: read sampling delay 1, in which only 1 bit is valid, used to control the closing time of DQS sampling window<br>7:0: read sampling delay is 0, in which only 1 bit is valid for control<br>DQS sampling window opening time |
| CONF_CTL_55[63:0]Offset: 0x370DDR2 667:0x0000000000000000 | | | | |
| PHY_OBS_REG_0_1 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 1 (read only) |
| PHY_OBS_REG_0_0 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signal for testing the 0th data set (read only) |
| CONF_CTL_56[63:0]Offset: 0x380DDR2 667:0x0000000000000000 | | | | |
| PHY_OBS_REG_0_3 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 3 (read only) |
| PHY_OBS_REG_0_2 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 2 (read only) |
| CONF_CTL_57[63:0]Offset: 0x390DDR2 667:0x0000000000000000 | | | | |
| PHY_OBS_REG_0_5 | 63:32 | 0 x000000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 5 (read only) |
| PHY_OBS_REG_0_4 | 31:0 | 0 x0000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 4 (read only) |
| CONF_CTL_58[63:0]Offset: 0x3a0DDR2 667:0x0000000000000000 | | | | |
| PHY_OBS_REG_0_7 | 63:32 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 7 (read only) |
| PHY_OBS_REG_0_6 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 6 (read only) |

| | | | | |
|---|---|---|---|---|
| CONF_CTL_59[63:0]Offset: 0x3b0DDR2 667:0x0000000000000000 | | | | |
| PHY_OBS_REG_0_8 | 31:0 | 0 x00000 | 0 x0 0 XFFFFFFFF | Observation signals for testing data group 8 (read only) |
| | | | | |
| CONF_CTL_114[63:0]Offset: 0x720DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_REQ | 56 | 0 x0 | 0 x0-0 x1 | User requests read-pass sampling training function. (write only) |
| RDLVL_GATE_PREA | 48 | 0 x0 | 0 x0-0 x1 | Open the lead sampling inspection during the read-pass sampling training |
| MBLE_CHECK_EN | | | | |
| RDLVL_GATE_EN | 40 | 0 x0 | 0 x0-0 x1 | Enable Read like a real-time Read through sampling training. After initialization, commands were sent to the particles for DQS sampling window training |
| RDLVL_EN | 32 | 0 x0 | 0 x0-0 x1 | 6. Like a rod, like a rod |
| RDLVL_BEGIN_DELAY_EN | 24 | 0 x0 | 0 x0-0 x1 | 6. Enable Read to look for the data sampling point feature |
| SWLVL_OP_DONE | 8 | 0 x0 | 0 x0-0 x1 | Used to indicate whether the software was ever finished. |
| CONF_CTL_115[63:0]Offset: 0x730DDR2 667:0x0000000000000000 | | | | |
| RDLVL_OFFSET_DIR_7 | 56 | 0 x0 | 0 x0-0 x1 | Data set 7 was reoriented like a medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| RDLVL_OFFSET_DIR_6 | 48 | 0 x0 | 0 x0-0 x1 | Data set 6 was reoriented like a medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| RDLVL_OFFSET_DIR_5 | 40 | 0 x0 | 0 x0-0 x1 | Data set 5 was reoriented like a medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| RDLVL_OFFSET_DIR_4 | 32 | 0 x0 | 0 x0-0 x1 | Data set 4 was reoriented like a medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and is 1 is added. |
| RDLVL_OFFSET_DIR_3 | 24 | 0 x0 | 0 x0-0 x1 | Data set 3 was reoriented like a medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| RDLVL_OFFSET_DIR_2 | 16 | 0 x0 | 0 x0-0 x1 | Data set 2 was reoriented like a medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |

| | | | | |
|---|---|---|---|---|
| RDLVL_OFFSET_DIR_1 | 8 | 0 x0 | 0 x0-0 x1 | Group 1. Read honored to have redirected the midpoint.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| RDLVL_OFFSET_DIR_0 | 0 | 0 x0 | 0 x0-0 x1 | The zeroth data set was reoriented like a midpoint.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| CONF_CTL_116[63:0]Offset: 0x740DDR2 667:0x0100000000000000 | | | | |
| AXI1_PORT_ORDERI | 57:56 | 0 x0 | 0 x0-0 x3 | Whether internal port 1 can be executed out of order is invalid for loongson 3 |
| NG, | | | | |
| AXI0_PORT_ORDERING | 49:48 | 0 x0 | 0 x0-0 x3 | Whether internal port 0 can be executed out of order |
| WRLVL_REQ | 40 | 0 x0 | 0 x0-0 x1 | The user requested that it be written. |
| WRLVL_INTERVAL_CT_EN | 32 | 0 x0 | 0 x0-0 x1 | Make it possible to Write a time interval feature |
| WEIGHTED_ROUND_ROBIN_WEIGHT_SHARING | 24 | 0 x0 | 0 x0-0 x1 | Per-port pair Shared arbitration for WRR |
| WEIGHTED_ROUND_ROBIN_LATENCY_The CONTROL | 16 | 0 x0 | 0 x0-0 x1 | Free-running or limited WRR deposition counters. |
| RDLVL_REQ | 8 | 0 x0 | 0 x0-0 x1 | The user requested that Read ever be trained. |
| RDLVL_OFFSET_DIR_8 | 0 | 0 x0 | 0 x0-0 x1 | Group 8. Read honored to have redirected the medium point.When is 0, the midpoint is calculated as minus rdlvl_offset_delay, and if is 1, then plus. |
| CONF_CTL_117[63:0]Offset: 0x750DDR2 667:0x0100000101020101 | | | | |
| WRLVL_CS | 57:56 | 0 x0 | 0 x0-0 x3 | A chipped signal indicating the current write-like operation |
| SW_LEVELING_MODE | 49:48 | 0 x0 | 0 x0-0 x3 | Define the software as a model of operation |
| RDLVL_CS | 41:40 | 0 x0 | 0 x0-0 x3 | A chip selective signal indicating the current Read polished operation |
| AXI2_W_PRIORITY | 33:32 | 0 x0 | 0 x0-0 x3 | The write priority of internal port 2 is not valid for loongson 3 |
| AXI2_R_PRIORITY | Thus for | 0 x0 | 0 x0-0 x3 | The read priority of internal port 2 is not valid for loongson 3 |
| AXI2_PORT_ORDERING | " | 0 x0 | 0 x0-0 x3 | Can internal port 2 be executed out of order? It is not valid for loongson 3 |

| | | | | |
|---|---|---|---|---|
| AXI1_W_PRIORITY | o | 0 x0 | 0 x0-0 x3 | The write priority of internal port 1 is not valid for loongson 3 |
| AXI1_R_PRIORITY | 1-0 | 0 x0 | 0 x0-0 x3 | The read priority of internal port 1 is not valid for loongson 3 |
| CONF_CTL_118[63:0]Offset: 0x760DDR2 667:0x0303030000020002 | | | | |
| AXI0_PRIORITY2_RELATIVE_PRIORITY | 59:56 | 0 x0 | 0 x0-0 xf | The relative priority of commands with internal port 0 priority 2 |
| AXI0_PRIORITY1_RELATIVE_PRIORITY | spoilers | 0 x0 | 0 x0-0 xf | The relative priority of commands with internal port 0 priority 1 |
| AXI0_PRIORITY0_RELATIVE_PRIORITY | 43:40 | 0 x0 | 0 x0-0 xf | The relative priority of commands with internal port 0 priority 0 |
| ADDRESS_MIRRORING | has | 0 x0 | 0 x0-0 xf | Indicates which slice supports the Address mirroring feature |
| TDFI_DRAM_CLK_DISABLE | they | 0 x0 | 0 x0 0 x7 | Delay setting from internal clock off to external clock off |
| BSTLEN | thou | 0 x0 | 0 x0 0 x7 | Sets the Burst length value sent to the memory module on the controller |
| ZQ_REQ | o | 0 x0 | 0 x0-0 x3 | User requests to start ZQ tuning |
| ZQ_ON_SREF_EXIT | 1-0 | 0 x0 | 0 x0-0 x3 | Defines the mode in which ZQ adjusts the functionality when exiting the self-refresh mode |
| CONF_CTL_119[63:0]Offset: 0x770DDR2 667:0x01010202020203 | | | | |
| AXI2_PRIORITY2_RELATIVE_PRIORITY | 59:56 | 0 x0 | 0 x0-0 xf | Internal port 2 priority 2 relative priority of the command, for Loongson 3 is invalid |
| AXI2_PRIORITY1_RELATIVE_PRIORITY | spoilers | 0 x0 | 0 x0-0 xf | Internal port 2 priority 1 relative priority of the command, for Loongson 3 is invalid |
| AXI2_PRIORITY0_RELATIVE_PRIORITY | 43:40 | 0 x0 | 0 x0-0 xf | Internal port 2 priority 0 relative priority of the command, for Loongson 3 is invalid |
| AXI1_PRIORITY3_RELATIVE_PRIORITY | has | 0 x0 | 0 x0-0 xf | Internal port 1 has a relative priority of 3 commands, for Loongson 3 is invalid |
| AXI1_PRIORITY2_RELATIVE_PRIORITY | he | 0 x0 | 0 x0-0 xf | Internal port 1 priority 2 command relative priority for Loongson 3 is invalid |
| AXI1_PRIORITY1_RELATIVE_PRIORITY | He hath | 0 x0 | 0 x0-0 xf | Internal port 1 priority 1 relative priority of the command, for Loongson 3 is invalid |
| AXI1_PRIORITY0_RELATIVE_PRIORITY | and | 0 x0 | 0 x0-0 xf | Internal port 1 priority 0 relative priority of the command, for Loongson 3 is invalid |
| AXI0_PRIORITY3_RELATIVE_PRIORITY | 3-0 | 0 x0 | 0 x0-0 xf | The relative priority of commands with internal port 0 priority 3 |

| | | | | |
|---|---|---|---|---|
| CONF_CTL_120[63:0]Offset: 0x780DDR2 667:0x0102020400040c01 | | | | |
| TDFI_DRAM_CLK_ENABLE | 59:56 | 0 x0 | 0 x0-0 xf | The delay from the internal clock to the output clock |
| TDFI_CTRL_DELAY | spoilers | 0 x0 | 0 x0-0 xf | The delay between the clock being active and the output command |
| RDLVL_GATE_DQ_ZERO_COUNT | 43:40 | 0 x0 | 0 x0-0 xf | Set the read-through sampling training to represent the zeros from 1 to 0<br>The number |
| RDLVL_DQ_ZERO_Cmount | has | 0 x0 | 0 x0-0 xf | Let's say that when I say Read like this, I mean the number of zeros from 1 to 0 |
| LOWPOWER_REFRESH_ENABLE | he | 0 x0 | 0 x0-0 xf | Enable the refresh function in low power mode |
| DRAM_CLASS | He hath | 0 x0 | 0 x0-0 xf | Defines the controller external memory type<br>110: DDR3<br>100: DDR2 |
| BURST_ON_FLY_BIT | and | 0 x0 | 0 x0-0 xf | The burst-on-fly bit in the pattern configuration emitted to DRAM |
| AXI2_PRIORITY3_RELATIVE_PRIORITY | 3-0 | 0 x0 | 0 x0-0 xf | Internal port 2 priority 3 command relative priority for Loongson 3 is invalid |
| CONF_CTL_121[63:0]Offset: 0x790DDR2 667:0x281900000f000303 | | | | |
| WLMRD | 61:56 | 0 x00 | 0 x0 x3f 0 | From the configuration of DRAM send mode to the latency of Write |
| WLDQSEN | 53:48 | 0 x00 | 0 x0 x3f 0 | From the configuration of DRAM send mode to the design of Write<br>Gate data sampling delay |
| LOWPOWER_CONTROL | 44:40 | 0 x00 | 0 x0 x1f 0 | Low power mode enabled<br>Bit 4: power down<br>Bit 3: power down external<br>Bit 2: self refresh<br>Bit 1: external<br>Bit 0: internal |
| LOWPOWER_AUTO_ENABLE | Took Jeremiah | 0 x00 | 0 x0 x1f 0 | Enable automatically enters low power mode when the controller is idle<br>The control bit is the same as LOWERPOWER_CONTROL |
| ZQCS_CHIP | he | 0 x0 | 0 x0-0 xf | Define the valid slice selection for the next ZQ |
| WRR_PARAM_VALUE_ERR | He hath | 0 x0 | 0 x0-0 xf | Errors/warningsrelatedtotheWRR<br>Parameters. (read-only) |

| | | | | |
|---|---|---|---|---|
| TDFI_WRLVL_DLL | " | 0 x00 | 0 x0-0 XFF | The minimum period between a read operation and a Write update the number of delay lines |
| TDFI_RDLVL_DLL | away | 0 x00 | 0 x0-0 XFF | A minimum period from a Read operation to a Read that updates the number of delay lines |
| CONF_CTL_122[63:0]Offset: 0x7a0DDR2 667:0x0000000000000000 | | | | |
| SWLVL_RESP_6 | 63:56 | 0 x00 | 0 x0-0 XFF | The polished response of dataset 6 |
| SWLVL_RESP_5 | 55:48 | 0 x00 | 0 x0-0 XFF | The polished response of group 5 |
| SWLVL_RESP_4 | 47:40 | 0 x00 | 0 x0-0 XFF | The polished response of group 4 |
| SWLVL_RESP_3 | 39:32 | 0 x00 | 0 x0-0 XFF | The polished response of data group 3 |
| SWLVL_RESP_2 | came | 0 x00 | 0 x0-0 XFF | The polished response of data group 2 |
| SWLVL_RESP_1 | Ephron; | 0 x00 | 0 x0-0 XFF | The polished response of group 1 |
| SWLVL_RESP_0 | " | 0 x00 | 0 x0-0 XFF | The polished response of the zeroth dataset |
| CONF_CTL_123[63:0]Offset: 0x7b0DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | 63:16 | | | |
| SWLVL_RESP_8 | " | 0 x00 | 0 x0-0 XFF | The polished response of data group 8 |
| SWLVL_RESP_7 | away | 0 x00 | 0 x0-0 XFF | The polished response of dataset 7 |
| CONF_CTL_124[63:0]Offset: 0x7c0DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_125[63:0]Offset: 0x7d0DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_CLK_ADJUST_3 | 63:56 | 0 x00 | 0 x0-0 XFF | In the third data set, read the starting value of the sampling training |
| RDLVL_GATE_CLK_ADJUST_2 | 55:48 | 0 x00 | 0 x0-0 XFF | In the second data set, read the starting value of the sampling training |
| RDLVL_GATE_CLK_ADJUST_1 | 47:40 | 0 x00 | 0 x0-0 XFF | In the first data set, read the starting value of the sampling training |
| RDLVL_GATE_CLK_ADJUST_0 | 39:32 | 0 x00 | 0 x0-0 XFF | In the 0th data set, read the starting value of the sampling training |
| CONF_CTL_126[63:0]Offset: 0x7e0DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_CLK_ADJUST_8 | 39:32 | 0 x00 | 0 x0-0 XFF | In the 8th data set, read the starting value of sampling training |
| RDLVL_GATE_CLK_ADJUST_7 | came | 0 x00 | 0 x0-0 XFF | In the seventh data set, read the starting value of the sampling training |
| RDLVL_GATE_CLK_ADJUST_6 | Ephron; | 0 x00 | 0 x0-0 XFF | In the sixth data set, read the starting value of the sampling training |

| | | | | |
|---|---|---|---|---|
| RDLVL_GATE_CLK_ADJUST_5 | " | 0 x00 | 0 x0-0 XFF | In the fifth data set, read the starting value of the sampling training |
| RDLVL_GATE_CLK_ADJUST_4 | away | 0 x00 | 0 x0-0 XFF | In the fourth data set, read the starting value of the sampling training |
| CONF_CTL_127[63:0]Offset: 0x7f0DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_128[63:0]Offset: 0x800DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_129[63:0]Offset: 0x810DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_130[63:0]Offset: 0x820DDR2 667:0x0420000c20400000 | | | | |
| TDFI_WRLVL_RESPLAT | 63:56 | 0 x00 | 0 x0-0 XFF | Write ever achieved the number of cycles that responded effectively |
| TDFI_RDLVL_RESPLAT | 39:32 | 0 x00 | 0 x0-0 XFF | The Read polished the number of cycles that responded effectively |
| REFRESH_PER_ZQ | Ephron; | 0 x00 | 0 x0-0 XFF | Number of refresh commands between automatic ZQCS commands |
| CONF_CTL_131[63:0]Offset: 0x830DDR2 667:0x0000000000000c0a | | | | |
| TMOD | " | 0 x00 | 0 x0-0 XFF | Number of cycles to be idle after DRAM mode configuration |
| CONF_CTL_132[63:0]Offset: 0x840DDR2 667:0x0000640064000000 | | | | |
| AXI1_PRIORITY_RELAX | 49:40 | 0 x000 | 0 x0 x3ff 0 | Counter value that triggers priority control release on internal port 1, yes<br>No effect on loongson 3 |
| AXI0_PRIORITY_RELAX [and] | 33:32 | 0 x0 | 0 x0-0 x3 | The counter value that triggers priority control relaxation on internal port 0 |
| AXI0_PRIORITY_RELAX [away] | came | 0 x00 | 0 x0-0 XFF | The counter value that triggers priority control relaxation on internal port 0 |
| CONF_CTL_133[63:0]Offset: 0x850DDR2 667:0x0000000000000064 | | | | |
| OUT_OF_RANGE_SOURCE_ID | 57:48 | 0 x000 | 0 x0 x3ff 0 | Access address overflow request ID number (read only) |
| ECC_U_ID | 41:32 | 0 x000 | 0 x0 x3ff 0 | Access the ID number (read only) of the 2 dislocation request |
| ECC_C_ID | " | 0 x000 | 0 x0 x3ff 0 | Access the ID number of the 1 dislocation request (read only) |
| AXI2_PRIORITY_RELAX | 9:0 | 0 x000 | 0 x0 x3ff 0 | Counter value that triggers priority control release on internal port 2, yes<br>No effect on loongson 3 |
| CONF_CTL_134[63:0]Offset: 0x860DDR2 667:0x0000004000000000 | | | | |

| | | | | |
|---|---|---|---|---|
| ZQCS | loathsome | 0 x000 | 0 x0 XFFF 0 | Number of cycles required by the ZQCS command |
| PORT_DATA_ERROR_ID | " | 0 x000 | 0 x0 x3ff 0 | ID number of internal port data error request (read only) |
| PORT_CMD_ERROR_ID | 9:0 | 0 x000 | 0 x0 x3ff 0 | Internal port command wrong ID number for request (read only) |
| CONF_CTL_135[63:0]Offset: 0x870DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_136[63:0]Offset: 0x880DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_137[63:0]Offset: 0x890DDR2 667:0x0000000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_138[63:0]Offset: 0x8a0DDR2 667:0x00000000001c001c | | | | |
| LOWPOWER_INTERNAL_CNT | 63:48 | 0 x0000 | 0 x0 0 XFFFF | Counts idle cycles to self-refresh with memory and controller CLK gating. |
| LOWPOWER_EXTERNAL_CNT | 47:32 | 0 x0000 | 0 x0 0 XFFFF | Counts idle cycles to self-refresh with memory clock gating. |
| AXI2_EN_SIZE_LT_WIDTH_INSTR | Caused the | 0 x0000 | 0 x0 0 XFFFF | Enables various narrow accesses on internal port 2, not valid for loongson 3 |
| AXI1_EN_SIZE_LT_WIDTH_INSTR | 15:0 | 0 x0000 | 0 x0 0 XFFFF | Enable internal port 1 on various narrow access for loongson 3 invalid |
| CONF_CTL_139[63:0]Offset: 0x8b0DDR2 667:0x0000000000000000 | | | | |
| LOWPOWER_POWER_DOWN_CNT | 15:0 | 0 x0000 | 0 x0 0 XFFFF | Number of idle cycles before entering Power Down mode |
| LOWPOWER_REFRESH_HOLD | Caused the | 0 x0000 | 0 x0 0 XFFFF | The number of idle cycles before the memory controller re-lock DLL in clock-gated mode |
| LOWPOWER_SELF_REFRESH_CNT | 47:32 | 0 x0000 | 0 x0 0 XFFFF | The number of idle cycles before entering memory self-refresh mode |
| CONF_CTL_140[63:0]Offset: 0x8c0DDR2 667:0x0004000000000000 | | | | |
| OBSOLETE | | | | |
| CONF_CTL_141[63:0]Offset: 0x8d0DDR2 667:0x00000000c8000000 | | | | |
| CKE_INACTIVE [and] | 55:32 | 0 x0000000 | 0 x0 0 XFFFFFFFF | High time interval from output DDR_RESET to CKE |
| CKE_INACTIVE [away] | came | 0 x00 | 0 x0-0 XFF | The interval from the output DDR_RESET value to the CKE value is low |

| | | | | |
|---|---|---|---|---|
| WRLVL_STATUS | 17:0 | 0 x00000 | 0 x0 x3ffff 0 | The last time you ever wrote like this. |
| CONF_CTL_142[63:0]Offset: 0x8e0DDR2 667:0x0000000000000050 | | | | |
| TRST_PWRON | 31:0 | 0 x0000000 | 0 x0 0 XFFFFFFFF | From start effective after 500 beats to DDR_RESET effective<br>The delay of |
| CONF_CTL_143[63:0]Offset: 0x8f0DDR2 667:0x0000000020202080 | | | | |
| DLL_CTRL_REG_2 [32] | " | 0 x0 | 0 x0-0 x1 | Output clock DLL enable signal, highly efficient |
| DLL_CTRL_REG_2 [31:0] | 31:0 | 0 x00000000 | 0 x0 0 XFFFFFFFF | Output clock DLL control<br>31:24: the delay of CLK4 and CLK5 on the output clock DLL<br>23:16: output clock DLL CLK2 and CLK3 delay |
| | | | | 15:8: the delay of CLK0 and CLK1 on the output clock DLL<br>7:0: outputs the precision value on the clock DLL |
| CONF_CTL_144[63:0]Offset: 0x900DDR2 667:0x0000000000000000 | | | | |
| RDLVL_ERROR_STATUS [go forth] | Go forth | 0 x00 | 0 x0 x3f 0 | Indicates the state of the RDLVL when an error occurred |
| RDLVL_ERROR_STATUS [31:0] | 31:0 | 0 x00000000 | 0 x0 0 XFFFFFFFF | Indicates the state of the RDLVL when an error occurred |
| CONF_CTL_145[63:0]Offset: 0x910DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_RESP_MASK [63:32] | 63:32 | 0 x00000000 | 0 x0 0 XFFFFFFFF | Read back mask in sampling training |
| RDLVL_GATE_RESP_MASK [31:0] | 31:0 | 0 x00000000 | 0 x0 0 XFFFFFFFF | Read back mask in sampling training |
| CONF_CTL_146[63:0]Offset: 0x920DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_RESP_MASK (71-64) | away | 0 x00 | 0 x0-0 XFF | Read back mask in sampling training |
| CONF_CTL_147[63:0]Offset: 0x930DDR2 667:0x0000000000000000 | | | | |
| RDLVL_RESP_MASK [63:32] | 63:32 | 0 x00000000 | 0 x0 0 XFFFFFFFF | 6. Read like a rod |
| RDLVL_RESP_MASK [31:0] | 31:0 | 0 x00000000 | 0 x0 0 XFFFFFFFF | 6. Read like a rod |
| CONF_CTL_148[63:0]Offset: 0x940DDR2 667:0x0301010000050500 | | | | |
| TDFI_RDLVL_EN | 59:56 | 0 x0 | 0 x0-0 xf | The smallest number of cycles that ever went from Read to Read |
| W2R_SAMECS_DLY | 50:48 | 0 x0 | 0 x0 0 x7 | An additional delay between write and read for the same selected signal |

| | | | | |
|---|---|---|---|---|
| W2R_DIFFCS_DLY | 42:40 | 0 x0 | 0 x0 0 x7 | An additional delay between write and read for different selective signals |
| LVL_STATUS | That which | 0 x0 | 0 x0 0 x7 | Write as well as Read as well as sample the state of the request for LVL_REQ interrupts (Read only) |
| RDLVL_EDGE | 24 | 0 x0 | 0 x0-0 x1 | In the Read polished operation, indicates that DQS rising edge is efficient or Falling edge effective |
| CKSRX | He hath | 0 x0 | 0 x0-0 x0 | Exit the clock cycle delay in self-refresh mode |
| CKSRE | and | 0 x0 | 0 x0-0 x0 | Clock cycle delay for entering self-refresh mode |
| RDLVL_RESP_MASK [71, 64] | away | 0 x00 | 0 x0-0 XFF | 6. Read like a rod |
| CONF_CTL_149[63:0]Offset: 0x950DDR2 667:0x0000000000000a03 | | | | |
| INT_MASK | From them | 0 x00 | 0 x0 x7ffff 0 | Interrupt mask [18] = or of all interrupt bits; [17] = end of user-initiated DLL synchronization; [16] = change of DLL lock signal (switch between locked and non-locked states); [15] = error of reading sampling clock; [14] = a read-write training operation is completed; [13] = a read-write training request has been initiated; [12] = a writing training result is wrong; [11] = error of a reading sampling training result; [10] = a reading training result is wrong; [9] = ODT enabled, and CAS Latency was 3. [8] = DRAM initialization is completed; [7] = internal port data error; [6] = internal port command error; [5] = multiple ECC two-digit errors found; [4] = finding a two-digit error in ECC once; [3] = multiple ECC one-bit errors found; [2] = one ECC error found; [1] = found multiple visits out of the physical space of memory; [0] = found an access out of the physical space of |

| | | | | |
|---|---|---|---|---|
| | | | | memory |
| TXPDLL | confess | 0 x0000 | 0 x0 0 XFFFF | DRAM TXPDLL parameter in cycles. |
| TDFI_WRLVL_EN | 3-0 | 0 x0 | 0 x0-0 xf | Write like the number of cycles that you can Write like the minimum number of read operations |
| CONF_CTL_150[63:0]Offset: 0x960DDR2 667:0x0604000000000000 | | | | |
| RDLAT_ADJ | 60:56 | 0 x00 | 0 x0 x1f 0 | PHY read delay period |
| WRLAT_ADJ | spoilers | 0 x0 | 0 x0-0 xf | PHY write delay period |
| SWLVL_START | 40 | 0 x0 | 0 x0-0 x1 | Like a chip, like a chip. The software worked like a chip. |
| SWLVL_LOAD | 32 | 0 x0 | 0 x0-0 x1 | Like a chip, like a chip. Like a chip. Like a chip. |
| SWLVL_EXIT | 24 | 0 x0 | 0 x0-0 x1 | Like a winnow, like a winnow. |
| INT_STATUS | 18:0 | 0 x00 | 0 x0 x7ffff 0 | Interrupt status (read only)<br>[18] = or of all interrupt bits;<br>[17] = end of user-initiated DLL synchronization; |

| | | | | |
|---|---|---|---|---|
| | | | | [16] = change of DLL lock signal (switch between locked and non-locked states);<br>[15] = error of reading sampling clock;<br>[14] = a read-write training operation is completed;<br>[13] = a read-write training request has been initiated; [12] = a writing training result is wrong;<br>[11] = error of a reading sampling training result; [10] = a reading training result is wrong;<br>[9] = ODT enabled, and CAS Latency was 3. [8] = DRAM initialization is completed;<br>[7] = internal port data error;<br>[6] = internal port command error;<br>[5] = multiple ECC two-digit errors found; [4] = finding a two-digit error in ECC once; [3] = multiple ECC one-bit errors found; [2] = one ECC error found;<br>[1] = found multiple visits out of the physical space of memory;<br>[0] = found an access out of the physical space of memory |
| CONF_CTL_151[63:0]Offset: 0x970DDR2 667:0x000000000003e805 | | | | |
| CONCURRENTAP_WR_ONLY | 56 | 0 x0 | 0 x0-0 x1 | Whether to prevent concurrent auto-precharge operations by waiting for write recovery time before read operation after write operation |
| CKE_STATUS | 48 | 0 x0 | 0 x0-0 x1 | Indicates CKE_STATUS (read only) |

| | | | | |
|---|---|---|---|---|
| INT_ACK | Even as | 0 x00 | 0 x0 x3ffff 0 | Break clear (write only)<br>[17] = end of user-initiated DLL synchronization;<br>[16] = change of DLL lock signal (switch between locked and non-locked states);<br>[15] = error of reading sampling clock;<br>[14] = a read-write training operation is completed;<br>[13] = a read-write training request has been initiated; [12] = a writing training result is wrong;<br>[11] = error of a reading sampling training result;<br>[10] = a reading training result is wrong; |
| | | | | [9] = ODT enabled, and CAS Latency was 3. [8] = DRAM initialization is completed;<br>[7] = internal port data error;<br>[6] = internal port command error;<br>[5] = multiple ECC two-digit errors found; [4] = finding a two-digit error in ECC once; [3] = multiple ECC one-bit errors found; [2] = one ECC error found;<br>[1] = found multiple visits out of the physical space of memory;<br>[0] = found an access out of the physical space of memory |
| DLL_RST_DELAY | confes s | 0 x0000 | 0 x0 0 XFFFF | DLL reset minimum number of cycles |
| DLL_RST_ADJ_DLY | away | 0 x00 | 0 x0-0 XFF | Configure the minimum number of cycles from the DLL precision to the end of the DLL reset |
| CONF_CTL_152[63:0]Offset: 0x980DDR2 667:0x0001010001000101 | | | | |
| ZQ_IN_PROGRESS | 56 | 0 x0 | 0 x0-0 x1 | Indicates that the ZQ operation is in progress (read only) |
| ZQCS_ROTATE | 48 | 0 x0 | 0 x0-0 x1 | Enables ZQCS (short ZQ) to be calibrated in turn.When the bit is 0, each ZQCS request command will correct all the slices in the system. When the position is 1, the system will correct only one slice when each ZQCS command comes, and the system will correct all the slices in turn. The ZQCS and REFRESH_PER_ZQ parameters should be set to match that bit |

| | | | | |
|---|---|---|---|---|
| WRLVL_REG_EN | 40 | 0 x0 | 0 x0-0 x1 | Enables you to write the wrlvl_delay register |
| WRLVL_EN | 32 | 0 x0 | 0 x0-0 x1 | Leverage the controller's Write capability |
| RESYNC_DLL_PER_AREF_EN | 24 | 0 x0 | 0 x0-0 x1 | Enables DLL automatic synchronization after each refresh command |
| RESYNC_DLL | 16 | 0 x0 | 0 x0-0 x1 | Issue a DLL synchronization command (write only) |
| RDLVL_REG_EN | 8 | 0 x0 | 0 x0-0 x1 | Enable to write the rdlvl_delay register |
| RDLVL_GATE_REG_EN | 0 | 0 x0 | 0 x0-0 x1 | Enable to write rdlvl_gate_delay register |
| CONF_CTL_153[63:0]Offset: 0x990DDR2 667:0x0101020202010100 | | | | |
| W2W_SAMECS_DLY | 58:56 | 0 x0 | 0 x0 0 x7 | The number of additional delay clock cycles from a write command selected on the same chip to a write command |
| W2W_DIFFCS_DLY | 50:48 | 0 x0 | 0 x0 0 x7 | The number of additional delay clock cycles from the write command to the write command selected for different slices |
| TBST_INT_INTERVAL | 42:40 | 0 x0 | 0 x0 0 x7 | Number of DRAM burst interrupt interval cycles |

| | | | | |
|---|---|---|---|---|
| R2W_SAMECS_DLY | That which | 0 x0 | 0 x0 0 x7 | The number of additional delay clock cycles from a read command to a write command selected for the same slice |
| R2W_DIFFCS_DLY | they | 0 x0 | 0 x0 0 x7 | The number of additional delay clock cycles from the read command to the write command selected for different slices |
| R2R_SAMECS_DLY | thou | 0 x0 | 0 x0 0 x7 | The number of additional delay clock cycles from the read command to the read command selected for the same slice |
| R2R_DIFFCS_DLY | 10:8 | 0 x0 | 0 x0 0 x7 | The number of additional delay clock cycles from the read command to the read command selected from different slices |
| AXI_ALIGNED_STROBE_DISABLE | The 2-0 | 0 x0 | 0 x0 0 x7 | When a transaction of a AXI port has one of the following characteristics: 1. The start and end addresses of the transaction are aligned according to the user word; 2.2. The transaction length is one user word (128 bits). Each AXI Strobe is not allowed to have a corresponding AXI port.<br>When set to 0, write operations are performed in read-modification-write order.<br>When set to 1, the write operation is treated as a standard write operation (non Read-modification-write order) |
| CONF_CTL_154[63:0]Offset: 0x9a0DDR2 667:0x0707040200060100 | | | | |
| TDFI_WRLVL_LOAD | 63:56 | 0 x0 | 0 x0-0 XFF | Write down the minimum number of clock cycles from the first write down to the first Load |
| TDFI_RDLVL_LOAD | 55:48 | 0 x0 | 0 x0-0 XFF | Read like the minimum number of clock cycles from the first read like the Load command |
| TCKESR | 44:40 | 0 x0 | 0 x0 x1f 0 | The minimum number of clock cycles in which CKE remains low from refresh entry to exit |
| TCCD | Took Jeremiah | 0 x0 | 0 x0 x1f 0 | The delay from the case # to the case # command |

| | | | | |
|---|---|---|---|---|
| TDFI_WRLVL_WW | " | 0 x0 | 0 x0 x3ff 0 | The minimum number of clock cycles between two consecutive times |
| TDFI_RDLVL_RR | 9:0 | 0 x0 | 0 x0 x3ff 0 | The minimum number of clock cycles between two consecutive times |
| CONF_CTL_156[63:0]Offset: 0x9c0DDR2 667:0x0a52000000000000 | | | | |
| MR0_DATA_0 | 62:48 | 0 x0 | 0 x0 x7fff 0 | The mode register 0 configuration value corresponding to the slice selection 0 |
| TDFI_PHYUPD_TYPE 3 | 45:32 | 0 x0 | 0 x0 x3fff 0 | Save DFI Tphyupd_type3 parameter (read only) |
| TDFI_PHYUPD_TYPE 2 | Was a | 0 x0 | 0 x0 x3fff 0 | Save DFI Tphyupd_type2 parameter (read only) |
| TDFI_PHYUPD_TYPE 1 | 13:0 | 0 x0 | 0 x0 x3fff 0 | Save DFI Tphyupd_type1 parameter (read only) |
| CONF_CTL_157[63:0]Offset: 0x9d0DDR2 667:0x00440a520a520a52 | | | | |
| MR1_DATA_0 | 62:48 | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 1 corresponding to shard selection 0 |
| MR0_DATA_3 | 46:32 | 0 x0 | 0 x0 x7fff 0 | The mode register 0 configuration value corresponding to the slice selection 3 |
| MR0_DATA_2 | and | 0 x0 | 0 x0 x7fff 0 | The mode register 0 configuration value corresponding to slice 2 |
| MR0_DATA_1 | 14:0 | 0 x0 | 0 x0 x7fff 0 | The mode register 0 configuration value corresponding to slice 1 |
| CONF_CTL_158[63:0]Offset: 0x9e0DDR2 667:0x0000004400440044 | | | | |
| MR2_DATA_0 | 62:48 | 0 x0 | 0 x0 x7fff 0 | The mode register 2 configuration value corresponding to the slice selection 0 |
| MR1_DATA_3 | 46:32 | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 1 corresponding to TAB 3 |
| MR1_DATA_2 | and | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 1 corresponding to chip selection 2 |
| MR1_DATA_1 | 14:0 | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 1 corresponding to chip selection 1 |
| CONF_CTL_159[63:0]Offset: 0x9f0DDR2 667:0x0000000000000000 | | | | |
| MR3_DATA_0 | 62:48 | 0 x0 | 0 x0 x7fff 0 | The mode register 3 configuration value corresponding to the slice selection 0 |
| MR2_DATA_3 | 46:32 | 0 x0 | 0 x0 x7fff 0 | The mode register 2 configuration value corresponding to TAB 3 |
| MR2_DATA_2 | and | 0 x0 | 0 x0 x7fff 0 | The mode register 2 configuration value corresponding to slice 2 |
| MR2_DATA_1 | 14:0 | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 2 corresponding to chip selection 1 |
| CONF_CTL_160[63:0]Offset: 0xa00DDR2 667:0x00ff000000000000 | | | | |
| DFI_WRLVL_MAX_DE LAY | 63:48 | 0 x0 | 0 x0 0 XFFFF | Like the largest delay line ever used by Hareware Write series |
| MR3_DATA_3 | 46:32 | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 3 corresponding to TAB 3 |

| | | | | |
|---|---|---|---|---|
| MR3_DATA_2 | and | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 3 corresponding to slice 2 |
| MR3_DATA_1 | 14:0 | 0 x0 | 0 x0 x7fff 0 | The configuration value of mode register 3 corresponding to chip selection 1 |
| CONF_CTL_161[63:0]Offset: 0xa10DDR2 667:0x0000000000000000 | | | | |
| RDLVL_BEGIN_DELA | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the third data set, Read ever went from the first 1 to |

| | | | | |
|---|---|---|---|---|
| Y_3 | | | | Number of delay units for 0 |
| RDLVL_BEGIN_DELAY_2 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the second data set, Read ever went from the first 1 to<br>Number of delay units for 0 |
| RDLVL_BEGIN_DELAY_1 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the first data set, Read ever went from the first 1 to<br>Number of delay units for 0 |
| RDLVL_BEGIN_DELAY_0 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the zeroth data set, Read ever went from the first one to<br>Number of delay units for 0 |
| CONF_CTL_162[63:0]Offset: 0xa20DDR2 667:0x0000000000000000 | | | | |
| RDLVL_BEGIN_DELAY_7 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the 7 data set, Read ever went from the first 1 to<br>Number of delay units for 0 |
| RDLVL_BEGIN_DELAY_6 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the 6th data set, Read ever went from the first 1 to<br>Number of delay units for 0 |
| RDLVL_BEGIN_DELAY_5 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the 5th data set, Read ever went from the first 1 to<br>Number of delay units for 0 |
| RDLVL_BEGIN_DELAY_4 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In data set 4, Read ever went from the first 1 to<br>Number of delay units for 0 |
| CONF_CTL_163[63:0]Offset: 0xa30DDR2 667:0x0000000000000000 | | | | |
| RDLVL_DELAY_2 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the second data set, Read imported the number of delay elements used |
| RDLVL_DELAY_1 | 47:32 | 0 x0 | 0 x0 0 XFFFF | The number of delay elements that Read used in the first data set |
| RDLVL_DELAY_0 | Caused the | 0 x0 | 0 x0 0 XFFFF | In data set 0, Read imported a delay unit<br>The number of |
| RDLVL_BEGIN_DELAY_8 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the 8th data set, Read ever went from the first 1 to<br>Number of delay units for 0 |
| CONF_CTL_164[63:0]Offset: 0xa40DDR2 667:0x0000000000000000 | | | | |
| RDLVL _DELAY_6 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In data set 6, Read imported the number of delay elements used |
| RDLVL_DELAY_5 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In data set 5, Read imported a delay unit<br>The number of |

| | | | | |
|---|---|---|---|---|
| RDLVL_DELAY_4 | Caused the | 0 x0 | 0 x0 0 XFFFF | In data set 4, Read imported a delay unit The number of |
| RDLVL_DELAY_3 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In data set 3, Read imported the number of delay elements used |
| CONF_CTL_165[63:0]Offset: 0xa50DDR2 667:0x0000000000000000 | | | | |
| RDLVL_END_DELAY_1 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the first data set, Read ever went from the first zero to zero The number of delay units of 1 |
| RDLVL_END_DELAY_0 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the zeroth data set, Read ever went from the first zero to zero The number of delay units of 1 |
| RDLVL_DELAY_8 | Caused the | 0 x0 | 0 x0 0 XFFFF | Number of delay units used by Read ever in data set 8 |
| RDLVL_DELAY_7 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In data set 7, Read imported the number of delay elements used |
| CONF_CTL_166[63:0]Offset: 0xa60DDR2 667:0x0000000000000000 | | | | |
| RDLVL_END_DELAY_5 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the fifth data set, Read ever went from the first zero to zero The number of delay units of 1 |
| RDLVL_END_DELAY_4 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the fourth data set, Read ever went from the first zero to zero The number of delay units of 1 |
| RDLVL_END_DELAY_3 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the third data set, Read ever went from the first zero to zero The number of delay units of 1 |
| RDLVL_END_DELAY_2 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the second data set, Read ever went from the first zero to zero The number of delay units of 1 |
| CONF_CTL_167[63:0]Offset: 0xa70DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_DELAY _0 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the data group 0, the number of delay units from the sampling time to the rising edge of the selected communication number |
| RDLVL_END_DELAY_8 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the 8th data set, Read ever went from the first zero to zero The number of delay units of 1 |

| | | | | |
|---|---|---|---|---|
| RDLVL_END_DELAY_7 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the 7 data set, Read ever went from the first zero to zero<br>The number of delay units of 1 |
| RDLVL_END_DELAY_6 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the sixth data set, Read ever went from the first zero to zero<br>The number of delay units of 1 |
| CONF_CTL_168[63:0]Offset: 0xa80DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_DELAY_4 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the fourth data set, the sampling time is extended to the rising edge of the selected communication number<br>Number of late units |
| RDLVL_GATE_DELAY_3 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the third data group, the sampling time is extended to the rising edge of the selected communication number<br>Number of late units |
| RDLVL_GATE_DELAY_2 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the second data group, the number of delay units from the sampling time to the rising edge of the selected communication number |
| RDLVL_GATE_DELAY_1 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the first data group, the number of delay units from the sampling time to the rising edge of the selected communication number |
| CONF_CTL_169[63:0]Offset: 0xa90DDR2 667:0x0000000000000000 | | | | |
| RDLVL_GATE_DELAY_8 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the 8th data group, the number of delay units from the sampling time to the rising edge of the selected communication number |
| RDLVL_GATE_DELAY_7 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the 7th data group, the number of delay units from the sampling time to the rising edge of the selected communication number |
| RDLVL_GATE_DELAY_6 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the sixth data group, the number of delay units from the sampling time to the rising edge of the selected communication number |
| RDLVL_GATE_DELAY_5 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the fifth data group, the sampling time is extended to the rising edge of the selected communication number<br>Number of late units |
| CONF_CTL_170[63:0]Offset: 0xaa0DDR2 667:0x0000ffff00000010 | | | | |

| | | | | |
|---|---|---|---|---|
| RDLVL_MIDPOINT_DELAY_0 | 63:48 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read polished the module, it was equal to the median between rdlvl_begin_delay_0 and rdlvl_end_delay_0. Otherwise, it was rdlvl_delay_0. |
| RDLVL_MAX_DELAY | 47:32 | 0 x0 | 0 x0 0 XFFFF | 6. The largest number of delay lines ever recorded by Read |
| RDLVL_GATE_REFRESH_INTERVAL | Caused the | 0 x0 | 0 x0 0 XFFFF | Maximum number of refresh commands between two automatic Gate Training (should be set to 0) |
| RDLVL_GATE_MAX_DELAY | 15:0 | 0 x0 | 0 x0 0 XFFFF | The maximum number of sampling delay lines |
| CONF_CTL_171[63:0]Offset: 0xab0DDR2 667:0x0000000000000000 | | | | |
| RDLVL_MIDPOINT_DELAY_4 | 63:48 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read learned like a rod, it was like Rdlvl_begin_delay_4 and rdlvl_end_delay_4, otherwise, equal to rdlvl_delay_4(read only) |
| RDLVL_MIDPOINT_DELAY_3 | 47:32 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read polished the module, it was equal to the median between rdlvl_begin_delay_3 and rdlvl_end_delay_3. Otherwise, it was rdlvl_delay_3. |
| RDLVL_MIDPOINT_DELAY_2 | Caused the | 0 x0 | 0 x0 0 XFFFF | When the Hardware read cultivated the module, it was equal to the median between rdlvl_begin_delay_2 and rdlvl_end_delay_2. Otherwise, it was rdlvl_delay_2. |
| RDLVL_MIDPOINT_DELAY_1 | 15:0 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read polished the module, it was equal to the median between rdlvl_begin_delay_1 and rdlvl_end_delay_1. Otherwise, it was rdlvl_delay_1. |
| CONF_CTL_172[63:0]Offset: 0xac0DDR2 667:0x0000000000000000 | | | | |
| RDLVL_MIDPOINT_D | 63:48 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read learned like a rod, it was like |
| ELAY_8 | | | | Rdlvl_begin_delay_8 and rdlvl_end_delay_8, otherwise, equal to rdlvl_delay_8(read only) |
| RDLVL_MIDPOINT_DELAY_7 | 47:32 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read polished the module, it was equal to the middle of rdlvl_begin_delay_7 and rdlvl_end_delay_7. Otherwise, it was rdlvl_delay_7. |

| | | | | |
|---|---|---|---|---|
| RDLVL_MIDPOINT_DELAY_6 | Caused the | 0 x0 | 0 x0 0 XFFFF | When the Hardware read cultivated the module, it was equal to the middle of rdlvl_begin_delay_6 and rdlvl_end_delay_6. Otherwise, it was rdlvl_delay_6. |
| RDLVL_MIDPOINT_DELAY_5 | 15:0 | 0 x0 | 0 x0 0 XFFFF | When the Hardware read polished the module, it was equal to the median between rdlvl_begin_delay_5 and rdlvl_end_delay_5. Otherwise, it was rdlvl_delay_5. |
| CONF_CTL_173[63:0]Offset: 0xad0DDR2 667:0x0000000000000000 | | | | |
| RDLVL_OFFSET_DELAY_3 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the third data set, the offset from the middle point of Read |
| RDLVL_OFFSET_DELAY_2 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the second data set, the offset from the middle point of Read |
| RDLVL_OFFSET_DELAY_1 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the first data set, the offset from the middle point of Read |
| RDLVL_OFFSET_DELAY_0 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the zeroth data set, the offset from the Read ever midpoint |
| CONF_CTL_174[63:0]Offset: 0xae0DDR2 667:0x0000000000000000 | | | | |
| RDLVL_OFFSET_DELAY_7 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the 7th data set, the offset from the middle point of Read |
| RDLVL_OFFSET_DELAY_6 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the sixth data set, the offset from the middle point of Read |
| RDLVL_OFFSET_DELAY_5 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the 5th data set, the offset from the middle point of Read |
| RDLVL_OFFSET_DELAY_4 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the fourth data set, the offset from the middle point of Read |
| CONF_CTL_175[63:0]Offset: 0xaf0DDR2 667:0x0000000000000000 | | | | |
| WRLVL_DELAY_1 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the first data group, control the number of write DQS via DLL delays |
| WRLVL_DELAY_0 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the 0th data group, control the number of write DQS via DLL delays |
| RDLVL_REFRESH_INTERVAL | Caused the | 0 x0 | 0 x0 0 XFFFF | Ever the maximum number of refresh commands between two auto Read instances<br>(should be set to 0) |
| RDLVL_OFFSET_DEL | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the 8th data set, the offset from the middle point of Read |
| AY_8 | | | | |
| CONF_CTL_176[63:0]Offset: 0xb00DDR2 667:0x0000000000000000 | | | | |
| WRLVL_DELAY_5 | 63:48 | 0 x0 | 0 x0 0 XFFFF | In the fifth data group, control the number of write DQS via DLL delays |

| | | | | |
|---|---|---|---|---|
| WRLVL_DELAY_4 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the fourth data set, control the number of write DQS via DLL delays |
| WRLVL_DELAY_3 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the third data group, control the number of write DQS via DLL delays |
| WRLVL_DELAY_2 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the second data set, control the number of write DQS via DLL delays |
| CONF_CTL_177[63:0]Offset: 0xb10DDR2 667:0x0000000000000000 | | | | |
| WRLVL_REFRESH_INTERVAL | 63:48 | 0 x0 | 0 x0 0 XFFFF | Ever the maximum number of refresh commands between two auto writes (should be set to 0) |
| WRLVL_DELAY_8 | 47:32 | 0 x0 | 0 x0 0 XFFFF | In the eighth data group, control the number of write DQS via DLL delays |
| WRLVL_DELAY_7 | Caused the | 0 x0 | 0 x0 0 XFFFF | In the seventh data group, control the number of write DQS via DLL delays |
| WRLVL_DELAY_6 | 15:0 | 0 x0 | 0 x0 0 XFFFF | In the sixth data set, control the number of write DQS via DLL delays |
| CONF_CTL_178[63:0]Offset: 0xb20DDR2 667:0x00000c2d00000c2d | | | | |
| TDFI_RDLVL_RESP | 63:32 | 0 x0 | 0 x0 0 XFFFF | Save the DFI Trdlvl_resp time parameter |
| TDFI_RDLVL_MAX | 31:0 | 0 x0 | 0 x0 0 XFFFF | Save the DFI Trdlvl_max time parameter |
| CONF_CTL_179[63:0]Offset: 0xb30DDR2 667:0x00000c2d00000c2d | | | | |
| TDFI_WRLVL_RESP | 63:32 | 0 x0 | 0 x0 0 XFFFF | Save the DFI Twrlvl_resp time parameter |
| TDFI_WRLVL_MAX | 31:0 | 0 x0 | 0 x0 0 XFFFF | Save the DFI Twrlvl_max time parameter |

# 9 HyperTransport controller

In loongson 3, the HyperTransport bus is used for external device connection and multi-chip interconnection.When used for peripheral connections, the user program is free to choose whether or not IO Cache consistency is supported (set by the address window Uncache window, see section x.4.2).In the Cache consistency support mode, IO devices are transparent to the Cache hierarchy for internal DMA access, which means that the program's Cache instructions do not need to maintain the consistency, but the hardware automatically maintains its consistency.When used for multi-chip interconnection, the HT0 controller (initial address 0x0C00_0000_0000 -- 0x0DFF_FFFF_FFFF) hardware automatically maintains Cache consistency between cpus, while the HT1 controller (initial address 0x0E00_0000_0000 -- 0x0FFF_FFFF_FFFF) does not support inter-chip Cache consistency.See section x.5 for details.

The HyperTransport controller supports up to 16 bit width in both directions and has a maximum operating frequency of 800Mhz.After the automatic initialization of the system to establish the connection, the user can modify the required running frequency and width by modifying the configuration register in the protocol, and then re-initialize, as detailed in section x.1.

The main features of loongson 3 HyperTransport controller are as follows:

Support 200/400/800 MHZ

Supports 8/16 bit widths

Each HT controller (HT0/HT1) can be configured as two 8-bit HT controllers

The bus control signal (including PowerOK, Rstn, LDT_Stopn) direction is configurable

Peripheral DMA space Cache/Uncache is configurable

The HT0 controller can be configured in Cache consistency mode when used for multi-chip interconnection

## 9.1 HyperTransport hardware setup and initialization

The HyperTransport bus consists of a transmission signal bus and a control signal pin, as shown in the following table

HyperTransport bus-related pins and their role.

Table 9-1 HyperTransport bus-related pin signals

| pin | The name of the | describe |
|---|---|---|
| {PCI_Config [7], PCI_Config [0]} | HT peripheral signal voltage control | 00: take HyperTransport peripheral signals as 1.8v signals, including HT_8x2, HT_Mode, HT_Powerok, HT_Rstn, HT_Ldt_Stopn, HT_Ldt_Reqn.<br>01: reserve.<br>10: take HyperTransport peripheral signals as 2.5v signals.<br>11: take HyperTransport peripheral signals as 3.3v signals. |
| HT0_8x2 | Bus width configuration | 1: the 16-bit HyperTransport bus is configured as two independent 8-bit buses, which are controlled by two independent controllers respectively. The address space is distinguished as<br>HT0_Lo: address[40] = 0;<br>HT0_Hi: address[40] = 1;<br>0: use the 16-bit HyperTransport bus as a 16-bit bus by HT0_Lo control, address space is the address of HT0_Lo, that is, address[40]<br>= 0; All signals of HT0_Hi are invalid. |
| HT0_Lo_mode | Master device mode | 1: set HT0_Lo as the main device mode. In this mode, bus control signals are driven by HT0_Lo, including HT0_Lo_Powerok, HT0_Lo_Rstn, HT0_Lo_Ldt_Stopn.In this mode, these control signals can also be bidirectional. At the same time, this pin determines the initial value of the register "Act as Slave". When this register is 0, the Bridge bit in the packet on the HyperTransport bus is 1, otherwise it is 0.In addition, when this register is 0, if the request address on the HyperTransport bus is not hit by the controller's receive window, it will be sent back to the bus as a P2P request, and if this register is 1, if it is not hit, it will be responded as an error request.<br>0: set HT0_Lo to slave device mode. In this mode, bus control signals are driven by other devices, including HT0_Lo_Powerok, HT0_Lo_Rstn, HT0_Lo_Ldt_Stopn.In this mode, these control signals are driven by each other's devices, or if they are not driven correctly, the HT bus<br>Not working properly. |
| HT0_Lo_Powerok | The bus Powerok | HyperTransport bus Powerok signal,<br>When HT0_Lo_Mode is 1, it is controlled by HT0_Lo;<br>When HT0_Lo_Mode is 0, it is controlled by the other device. |
| HT0_Lo_Rstn | Bus Rstn | HyperTransport bus Rstn signal,<br>When HT0_Lo_Mode is 1, it is controlled by HT0_Lo;<br>When HT0_Lo_Mode is 0, it is controlled by the other device. |
| HT0_Lo_Ldt_Stopn | The bus Ldt_Stopn | HyperTransport bus Ldt_Stopn signal,<br>When HT0_Lo_Mode is 1, it is controlled by HT0_Lo;<br>When HT0_Lo_Mode is 0, it is controlled by the other device. |
| HT0_Lo_Ldt_Reqn | The bus Ldt_Reqn | HyperTransport bus Ldt_Reqn signal, |
| HT0_Hi_mode | Master device mode | 1: set HT0_Hi to the main device mode. In this mode, bus control signals 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 bidirectional. At the same time, this pin determines the initial value of the register "Act as Slave". When this register is 0, the Bridge bit in the packet on the HyperTransport bus is 1, otherwise it is 0.In addition, when this register is 0, if the request address on the HyperTransport bus is not hit by the controller's receive window, it will be sent back to the bus as a P2P request, and if this register is 1, if it is not hit, it will be responded as an error request.<br>0: set HT0_Hi to slave device mode. In this mode, bus control signals are driven by other devices, including HT0_Hi_Powerok, HT0_Hi_Rstn, HT0_Hi_Ldt_Stopn.In this mode, these controls<br>The signal is driven by the other device, and if it is not driven correctly, the HT bus will not work correctly. |

| | | |
|---|---|---|
| HT0_Hi_Powerok | The bus Powerok | When the HyperTransport main line Powerok number, HT0_Lo_Mode is 1, it is controlled by HT0_Hi; When HT0_Lo_Mode is 0, it is controlled by the other device. When HT0_8x2 is 1, control the high 8-bit bus; Invalid when HT0_8x2 is 0. |
| HT0_Hi_Rstn | Bus Rstn | When Rstn number of HyperTransport main line, HT0_Lo_Mode is 1, it is controlled by HT0_Hi; When HT0_Lo_Mode is 0, it is controlled by the other device. When HT0_8x2 is 1, control the high 8-bit bus; Invalid when HT0_8x2 is 0. |
| HT0_Hi_Ldt_Stopn | The bus Ldt_Stopn | HyperTransport main line Ldt_Stopn, HT0_Lo_Mode = 1, is controlled by HT0_Hi; When HT0_Lo_Mode is 0, it is controlled by the other device. When HT0_8x2 is 1, control the high 8-bit bus; Invalid when HT0_8x2 is 0. |
| HT0_Hi_Ldt_Reqn | The bus Ldt_Reqn | HyperTransport bus Ldt_Reqn signal, When HT0_8x2 is 1, control the high 8-bit bus; Invalid when HT0_8x2 is 0. |
| HT0_Rx_CLKp [1:0] | CLK [1:0] | HyperTransport bus CLK signal |

| | | |
|---|---|---|
| HT0_Rx_CLKn HT0_Tx_CLKp [1:0] [1:0] HT0_Tx_CLKp (1-0) | | When HT0_8x2 is 1, CLK[1] is controlled by HT0_Hi<br>CLK[0] is controlled by HT0_Lo<br>When HT0_8x2 is 0, CLK[1:0] is controlled by HT0_Lo |
| HT0_Rx_CTLp HT0_Rx_CTLn [1:0] [1:0] HT0_Tx_CTLp [1:0] HT0_Tx_CTLn (1-0) | CTL (1-0) | HyperTransport bus CTL signal<br>When HT0_8x2 is 1, CTL[1] is controlled by HT0_Hi<br>The CTL[0] is controlled by HT0_Lo<br>When HT0_8x2 is 0, CTL[1] is invalid<br>The CTL[0] is controlled by HT0_Lo |
| HT0_Rx_CADp HT0_Rx_CADn [15:0] [15:0] HT0_Tx_CADp HT0_Tx_CADn [15:0] [15:0] | CAD [15:0] | HyperTransport bus CAD signal<br>When HT0_8x2 is 1, CAD[15:8] is controlled by HT0_Hi<br>CAD[7:0] is controlled by HT0_Lo<br>When HT0_8x2 is 0, CAD[15:0] is controlled by HT0_Lo |

The initialization of HyperTransport starts automatically after each reset is completed. After cold startup, the HyperTransport bus will automatically work at the lowest frequency (200Mhz) and minimum width (8bit), and attempt to perform the bus initialization handshake.The state of whether the initialization is Complete or not can be read by the register "Init Complete" (see section 9.5.2).After initialization, the Width of the bus can be read from the registers "Link Width Out" and "Link Width In" (see section 9.5.2).After initialization is complete, the user can back to register "Link Width Out", "the Link Width In" and "Link Freq" programming, at the same time, the corresponding register of needs of the other equipment programming, programming after the completion of the heat need to reset the bus or through HT_Ldt_Stopn signal to initialize the bus operation, In order to make programming values In the re-initialization after effect.The HyperTransport bus will operate at the new frequency and width after reinitialization.It is important to note that the configuration of devices at both ends of HyperTransport needs to be one-to-one, otherwise the HyperTransport interface will not work properly.

## 9.2 HyperTransport protocol support

The HyperTransport bus supports most of the commands in the 1.03 protocol, and some extended instructions are added to the extended conformance protocol supported by multi-chip interconnection. For both modes, the HyperTransport receiver can receive commands as shown in the following table.It is important to note that the atomic operation command of the HyperTransport bus is not supported.

Table 9-2 commands that the HyperTransport receiver can receive

| coding | channel | The command | The standard model | Extension (consistency) |
|---|---|---|---|---|
| 000000 | - | The NOP | Empty packet or flow control | |
| 000001 | NPC | FLUSH | No operation | |
| x01xxx | NPC<br>The or<br>The PC | The Write | Bit 5:0 - Nonposted<br>1 - Posted<br>bit 2:0 - Byte<br>1 -- Doubleword | Bit 5: must be 1, POSTED<br>Bit 2:0 -- Byte<br>1 - Doubleword |

| | | | | |
|---|---|---|---|---|
| | | | bit 1: Don't Care<br>Bit 0: Don't Care | Bit 1: Don't Care<br>Bit 0: must be 1 |
| 01 XXXX | NPC | The Read | Bit 3: Don't Care<br>bit 2:0 -- Byte<br>1 -- Doubleword<br>bit 1: Don't Care<br>Bit 0: Don't Care | Bit 3: Don't Care<br>bit 2:0 -- Byte<br>1 -- Doubleword<br>bit 1: Don't Care<br>Bit 0: must be 1 |
| 110000 | R | RdResponse | Read operation return | |
| 110011 | R | TgtDone | Write operation return | |
| 110100 | The PC | WrCoherent | ---- | Write command extension |
| 110101 | The PC | WrAddr | ---- | Write address extension |
| 111000 | R | RespCoherent | ---- | Read response extension |
| 111001 | NPC | RdCoherent | ---- | Read command extension |
| 111010 | The PC | Broadcast | No operation | |
| 111011 | NPC | RdAddr | ---- | Read address extension |
| 111100 | The PC | A FENCE | Guarantee order relation | |
| 111111 | - | The Sync/Error | The Sync/Error | |

For the sending side, the commands sent out in both modes are shown in the following table.

Table 9-3 commands sent out in both modes

| coding | channel | The command | The standard model | Extension (consistency) |
|---|---|---|---|---|
| 000000 | - | The NOP | Empty packet or flow control | |
| x01x0x | NPC<br>The or PC | The Write | Bit 5:0 - Nonposted<br>1 - Posted<br>bit 2:0 - Byte<br>1 - Doubleword<br>Bit 0: must be 0 | Bit 5: must be 1, POSTED<br>Bit 2:0 -- Byte<br>1 - Doubleword<br>Bit 0: must be 1 |
| 010 x0x | NPC | The Read | Bit 2:0 -- Byte<br>1 - Doubleword<br>Bit 0: Don't Care | Bit 2:0 -- Byte<br>1 - Doubleword<br>Bit 0: must be 1 |
| 110000 | R | RdResponse | Read operation return | |
| 110011 | R | TgtDone | Write operation return | |
| 110100 | The PC | WrCoherent | ---- | Write command extension |
| 110101 | The PC | WrAddr | ---- | Write address extension |
| 111000 | R | RespCoherent | ---- | Read response extension |
| 111001 | NPC | RdCoherent | ---- | Read command extension |
| 111011 | NPC | RdAddr | ---- | Read address extension |
| 111111 | - | The Sync/Error | Will only forward | |

## 9.3 HyperTransport interrupt support

The HyperTransport controller provides 256 interrupt vectors to support Fix, Arbitor, and other types of interrupts, but no hardware automatic EOI support.For these two types of interrupts, the controller will automatically write to the interrupt register after receiving and notify the system interrupt controller according to the Settings of the interrupt mask

register.For the specific interrupt control, see the interrupt control register group in section 5.

In addition, the controller has special support for PIC interrupts to speed up this type of interrupt handling.

A typical PIC interrupt is completed by the following steps: (1) the PIC controller sends a PIC interrupt request to the system;The system sends interrupt vector query to PIC controller;(3) PIC controller sends interrupt vector number to the system;The system clears the corresponding interrupt on the PIC controller.The PIC controller will not issue the next interrupt to the system until all 4 steps above have been completed.for

Loongson 3 HyperTransport controller will automatically process the first 3 steps and write PIC interrupt vector to the corresponding position in 256 interrupt vectors.After the interrupt is processed by the software system, the fourth step is required, which is to send a clear interrupt to the PIC controller.Then the processing of the next interrupt begins.

## 9.4 HyperTransport address window

### 9. 4. 1 HyperTransport space

In loongson 3 processor, the default four HyperTransport address Windows are as follows:

Table 9-4 default addresses for the four HyperTransport address Windows

| Base address | End address | The size of the | define |
|---|---|---|---|
| 0 x0c00_0000_0000 | 0 x0cff_ffff_ffff | One Tbytes | HT0_LO window |
| 0 x0d00_0000_0000 | 0 x0dff_ffff_ffff | One Tbytes | HT0_HI window |
| 0 x0e00_0000_0000 | 0 x0eff_ffff_ffff | One Tbytes | HT1_LO window |
| 0 x0f00_0000_0000 | 0 x0fff_ffff_ffff | One Tbytes | HT1_HI window |

By default (the system address window is not configured separately), the software accesses each HyperTransport interface through the above address space. In addition, the software can configure the address window on the crossover switch to be accessed from another address space (see section 2.5).The distribution of specific address Windows within each HyperTransport interface's 40-bit address space is described in the following table.

The address window distribution of the loong chip 3 processor HyperTransport interface protocol is as follows:

Table 9-5 address window distribution of the HyperTransport interface of loongson 3 processor

| Base address | End address | The size of the | define |
|---|---|---|---|
| 0 x00_0000_0000 | 0 xfc_ffff_ffff | 1012 Gbytes | MEM space |
| 0 xfd_0000_0000 | 0 xfd_f7ff_ffff | 3968 Mbytes | reserve |
| 0 xfd_f800_0000 | 0 xfd_f8ff_ffff | 16 Mbytes | interrupt |
| 0 xfd_f900_0000 | 0 xfd_f90f_ffff | 1 Mbyte | PIC interrupt response |
| 0 xfd_f910_0000 | 0 xfd_f91f_ffff | 1 Mbyte | System information |
| 0 xfd_f920_0000 | 0 xfd_faff_ffff | 30 Mbytes | reserve |
| 0 xfd_fb00_0000 | 0 xfd_fbff_ffff | 16 Mbytes | HT controller configuration space |
| 0 xfd_fc00_0000 | 0 xfd_fdff_ffff | 32 Mbytes | I/O space |
| 0 xfd_fe00_0000 | 0 xfd_ffff_ffff | 32 Mbytes | HT bus configuration space |
| 0 xfe_0000_0000 | 0 xff_ffff_ffff | 8 Gbytes | reserve |

## 9. 4. 2 HyperTransport controller internal window configuration

The loongson 3 processor HyperTransport interface provides a variety of rich address Windows for users to use, including

Here are a few:

Table 9-6 address window provided in the HyperTransport interface of loongson 3 processor

| The address window | Window number | Accept the bus | role | note |
|---|---|---|---|---|
| Receiving window<br>See window configuration 9.5.4) | 3 | HyperTransport | Judge whether to accept An access emitted on the HyperTransport bus. | In the main bridge mode (i.e., act_as_slave is 0 in the configuration register), only the access that falls in these address Windows will be responded to by the internal bus. Other access will be considered as P2P access and sent back to the HyperTransport bus.When in device mode (that is, act_as_slave is 1 in the configuration register), only the accesses that fall into these address Windows will be received and processed by the internal bus, and other accesses will be returned with protocol errors. |
| The Post window<br>See window configuration 9.5.8) | 2 | Inside the bus | Determines whether Write access from the internal bus to the HyperTransport bus is treated as Post Write | Outgoing Write access that falls into these address Spaces will be treated as Post Write.<br>In the Post Write: HyperTransport protocol, this Write access does not have to wait for the Write response, that is, after the controller issues the Write access to the bus, the Write access to the processor completes the response. |

| | | | | |
|---|---|---|---|---|
| Prefetch window<br>See window configuration 9.5.9) | 2 | Inside the bus | To determine whether to receive internal Cache access, fetch access. | When the processor core is out of order, some guess reads or fetches are given to the bus, which is wrong for some IO Spaces.By default, this access HT controller will return directly without access to the HyperTransport bus.These Windows enable such access to the HyperTransport bus. |
| Uncache window<br>See window configuration 9.5.10) | 2 | HyperTransport | Determine if the access on the HyperTransport bus is to be accessed as Uncache on the inside | IO DMA access inside loongson 3 processor, in case of Cache access, will be judged as a hit by the second level Cache, so as to maintain its IO consistency information.Through the configuration of these Windows, it is possible to make the access hit in these Windows access memory directly in the manner of Uncache, without maintaining its IO consistency information through hardware. |

## 9.5 Configuration register

The configuration register module is mainly used to control the access of the configuration register from the AXI SLAVE or HT RECEIVER, the external interrupt processing, and the storage of a large number of configuration registers visible to the software to control the various working modes of the system.

First, the access and storage of the configuration registers used to control the various behaviors of the HT controller are in this module, whose access offset address is 0xFD_FB00_0000 to 0xFD_FBFF_FFFF at the AXI end.The visible registers of all software in this module are shown in the following table:

Table 9-7 all software visible registers in this module

| offset | The name of the | describe |
|---|---|---|
| 0 x30 | | |
| 0 x34 | | |
| 0 x38 | | |
| 0 x3c | Bridge Control | Bus Reset Control |
| 0 x40 | Capability Registers | Command, Capabilities Pointer, Capability ID |
| 0 x44 | | Link Config, Link Control |
| 0 x48 | | Revision ID, Link Freq, Link Error, Link Freq Cap |
| 0 x4c | | Feature Capability |
| 0 x50 | Custom register | MISC |
| 0 x54 | Receive diagnostic register | The receiver samples the received cad and CTL values |
| 0 x58 | Interrupt routing register | Interrupt routing control register (LS3A1000E and above) |
| 0 x5c | Receive buffer register | Set the initial value of the receive buffer (LS3A1000E and above) |
| 0 x60 | Receive address window configuration register | HT bus receive address window 0 enable (external access) |
| 0 x64 | | HT bus receiving address window 0 base address (external access) |
| 0 x68 | | HT bus receive address window 1 enable (external access) |
| 0 x6c | | HT bus receiving address window 1 base address (external access) |
| 0 x70 | | HT bus receive address window 2 enable (external access) |
| 0 x74 | | HT bus receiving address window 2 base address (external access) |
| 0 x78 | | |
| 0 x7c | | |
| 0 x80 | Interrupt vector register | HT bus interrupt vector register [31:0] |
| 0 x84 | | HT bus interrupt vector register [63:32] |
| 0 x88 | | HT bus interrupt vector register [95:64] |
| 0 x8c | | HT bus interrupt vector register [127:96] |
| 0 x90 | | HT bus interrupt vector register [159:128] |
| 0 x94 | | HT bus interrupt vector register [191:160] |
| 0 x98 | | HT bus interrupt vector register [223:192] |
| 0 x9c | | HT bus interrupt vector register [255:224] |
| 0 xa0 | Interrupt enable register | HT bus interrupts enable register [31:0] |
| 0 xa4 | | HT bus interrupt enabled register [63:32] |
| 0 xa8 | | HT bus interrupt enabled register [95:64] |
| 0 xac | | HT bus interrupt enabled register [127:96] |
| 0 xb0 | | HT bus interrupt enabled register [159:128] |
| 0 xb4 | | HT bus interrupt enabled register [191:160] |
| 0 xb8 | | HT bus interrupt enabled register [223:192] |
| 0 XBC | | HT bus interrupt enabled register [255:224] |
| 0 xc0 | Interrupt Discovery & The Configuration | Interrupt Capability |
| 0 xc4 | | DataPort |
| 0 xc8 | | IntrInfo [31:0] |
| 0 XCC | | IntrInfo [63:32] |
| 0 xd0 | Configure the register in the POST address | HT bus POST address window 0 enable (internal access) |
| 0 xd4 | | HT bus POST address window 0 base address (internal access) |
| 0 xd8 | | HT bus POST address window 1 enable (internal access) |
| 0 XDC | | HT bus POST address window 1 base address (internal access) |

| | window | |
|---|---|---|
| 0 xe0-0xfc | Pre-fetch address window configuration register | HT bus prefetching address window 0 enable (internal access) |
| 0 xe4 | | HT bus can prefetch address window 0 base address (internal access) |
| 0 xe8 | | HT bus prefetching address window 1 enable (internal access) |
| 0 xec | | Ht bus prefetch address window 1 base address (internal access) |
| 0 xf0 | Uncache address window configuration register | HT bus Uncache address window 0 enable (internal access) |
| 0 xf4 | | HT bus Uncache address window 0 base address (internal access) |
| 0 xf8 | | HT bus Uncache address window 1 enable (internal access) |
| 0 XFC | | HT bus Uncache address window 1 base address (internal access) |

The specific meaning of each register is shown in the following section:

## 9.5.1 Bridge Control

Offset: 0x3C

Reset value: 0x00200000

Name: Bus Reset Control

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| For calamity | Reserved | 4 | 0 x0 | | reserve |
| 22 | The Reset | 12 | 0 x0 | R/W | Bus reset control:<br>0 >1: HT_RSTn set 0, bus reset<br>1 >0: HT_RSTn set 1, bus unreset |
| Lift up | Reserved | 4 | 0 x0 | | reserve |
| 17 | B_interleave | 1 | 0 x0 | R/W | Does the write response channel allow out-of-order execution (LS3A1000E and above)<br>When using multi-chip interconnection mode, 1 must be set |
| 16 | Nop_interleave | 1 | 0 x0 | R/W | Does flow control channel allow out-of-order execution (LS3A1000E and above)<br>When using multi-chip interconnection mode, 1 must be set |
| 15:0 | Reserved | 16 | 0 x0 | | reserve |

## 9.5.2 Capability Registers

Offset: 0x40

Reset value: 0x20010008

Name: Command, Capabilities Pointer, Capability ID

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| take | The HOST/Sec | 3 | 0 x1 | R | The Command format is HOST/Sec |
| Forepart thereof | Reserved | 2 | 0 x0 | R | reserve |
| 26 | Act as a Slave | 1 | 0 x0<br>/ 0 x1 | R/W | The HOST/SLAVE mode<br>The initial value is determined by the pin HOSTMODE<br>HOSTMODE: 0<br>HOSTMODE dropdown: 1 |

| | | | | | |
|---|---|---|---|---|---|
| 25 | Reserved | 1 | 0 x0 | | reserve |
| 24 | The Host Hide | 1 | 0 x0 | R/W | Whether to disable register access from HT bus |
| 23 | Reserved | 1 | 0 x0 | | reserve |
| " | The Unit ID | 5 | 0 x0 | R/W | HOST mode: can be used to record the number of use ids<br>SLAVE mode: record self Unit ID |
| 17 | A Double Ended | 1 | 0 x0 | R | The dual HOST mode is not used |
| 16 | A Warm Reset | 1 | 0 x1 | R | The reset in Bridge Control adopts the thermal reset mode |
| " | "Capabilities Pointer | 8 | 0 xa0 | R | The next Cap register offset address |
| away | Capability ID | 8 | 0 x08 | R | HyperTransport capability ID |

Offset: 0x44

Reset value: 0x00112000

Name: Link Config, Link Control

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | Reserved | 1 | 0 x0 | | reserve |
| he | The Link Width Out | 3 | 0 x0 | R/W | Sending end width<br>The value after 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<br>000:8 bit mode<br>001:16 bit mode |
| 27 | Reserved | 1 | 0 x0 | | reserve |
| they | The Link Width In | 3 | 0 x0 | R/W | Receiver width<br>The value after 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 |
| 23 | Dw Fc out | 1 | 0 x0 | R | Sending terminal does not support double word flow control |
| Lift up | Max Link Width out | 3 | 0 x1 | R | HT bus sending end maximum width: 16bits |
| 19 | Dw Fc In | 1 | 0 x0 | R | The receiver does not support double word flow control |
| thou | Max Link Width In | 3 | 0 x1 | R | HT bus receiver maximum width: 16bits |
| The lowest | Reserved | 2 | 0 x0 | | reserve |
| 13 | LDTSTOP# Tristate Enable | 1 | 0 x1 | R/W | Whether to close HT PHY when HT bus enters HT Disconnect state<br>1: closed<br>0: not closed |
| 12:10 | Reserved | 3 | 0 x0 | | reserve |
| 9 | CRC Error (hi) | 1 | 0 x0 | R/W | CRC errors occur in the high 8 bit |
| 8 | CRC Error (lo) | 1 | 0 x0 | R/W | CRC errors occur in the lower 8 bits |
| 7 | Trans off | 1 | 0 x0 | R/W | HT PHY close control<br>In 16-bit bus mode of operation<br>1: close high/low 8-bit HT PHY<br>0: enable low 8-bit HT PHY,<br>high 8-bit HT PHY controlled<br>by bit 0 |
| 6 | The End of the Chain | 0 | 0 x0 | R | HT bus terminal |

| | | | | | |
|---|---|---|---|---|---|
| 5 | Init Complete | 1 | 0 x0 | R | Whether the HT bus initialization is completed |
| 4 | The Link Fail | 1 | 0 x0 | R | Indicating connection failure |
| 3:2 | Reserved | 2 | 0 x0 | | reserve |
| 1 | CRC<br>Flood the Enable | 1 | 0 x0 | R/W | Whether flood HT bus occurs when CRC error occurs |
| 0 | Trans off (hi) | 1 | 0 x0 | R/W | High 8-bit PHY closes control when running 8-bit protocol using 16-bit HT bus<br>1: close high 8-bit HT PHY<br>0: enable high 8 bit HT PHY |

Offset: 0x48

Reset value: 0x80250023

Name: Revision ID, Link Freq, Link Error, Link Freq Cap

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | The Link Freq Cap | 16 | 0 x0025 | R | Supported HT bus frequency<br>200Mhz, 400Mhz, 800Mhz, |
| The lowest | Reserved | 2 | 0 x0 | | reserve |
| 13 | Over Flow Error | 1 | 0 x0 | R | HT bus packet overflow |
| 12 | Protocol Error | 1 | 0 x0 | R/W | Protocol error,<br>An unrecognized command received on the HT bus |
| and | The Link Freq | 4 | 0 x0 | R/W | HT bus operating frequency<br>The value of this register will take effect after the next hot reset or HT Disconnect<br>0000-200 m<br>0010-400 m<br>0101-800 m |
| away | Revision ID | 8 | 0 x23 | R/W | Version number: 1.03 |

Offset: 0x4C

Reset value: 0x00000002

Name: Feature Capability

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Is wasted | Reserved | 25 | 0 x0 | | reserve |
| 8 | Extended the Register | 1 | 0 x0 | R | There is no |
| The log | Reserved | 3 | 0 x0 | | reserve |
| 3 | Extended CTL Time | 1 | 0 x0 | R | Don't need |
| 2 | CRC Test Mode | 1 | 0 x0 | R | Does not support |
| 1 | LDTSTOP# | 1 | 0 x1 | R | Support LDTSTOP# |
| 0 | Isochronous Mode | 1 | 0 x0 | R | Does not support |

### 9.5.3 Custom register

Offset: 0x50

Reset value: 0x00904321

Name: MISC

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 31 | Reserved | 1 | 0 x0 | | reserve |
| 30 | Ldt Stop Gen | 1 | 0 x0 | R/W | Take the bus into LDT DISCONNECT mode<br>The correct way is: 0 minus >, 1 |
| 29 | Ldt the Req Gen | 1 | 0 x0 | R/W | Waken HT bus from LDT DISCONNECT, set LDT_REQ_n<br>The right way to do it is to put a 0 and then a 1:0 minus >, 1<br>In addition, direct read and write requests to the bus can also automatically wake up the bus |
| A domain | A domain name | A wide | Reset value | access | describe |
| throughout | Interrupt the Index | 5 | 0 x0 | R/W | Redirect interrupts other than standard interrupts to which interrupt vector (including SMI, NMI, INIT, INTA, INTB, INTC, INTD)<br>There are 256 interrupt vectors in total. This register represents interrupts<br>The height of the vector is 5 bits, and the internal interrupt vector is as follows:<br>000: SMI<br>001: NMI<br>010: INIT<br>011: Reservered<br>100: INTA<br>101: intb.br deal<br>110: INTC<br>111: INTD |
| 23 | Dword Write | 1 | 0 x1 | R/W | For 32/64/128/256 bit write access, whether to use Dword Write command format<br>1: Write using Dword<br>0: use Byte Write (with MASK) |
| 22 | Coherent Mode | 1 | 0 x0 | R | Processor consistency mode<br>Determined by pin ICCC_EN |
| 21 | Not Care Seqid | 1 | 0 x0 | R/W | Whether you don't care about HT |
| 20 | The Not Axi2Seqid | 1 | 0 x1 | R | Whether to convert the commands on the Axi bus to a different SeqID or not, if not, all read and write commands will take the Fixed ID number in the Fixed SeqID<br>1: no conversion<br>0: conversion |
| He hath | Fixed Seqid | 4 | 0 x0 | R/W | Configure the HT bus emitted when Not Axi2Seqid is valid Seqid |
| " | Priority Nop | 4 | 0 x4 | R/W | HT bus Nop flow control packet priority |
| and | Specify the NPC | 4 | 0 x3 | R/W | Non Post channel read and write priority |
| The log | Priority RC | 4 | 0 x2 | R/W | The Response channel reads and writes first |
| 3-0 | Priority PC | 4 | 0 x1 | R/W | Post channel read and write priority<br>0x0: highest priority<br>0xF: lowest priority<br>The priority strategy of increasing with time is adopted for each channel, and this memory set is used to configure the initial priority of each channel |

## 9.5.4 Receive diagnostic register

Offset: 0x54

Reset value: 0x00000000

Name: receive diagnostic register

| A | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|

| domain | | | | | |
|---|---|---|---|---|---|
| 0 | Sample_en | 1 | 0 x0 | R/W | Enable to sample input cad<br>and CTL 0x0: disabled<br>0 x1: can make |
| " | rx_ctl_catch | 24 | 0 x0 | R/W | Save the input CTL from the sample<br>(0, 2, 4, 6) corresponds to the four phases of CTL0 sampling<br>(1, 3, 5, 7) corresponds to the four phases of CTL1 sampling |
| Caused the | rx_cad_phase_0 | 24 | 0 x0 | R/W | Save the values of the input CAD[15:0] from the sample |

### 9.5.5 Interrupt routing mode selection register (LS3A1000E and above)

Offset: 0x58
Reset value: 0x00000000
Name: interrupt routing selection register

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| o | ht_int_stripe | 2 | 0 x0 | R/W | Corresponding to the three interrupt routing methods, see 9.5.7 interrupt vector register for details<br>0x0: ht_int_stripe_1 0x1: ht_int_stripe_2<br>0 x2: ht_int_stripe_4 |

### 9.5.6 Receive buffer initial register (LS3A1000E and above)

Offset: 0x5c
Reset value: 0x07778888
Name: receive buffer initialization configuration register

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| he | rx_buffer_r_data | 4 | 0 x7 | R/W | Receive buffer's read data buffer initialization information |
| Behold, | rx_buffer_npc_data | 4 | 0 x7 | R/W | Receive the NPC data buffer initialization information for the buffer |
| He hath | rx_buffer_pc_data | 4 | 0 x7 | R/W | Receive the buffer's PC data buffer initialization information |
| " | rx_buffer_b_cmd | 4 | By 8 0 | R/W | The bresponse command buffer initializes the message to receive the buffer Interest rates |
| and | rx_buffer_r_cmd | 4 | By 8 0 | R/W | Receive buffer's read command buffer initialization information |
| The log | rx_buffer_npc_cmd | 4 | By 8 0 | R/W | Receive the NPC command buffer initialization information for the buffer |
| 3-0 | rx_buffer_pc_cmd | 4 | By 8 0 | R/W | Receive the buffer's PC command buffer initialization information |

### 9.5.7 Receive address window configuration register

The address window hit formula in this controller is as follows:

Hit = (BASE & MASK) == (ADDR & MASK)

Addr_out = TRANS_EN? TRANS b0 ADDR & ~MASK: ADDR

It is worth mentioning that, when configuring the address window register, the MASK should be all 1 and all 0.MASK

The actual number of zeros in the address window represents the size of the address window.

The address of this window is the address received on the HT bus.The HT address that

falls in this window will be sent to the CPU, and commands from other addresses will be forwarded back to the HT bus as P2P commands.

Offset: 0x60

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_rx_image0_en | 1 | 0 x0 | R/W | HT bus receives address window 0, enabling signal |
| 30 | Ht_rx_image0_ trans_en | 1 | 0 x0 | R/W | HT bus receives address window 0, mapping enable signal |
| 29:0 | Ht_rx_image0_ trans [53:24] | 16 | 0 x0 | R/W | HT bus receive address window 0, the mapped address [53:24], for LS3A1000D and below version, [29:23] fixed as 0 |

Offset: 0x64

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_rx_image0_ base [they] | 16 | 0 x0 | R/W | HT bus receive address window 0, address base address [39:24] |
| 15:0 | Ht_rx_image0_ mask [they] | 16 | 0 x0 | R/W | HT bus receiving address window 0, address shielded [39:24] |

Offset: 0x68

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_rx_image1_en | 1 | 0 x0 | R/W | HT bus receives address window 1, enabling signal |
| 30 | Ht_rx_image1_ trans_en | 1 | 0 x0 | R/W | HT bus receives address window 1, mapping the enable signal |
| 29:0 | Ht_rx_image1_ trans [53:24] | 16 | 0 x0 | R/W | HT bus receive address window 1, the mapped address [53:24], for LS3A1000D and below version, [29:23] fixed as 0 |

Offset: 0x6c

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_rx_image1_ base [they] | 16 | 0 x0 | R/W | HT bus receive address window 1, address base address [39:24] |
| 15:0 | Ht_rx_image1_ mask [they] | 16 | 0 x0 | R/W | HT bus receive address window 1, address mask [39:24] |

Offset: 0x70

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_rx_image2_en | 1 | 0 x0 | R/W | HT bus receives address window 2, enabling signal |
| 30 | Ht_rx_image2_ trans_en | 1 | 0 x0 | R/W | The HT bus receives the address window 2, mapping the enable signal |
| 29:0 | Ht_rx_image2_ trans [53:24] | 16 | 0 x0 | R/W | HT bus receive address window 2, the translated address [53:24], for LS3A1000D and below version, [29:23] fixed as 0 |

Offset: 0x74
Reset value: 0x00000000
Name: HT bus receiving address window 2 base address (external access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_rx_image2_ base [they] | 16 | 0 x0 | R/W | HT bus receive address window 2, address base address [39:24] |
| 15:0 | Ht_rx_image2_ mask [they] | 16 | 0 x0 | R/W | HT bus receive address window 2, address mask [39:24] |

## 9.5.8 Interrupt vector register

Interrupt vector register a total of 256, of which exclude HT Fixed on the bus and Arbitrated, interruption of PIC 256 map directly to the interrupt vector, other plants, such as SMI, NMI, INIT, INTA, intb.br deal, a steady, 0 x50 [doth INTD can register mapped to any one of eight interrupt vector, the order of the map for {INTD, steady, intb.br deal, INTA, 1 'b0, INIT, NMI, SMI}.The corresponding value of Interrupt vector is {Interrupt Index, internal vector [2:0]}.

For LS3A1000E and above versions, 256 interrupt vectors are mapped to different interrupt lines according to different interrupt routing mode selection register configuration. The specific mapping mode is as follows:

Ht_int_stripe_1:

[0,1,2,3... 63] corresponds to the interrupted line 0 /HT HI corresponds to the interrupted line 4

[64,65,66,67... 127] corresponding to interrupted line 1 /HT HI corresponding to interrupted line 5

[128,129,130,131... 191] corresponding to discontinuous line 2 /HT HI corresponding to discontinuous line 6

[192,193,194,195... 255] corresponding to interrupted line 3 /HT HI corresponding to interrupted line 7 ht_int_stripe_2:

[0,2,4,6... 126] corresponds to the interrupted line 0 /HT HI corresponds to the interrupted line 4

[1,3,5,7... 127] corresponds to interrupted line 1 /HT HI corresponds to interrupted line 5

[128,130,132,134... 254] corresponding to discontinuous line 2 /HT HI corresponding to discontinuous line 6

[129,131,133,135... 255] corresponding to discontinuous line 3 /HT HI corresponding to discontinuous line 7 ht_int_stripe_4:

[0,4,8,12... 252] corresponds to the interrupted line 0 /HT HI corresponds to the interrupted line 4

[1,5,9,13... 253] corresponds to 1 /HT HI corresponds to 5

[2,6,10,14... 254] corresponding to discontinuous line 2 /HT HI corresponding to discontinuous line 6

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

The following interrupt vector description corresponds to ht_int_stripe_1, and two other ways can be obtained as described above.For LS3A1000D and below, you can only use ht_int_stripe_1.

Offset: 0x80
Reset value: 0x00000000
Name: HT bus interrupt vector register [31:0]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [31:0] | 32 | 0 x0 | R/W | HT bus interrupt vector register [31:0], It's 0 over HT HI and it's 4 |

Offset: 0x84
Reset value: 0x00000000
Name: HT bus interrupt vector register [63:32]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [63:32] | 32 | 0 x0 | R/W | HT bus interrupt vector register [63:32], It's 0 over HT HI and it's 4 |

Offset: 0x88
Reset value: 0x00000000
Name: HT bus interrupt vector register [95:64]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [95, 64] | 32 | 0 x0 | R/W | HT bus interrupt vector register [95:64], This corresponds to 1 /HT HI and this corresponds to 5 |

Offset: 0x8c
Reset value: 0x00000000
Name: HT bus interrupt vector register [127:96]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [127, 96] | 32 | 0 x0 | R/W | HT bus interrupt vector register [127:96], This corresponds to 1 /HT HI and this corresponds to 5 |

Offset: 0x90
Reset value: 0x00000000
Name: HT bus interrupt vector register [159:128]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [159, 128] | 32 | 0 x0 | R/W | HT bus interrupt vector register [159:128], corresponding to the discontinuous line 2 /HT HI corresponding to the discontinuous line 6 |

Offset: 0x94
Reset value: 0x00000000
Name: HT bus interrupt vector register [191:160]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [191, 160] | 32 | 0 x0 | R/W | HT bus interrupt vector register [191:160], So this corresponds to 2 /HT HI and this corresponds to 6 |

Offset: 0x98
Reset value: 0x00000000
Name: HT bus interrupt vector register [223:192]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [223, 192] | 32 | 0 x0 | R/W | HT bus interrupt vector register [223:192], So this corresponds to 3 /HT HI and this corresponds to 7 |

Offset: 0x9c
Reset value: 0x00000000
Name: HT bus interrupt vector register [255:224]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [255, 224] | 32 | 0 x0 | R/W | HT bus interrupt vector register [255:224], So this corresponds to 3 /HT HI and this corresponds to 7 |

## 9.5.9 Interrupt enable register

There are 256 interrupt enabled registers, corresponding to the interrupt vector register. Set 1 to open the corresponding interrupt, and set 0 to interrupt shielding.

Offset: 0xa0
Reset value: 0x00000000
Name: HT bus interrupt enabled register [31:0]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [31:0] | 32 | 0 x0 | R/W | HT bus interrupts enable register [31:0], It's 0 over HT HI and it's 4 |

Offset: 0xa4
Reset value: 0x00000000
Name: HT bus interrupt enabled register [63:32]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [63:32] | 32 | 0 x0 | R/W | HT bus interrupts enable register [63:32], It's 0 over HT HI and it's 4 |

Offset: 0xa8
Reset value: 0x00000000
Name: HT bus interrupt enabled register [95:64]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [95, 64] | 32 | 0 x0 | R/W | HT bus interrupts enable register [95:64], This corresponds to 1 /HT HI and this corresponds to 5 |

Offset: 0xac
Reset value: 0x00000000
Name: HT bus interrupt enabled register [127:96]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [127, 96] | 32 | 0 x0 | R/W | HT bus interrupts enable register [127:96], This corresponds to 1 /HT HI and this corresponds to 5 |

Offset: 0xb0

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [159, 128] | 32 | 0 x0 | R/W | HT bus interrupts enable register [159:128], So this corresponds to 2 /HT HI and this corresponds to 6 |

Offset: 0xb4

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [191, 160] | 32 | 0 x0 | R/W | HT bus interrupts enable register [191:160], So this corresponds to 2 /HT HI and this corresponds to 6 |

Offset: 0xb8

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [223, 192] | 32 | 0 x0 | R/W | HT bus interrupts enable register [223:192], So this corresponds to 3 /HT HI and this corresponds to 7 |

Offset: 0xbc

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [255, 224] | 32 | 0 x0 | R/W | HT bus interrupts enable register [255:224], So this corresponds to 3 /HT HI and this corresponds to 7 |

## 9.5.10 Interrupt Discovery & Configuration

Offset: 0xc0

Reset value: 0x80000008

Name: Interrupt Capability

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| came | "Capabilities Pointer | 8 | 0 x80 | R | Interrupt discovery and configuration block |
| Ephron; | The Index | 8 | 0 x0 | R/W | Read register offset address |
| " | "Capabilities Pointer | 8 | 0 x0 | R | "Capabilities Pointer |
| away | Capability ID | 8 | 0 x08 | R | Hypertransport Capablity ID |

Offset: 0xc4

Reset value: 0x00000000

Name: Dataport

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | Dataport | 32 | 0 x0 | R/W | When the last register Index is 0x10, this register reads and writes<br>The result is register 0xa8, otherwise 0xac |

Offset: 0xc8

Reset value: 0xF8000000

Name: IntrInfo [31:0]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| came | IntrInfo [came] | 32 | 0 xf8 | R | reserve |
| Isle; | IntrInfo [great] | 22 | 0 x0 | R/W | IntrInfo[23:2], the value of IntrInfo when sending PIC interrupt<br>It's used to represent interrupt vectors |
| 1-0 | Reserved | 2 | 0 x0 | R | reserve |

Offset: 0xcc

Reset value: 0x00000000

Name: IntrInfo [63:32]

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31:0 | IntrInfo [63:32] | 32 | 0 x0 | R | reserve |

## 9.5.11 Configure the register in the POST address window

The address window hit formula in this controller is as follows:

Hit = (BASE & MASK) == (ADDR & MASK)

It is worth mentioning that, when configuring the address window register, the MASK should be all 1 and all 0. 0 in the MASK

Is the size of the address window.

The address of this window is the one received on the AXI bus.All WRITE accesses that fall into this window will immediately be returned in the AXI B channel and sent to the HT bus in the command format of POST WRITE.WRITE requests that are not in this window are sent to the HT bus as NONPOST WRITE and wait for the HT bus to respond before returning to the AXI bus.

Offset: 0xd0

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_post0_en | 1 | 0 x0 | R/W | HT bus POST address window 0, enable signal |
| then | Reserved | 15 | 0 x0 | | reserve |

| | | | | | |
|---|---|---|---|---|---|
| 15:0 | Ht_post0_trans [they] | 16 | 0 x0 | R/W | HT bus POST address window 0, translated after the address [they] |

Offset: 0xd4
Reset value: 0x00000000
Name: HT bus POST address window 0 base address (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_post0_base [they] | 16 | 0 x0 | R/W | HT bus POST address window 0, address base address [39:24] |
| 15:0 | Ht_post0_mask [they] | 16 | 0 x0 | R/W | HT bus POST address window 0, address masked [39:24] |

Offset: 0xd8
Reset value: 0x00000000
Name: HT bus POST address window 1 enable (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| A domain | A domain name | A wide | Reset value | access | describe |
| 31 | ht_post1_en | 1 | 0 x0 | R/W | HT bus POST address window 1, enable signal |
| then | Reserved | 15 | 0 x0 | | reserve |
| 15:0 | Ht_post1_trans [they] | 16 | 0 x0 | R/W | HT bus POST address window 1, translated after the address [they] |

Offset: 0xdc
Reset value: 0x00000000
Name: HT bus POST address window 1 base address (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_post1_base [they] | 16 | 0 x0 | R/W | HT bus POST address window 1, address base address [39:24] |
| 15:0 | Ht_post1_mask [they] | 16 | 0 x0 | R/W | HT bus POST address window 1, address masked [39:24] |

## 9.5.12 Pre-fetch address window configuration register

The address window hit formula in this controller is as follows:

Hit = (BASE & MASK) == (ADDR & MASK)

It is worth mentioning that, when configuring the address window register, the MASK should be all 1 and all 0. 0 in the MASK

Is the size of the address window.

The address of this window is the one received on the AXI bus.The CACHE access will only be sent to the HT bus if it falls into this window. Other access or CACHE access will not be sent to the HT bus, but will be returned immediately. If it is a read command, it will return invalid read data of the corresponding number.

Offset: 0xe0

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_prefetch0_en | 1 | 0 x0 | R/W | HT bus can prefetch address window 0, enable signal |
| then | Reserved | 15 | 0 x0 | | reserve |
| 15:0 | Ht_prefetch0_trans [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 0, after translation address [they] |

Offset: 0xe4

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_prefetch0_ base [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 0, address base address [39:24] An address |
| 15:0 | Ht_prefetch0_ mask [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 0, address mask [39:24] |

Offset: 0xe8

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| A domain | A domain name | A wide | Reset value | access | describe |
| 31 | ht_prefetch1_en | 1 | 0 x0 | R/W | HT bus can prefetch address window 1 to enable the signal |
| then | Reserved | 15 | 0 x0 | | reserve |
| 15:0 | Ht_prefetch1_ trans [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 1, after the translation of the address [they] |

Offset: 0xec

Reset value: 0x00000000

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

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_prefetch1_ base [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 1, address base address [39:24] |
| 15:0 | Ht_prefetch1_ mask [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 1, address mask [39:24] |

## 9.5.13 UNCACHE address window configuration register

The address window hit formula in this controller is as follows:

Hit = (BASE & MASK) == (ADDR & MASK)

Addr_out = TRANS_EN? TRANS b0 ADDR & ~MASK: ADDR

It is worth mentioning that, when configuring the address window register, the MASK

should be all 1 and all 0. 0 in the MASK

Is the size of the address window.

The address of this window is the address received on the HT bus.A read or write command that falls at the address of this window will not be sent to the second level CACHE or invalidate the first level CACHE, but will be sent directly to memory or another address space.This means that read and write commands in the address window will not maintain IO CACHE consistency.This window is for operations that are not hit in the CACHE and can improve memory efficiency, such as access to video memory.

Offset: 0xf0
Reset value: 0x00000000
Name: HT bus Uncache address window 0 enable (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_uncache0_en | 1 | 0 x0 | R/W | HT bus uncache address window 0, enable signal |
| 30 | Ht_uncache0_ trans_en | 1 | 0 x0 | R/W | HT bus uncache address window 1, mapping enable signal |
| 29:0 | Ht_uncache0_ trans [53:24] | 16 | 0 x0 | R/W | HT bus uncache address window 0, translated address [53:24], for LS3A1000D and below, [29:23] Fixed 0 |

Offset: 0xf4
Reset value: 0x00000000
Name: HT bus Uncache address window 0 base address (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_uncache0_ base [they] | 16 | 0 x0 | R/W | HT bus uncache address window 0, address base address [they] |
| 15:0 | Ht_uncache0_ mask [they] | 16 | 0 x0 | R/W | HT bus uncache address window 0, address mask [they] |

Offset: 0xf8
Reset value: 0x00000000
Name: HT bus Uncache address window 1 enable (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| 31 | ht_uncache1_en | 1 | 0 x0 | R/W | HT bus uncache address window 1, enable the signal |
| 30 | Ht_uncache1_ trans_en | 1 | 0 x0 | R/W | HT bus uncache address window 1, mapping enable signal |
| 29:0 | Ht_uncache1_ trans [53:24] | 16 | 0 x0 | R/W | HT bus uncache address window 1, translated address [53:24], for LS3A1000D and below, [29:23] Fixed 0 |

Offset: 0xfc
Reset value: 0x00000000
Name: HT bus Uncache address window 1 base address (internal access)

| A domain | A domain name | A wide | Reset value | access | describe |
|---|---|---|---|---|---|
| Caused the | Ht_uncache1_ base [they] | 16 | 0 x0 | R/W | HT bus uncache address window 1, address base address [they] |

| | | | | | |
|---|---|---|---|---|---|
| 15:0 | Ht_uncache1_ mask [they] | 16 | 0 x0 | R/W | HT bus uncache address window 1, address mask [they] |

## 9.5.14 The HyperTransport bus conpoints the access methods to the space

The protocol of the HyperTransport interface software layer is basically the same as that of the PCI protocol, and the configuration access may be slightly different because it is directly related to the underlying protocol. As shown in table 9-5, the configuration access space is located at addresses 0xFD_FE00_0000 ~ 0xFD_FFFF_FFFFh.For configuration access in the PCI protocol, the following implementation is implemented in loongson 3.

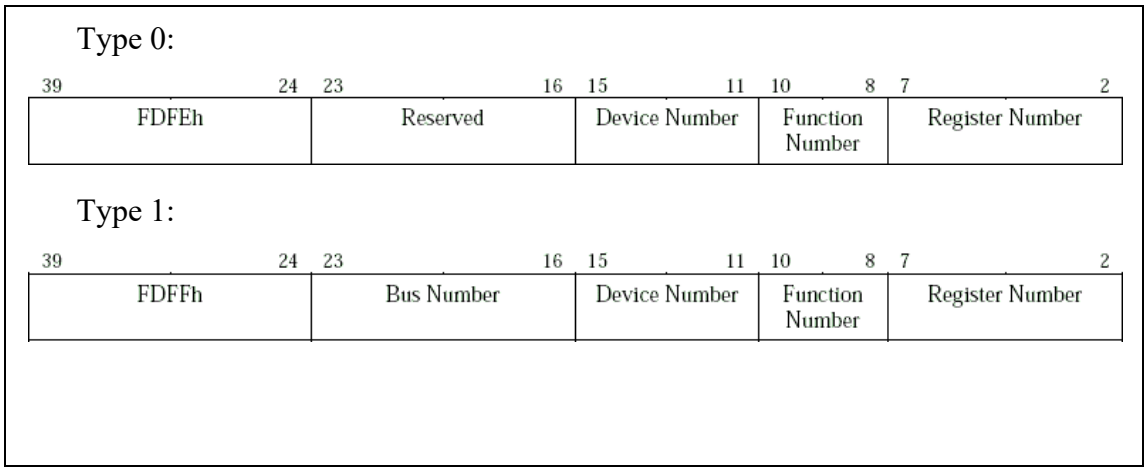



Figure 9-1 configuration access of HT protocol in loongson 3

## 9.6 HyperTransport multi-processor support

The loongson 3 processor USES the HyperTransport interface to interconnect multiple processors, and the hardware can automatically maintain the consistency requests between the four chips.Here are two ways to interconnect multiple processors:

**Four pieces of loong core no. 3 interconnection structure**

Four pieces of CPU are connected to form a ring structure.Each CPU USES two 8-bit controllers of HT0 to connect with two adjacent ones, among which HTx_LO is the main device and HTx_HI is the slave device. Thus, the following interconnection structure can be obtained:

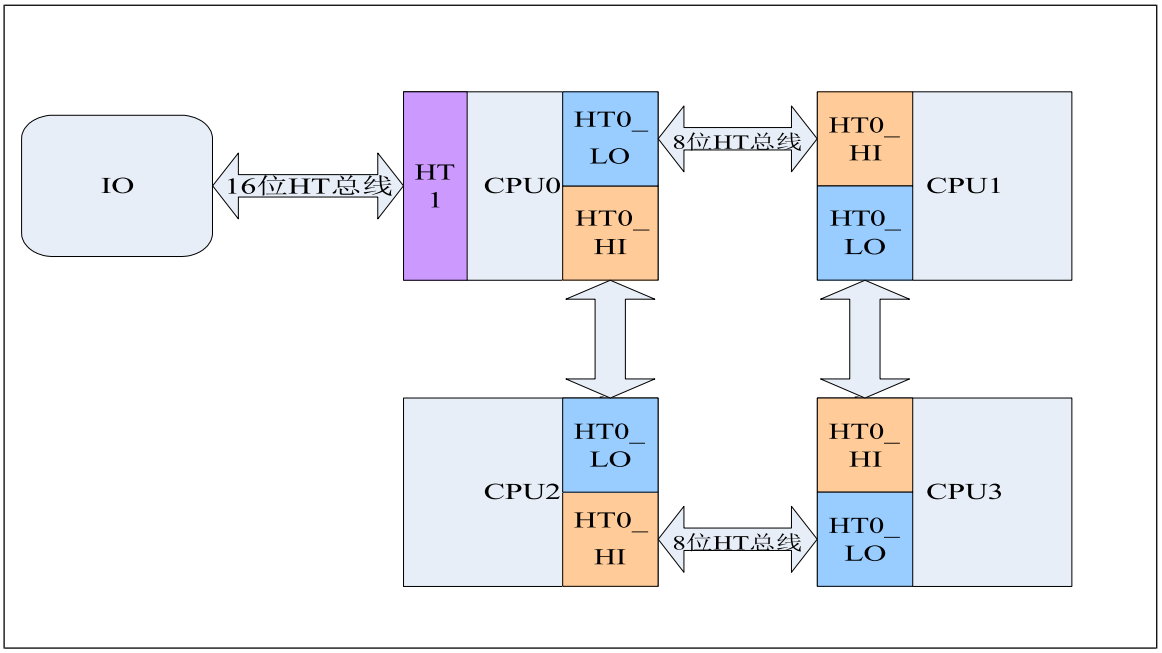



FIG. 9-2 interconnection structure of four-piece loongson no. 3

**Loongson 3 interconnection routing**

Loongson 3 interconnection routing adopts simple x-y routing method.In other words, when routing, first X, then Y, take four chips as an example, ID

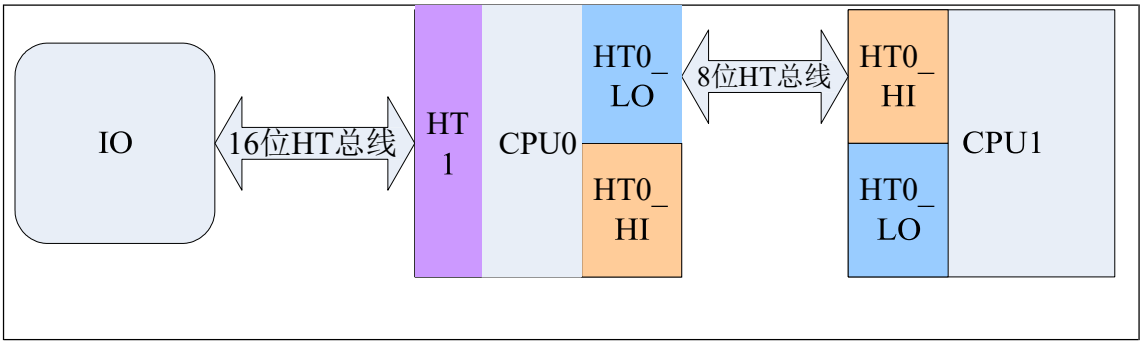
The Numbers are 0, 01, 10, 11.If a request is made from 11 to 00, route 11 to 00, starting with X ^ 2

We go from 11 to 10, then we go from 10 to 00 in the Y direction.When the request response returns from 00 to 11, the route goes first in the X direction, from 00 to 01, and then in the Y direction, from 01 to 11.As you can see, these are two different routing lines. Because of the characteristics of this algorithm, we will take a different approach when building the interconnection between two chips.

**Two - piece loong core 3 interconnection structure**

Because of the nature of the fixed routing algorithm, there are two different approaches to building a two-chip interconnection. The first is the use of 8-bit HT bus interconnection.In this mode of interconnection, only 8-bit HT interconnection can be used between two processors.The Numbers of the two chips are 00 and 01 respectively. According to the routing algorithm, we can know that when the two chips visit each other, they all pass through the 8-bit HT bus consistent with the four-chip interconnection.As follows:

Figure 9-3 8-bit interconnection structure of two pieces of loong core no. 3




However, our HT bus can be in 16-bit mode at its widest, so the connection mode that maximizes the bandwidth should be 16

Bit interconnection. In loongson iii, 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 according to the routing table as before, so that we can use the 16-bit bus when interconnecting.Therefore, we only need to properly configure the 16-bit mode of CPU0 and CPU1 and properly connect the high-low bit bus to interconnect using the 16-bit HT bus.This interconnection structure can also use the 8-bit HT bus protocol to access each other.The resulting interconnection structure is as follows:

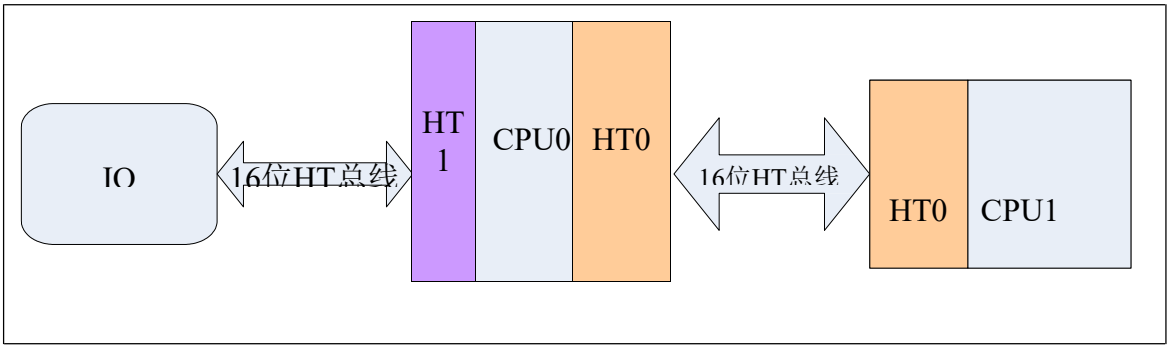



Figure 9-4 16-bit interconnection structure of two pieces of loongson 3

# 10 Low speed IO controller configuration

Loongson 3 I/O controller includes PCI/ pci-x controller, LPC controller, UART controller, SPI controller, GPIO and configuration register.These I/O controllers share a AXI port, and CPU requests are decoded to the corresponding device.

## 10.1 PCI controller/PCI - X

Loongson 3's PCI/pci-x controller can either control the entire system as the main bridge or work on the PCI/pci-x bus as a common PCI/pci-x device.Its implementation conforms to pci-x 1.0b and PCI 2.3 specifications.The loongson 3's PCI/ pci-x controller also has a PCI/ pci-x arbitrator built in.

The configuration header for the PCI/ pci-x controller is located at 256 bytes starting at 0x1FE00000, as shown in table 13-1.

Table 10-1 PCIX controller configuration header

| The byte 3 | 2 bytes | 1 byte | Byte 0 | address |
|---|---|---|---|---|
| The Device ID | | Vendor ID | | 00 |
| The Status | | The Command | | 04 |
| The Class Code | | | Revision ID | 08 |
| BIST | The Header Type | Latency Timer | CacheLine Size | 0 c |
| The Base Address Register 0 | | | | 10 |
| The Base Address Register 1 | | | | 14 |
| The Base Address Register 2 | | | | 18 |
| The Base Address Register 3 | | | | 1 c |
| The Base Address Register 4 | | | | 20 |
| The Base Address Register 5 | | | | 24 |
| | | | | 28 |
| Subsystem ID | | Subsystem Vendor ID | | 2 c |
| | | | | 30 |
| | | | "Capabilities Pointer | 34 |
| | | | | 38 |
| Maximum Latency | A Minimum Grant | Interrupt Pin | Interrupt Line | 3 c |
| Implementation Specific Register (ISR40) | | | | 40 |
| Implementation Specific Register (ISR44) | | | | 44 |
| Implementation Specific Register (ISR48) | | | | 48 |
| Implementation Specific Register (ISR4C) | | | | 4 c |
| Implementation Specific Register (ISR50) | | | | 50 |
| Implementation Specific Register (ISR54) | | | | 54 |
| Implementation Specific Register (ISR58) | | | | 58 |

| | |
|---|---|
| | . |
| PCIX Command Register | E0 |
| PCIX Status Register | E4 |

Loongson 3A1000's PCIX controller supports three 64-bit Windows, including {BAR1, BAR0}, {BAR3, BAR2},

{BAR5, BAR4} three pairs of register configuration Windows 0, 1, 2 base addresses.The size, enabling, and other details of the window are controlled by the other three corresponding registers, PCI_Hit0_Sel, PCI_Hit1_Sel, and PCI_Hit2_Sel. See table 2 for the specific bit fields.

Table 10-2 PCI control registers

| A dom ain | The field name | acce ss | Rese t val ue | instr uctio ns |
|---|---|---|---|---|
| REG_40 | | | | |
| 31 | tar_read_io | Read and write (write 1 Qing) | 0 | The target side receives access to IO or the unprefetched region |
| 30 | tar_read_discard | Read and write (write 1 Qing) | 0 | The delay request on the target side is discarded |
| 29 | tar_resp_delay | Read and write | 0 | Target accesses when to give delay/split 0: after timeout 1: immediately |
| 28 | tar_delay_retry | Read and write | 0 | Target accesses the retry policy 0: according to internal logic (see 29 bits) 1: try again immediately |
| 27 | tar_read_abort_en | Read and write | 0 | If target times out for an internal read request, do you want to respond with target-abort |
| " | Reserved | - | 0 | |
| 24 | tar_write_abort_en | Read and write | 0 | If target times out for an internal write request, do you want to respond with target-abort |
| 23 | tar_master_abort | Read and write | 0 | Whether master-abort is allowed |

| | | | | |
|---|---|---|---|---|
| Lift up | tar_subseq_timeout | Read and write | 000 | Target subsequent delay timeout<br>000:8 cycles<br>Others: not supported |
| He hath | tar_init_timeout | Read and write | 0000 | Target initial delay timeout<br>PCI mode<br>0:16 cycles<br>1-7: disable the counter<br>8-15:8 to 15 cycles<br>The timeout count is fixed at 8 cycles in PCIX mode, where the configuration has the greatest impact<br>Delay access number<br>0: 8 delay access<br>8: 1 delay access<br>9: 2 delay access<br>10: 3 delay access<br>11: 4 delay access<br>12: 5 delay access<br>13: 6 delay access<br>14: 7 delay access |
| | | | | 15: 8 delay access |
| Indeed , | tar_pref_boundary | Read and write | 000 h. | Prefetching boundary configuration (in 16 bytes)<br>FFF: 64KB to 16byte FFE:<br>64KB to 32byte<br>FF8:64 KB to 128 byte |
| 3 | tar_pref_bound_en | Read and write | 0 | Configuration using tar_pref_boundary<br>0: prefetch to device boundary<br>1: using tar_pref_boundary |
| 2 | Reserved | - | 0 | |
| 1 | tar_splitw_ctrl | Read and write | 0 | Target split write control<br>0: blocks access other than Posted Memory Write<br>1: block all access until split is completed |
| 0 | mas_lat_timeout | Read and write | 0 | Disable mater access timeout<br>0: allows master access timeout<br>1: not allowed |
| REG_44 | | | | |
| 31:0 | Reserved | - | - | |
| REG_48 | | | | |
| 31:0 | tar_pending_seq | Read and write | 0 | The request number vector that<br>target has not processed can be<br>marked with 1 |
| REG_4C | | | | |
| charm | Reserved | - | - | |
| 29 | mas_write_defer | Read and write | 0 | Allow subsequent reads to override previous incomplete writes (for PCI only) |

| | | | | |
|---|---|---|---|---|
| 28 | mas_read_defer | Read and write | 0 | Allow subsequent reads and writes to override previous incomplete reads<br>(for PCI only) |
| 27 | mas_io_defer_cnt | Read and write | 0 | The maximum number of external IO requests<br>Zero: by the control<br>1:1. |
| they | mas_read_defer_cnt | Read and write | 010 | Master supports maximum number of external reads (for PCI only)<br>Zero: 8<br>1-7:1-7<br>Note: a dual address cycle access accounts for two items |
| Ephron; | err_seq_id | read-only | 00 h | The target/master error number |
| 15 | err_type | read-only | 0 | Target /master error command type<br>Zero: |
| 14 | err_module | read-only | 0 | Wrong module<br>0: target<br>1: master |
| 13 | system_error | Read and write | 0 | Target /master system error (write 1 clear) |
| 12 | data_parity_error | Read and write | 0 | Target /master data parity error (write 1 clear) |
| 11 | ctrl_parity_error | Read and write | 0 | Target /master address odd and even wrong (write 1 clear) |

| | | | | |
|---|---|---|---|---|
| 10:0 | Reserved | - | - | |
| REG_50 | | | | |
| 31:0 | mas_pending_seq | Read and write | 0 | The request bit vector that the master didn't finish processing<br>The corresponding bit can be written 1 |
| REG_54 | | | | |
| 31:0 | mas_split_err | Read and write | 0 | Split returns the error request bit vector |
| REG_58 | | | | |
| charm | Reserved | - | - | |
| then | tar_split_priority | Read and write | 0 | Target split returns the priority<br>Zero is the highest, three is the lowest |
| But after | mas_req_priority | Read and write | 0 | The external priority of master<br>Zero is the highest, three is the lowest |
| 25 | Priority_en | Read and write | 0 | Arbitrate algorithm (arbitrate between master's access and target's split return)<br>0: fixed priority<br>1: rotary |
| Where upon certain | reserve | - | - | |
| 17 | mas_retry_aborted | Read and write | 0 | Master retry cancel (write 1 clear) |
| 16 | mas_trdy_timeout | Read and write | 0 | Master TRDY timeout count |
| " | mas_retry_value | Read and write | 00 h | Number of master retries<br>0: infinite retry<br>1-255:1-255 |
| away | mas_trdy_count | Read and write | 00 h | Master TRDY timeout counter<br>0: disable<br>1-255:1-255 |

Before initiating configuration space reads and writes, the application should configure the PCIMap_Cfg register to tell the controller the type of configuration operation to initiate and the value on the high 16-bit address line.The configuration header for the corresponding device can then be accessed by reading and writing to the 2K space starting at 0x1fe80000.The device number is obtained by encoding from low to high priority according to PCIMap_Cfg[15:0].

The configuration action address generation is shown in figure 10-1.

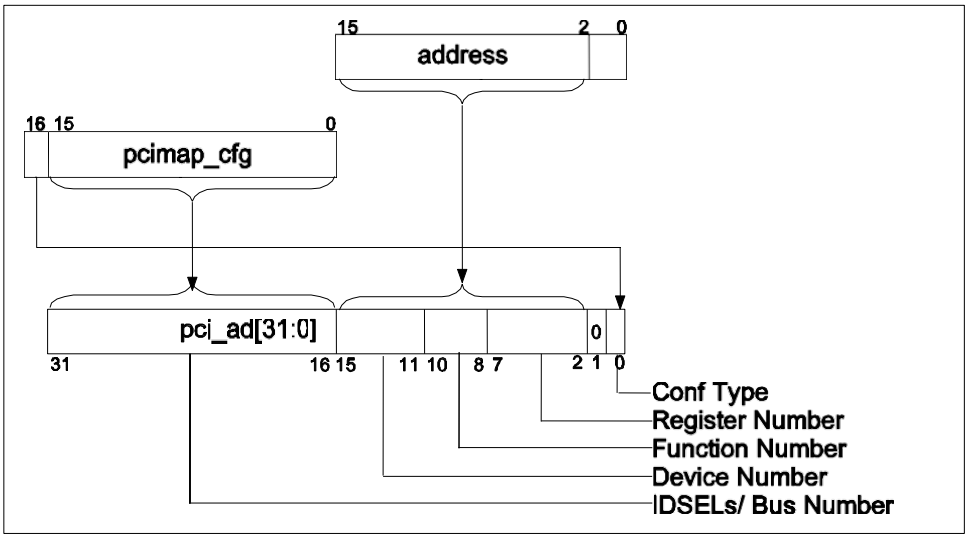



Figure 10-1 configure read and write bus address generation

The PCI/PCIX arbitrator implements two-stage rotation arbitration, bus docking and isolation of damaged master devices. Its configuration and status are shown in the PXArb_Config and PXArb_Status registers. PCI/PCIX bus request and reply line assignment is shown in table 13-3.

Table 10-3 PCI/PCIX bus request and reply line allocation

| Request and reply lines | describe |
|---|---|
| 0 | Internal integrated PCI/PCIX controller |
| 7:1 | External requests 6~0 |

The route-based arbitration algorithm provides two levels, the second level as a whole as a member of the first level scheduling together. When multiple devices apply for the bus at the same time, the first level device is transferred once per cycle, and the highest priority device in the second level can get the bus.

The mediators are designed to be switched whenever conditions permit, which can make them abnormal for some PCI devices that do not conform to the protocol.Using a forced priority allows these devices to occupy the bus through continuous requests.

Bus docking refers to whether one of the devices is selected to give a permit signal when no device requests to use the bus. Directly initiating bus operations can improve efficiency for devices that are already permitted.Loongson 2F's PCI arbitrator offers two docking modes: the last master and the default master.If you can't dock on special occasions, you can set the mediator to dock to the default master device 0 (internal controller) with a dependency delay of 0.

## 10.2 LPC controller

LPC controller has the following characteristics:

Comply with LPC1.1 specification

Support for LPC access timeout counters

Memory Read, Memory write access types are supported

Firmware Memory Read, Firmware Memory Write access type (single byte)

I/O read and I/O write access types are supported

Support for Memory access type address translation

Support for the Serizlized IRQ specification, providing 17 interrupt sources

The address space distribution of LPC controller is shown in table 4:

Table 10-4 address space distribution of LPC controller

| Address name | Address range | The size of the |
|---|---|---|
| LPC Boot | 0 x1fc0_0000 x1fd0_0000 0 | 1 mbyte |
| LPC Memory | 0 x1c00_0000 x1e00_0000 0 | 32 mbyte |
| LPC I/O | 0 x1ff0_0000 x1ff1_0000 0 | 64 kbyte |
| LPC Register | 0 x1fe0_0200 x1fe0_0300 0 | 256 byte |

The LPC Boot address space is the address space first accessed by the processor when the system is started. This address space supports LPC Memory or Firmware Memory access types.The type of access emitted at system startup is controlled by LPC_ROM_INTEL pins. LPC_ROM_INTEL pins send out LPC Firmware Memory access when pulled up, and LPC_ROM_INTEL pins send out LPC Memory access type when pulled down.

The LPC Memory address space is the address space that the system accesses with Memory/Firmware Memory. The type of Memory access issued by an LPC controller is determined by the LPC controller's configuration register, LPC_MEM_IS_FWH. The address sent by the processor to this address space can be translated. The converted address is set by the LPC controller's configuration register LPC_MEM_TRANS.

The access that the processor sends to the LPC I/O address space is sent to the LPC bus

according to the LPC I/O access type.The address is 16 bits below the address space.

The LPC controller configuration registers have a total of 3 32-bit registers. The meaning of the configuration register is shown in table 13-5:

Table 10-5 LPC configuration register meanings

| A domain | The field name | access | Reset value | instructions |
|---|---|---|---|---|
| REG0 | | | | |
| REG0 [hands] | SIRQ_EN | Read and write | 0 | SIRQ enables control |
| REG0 [take] | LPC_MEM_TRANS | Read and write | 0 | LPC Memory space address translation control |
| REG0 [15:0] | LPC_SYNC_TIMEOUT | Read and write | 0 | LPC access timeout counter |
| REG1 | | | | |
| REG1 [hands] | LPC_MEM_IS_FWH | Read and write | 0 | LPC Memory space Firmware<br>Memory access type Settings |
| REG1 [17:0] | LPC_INT_EN | Read and write | 0 | LPC SIRQ interrupt enablement |
| REG2 | | | | |
| REG2 [17:0] | LPC_INT_SRC | Read and write | 0 | LPC SIRQ interrupts source indication |
| REG3 | | | | |
| REG3 [17:0] | LPC_INT_CLEAR | write | 0 | LPC SIRQ interrupt clear |

## 10.3 UART controller

UART controllers have the following characteristics

Full duplex asynchronous data receive/send

Programmable data format

16 bit programmable clock counter

Receive timeout detection is supported

A multi-interrupt system with arbitration

Work in FIFO mode only

Register and function compatible NS16550A

This module has two parallel working UART interfaces, the function registers are exactly the same, but the base address is different.

The physical base address of the UART0 register is 0x1FE001E0. The physical base address of UART1 register is 0x1FE001E8.

### 10.3.1 Data register (DAT)

Chinese name: data transfer register register bit width: [7:0]

Offset: 0x00
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| away | Tx FIFO | 8 | W. | Data transfer register |

### 10.3.2 Interrupt enabled register (IER)

Chinese name: interrupt enabled
register register bit width: [7:0]

Offset: 0x01
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| The log | Reserved | 4 | RW | reserve |
| 3 | IME | 1 | RW | Modem status interrupted to enable '0' -- off '1' -- on |
| 2 | ILE | 1 | RW | Receiver line status interruption enables' 0 '-- off' 1 '-- on |

| | | | | |
|---|---|---|---|---|
| 1 | ITxE | 1 | RW | Transfer save register for air break enable '0' -- close '1' -- hit<br>open |
| 0 | IRxE | 1 | RW | Receive valid data interrupt enables' 0 '-- close' 1 '-- open |

### 10.3.3 Interrupt identification register (IIR)

Chinese name: interrupt source register

Register bit width: [7:0]

Offset: 0x02

Reset value: 0xc1

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| The log | Reserved | 4 | R | reserve |
| 3:1 | II | 3 | R | Interrupt source representation bits, as shown in the table below |
| 0 | INTp | 1 | R | Interrupt representation bit |

Interrupt control menu

| Bit 3 | 2 - | Bit 1 | priority | Interrupt type | The interrup t source | Interrupt reset control |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 st | Receiving line<br>state | Parity, overflow, or frame error, or<br>Interrupt interrupted | Read the LSR |
| 0 | 1 | 0 | 2 nd | Received valid<br>data | The number of characters of FIFO is reached<br>The level of the trigger | FIFO has a low number of characters<br>In the trigger value |
| 1 | 1 | 0 | 2 nd | Receive a timeout | There is at least one<br>character in FIFO, but no<br>operation, including read<br>and write operations, in four<br>character times<br>As a | Read receive FIFO |
| 0 | 0 | 1 | 3 rd | Transmission save send<br>The store is empty | The transfer save register is empty | Write data to THR or<br>More IIR |
| 0 | 0 | 0 | 4 th | Modem state | CTS, DSR, RI or DCD. | Read MSR |

### 10.3.4 FIFO control register (FCR)

Chinese name: FIFO control

register register bit width: [7:0]

Offset: 0x02
Reset value: 0xc0

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| but | TL | 2 | W. | The trigger value of '00' -- 1 byte '01' -- 4 bytes for receiving FIFO's interrupt application '10' -- 8 bytes' 11 '-- 14 bytes |
| o | Reserved | 3 | W. | reserve |
| 2 | Txset | 1 | W. | '1' clears the content of sending FIFO and reset its logic |
| 1 | Rxset | 1 | W. | '1' clears the contents of the received FIFO and reset its logic |
| 0 | Reserved | 1 | W. | reserve |

### 10.3.5 Line control register (LCR)

Chinese name: circuit control
register register bit width: [7:0]

Offset: 0x03
Reset value: 0x03

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| 7 | dlab | 1 | RW | Frequency divider latch access bit '1' - access operation frequency divider latch '0' - access operation normal register |
| 6 | BCB | 1 | RW | Interrupt control bit '1' - the output of the serial port is set to 0(interrupted state). '0' - normal operation |
| 5 | .spb | 1 | RW | Specifies parity bits '0' - do not specify parity bits '1' - if the LCR[4] bit is 1, the transmission and check parity bits are 0. If the LCR[4] bit is 0, the transmission and check parity bit is 1. |

| | | | | |
|---|---|---|---|---|
| 4 | eps | 1 | RW | Parity bit selection<br>'0' - has an odd number of 1's in each character<br>(including data and parity bits)<br>'1' - there are an even number of 1's in each character |
| 3 | PE | 1 | RW | Parity bit enabled<br>'0' - no parity bits<br>'1' - the parity bit is generated on the output, and<br>the parity bit is judged on the input |
| 2 | sb | 1 | RW | Defines the number of bits that generate the stop bit |
| | | | | '0' - 1 stop bit<br>'1' - 1.5 stop bits at 5 character length, others<br>The length is 2 stop bits |
| 1-0 | bec | 2 | RW | Sets the number of bits per character<br>'00' -- 5 '01' -- 6<br>'10' - 7 '11' - 8 ' |

### 10.3.6 MODEM control register (MCR)

Chinese name: Modem control
register register bit width: [7:0]

Offset: 0x04
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| 7:5 | Reserved | 3 | W. | reserve |

| | | | | |
|---|---|---|---|---|
| 4 | Loop | 1 | W. | Loop mode control bit<br>'0' - normal operation<br>'1' - loop mode.In loopback mode, TXD output is always 1, and the output shift register is connected directly to the input shift register.The other links are as follows.<br>DTR □ DSR RTS<br>□ CTS<br>Out1 □ RI<br>Out2 □ DCD |
| 3 | OUT2 | 1 | W. | Connect to DCD input in loopback mode |
| 2 | The OUT1 | 1 | W. | Connect to RI input in loop mode |
| 1 | RTSC | 1 | W. | RTS signal control bit |
| 0 | DTRC | 1 | W. | DTR signal control bit |

### 10.3.7 Line status register (LSR)

Chinese name: line status register
register bit width: [7:0]
Offset: 0x05
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| 7 | The ERROR | 1 | R | Error representation bit<br>'1' - at least one with a parity bit error, frame error, or interrupt.<br>'0' - no errors |
| 6 | TE | 1 | R | The null transport represents the bit<br>'1' - transfer FIFO and transfer shift registers are empty. Zero when writing data to the transmitted FIFO<br>'0' -- has data |

| | | | | |
|---|---|---|---|---|
| 5 | TFE | 1 | R | Transmission of FIFO bit empty represents bit<br>'1' -- the current transmission FIFO is empty, and when data is written to the transmission FIFO, it will be zero<br>'0' -- has data |
| 4 | BI | 1 | R | Interrupts interrupt bits<br>'1' - received start bit + data + odd and even bit + stop bit are all 0, that is, there is interrupt interrupt<br>'0' -- no interruptions |
| 3 | FE | 1 | R | Frame errors represent bits<br>'1' - received data without stop bits<br>'0' - no errors |
| 2 | PE | 1 | R | A parity bit error represents a bit<br>'1' - there is a parity error in the received data<br>'0' - no parity errors |
| 1 | OE | 1 | R | Data overflow represents bits<br>'1' -- data overflow<br>'0' - no overflow |
| 0 | Dr. | 1 | R | Receiving data effectively represents bits<br>'0' - no data in a FIFO<br>'1' - data in FIFO |

When this register is read, LSR[4:1] and LSR[7] are cleared, LSR[6:5] is cleared when writing data to transmit FIFO, and LSR[0] judges the receiving FIFO.

### 10.3.8 MODEM status register (MSR)

Chinese name: Modem status register
register bit width: [7:0]
Offset: 0x06
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| 7 | CDCD | 1 | R | DCD input the inverse of the value, or in loop mode to Out2 |
| 6 | CRI | 1 | R | The inverse of the RI input value, or connected to OUT1 in loopback mode |
| 5 | CDSR | 1 | R | The inverse of the DSR input value, or in loopback |

| | | | | |
|---|---|---|---|---|
| | | | | mode connected to DTR |
| 4 | CCTS | 1 | R | Invert the input value of the CTS, or connect to the RTS in loopback mode |
| 3 | DDCD | 1 | R | DDCD indicating a |
| 2 | TERI | 1 | R | RI edge detection. RI state changes from low to high |
| 1 | DDSR | 1 | R | DDSR indicating a |
| 0 | DCTS | 1 | R | DCTS indicating a |

### 10.3.9 Frequency divider latch

Chinese name: frequency division latch 1
Register bit width: [7:0]
Offset: 0x00
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| away | LSB | 8 | RW | Store the lower 8 bits of the frequency divider latch |

Chinese name: frequency division latch 2
Register bit width: [7:0]
Offset: 0x01
Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| away | The MSB | 8 | RW | Store the high 8 bits of the frequency divider latch |

## 10.4 SPI controller

SPI controller has the following characteristics:

Full duplex synchronous serial port data transmission

Support for variable-length byte transfers up to 4

Master mode support

A mode failure generates an error flag and issues an interrupt request

Double buffer receiver

Polarity and phase programmable serial clock

SPI can be controlled in wait mode

The SPI controller module register physical address is 0x1FE001F0.

### 10.4.1 Control register (SPCR)

Chinese name: control

register register bit width:

[7:0]

Offset: 0x00

Reset value: 0x10

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| 7 | Spie | 1 | RW | Interrupt output to make the signal highly efficient |
| 6 | The spe | 1 | RW | The system works to make the signal highly efficient |
| 5 | Reserved | 1 | RW | reserve |
| 4 | MSTR | 1 | RW | Select bit in master mode, this bit is always 1 |
| 3 | cpol | 1 | RW | Clock polarity |
| 2 | cpha | 1 | RW | Clock phase 1 is in opposite phase and 0 is the same |
| 1-0 | SPR | 2 | RW | Sclk_o frequency divider setting, to be used with sper's spre |

### 10.4.2 Status register (SPSR)

Chinese name: status

register register bit width:

[7:0]

Offset: 0x01

Reset value: 0x05

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| 7 | spif | 1 | RW | The interrupt sign bit 1 indicates that there is an interrupt request, and if you write 1, it will clear zero |
| 6 | wcol | 1 | RW | Write register overflow bit 1 indicates overflow, write 1<br><br>reset |
| when | Reserved | 2 | RW | reserve |
| 3 | wffull | 1 | RW | Write register full flag 1 to indicate full |
| 2 | wfempty | 1 | RW | Write register empty flag 1 to indicate empty |
| 1 | rffull | 1 | RW | Read register full flag 1 to indicate full |
| 0 | rfempty | 1 | RW | Read register empty flag 1 to indicate empty |

### 10.4.3 Data register (TxFIFO)

Chinese name: data transfer

register register bit width: [7:0]

Offset: 0x02

Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| away | Tx FIFO | 8 | W. | Data transfer register |

### 10.4.4 External register (SPER)

Chinese name: external

register register bit width:

[7:0]

Offset: 0x03

Reset value: 0x00

| A domain | A domain name | A wide | access | describe |
|---|---|---|---|---|
| but | icnt | 2 | RW | How many bytes have been transmitted before the interrupt request signal is sent<br><br>00 -- 1 byte 01 -- 2 bytes<br><br>10-3 bytes 11-3 bytes |
| 5-2 | Reserved | 4 | RW | reserve |
| 1-0 | spre | 2 | RW | Set the frequency division ratio with the Spr |

Frequency division coefficient:

| 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 coefficient | 2 | 4 | 16 | 32 | 8 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |

## 10.5 IO controller configuration

The configuration register is used to configure the address window of the PCI/ pic-x controller, the arbitrator, and the GPIO controller.These registers are listed in table 10-6 and detailed descriptions of registers are given in table 10-7.This part of the register is base address 0x1FE00100.

Table 10-6 IO control registers

| address | registe r | instructions |
|---|---|---|
| 00 | PonCfg | The electric configuration |
| 04 | GenCfg | General configuration |
| 08 | reserve | |
| 0 c | reserve | |
| 10 | PCIMap | PCI mapping |
| 14 | PCIX_Bridge_Cfg | PCI/X bridge related configuration |
| 18 | PCIMap_Cfg | PCI concatenates the read and write device address |
| 1 c | GPIO_Data | GPIO data |
| 20 | GPIO_EN | GPIO direction |
| 24 | reserve | |
| 28 | reserve | |
| 2 c | reserve | |
| 30 | reserve | |
| 34 | reserve | |
| 38 | reserve | |
| 3 c | reserve | |
| 40 | Mem_Win_Base_L | Prefetch window base address low 32 bits |
| 44 | Mem_Win_Base_H | Prefetch window base address height 32 bits |
| 48 | Mem_Win_Mask_L | Prefetch window mask lower 32 bits |
| 4 c | Mem_Win_Mask_H | Prefetch window mask height 32 bits |
| 50 | PCI_Hit0_Sel_L | PCI window 0 controls low 32 bits |
| 54 | PCI_Hit0_Sel_H | PCI window 0 controls the height of 32 bits |

| | | |
|---|---|---|
| 58 | PCI_Hit1_Sel_L | PCI window 1 controls the low 32-bit |
| 5 c | PCI_Hit1_Sel_H | PCI window 1 controls the high 32-bit |
| 60 | PCI_Hit2_Sel_L | PCI window 2 controls low 32 bits |
| 64 | PCI_Hit2_Sel_H | PCI window 2 controls the high 32-bit |
| 68 | PXArb_Config | PCIX arbitrator configuration |
| 6 c | PXArb_Status | PCIX arbitrator state |
| 70 | | |
| 74 | | |
| 78 | | |
| 7 c | | |
| 80 | Chip Config | Chip configuration register |
| 84 | | |
| 88 | | |
| 8 c | | |
| 90 | Chip Sample | Chip sampling register |

Table 10-7 registers are described in detail

| A domain | The field name | access | Reset value | instructions |
|---|---|---|---|---|
| CR00: PonCfg | | | | |
| 15:0 | pcix_bus_dev | read-only | Lio_ad [away] | The total amount used by the CPU in the PCIX Agent mode<br>Line and equipment number |
| " | reserve | read-only | Lio_ad [or] | |
| Ephron; | pon_pci_configi | read-only | pci_configi | PCI_Configi pin value |
| came | reserve | read-only | | |
| CR04: reserve | | | | |
| 31:0 | reserve | read-only | 0 | |
| CR08: reserve | | | | |
| 31:0 | reserve | read-only | 0 | |
| CR10: PCIMap | | | | |
| 5-0 | trans_lo0 | Read and write | 0 | The PCI_Mem_Lo0 window maps the address six bits higher |
| but | trans_lo1 | Read and | 0 | The PCI_Mem_Lo1 window maps the address six bits higher |

| | | | | |
|---|---|---|---|---|
| | | write | | |
| " | trans_lo2 | Read and write | 0 | The PCI_Mem_Lo2 window maps the address to a height of 6 bits |
| all | reserve | read-only | 0 | |
| CR14: PCIX_Bridge_Cfg | | | | |
| 5-0 | pcix_rgate | Read and write | 6 'h18 | PCIX mode to DDR2 read number threshold |
| 6 | pcix_ro_en | Read and write | 0 | The PCIX bridge allows writing over reading |
| all | reserve | read-only | 0 | |
| CR18: PCIMap_Cfg | | | | |
| 15:0 | dev_addr | Read and write | 0 | PCI configuration reads and writes when the AD line is 16 bits high |
| 16 | conf_type | Read and write | 0 | Configure the types of reads and writes |
| 31:17 | reserve | read-only | 0 | |
| CR1C: GPIO_Data | | | | |
| 15:0 | gpio_out | Read and write | 0 | GPIO outputs data |
| Caused the | gpio_in | Read and write | 0 | GPIO enters data |
| CR20: GPIO_EN | | | | |
| 15:0 | gpio_en | Read and write | 16 'hFFFF | High is the input and low is the output |
| Caused the | reserve | read-only | 0 | |
| CR3C: reserve | | | | |
| 31:0 | reserve | read-only | 0 | reserve |
| CR24, 2 c, 30,34,38: reservations | | | | |
| As shown in table 11 | | | | |
| CR50, 60 (54/58, 5 C / : _Sel_ PCI_Hit * * | | | | |
| 0 | reserve | read-only | 0 | |
| 2:1 | pci_img_size | Read and write | 2 'b11 | 00:32; 10:64; Others: invalid |
| 3 | pref_en | Read and write | 0 | Prefetching can make |
| 4 | reserve | read-only | 0 | |
| 62:12 | bar_mask | Read and write | 0 | Window size mask (high 1, low 0) |
| 63 | burst_cap | Read and | 1 | Whether to allow burst transmission |

| | | | | |
|---|---|---|---|---|
| | | write | | |
| CR68: PXArb_Config | | | | |
| 0 | device_en | Read and write | 1 | External device allowed |
| 1 | disable_broken | Read and write | 0 | Disable damaged master devices |
| 2 | default_mas_en | Read and write | 1 | The bus is docked to the default main device<br>0: dock to the last main device<br>1: dock to the default main device |
| o | default_master | Read and write | 0 | The bus is docked with the default main device number |
| but | park_delay | Read and write | 2 'bl1 | The delay from the beginning of no device request bus to the triggering of docked default device behavior<br>00:0 cycle<br>01:8 cycles<br>10:32 cycle<br>11:128 cycles |
| " | level | Read and write | 8 'h01-2 | A device in the first stage |
| Ephron; | rude_dev | Read and write | 0 | Force priority devices |
| | | | | The PCI device corresponding to the bit of 1 can be obtained after the bus<br>To occupy the bus with continuous requests |
| away | reserve | read-only | 0 | |
| CR6C: PXArb_Status | | | | |
| away | broken_master | read-only | 0 | Damaged master device (reset when changing disabled policy) |
| 10:8 | Last_master | read-only | 0 | Finally, the main device of the bus is used |
| lustful | reserve | read-only | 0 | |
| CR80: Chip config | | | | |
| The 2-0 | Freq_scale_ctrl | Read and write | 3 'b111 | Processor core frequency division |
| 3 | DDR_Clksel_en | Read and write | 1 'b0 | Whether to use software to configure DDR octave |
| 8 | Disable_ddr2_confspace | Read and write | 1 'b0 | Whether to disable DDR configuration space |
| 9 | DDR_buffer_cpu | Read and write | 1 'b0 | Whether to turn on the DDR read access buffer |
| 12 | Core0_en | Read and write | 1 'b1 | Whether to enable processor core 0 |
| 13 | Core1_en | Read | 1 'b1 | Whether to enable processor core 1 |

| | | | | |
|---|---|---|---|---|
| | | and write | | |
| 14 | Core2_en | Read and write | 1 'b1 | Whether to enable processor core 2 |
| 15 | Core3_en | Read and write | 1 'b1 | Whether to enable processor core 3 |
| 16 | Mc0_en | Read and write | 1 'b1 | Whether to enable DDR controller 0 |
| 17 | Mc1_en | Read and write | 1 'b1 | Whether to enable DDR controller 1 |
| 18 | DDR_reset0 | Read and write | 1 'b1 | Software reset DDR controller 0 |
| 19 | DDR_reset1 | Read and write | 1 'b1 | Software reset DDR controller 1 |
| 22 | HT0_en | Read and write | 1 'b1 | Whether HT controller 0 is enabled |
| 23 | HT1_en | Read and write | 1 'b1 | Whether to enable HT controller 1 1: on 0: disable |
| throughout | DDR_Clksel | Read and write | 5 'b11111 | Software configuration DDR clock frequency doubling relationship (when DDR_Clksel_en is 1) |
| take | HT_freq_scale_ctrl0 | Read and write | 3 'b111 | HT controller 0 frequency division |
| That which | HT_freq_scale_ctrl0 | Read and write | 3 'b111 | HT controller 1 frequency division |
| 35 | Mc0_prefetch_disable | Read and write | 1 'b0 | Whether or not to disable MC0 prefetch (which has different performance effects on different program behaviors) |
| 36 | Mc1_prefetch_disable | Read and write | 1 'b0 | Whether to disable MC1 prefetch (which has different performance effects on different program behaviors) |
| other | | read-only | | reserve |
| CR90: Chip Sample | | | | |
| 15:0 | Pad2v5_ctrl | Read and write | 16 'h780 | 2 v5pad control |
| Caused the | Pad3v3_ctrl | Read and write | 16 'h780 | 3 v3pad control |
| 47:32 | Sys_clksel | read-only | | On board frequency doubling setting |
| spoilers | Bad_ip_core | read-only | | Whether 4 processor cores are bad or not |
| 53:52 | Bad_ip_ddr | read-only | | Whether 2 DDR controllers are bad |
| 57:56 | Bad_ip_ht | read-only | | Is the 2 HT controllers bad |

| | | | | |
|---|---|---|---|---|
| A 102-96 | Thsens0_out | read-only | | The temperature sensor has a temperature of 0, which is used to monitor the temperature near the second-level cache with an accuracy of +/-6 ℃ |
| 103 | Thsens0_overflow | read-only | | Temperature sensor 0 temperature overflow (over 128 degrees) |
| A 110-104 | Thsens1_out | read-only | | Temperature sensor 1 temperature, used to monitor the temperature near the processor core, precision is +/-6 degrees Celsius |
| 111 | Thsens1_overflow | read-only | | Temperature sensor 1 temperature overflow (over 128 degrees) |
| other | | read-only | | reserve |

# The second part

## System software programming guide

# 11 Configuration and use of interrupts

## 11.1 Interrupted flow

The loongson 3A1000 process for handling interrupts, from external interrupt requests to kernel handling of interrupts, is the same.The diagram below shows the flow chart of the interrupt handling of the 3a-690e board card.

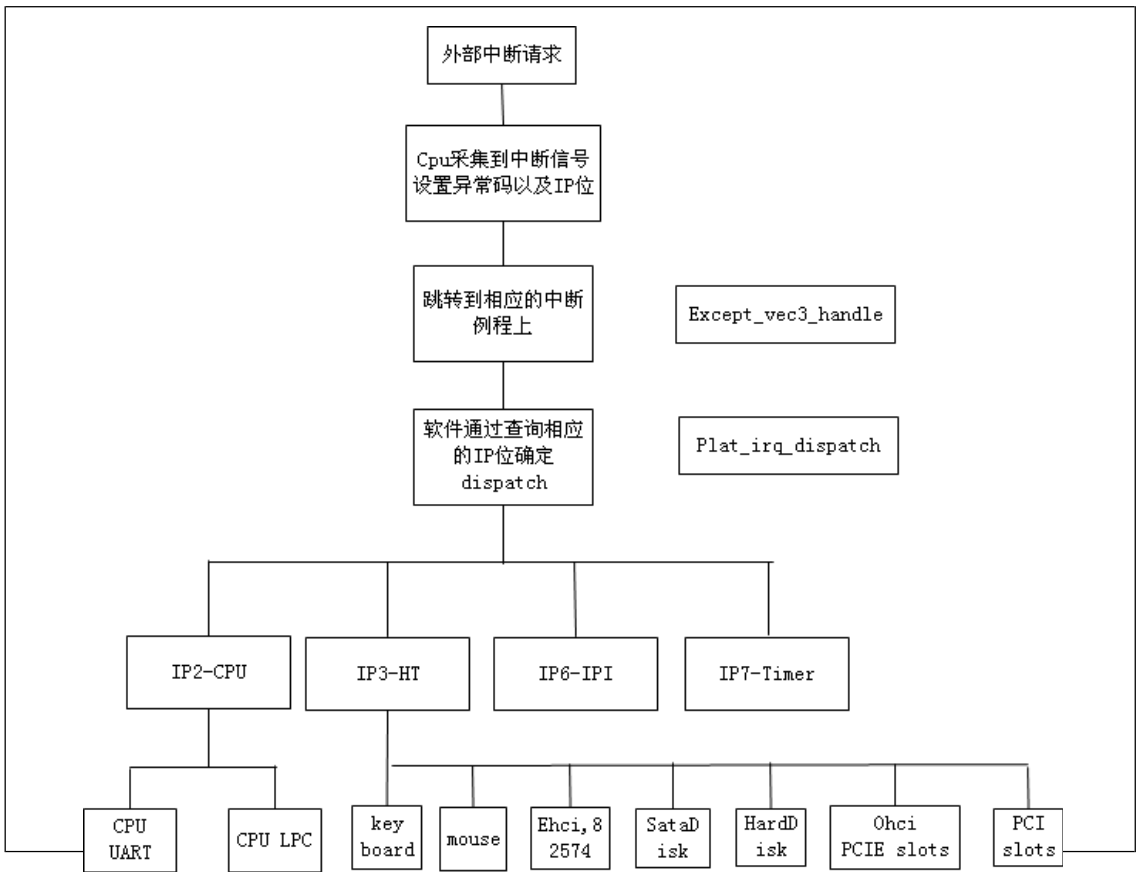



Figure 11-1 3a-690e interrupt flow chart

## 11.2 Interrupt routing and interrupt enabling

The loongson 3A1000 chip supports up to 32 interrupt sources, which are managed in a unified manner. As shown in the figure below, any IO interrupt source can be configured to enable or disable, trigger, and route the target processor core interrupt pins.

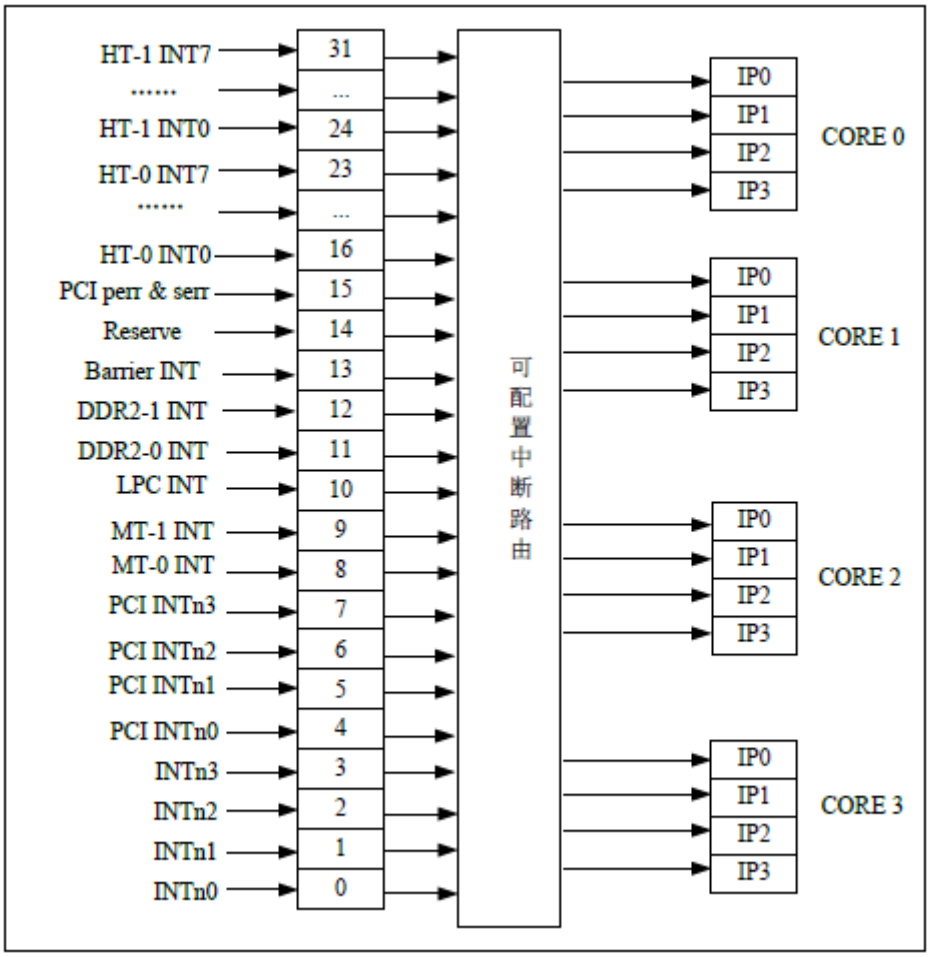



Figure 11-2 schematic diagram of interru 1

### 11.2.1 Interrupt routing

With four processor cores integrated into the loson 3A1000, the 32-bit interrupt source described above can be configured to select the target processor core to interrupt.Further, the interrupt source can choose to route to any of the processor core interrupts INT0 through INT3, IP2 through IP5 corresponding to CP0_Status.In other words, IP0~IP3 of CORE0~CORE3 as shown in the figure above corresponds to IP2~IP5 of CP0_Status. Each of the 32 I/O interrupt sources corresponds to an 8-bit routing controller, whose format and address are shown in tables 1 and 2 below. The routing register USES a vector approach for routing, such as 0x48 to indicate routing to INT2 of processor no. 3.

Table 11-1 description of interrupt routing registers

| A domain | instructions |
|---|---|
| 3-0 | Routing the processor core vector number |

| The log | Routing the processor core interrupt pin vector number |
|---|---|

Table 11-2 interrupt routing register addresses

| The name of the | Address offset | describe | The name of the | Address offset | describe |
|---|---|---|---|---|---|
| Entry0 | 0 x1400 | Sys_int0 | Entry16 | 0 x1410 | HT0 - int0 |
| Entry1 | 0 x1401 | Sys_int1 | Entry17 | 0 x1411 | HT0 - int1 |
| Entry2 | 0 x1402 | Sys_int2 | Entry18 | 0 x1412 | HT0 - int2 |
| Entry3 | 0 x1403 | Sys_int3 | Entry19 | 0 x1413 | HT0 - int3 |
| Entry4 | 0 x1404 | Pci_int0 | Entry20 | 0 x1414 | HT0 - int4 |
| Entry5 | 0 x1405 | Pci_int1 | Entry21 | 0 x1415 | HT0 - int5 |
| Entry6 | 0 x1406 | Pci_int2 | Entry22 | 0 x1416 | HT0 - int6 |
| Entry7 | 0 x1407 | Pci_int3 | Entry23 | 0 x1417 | HT0 - int7 |
| Entry8 | 0 x1408 | Matrix int0 | Entry24 | 0 x1418 | HT1 - int0 |
| Entry9 | 0 x1409 | Matrix int1 | Entry25 | 0 x1419 | HT1 - int1 |
| Entry10 | 0 x140a | Lpc int | Entry26 | 0 x141a | HT1 - int2 |
| Entry11 | 0 x140b | Mc0 | Entry27 | 0 x141b | HT1 - int3 |
| Entry12 | 0 x140c | Mc1 | Entry28 | 0 x141c | HT1 - int4 |
| Entry13 | 0 x140d | The Barrier | Entry29 | 0 x141d | HT1 - int5 |
| Entry14 | 0 x140e | reserve | Entry30 | 0 x141e | HT1 - int6 |
| Entry15 | 0 x140f | Pci_perr/serr | Entry31 | 0 x141f | HT1 - int7 |

For the convenience of understanding, the configuration of the 3a-690e and 3a-kd60 board CARDS on the interrupt route is given below:

A) 3 a - 690 e

The hardware connection is "CPU serial port + HT1 connecting the north and south bridge".The routing setting is:

```
/* Route the LPC interrupt to Core0 INT0, corresponding to Cp0_Status of IP2*/
* (volatile unsigned char *) 0 x900000003ff0140a = 0 x11;

/* Route the HT1 interrupt to Core0 INT1, corresponding to Cp0_Status of IP3*/
0 x900000003ff01418 * (volatile unsigned char *)    =   0 x21;
0 x900000003ff01419 * (volatile unsigned char *)    =   0 x21;
0 x900000003ff0141a * (volatile unsigned char *)    =   0 x21;
```

0 x900000003ff0141b * (volatile unsigned char *) = 0 x21;

0 x900000003ff0141c * (volatile unsigned char *) = 0 x21;

0 x900000003ff0141d * (volatile unsigned char *) = 0 x21;

```
* (volatile unsigned char *) 0 x900000003ff0141e = 0 x21;
* (volatile unsigned char *) 0 x900000003ff0141f = 0 x21;
```

B) 3 a - KD60

The hardware connection is "CPU serial port + PCI + CPU" 8259, and the routing is set as:

```
/* Route the LPC interrupt to Core0 INT0, corresponding to Cp0_Status of IP2*/
* (volatile unsigned char *) 0 x900000003ff0140a = 0 x11;

/* Route the I8259 interrupt to Core0 INT1, corresponding to Cp0_Status of IP3*/
*(volatile unsigned char *)(0x900000003ff01400) = 0x21;

/* Route PCI interrupt to Core0 INT3, corresponding to Cp0_Status of IP5*/
*(volatile unsigned char *)(0x900000003ff01404) = 0x81;
```

### 11.2.2 The interrupt enable

The interruption-related configuration registers control the corresponding interrupts in the form of bits. See table 11-3 for the connection and property configuration of interrupt control bits.The interrupt Enable configuration has three registers: Intenset, Intenclr, and Inten.The Intenset sets the interrupt enable, and the interrupt corresponding to the bit write 1 in the Intenset register is enabled. The Intenclr clears interrupts to enable, and the interrupt corresponding to the Intenclr register write 1 is cleared. The Inten register reads the current status of each interrupt enable. Interrupt signals in the form of pulses (such as PCI_SERR) are selected by the Intedge configuration register, with write 1 for pulse trigger and write 0 for level trigger.The interrupt handler can clear the pulse record by the corresponding bit of the Intenclr.

Table 11-3 interrupt control bit connection and property configuration

| The name of the | Address offset | desc ribe |
|---|---|---|
| Intisr | 0 x1420 | 32-bit interrupt status register |
| Inten | 0 x1424 | 32-bit interrupt enabled status register |
| Intenset | 0 x1428 | The 32-bit setting enables the register |
| Intenclr | 0 x142c | 32-bit clear enable register |
| Intedge | 0 x1438 | 32-bit trigger mode register |
| CORE0_INTISR | 0 x1440 | Routing to the 32-bit interrupt state of CORE0 |
| CORE1_INTISR | 0 x1448 | Routing a 32-bit interrupt state to CORE1 |

| | | |
|---|---|---|
| CORE2_INTISR | 0 x1450 | Routing a 32-bit interrupt state to CORE2 |
| CORE3_INTISR | 0 x1458 | Routing a 32-bit interrupt state to CORE3 |

Not only do you need an enabling IO controller, specific to the connected IO, but you also need a single interrupt controller if it has its own interrupt controller

Only enable, such as LPC interrupt controller, HT interrupt controller, the specific register configuration can see the register manual.Some interrupt controllers that may need to be enabled are listed below:

```
/* Enable the IO interrupt controller, LPC (10) and HT (16~31) */
t=*(volatile unsigned int*)0x900000003ff01428;
*(volatile unsigned int*)0x900000003ff01428 = t | (0xffff << 16) | (0x1 << 10);

/* Enable LPC interrupt controller*/
*(volatile unsigned int*)(0xffffffffffbfe00200 + 0x00)=0x80000000;
/* the 18-bit interrpt enable bit */
*(volatile unsigned int*)(0xffffffffffbfe00200 + 0x04) = 0x0;

/* Enable HT interrupt, only used 7 interrupt vectors*/
*(volatile unsigned int*)0x90000EFDFB0000A0 = 0xffffff7f;
```

## 11.3 Interrupt to distribute

When an interrupt occurs, the hardware sets the cause register's Excode domain and the associated IP bits.Then into the software processing process, the software queries the Excode domain to determine which type of exception, to choose which exception handling routine to use.If it's the hardware interrupt we're talking about, it goes into the platform-specific interrupt dispatch function. The interrupt distribution function then performs primary interrupt distribution according to the IP bit and IM bit (interrupt masking) of the cause register, and then performs secondary distribution according to the bit field of the interrupt number, and then performs the specific do_IRQ interrupt operation.

At startup, the kernel maps each exception vector to each exception handler routine. There are 32 types of exceptions, and we're just going to talk about exception number 0, which is hardware interrupts.In the trap_init() function, the generic entry address for the exception is set to 0x80000180, which holds a function pointer of expect_vec3_generic, as shown in the kernel arch/ MIPS /kernel/ genex.s.Expect_vec3_generic () functions are entered into the corresponding routine functions based on the Excode code obtained.In trap_init(), the exception code 0, that is, the hardware interrupt, is related to the handle_int

routine. For handle_int, see kernel arch/ MIPS /kernel/ genex.s. Hanle_int eventually jumps to the plat_irq_diapatch() platform-specific interrupt distribution function, which does the interrupt level distribution.

Take 3a-690e as an example, IP0 and IP1 of Cause register correspond to soft interrupt, IP6 corresponds to intercore interrupt, IP7 corresponds to clock interrupt, IP2 corresponds to Cpu serial port and LPC interrupt, and IP3 corresponds to the interrupt of routing to HT1. HT1 is connected to the north bridge 690E, the north bridge is connected to the south bridge SB600, and the interrupts of the north and south Bridges are all controlled by the 8259 on the south bridge. The interrupts of each peripheral such as USB and sata are routed to the 8259 controller, as shown in arch/ MIPS /kernel/fixup_ev3a.c

In the godson3a_smbus_fixup function, where the Register is defined in AMD SB600 Register Reference handmanual. PDF. Pcibios_map_irq function is the scanning and interrupt number assignment of PCI and PCIE slots. Other functions in the file fixup_ev3a.c are mainly the interrupt number assignment of some fixed PCI devices. Details of interrupt allocation and routing to 8259 can be found in the code and notes of the file fixup_ev3a.c.

When the distribution is complete, run the do_IRQ() function and jump to the corresponding specific driver for execution.

# 12 Configuration and use of serial port

## 12.1 Optional serial port

As a communication interface, serial port is mainly used for system debugging. The principle of its work is to configure the serial port's baud rate, data bit, stop bit, check bit related registers, so that the serial port can send and receive bytes by bit.

Currently, there are two types of UART available on 3A1000: one is the UART of CPU, including UART0 and UART1. The base address of the serial port register of UART0 is 0xbfe001e0, and the base address of the serial port register of UART1 is 0xbfe001e8, with baud rates of 115200.There is also a class of UART for LPC, whose base address is 0xbff003f8, with a baud rate of 57600.In the setting, the data bits are all set to 8 bits, and the stop bit is 1 bit. There is no check and no flow control.

## 12.2 PMON's serial port configuration

In pmon, the macro USE_LPC_UART is used to distinguish between the two types of serial ports. The main files involved in setting up pmon are start.S and tgt_machdep.c.

Take the CPU's UART0 as an example. First, there is a function to initialize the serial port in pmon's startup.s (see the UART controller section for the use of registers) :

The LEAF (initserial)

```
    Lia0, GS3_UART_BASE
    Li      T1, t1
    sb      128, 3          // access the frequency division register
    li      (a0) t1,        # divider, the highest possible baud rate
    sb      0 x12           // frequency division register 1, which stores the lower 8 bits
                            of the frequency division register, and the calculation
            T1, 0 (a0)      formula is
                            // working clock /(baud rate *16), in this case,
                            33M/(115200*16)
    li      T1, 0 x0
                            # divider, the highest possible baud rate
    Sb      T1, 1           // frequency division register 2, store the high 8 bits of
                            frequency division register
    li      (a0) t1,
    sb      t1, 3 3
                            // the data bit is 8 bits
            (a0)
```

```
Li          T1, t1, 0
sb          1 (a0)          // no interrupts are used
Lit1, 71
Sbt1,2(a0)// set the FIFO control register
jrra
The nop
END (initserial)
```

GS3_UART_BASE is also defined in the start.s file, 0xbfe001e0.
In tgt_machdep.c, the structure ConfigEntry also gives the setting of UART, and the function ns16550 is also the sending and receiving processing function of UART.

## 12.3 Serial port configuration for the Linux kernel

In the Linux kernel to a serial port configuration is mainly has three parts: include/asm/serial h, the arch/MIPS/kernel / 8250 - platform. C, the arch/MIPS/lemote ev3a/dbg_io. C.These three configuration files relate to the setting of the serial port base address, depending on the specific use of the serial port. In the kernel, there is also a macro definition CONFIG_CPU_UART in arch/ MIPS /Kconfig to select whether to choose LPC serial port or CPU serial port.Even if the CPU UART is selected, whether it is UART0 or UART1 still needs to check whether the above three files have correctly defined the serial port base address.

In the arch/MIPS/lemote ev3a/dbg_io. C is mainly used in the kernel boot process, the interrupt initialization phase, for the convenience of kernel debugging using a simple serial printing way, including function prom_printf () in the kernel debugging stage will often be used to, it will call putDebugChar () going to output characters one by one to print to the console.Include /asm/serial.h provides a macro definition for the serial driver/serial/8250.c, such as UART0, which USES the CPU, as follows:

```
# define STD_SERIAL_PORT_DEFNS \
/ * UART_CLK PORT_IRQ FLAGS * /
{, baud_base = BASE_BAUD, irq = 58,
\
. Flags = STD_COM_FLAGS,.iomem_base = (u8*)(0xffffffffffffbfe001e0),\
. Io_type = SERIAL_IO_MEM}
```

The driver used is the standard 8250/16x50 series serial driver.The serial port interrupt

number is 58, according to the mining

CPU serial port or LPC serial port are used, and the distribution of interrupts is slightly different. The part about serial port interrupts in irq.c is as follows:

```
... .
} else if (pending & CAUSEF_IP2) {// For LPC
    #ifdef CONFIG_CPU_UART
            Do_IRQ (58);
  # the else
            Irq = *(volatile unsigned int*)(0xffffffffbfe00200 + 0x08);   If
            ((irq & 0 x2))
                    Do_IRQ
            (1);   If ((irq & 0
            x1000))
                    Do_IRQ (12);
            If ((irq & 0 x10))
    # endif           Do_IRQ (58);
```

If it is the serial port of CPU, directly deal with no. 58 interrupt, but if it is the LPC serial port, it needs to read the relevant bit field of LPC controller to judge whether the interrupt is the keyboard interrupt, the mouse interrupt or the serial port interrupt, and the serial port interrupt number is still no. 58.

# 13 EJTAG debugging

## 13.1 Introduce EJTAG

EJTAG(Enhanced JTAG) is a specification defined by MIPS according to the basic construction and function extension of IEEE1149.1 protocol. It is a hardware/software subsystem used to support on-chip debugging. The EJTAG specification is well integrated with the original MIPS architecture by defining a new debugging mode, including dedicated instructions, running modes, and address Spaces. The basic idea of debug mode is to use the exception handling mechanism so that the debugged software cannot be aware of the presence of the debugger.In addition, the processor extends the address space in debug mode to access debug control registers and debug interfaces mapped to memory regions.The following figure shows a common EJTAG debugging system composition, including:

- Debug Host: run the Debug application and control the EJTAG cable
- Target Board: contains the Board card of the debugged chip and provides EJTAG interface

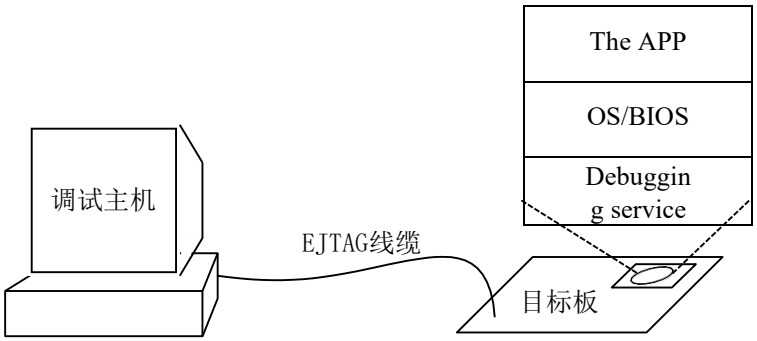



Figure 13-1 EJTAG debugging system

The debug host can use EJTAG line to put the processor on the target board into debug mode to execute the debug service routine.The debugging service has two entries, 0xbfc00480 located in the BIOS and 0xff200200 located in the EJTAG debugging memory interface, which are selected by the debugging host.

The EJTAG specification maps two pieces of the processor's debug mode address space to the debug control register (drseg) and the debug memory interface (dmseg).When the processor performs the debugging service to access the debugging interface, the access requests issued are blocked in the debugging interface.At this point, the debug host is able to detect the memory interface's memory access request and respond to it.In simple terms, the debug service program can access a portion of the memory space located on the debug host.

With the memory space controlled by the debugging host, the debugging service can perform almost any imaginable function, because the code of the debugging service program itself can be provided by the debugging host (currently loongson 3A1000 and 2G samples can't get a reference from dmseg, but they can execute memory access instructions).

The PC sampling function of EJTAG specification is also supported in longson 3a1000/2g.

## 13.2 EJTAG tool

### 13.2.1 Environment to prepare

- EJTAG wire, parallel port to 14 EJTAG port
- Linux debug host with parallel port and root access
- Add the debug service in PMON

### 13.2.2 PC sample

Running jtag_1/4/16 on the debug host completes a PC sampling of the processor core.The program suffix refers to the number of processor cores in EJTAG interface, one 2G is 1, one 3A1000 is 4, and four 3A1000 strings together are 16. PC sampling does not require debugging the server.

The PC sampling value in 3a1000/2g is not accurate, and there is jitter. Users should ignore the lower two bits of the sampling result. The real PC pointer in the processor core may be 0~4 instructions behind the program flow of the corresponding instruction of the sampled PC, that is, if the transfer is included, a real PC may fall near the transfer target.

PC sampling should not stand still without a regular cycle or a software-controlled stop clock.If it does, it means the processor is not running properly.

### 13.2.3 Read/write memory

Run pracc_1/4/16 and specify in the parameters the processor core number, access type, access address, value to write, and so on to perform the debugging service.Its working principle is to initialize a parameter area in the memory space of debugging according to the task to be completed, and control the execution of debugging service by cooperating with the debugging service program.The debug service first saves several registers to the debug host, then runs on these empty registers, reads out the relevant parameters and executes, and finally restores the previously saved registers.

It is important to note that this functionality requires that the processor core performing the debugging service also be able to execute instructions. If the operation is abnormal, there may be a hardware failure.

### 13.2.4 Execute instructions

Read the meaning of the bfc00480| value

```
cpu@ubuntu:~/ejtag$ ./pracc_4 0 0xffffffffbfc00480 d r
target = 0, addr = ffffffffbfc00480, dword read
breaking...
ctrl = 00049000
stat = 00008008
pracc write, size=3,   address = 000000000000000f, value = 0000000000000000 |t1
pracc write, size=3,   address = 0000000000000017, value = 0000000000000000 |t2
pracc write, size=3,   address = 000000000000001f, value = ffffffff80e1060 |t3
pracc write, size=3,   address = 0000000000000027, value = ffffffff8000b899 |t4
pracc write, size=3,   address = 000000000000002f, value = ffffffffffffffff |t5
pracc write, size=3,   address = 0000000000000037, value = ffffffff80110000 |t6
pracc write, size=3,   address = 000000000000003f, value = ffffffff800f7428 |t7
pracc write, size=3,   address = 0000000000000047, value = 00000000340000e0|status
pracc write, size=3,   address = 000000000000004f, value = 0000000080034483|config
pracc write, size=3,   address = 0000000000000057, value = 0000000040008000|cause
pracc write, size=3,   address = 000000000000005f, value = 0000000000000033 |a0
pracc write, size=3,   address = 0000000000000067, value = ffffffff80120000 |a1
pracc write, size=3,   address = 000000000000006f, value = 0000000000000000 |a2
pracc read, size=3,   address = 0000000000000207, value = fff3ffffbfc00480
pracc write, size=3,   address = 0000000000000107, value = fff3ffffbfc00480
pracc write, size=3,   address = 0000000000000107, value = 0000000000000003
pracc write, size=3,   address = 0000000000000107, value = 00000000000001fc
pracc write, size=3,   address = 0000000000000107, value = 0000000000000001
pracc write, size=3,   address = 0000000000000107, value = ffffffffbfc00480
pracc write, size=3,   address = 000000000000020f, value = 3c08ff2040a8f800
return  3c08ff2040a8f800
```

- Write bfc00480.

```
cpu@ubuntu:/home/cpu/ejtag$ ./pracc_4 0 0xffffffffbfc00480 d w 0x0
target = 0, addr = ffffffffbfc00480, dword write with 0000000000000000
press <enter> to confirm..
breaking...
ctrl = 00049000
stat = 60008008
pracc write, size=3,   address = 000000000000000f, value = 0000000000000000
pracc write, size=3,   address = 0000000000000017, value = 0000000000000000
pracc write, size=3,   address = 000000000000001f, value = ffffffff80e1060
pracc write, size=3,   address = 0000000000000027, value = ffffffff8000b899
pracc write, size=3,   address = 000000000000002f, value = ffffffffffffffff
pracc write, size=3,   address = 0000000000000037, value = ffffffff80110000
pracc write, size=3,   address = 000000000000003f, value = ffffffff800f7428
pracc write, size=3,   address = 0000000000000047, value = 00000000340000e0
pracc write, size=3,   address = 000000000000004f, value = 0000000080034483
pracc write, size=3,   address = 0000000000000057, value = 0000000040008000
pracc write, size=3,   address = 000000000000005f, value = ffffffff8ec08c00
pracc write, size=3,   address = 0000000000000067, value = ffffffff8ec08400
pracc write, size=3,   address = 000000000000006f, value = 0000000000000000
pracc read, size=3,   address = 0000000000000207, value = 0000000000000000
pracc write, size=3,   address = 0000000000000107, value = 0000000000000000
pracc read, size=3,   address = 0000000000000217, value = fff3ffffbfc00480
pracc read, size=3,   address = 000000000000021f, value = 0000000000000000
pracc write, size=3,   address = 0000000000000107, value = fff3ffffbfc00480
pracc write, size=3,   address = 0000000000000107, value = 0000000000000003
pracc write, size=3,   address = 0000000000000107, value = 00000000000001fc
```

```
pracc write, size=3,   address = 0000000000000107, value = 0000000000000001
pracc write, size=3,   address = 0000000000000107, value = ffffffffbfc00480
pracc write, size=3,   address = 000000000000021f, value = 0000000000000000
pracc read, size=3,   address = 000000000000000f, value = 0000000000000000
pracc read, size=3,   address = 0000000000000017, value = 0000000000000000
pracc read, size=3,   address = 000000000000001f, value = ffffffff80e1060
pracc read, size=3,   address = 0000000000000027, value = ffffffff8000b899
pracc read, size=3,   address = 000000000000002f, value = ffffffffffffffff
pracc read, size=3,   address = 0000000000000037, value = ffffffff80110000
pracc read, size=3,   address = 000000000000003f, value = ffffffff800f7428
```

debug service code annotations
# start. S
/ * the Debug exception * /
Bfc00480 align7 / * * /
/ / / / / / / / / / / / / / / / / / / / / / / / / /
/ / / / / / / / / / / / / / / / # define
COP_0_DESAVE $31
. Set mips64
/ / save the context
Dmtc0t0, COP_0_DESAVE// saves a register for the dmseg pointer
Luit0, 0 xff20 / /
Sdt1, 0 x08 (t0) / / pressure stack
Sdt2, 0 x10 (t0)
Sdt3, 0 x18 (t0)
Sdt4, 0 x20 (t0)
Sdt5, 0 x28 (t0)
Sdt6, 0 x30 (t0)
Sdt7, 0 x38 (t0)
Dmfc0t1, COP_0_STATUS_REG// outputs several cp0 registers
Sdt1, 0 x40 (t0)
Dmfc0t1, COP_0_CONFIG sdt1,
0 x48 (t0)
Dmfc0t1, COP_0_CAUSE_REG sdt1,
0 x50 (t0)
Sda0, 0x58(t0)// other registers of interest
Sda1, 0 x60 (t0)
Sda2, 0 x68 (t0)

# define _t19
# define _t210
# define _t311
# define _t412
\ #define dextu(dest, SRC, MSBD, DLSB) \
The word
(26) (0 x1f < < | ((SRC & 0 x1f) < < 21) | ((dest & 0 x1f) < < 16) | (((MSBD) & 0 x1f) < < 11) | (((DLSB) & 0 x1f) < < 6)
| (0 x2))
\ #define dinsu(dest, SRC, DMSB, DLSB) \
Word (26) (0 x1f < < | ((SRC & 0 x1f) < < 21) | ((dest & 0 x1f) < < 16) | (((DMSB) & 0 x1f) < < 11) | (((DLSB) & 0 x1f) < < 6) | (0 x6))

/ / exec_main
Ldt1, 0x200(t0)// read parameter addr/size/count beqzt1, read_end

Sdt1, 0 x100 (t0) / / debug...
Dextu (_t2 _t1, 2-1, 48-32)

```
        sd          T2, 0 x100 (t0)                 / / the debug...
        Dextu       (_t3, _t1, 9-1, 50-32)
        sd          t3, 0x100(t0)                   / / the debug...
        Dextu       (_t4, _t1, 1-1, 47-32)
        sd          t4, 0x100(t0)                   / / the debug...
        dsubu        T4, $0, t4                     / / sign bit the
                                                    extend
        Dinsu (_t1, _t4, 58-32, 48-32) // address back sdt1,
        0x100(t0)// debug...

        // case t2 0,1,2,3-> lb,lh,lw,ld
        beqzlt2, 1f
        Lbt5, 0x0(t1)
        addiut2, t2, minus 1


        Beqzlt2, 1 f
        Lht5, 0x0(t1)
        addiut2, t2, -1


        Beqzlt2, 1 f
        Lwt5, 0x0(t1)
        addiut2, t2, -1


        Ldt5, 0 x0 (t1)
1:
        Sdt5, 0x208(t0)// reads return value
 Read_end:
       / / write
        Ldt1, 0x210(t0)// addr/size/count beqzt1, write_end

        Ldt5, 0x218(t0)// write parameter
        Sdt1, 0 x100 (t0) / / debug...
        Dextu (_t2 _t1, 2-1, 48-32)
        Sdt2, 0 x100 (t0) / / debug...
        Dextu (_t3 _t1, 9-1, 50-32)
        Sdt3, 0 x100 (t0) / / debug...
        Dextu (_t4 _t1, 1-1, 47-32)
        Sdt4, 0 x100 (t0) / / debug...
        Dsubut4, $0, t4// sign bit extend dinsu(_t1, _t4, 58-32, 48-32)
        // address back

        Sdt1, 0 x100 (t0) / / debug...
        // case t2 0,1,2,3-> sb,sh,sw,sd
        beqzlt2, 1f
        Sbt5, 0x0(t1)
        addiut2, t2, -1


        Beqzlt2, 1 f
        Sht5, 0x0(t1)
        addiut2, t2, -1


        Beqzlt2, 1 f
        Swt5, 0x0(t1)
```

```
        addiut2, t2, -1


        Sdt5, 0 x0 (t1)
1:
        Sdt5, 0x218(t0)// write response
Write_end:
```

```
/ / restore context
ld      T1, 0 x08 (t0)          / / return stack
ld      T2, 0 x10 (t0)
L       T4, t3, 0 x18
d       (t0) 0 x20
ld      (t0)
ld      T5, 0 x28 (t0)
L       T7 has t6, 0
d       x30 (t0), 0
ld      x38 (t0)
dmfc0    T0, COP_0_DESAVE
deret                           // debug
                                exception
                                returned
```

### 13.2.5 Online GDB debugging

Due to the bugs in the current sample, the online debugging function needs to make some changes in the common MIPS debugging platform and add the corresponding debugging service program.This functionality has not yet been implemented.

# 14 Configure the translation in the address window

Loongson 3A1000 adopts a two-stage crossover switch structure, and the two-stage crossover switch window can be configured separately to control sending a specific address to a specific receiving end for processing.In addition, the HyperTransport controller also has internal control over the chip's internal and external accessible address Windows.

## 14.1 First and second level cross switch address window configuration method

Each primary port on the crossover switch has eight address Windows to configure.Each address window is composed of three 64-bit registers, BASE, MASK and MMAP. BASE is aligned with K bytes, that is, the space divided by each address window is at least 1KB. MASK is the window MASK; MMAP is the mapped address of the window; meanwhile, [2:0] represents the number of the corresponding target slave port; MMAP[4] represents the allowed reference; MMAP[5] represents the allowed block read; MMAP[7] represents the enabled window.

The judgment of window hit is as follows:

Main port address & MASK == BASE

The formula of translation from port address after mapping is as follows:

Slave port address = primary port address & (~(MASK))|MMAP & MASK

Note that MMAP[4] and MMAP[5] must be 1 for a level 1 crossover switch.For secondary cross-switches, slave ports that do not allow Cache access or referential access can set MMAP[4] or MMAP[5] to 0.

In addition, if a level 1 cross-switch is used to map the level 2 Cache address, the mapped address, i.e., the "slave port address", must be the same as the pre-mapped address, i.e., the "primary port address".Configurations mapped to HyperTransport addresses and secondary cross-switches are not subject to this constraint.

## 14.2 Level 1 cross switch address window

The level 1 crosswalk has the default route setting, which is not displayed by the address window configuration register, and only addresses that are not hit by any address window are interpreted by the default route.

For the primary port of the first-level crossover switch, that is, the primary device side that issues the

request, it includes the following:

0 Primary port no. : processor core 0

1 Primary port no. 1: processor core 1

2 Main port no. : processor core 2

3 Main port no. : processor core 3

6 Primary port no. : HyperTransport 0

7 Primary port no. 1: HyperTransport 1

For the slave port of the first-level crossover switch, that is, the slave device side that issues the request, it includes the following:

0 Number from port: level 2 Cache 0

1 Number from port: level 2 Cache 1

2 Number from port: level 2 Cache 2

3 Number from port: level 2 Cache 3

6 Port number: HyperTransport 0

7 Port number: HyperTransport 1

The address window configuration of each primary port is independent of each other, with eight configurable Windows each.Configuration Windows take precedence

Descending, starting from configuration window 0, the first hit window routes the address.The priority order is as follows:

| | | |
|---|---|---|
| The highest | Configuration window 0 | |
| | Configuration window 1 | |
| | Configuration window 2 | |
| | Configuration window 3 | |
| | Configuration window 4 | |
| | Configuration window 5 | |
| | Configuration window 6 | |
| | Configuration window 7 | |
| The minimum | System default window | See the table below |

The system default window takes effect only if all eight configuration Windows do not hit a particular address.That is to say, before the configuration window is configured, all read and write requests will be routed according to the "system default window". The "system default window" is explained in the following table:

| The starting address | End address | The target | instructions |
|---|---|---|---|
| 0 xn000_0000_0000 | 0 xnfff_ffff_ffff | HyperTransport 0 | When n! = when NODE_ID |
| 0 x0000_0000_0000 | 0 x0bff_ffff_ffff | Level 2 Cache | Depending on the configuration of scid_sel, you choose which two are mapped to the different four levels<br>In the Cache.See section 2.4 of the user's manual for details. |
| | | | For example,<br>When scid_sel = 0,<br>0x000: route to level 2 Cache 0<br>0x020: route to level 2 Cache 1<br>0x040: route to level 2 Cache 2<br>0x060: route to level 2 Cache 3<br><br>When scid_sel = 2,<br>0x000: route to level 2 Cache 0<br>0x400: route to level 2 Cache 1<br>0x800: route to level 2 Cache 2<br>0xc00: route to level 2 Cache 3 |
| 0 x0c00_0000_0000 | 0 x0dff_ffff_ffff | HyperTransport 0 | |
| 0 x0e00_0000_0000 | 0 x0fff_ffff_ffff | HyperTransport 1 | |

## 14.3 Level 1 cross switch address window configuration timing

The configuration of the address window of the level 1 cross switch is very strict for the window configuration that needs to route to the level 2 Cache request. Before and after the configuration, the data consistency between the level 2 Cache and the level 1 Cache should be guaranteed. In other words, the following situations are not allowed after the configuration:

There is a backup in the secondary Cache before configuration, and a backup in the primary Cache, but after configuration, requests for this backup address are routed to other slave ports.

This will eventually result in a level 1 or 2 Cache data mismatch.Therefore, the best time to configure the various Windows is before the system performs a Cache operation.In other cases if you need to configure a level 1 cross switch you have to make sure that this does not happen.

Note that changing the value of scid_sel after the Cache operation itself can cause this inconsistency, because the address space has changed with the mapped second-level Cache number.

## 14.4 Secondary cross-switch address window

The secondary crosswalk also has default routing, and all addresses that are not hit by any address window are routed to slave port 3, the system configuration register space.

For the primary port of the secondary crosswalker, that is, the primary device end of the outgoing request, it includes the following:

- Primary port 0: level 2 Cache 0-3
- No. 1 primary port: PCI primary port

For the slave port of the secondary cross-switch, that is, the slave device side that issues the request, it includes the following:

- Port 0: memory controller 0
- No. 1 slave port: memory controller 1
- No. 2 slave port: low-speed device interface, including PCI slave port, LPC interface, UART interface, SPI interface and PCI register space
- No. 3 slave port: system configuration register

Compared with the first-level crossover switch, the second-level crossover switch has a weaker limit on the configuration of the address window, and the software is mainly responsible for ensuring that there will be no errors in the access content after the reconfiguration of the address window.

For example, the system address 0x0000_0000_0000 -- 0x0000_0FFF_FFFF was previously mapped to 0x0000_0000_0000 -- 0x0000_0FFF_FFFF on memory controller 0, and some read and write operations were performed using this address to store some valid data.After the address window is configured, the system address 0x0000_2000_0000 is added -- 0x0000_2FFFF_FFFF maps to 0x0000_0000_0000 -- 0x0000_0FFF_FFFF of memory controller 0.An access to 0x0000_2000_0000 at this time will result in the original enabled The value stored at address 0x0000_0000_0000.

During this process, it is important to note that the contents of the second-level Cache are not changed according to the address window configuration, that is, if the original write access to 0x0000_0000_0000 is in Cache mode, it is likely that the access to 0x0000_20000_0000 will get an old value after the address window is updated.

## 14.5 Special handling of address window configuration

Since the core of the loongson 3A1000 processor will be exposed to some guesswork, the guesswork may fall into any address space.However, not all devices are allowed to be accessed by guess, especially for PCI devices, a guess read access is likely to cause the destruction of a "read clear" register data, and it may also cause some illegal access to the address can not return normally, resulting in processor crash.

We prevent this by configuring the first - and second-level cross-switches, disabling some unguessable access to the address space.For example, we prevent guest-access to the PCI space 0x1000_0000 by setting the secondary crossover switch as follows, but allow guest-access to 0x1FC0_0000 at the same time.

| | The BASE | MASK | MMAP |
|---|---|---|---|
| Window 0 | 0 x0000_0000_1000_0000 | 0 xffff_ffff_f000_0000 | 0 x0000_0000_1000_0082 |
| Window 1 | 0 x0000_0000_1fc0_0000 | 0 xffff_ffff_fff0_0000 | 0 x0000_0000_1fc0_00f2 |

For a level 1 crossover switch, the mapping to the level 2 Cache address is also affected by scid_sel, and the two should not conflict.

If you need to map an address space to a second-level Cache, consider the second-level Cache hash.This maps the address space to each level of the Cache.Since each address window can correspond to only one slave port, mapping the second-level Cache requires four address window mappings.

In addition, since the address window has a minimum unit of 1KB, you need to set the value of scid_sel to more than 2 if you want to configure the second-level Cache space at this time, namely, hash the address with more than 10 bits.In the example in the following table, a level 1 cross-switch is configured to map all address accesses issued by the processor core to a level 2 Cache.

Scid_sel = 2

| | The BASE | MASK | MMAP |
|---|---|---|---|
| Window 4 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_00f0 |
| Window 5 | 0 x0000_0000_0000_0400 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_04f1 |
| Window 6 | 0 x0000_0000_0000_0800 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_08f2 |
| Windows 7 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_0cf3 |

From this window we know that any address that has not been hit in window 0-3 will be hit in the four Windows, and according to the

Scid_sel hashes the entire address space into the four correct second-level caches.

## 14.6 HyperTransport address window

The HyperTransport controller can not only send read and write access outward, but also realize DMA access of external devices to the internal memory of the processor. The two access address Spaces are independent of each other, as described below.

### 14.6.1 Processor core external access address window

There are four HyperTransport controllers in the chip, HT0_LO, HT0_HI, HT1_LO and HT1_HI. Among them, HT0_LO shares a physical interface with HT0_HI, and HT1_LO shares a physical interface with HT1_HI. When the chip pin HTx_8x2 is set low, only HTx_LO is visible to users, while the address space of HTx_HI is invalid.According to the default route of the first-level crossover switch, the address space of each controller is as follows:

| Base address | End address | The size of the | define | instructions |
|---|---|---|---|---|
| 0 x0c00_0000_0000 | 0 x0cff_ffff_ffff | One Tbytes | HT0_LO window | |
| 0 x0d00_0000_0000 | 0 x0dff_ffff_ffff | One Tbytes | HT0_HI window | HT0_8x2 =1 |
| 0 x0e00_0000_0000 | 0 x0eff_ffff_ffff | One Tbytes | HT1_LO window | |
| 0 x0f00_0000_0000 | 0 x0fff_ffff_ffff | One Tbytes | HT1_HI window | HT1_8x2 =1 |

Each HyperTransport controller has a 40-bit address space, according to the HyperTransport protocol, which will this

The 40-bit addresses are divided into the following:

| Base address | End address | The size of the | define |
|---|---|---|---|
| 0 x00_0000_0000 | 0 xfc_ffff_ffff | 1012 Gbytes | MEM space |
| 0 xfd_0000_0000 | 0 xfd_f7ff_ffff | 3968 Mbytes | reserve |
| 0 xfd_f800_0000 | 0 xfd_f8ff_ffff | 16 Mbytes | interrupt |
| 0 xfd_f900_0000 | 0 xfd_f90f_ffff | 1 Mbyte | PIC interrupt response |
| 0 xfd_f910_0000 | 0 xfd_f91f_ffff | 1 Mbyte | System information |
| 0 xfd_f920_0000 | 0 xfd_faff_ffff | 30 Mbytes | reserve |
| 0 xfd_fb00_0000 | 0 xfd_fbff_ffff | 16 Mbytes | HT controller configuration space |
| 0 xfd_fc00_0000 | 0 xfd_fdff_ffff | 32 Mbytes | I/O space |
| 0 xfd_fe00_0000 | 0 xfd_ffff_ffff | 32 Mbytes | HT bus configuration space |
| 0 xfe_0000_0000 | 0 xff_ffff_ffff | 8 Gbytes | reserve |

Among them, MEM space, I/O space and HT bus configuration space correspond to the three access forms on the traditional PCI space, namely PCI MEM access, PCI IO access and PCI configuration access. HT controller configuration space mainly provides HT interrupt vector and internal window configuration and other functions.

HT bus configuration space, according to the external device "bus number", "device number", "function number", "register offset" to the following rules for direct read and write access to the address.

Type 0:

| 39 24 | 23 16 | 15 11 | 10 8 | 7 2 |
|---|---|---|---|---|
| FDFEh | Reserved | Device Number | Function Number | Register Number |

Type 1:

| 39 24 | 23 16 | 15 11 | 10 8 | 7 2 |
|---|---|---|---|---|
| FDFFh | Bus Number | Device Number | Function Number | Register Number |

## 14.6.2 External devices access the address window on the processor chip memory DMA

To protect the memory data in the processor chip, the HyperTransport controller provides a set of Windows for DMA access to external devices, namely the "receive address window" in section 9.5.4 of the user manual, and only the DMA places that fall into this set of Windows

The address is actually operated on in the memory space of the chip, otherwise it will give an error reply to the peripheral that initiated the access.

This set of Windows is determined by the following parameters. The window hit rule is as follows:

Hit = ht_rx_image_en &&

(DMA address &ht_rx_image_mask)==(ht_rx_image_base& ht_rx_image_mask)

Address_out = ht_rx_image_trans_en?

Ht_rx_image_trans | DMA address; ht_rx_image_mask: DMA address the priority of different address Windows decreases in turn.

### 14.6.3 Low speed device address window

Low-speed device space includes PCI, LPC, UART, SPI and other devices. The internal address of this space is divided as follows:

| Address name | Address range | The size of the |
|---|---|---|
| LPC Memory | 0 x1dff_ffff x1c00_0000-0 | 32 MByte |
| LPC Boot | 0 x1fcf_ffff x1fc0_0000-0 | 1 MByte |
| PCI I/o space | 0 x1fdf_ffff x1fd0_0000-0 | 1 MByte |
| PCI controller configuration space | 0 x1fe0_00ff x1fe0_0000-0 | 256 Byte |
| IO register space | 0 x1fe0_01df x1fe0_0100-0 | 256 Byte |
| UART 0 | 0 x1fe0_01e7 x1fe0_01e0-0 | 8 Byte |
| UART 1 | 0 x1fe0_01ef x1fe0_01e8-0 | 8 Byte |
| SPI | 0 x1fe0_01ff x1fe0_01f0-0 | 16 Byte |
| LPC Register | 0 x1fe0_02ff x1fe0_0200-0 | 256 Byte |
| PCI configuration space | 0 x1fe8_ffff x1fe8_0000-0 | 64 KByte |
| LPC I/O | 0 x1ff0_ffff x1ff0_0000-0 | 64 Kbyte |
| PCI MEM | other | |

## 14.7 Address space configuration example analysis

The following sections describe the configuration of PMON facing a two-stage crossover switch.In the following example, the HT device is connected using the HT1 interface and the HT0 interface is suspended.

### 14.7.1 A level 1 cross switch example 1

One configuration of a level 1 crossover switch is as follows:

| | The BASE | MASK | MMAP |
|---|---|---|---|
| Window 0 | 0 x0000_0000_1800_0000 | 0 xffff_ffff_fc00_0000 | 0 x0000_0efd_fc00_00f7 |
| Window 1 | 0 x0000_0000_1000_0000 | 0 xffff_ffff_f800_0000 | 0 x0000_0e00_1000_00f7 |
| Window 2 | 0 x0000_0000_1e00_0000 | 0 xffff_ffff_ff00_0000 | 0 x0000_0e00_0000_00f7 |
| Window 3 | | | |
| Window 4 | 0 x0000_0c00_0000_0000 | 0 xffff_fc00_0000_0000 | 0 x0000_0c00_0000_00f7 |
| Window 5 | | | |
| Window 6 | 0 x0000_1000_0000_0000 | 0 x0000_1000_0000_0000 | 0 x0000_1000_0000_00f7 |
| Windows 7 | 0 x0000_2000_0000_0000 | 0 x0000_2000_0000_0000 | 0 x0000_2000_0000_00f7 |

Let's analyze what each configuration window does.

Window 0, converts the address of 0x1800_0000 to 0x0000_0EFD_FC00_0000 and routes to the HT1 controller.In this way, the HT IO space and HT configuration space that originally needed 64-bit address to be accessed can be mapped directly with 32-bit address space, and these Spaces can be accessed with 32-bit address after mapping.

| | Pre-conversion address | Converted address | instructions |
|---|---|---|---|
| Address 0 | 0 x0000_0000_18xx_xxxx | 0 x0000_0efd_fcxx_xxxx | HT I/o space |
| Address 1 | 0 x0000_0000_19xx_xxxx | 0 x0000_0efd_fdxx_xxxx | |
| Address 2 | 0 x0000_0000_1axx_xxxx | 0 x0000_0efd_fexx_xxxx | HT configuration space: Type 0 |
| Address 3 | 0 x0000_0000_1bxx_xxxx | 0 x0000_0efd_ffxx_xxxx | HT configuration space: Type 1 |

Window 1, converts the address of 0x1000_0000 to 0x0000_0E00_1000_0000 and routes to the HT1 controller.This actually maps parts of the HT MEM space that previously needed to be accessed using 64-bit addresses directly using 32-bit address Spaces, which can then be accessed using 32-bit addresses.Although the entire HT MEM space is not mapped, up to 128MB of HT MEM space can already be used here.

| | Pre-conversion address | Converted address | instructions |
|---|---|---|---|
| Address | 0 x0000_0000_1xxx_xxxx | 0 x0000_0e00_1xxx_xxxx | HT MEM space |

| | | | |
|---|---|---|---|
| 0 | | | |

Window 2, converts the address 0x1E00_0000 to 0x0000_0E00_0000_0000 and routes to the HT1 controller.In this way, the minimum 16MB address of HT MEM space that previously required 64-bit address access is mapped directly using 32-bit address space, and these Spaces can be accessed using 32-bit address after mapping.the

Therefore, this part of the address should be mapped, because some traditional devices need to use this reserved space for fixed decoding, such as video card devices and so on.

| | Pre-conversion address | Converted address | instructions |
|---|---|---|---|
| Address 0 | 0 x0000_0000_1exx_xxxx | 0 x0000_0e00_1exx_xxxx | HT MEM space is 16MB low |

The setting of the previous Windows is to directly use 32-bit address in PMON, and to achieve the access of HT space and HT device without TLB conversion, so as to facilitate the compatible design of the software version. In Linux system, since 64-bit address can be directly used for access, it is not required to convert these addresses.However, it is important to note that, based on how Linux handles HT MEM Spaces, the transformation of HT MEM Spaces is still required to simplify access to 32-bit addressing peripherals.

The remaining Windows are used to route all addresses of unresponsive devices to the HT1 controller, which responds.These addresses do not appear spontaneously during normal program execution, but due to the processor core's guess execution, any access to the address may occur, which may cause the processor to crash if the correct response is not obtained.The HT controller in loongson 3A1000 can correctly identify and handle such access.So you need to route all of these potential guess accesses to the HT controller.

Addresses other than these exceptions are routed to the secondary Cache using the default routing method, which is then forwarded to the secondary cross-switch.

### 14.7.2 A level 1 cross switch example 2

Another configuration of the level 1 crossover switch is as follows:

Scid_sel = 2

| | The BASE | MASK | MMAP |
|---|---|---|---|
| Window 0 | 0 x0000_0000_1800_0000 | 0 xffff_ffff_fc00_0000 | 0 x0000_0efd_fc00_00f7 |
| Window 1 | 0 x0000_0000_1000_0000 | 0 xffff_ffff_f800_0000 | 0 x0000_0e00_1000_00f7 |
| Window 2 | 0 x0000_0000_1e00_0000 | 0 xffff_ffff_ff00_0000 | 0 x0000_0e00_0000_00f7 |
| Window 3 | 0 x0000_0e00_0000_0000 | 0 xffff_fe00_0000_0000 | 0 x0000_0e00_0000_00f7 |
| Window 4 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_00f0 |
| Window 5 | 0 x0000_0000_0000_0400 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_04f1 |
| Window 6 | 0 x0000_0000_0000_0800 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_08f2 |
| Windows 7 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_0c00 | 0 x0000_0000_0000_0cf3 |

In this configuration, the first three Windows are the same as the previous configuration, which I won't cover here.

Window 3 routes all 0x0000_0E00_0000_0000 addresses to the HT1 controller, which would be the default route. This is configured here because in Windows 4-7, all addresses are routed to four different level 2 caches, so the default route is no longer valid.

The configuration of Windows 4-7 is explained in section 1.5, the most important of which is that the routing to the secondary caches should be consistent with the configuration of scid_sel.The purpose of this configuration is the same as the first configuration, to prevent some address that is not responding to the device from causing the processor to crash.

### 14.7.3 Example 1 of a two-level cross switch

One configuration of the secondary crosswise switch is as follows. Only one memory controller is used in this configuration. Use on memory

256 MB of space.

| | The BASE | MASK | MMAP |
|---|---|---|---|
| Window 0 | 0 x0000_0000_1000_0000 | 0 xffff_ffff_f000_0000 | 0 x0000_0000_1000_0082 |
| Window 1 | 0 x0000_0000_1fc0_0000 | 0 xffff_ffff_fff0_0000 | 0 x0000_0000_1fc0_00f2 |
| Window 2 | 0 x0000_0000_0000_0000 | 0 xffff_ffff_f000_0000 | 0 x0000_0000_0000_00f0 |
| Window 3 | | | |
| Window 4 | | | |
| Window 5 | | | |
| Window 6 | | | |
| Windows 7 | | | |

Window 0 opens uncache and non-fetchable access to the low-speed device space, so that you can ensure that any access that falls into this window is the access your application needs to make.

Window 1 opens all types of access to the BOOT space in the low-speed device space, that is, normal access, including cache access and picker access.

Window 2 opens the lower 256MB space on memory controller 0, allowing all types of access.All other accesses are routed to the system configuration register space as the default route.

Combined with the configuration of a level 1 cross switch in 1.8.1, the full chip address space is obtained as follows:

| | The starting address | End address | instructions |
|---|---|---|---|
| Address 0 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0fff_ffff | Memory controller 0 |
| Address 1 | 0 x0000_0000_1000_0000 | 0 x0000_0000_17ff_ffff | HT1 MEM space |
| Address 2 | 0 x0000_0000_1800_0000 | 0 x0000_0000_19ff_ffff | HT1 IO space |
| Address 3 | 0 x0000_0000_1a00_0000 | 0 x0000_0000_1bff_ffff | HT1 configuration space |
| Address 4 | 0 x0000_0000_1c00_0000 | 0 x0000_0000_1dff_ffff | LPC Memory |
| Address five | 0 x0000_0000_1fc0_0000 | 0 x0000_0000_1fcf_ffff | LPC Boot |

| | | | |
|---|---|---|---|
| Address 6 | 0 x0000_0000_1fd0_0000 | 0 x0000_0000_1fdf_ffff | PCI I/o space |
| Address 7 | 0 x0000_0000_1fe0_0000 | 0 x0000_0000_1fe0_00ff | PCI controller configuration space |
| Address 8 | 0 x0000_0000_1fe0_0100 | 0 x0000_0000_1fe0_01df | IO register space |
| Address 9 | 0 x0000_0000_1fe0_01e0 | 0 x0000_0000_1fe0_01e7 | UART 0 |
| Address 10 | 0 x0000_0000_1fe0_01e8 | 0 x0000_0000_1fe0_01ef | UART 1 |
| Address 11 | 0 x0000_0000_1fe0_01f0 | 0 x0000_0000_1fe0_01ff | SPI |
| Address 12 | 0 x0000_0000_1fe0_0200 | 0 x0000_0000_1fe0_02ff | LPC Register |
| Address 13 | 0 x0000_0000_1fe8_0000 | 0 x0000_0000_1fe8_ffff | PCI configuration space |
| Address 14 | 0 x0000_0000_1ff0_0000 | 0 x0000_0000_1ff0_ffff | LPC I/O |
| Address 15 | 0 x0000_0c00_0000_0000 | 0 x0000_0fff_ffff_ffff | HT1 controller, various Spaces |
| Address 16 | 0 x0000_1000_0000_0000 | 0 x0000_3fff_ffff_ffff | HT1 controller, guess space |
| Address the 17th | Other address | | System configuration space |

### 14.7.4 Example 2 of a secondary cross switch

Another configuration of the secondary crosswise switch is as follows. Two memory controllers are used in this configuration. Each memory controller USES 1GB of memory space

| | The BASE | MASK | MMAP |
|---|---|---|---|
| Window 0 | 0 x0000_0000_1000_0000 | 0 xffff_ffff_f000_0000 | 0 x0000_0000_1000_0082 |
| Window 1 | 0 x0000_0000_1fc0_0000 | 0 xffff_ffff_fff0_0000 | 0 x0000_0000_1fc0_00f2 |
| Window 2 | 0 x0000_0000_0000_0000 | 0 xffff_ffff_f000_0000 | 0 x0000_0000_0000_00f0 |
| Window 3 | | | |
| Window 4 | 0 x0000_0000_8000_0000 | 0 xffff_ffff_c000_0000 | 0 x0000_0000_0000_00f0 |

Window 0 opens uncache and non-fetchable access to the low-speed device space, so that you can ensure that any access that falls into this window is the access your application needs to make.

Window 1 opens all types of access to the BOOT space in the low-speed device space, that is, normal access, including cache access and picker access.

Window 2 opens the lower 256MB space on memory controller 0, allowing all types of access.

4 opened the window all the 1 gb of space on the memory controller 0, 0 x8000_0000-0 xbfff_ffff address system used for access, it is important to note that the system zero

x8000_0000-0 x8fff_ffff x0000_0000 space and 0-0 x0fff_ffff space collocated, in order to guarantee the correctness of the data, the system software must make the pledge that we shall use only one address to access this for Linux, for example,0x0000_0000 on the system must be used

The address of 0x0FFF_FFFF, then, for this space, 0x8000_0000-8fff_ffff access is Prohibited.

Window 6 opens all 1GB of space on memory controller 1, which is accessed by the system using 0xC0000_0000 -- 0xFFFF_FFFF.

Combined with the configuration of the one-level cross-switch in 1.8.2, the full chip address space distribution is obtained as follows:

| | The starting address | End address | instructions |
|---|---|---|---|
| Address 0 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0fff_ffff | Memory controller 0 |
| Address 1 | 0 x0000_0000_1000_0000 | 0 x0000_0000_17ff_ffff | HT1 MEM space |
| Address 2 | 0 x0000_0000_1800_0000 | 0 x0000_0000_19ff_ffff | HT1 IO space |
| Address 3 | 0 x0000_0000_1a00_0000 | 0 x0000_0000_1bff_ffff | HT1 configuration space |
| Address 4 | 0 x0000_0000_1c00_0000 | 0 x0000_0000_1dff_ffff | LPC Memory |
| Address five | 0 x0000_0000_1fc0_0000 | 0 x0000_0000_1fcf_ffff | LPC Boot |
| Address 6 | 0 x0000_0000_1fd0_0000 | 0 x0000_0000_1fdf_ffff | PCI I/o space |
| Address 7 | 0 x0000_0000_1fe0_0000 | 0 x0000_0000_1fe0_00ff | PCI controller configuration space |
| Address 8 | 0 x0000_0000_1fe0_0100 | 0 x0000_0000_1fe0_01df | IO register space |
| Address 9 | 0 x0000_0000_1fe0_01e0 | 0 x0000_0000_1fe0_01e7 | UART 0 |
| Address 10 | 0 x0000_0000_1fe0_01e8 | 0 x0000_0000_1fe0_01ef | UART 1 |
| Address 11 | 0 x0000_0000_1fe0_01f0 | 0 x0000_0000_1fe0_01ff | SPI |
| Address 12 | 0 x0000_0000_1fe0_0200 | 0 x0000_0000_1fe0_02ff | LPC Register |
| Address 13 | 0 x0000_0000_1fe8_0000 | 0 x0000_0000_1fe8_ffff | PCI configuration space |
| Address 14 | 0 x0000_0000_1ff0_0000 | 0 x0000_0000_1ff0_ffff | LPC I/O |
| Address 15 | 0 x0000_0000_8000_0000 | 0 x0000_0000_bfff_ffff | Memory controller 0 |
| Address 16 | 0 x0000_0000_c000_0000 | 0 x0000_0000_ffff_ffff | Memory controller 1 |
| Address the 17th | 0 x0000_0e00_0000_0000 | 0 x0000_0fff_ffff_ffff | HT1 controller, various Spaces |
| Address 18 | Other address | | System configuration space |

# 15 System memory space distribution design

## 15. 1 System memory space

Within the godson 3 a1000 processor has two memory controller, if able to address space distribution on the two memory controller, the system average access delay and average access bandwidth will bring benefit, but is limited by cross number configuration window switch on, certain rules must be taken to address the space distribution methods to make certain designs.

Based on the above considerations, the following rules can be used to design the memory address space in order to maintain the different memory space sizes of the Linux system and to ensure that the memory space distribution is simple and beautiful.Of course, you can also customize the memory space distribution rules according to the needs of the system designer.

（1） Regardless of memory size, 0x0000_0000-0x0FFF_FFFF must be kept at the lower 256MB Space;

（2） To set aside the necessary direct access address space for the IO device, 0x1000_0000 -- 0x1FFF_FFFF Reserve the address space not used as a space;

（3） Therefore, the rest of 1GB and above memory space is defined according to the following formula:

Base = Size + 0x1000_0000

Limit = Size + size-1

Where, Base and Limit are the Base address and high address of this space respectively, and Size is the Size of all memory.

For example, if the memory size is 1GB, the address space in the system is stored in the following table:

| | The starting address | End address | instructions |
|---|---|---|---|
| Address 0 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0fff_ffff | 0-256 MB |
| Address 1 | 0 x0000_0000_5000_0000 | 0 x0000_0000_7fff_ffff | 256 MB, 1 gb |

If the memory size is 2GB, the address space stored in the system is shown in the following table:

| | The starting address | End address | instructions |
|---|---|---|---|
| Address 0 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0fff_ffff | 0-256 MB |
| Address 1 | 0 x0000_0000_9000_0000 | 0 x0000_0000_ffff_ffff | 256 MB to 2 gb |

If the memory size is 4GB, the address space stored in the system is shown in the following table:

| | The starting address | End address | instructions |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Address 0 | 0 x0000_0000_0000_0000 | 0 x0000_0000_0fff_ffff | 0-256 MB |
| Address 1 | 0 x0000_0001_1000_0000 | 0 x0000_0001_ffff_ffff | 256 MB - 4 gb |

And so on.

In addition, to make the two memory controllers interleavable, we configure the memory address space on the secondary crosswise switch as follows.

| instructions | | window | | |
|---|---|---|---|---|
| Used to enable access to the BIOS space | | 0 | The BASE | 0 x00000000_1fc00000 |
| | | | MASK | 0 xffffffff_fff00000 |
| | | | MMAP | 0 x00000000_1fc000f2 |
| Used to enable access to a PCI space (only non-referential UNCACHE access is allowed) | | 1 | The BASE | 0 x00000000_10000000 |
| | | | MASK | 0 xffffffff_f0000000 |
| | | | MMAP | 0 x00000000_10000082 |
| Used to enable access to the lower 256M space of the chip | MC0 single channel 256 MB and above | 2 | The BASE | 0 x00000000_00000000 |
| | | | MASK | 0 xffffffff_f0000000 |
| | | | MMAP | 0 x00000000_000000f0 |
| | | 3 | | |
| | Dual channel 256MB x 2 and above (interleaved with address [10]) | 2 | The BASE | 0 x00000000_00000000 |
| | | | MASK | 0 xffffffff_f0000400 |
| | | | MMAP | 0 x00000000_000000f0 |
| | | 3 | The BASE | 0 x00000000_00000400 |
| | | | MASK | 0 xffffffff_f0000400 |
| | | | MMAP | 0 x00000000_000000f1 |
| Used to enable access to the memory high address space | MC0 single channel 256M | 4 | | |
| | | 5 | | |
| | | 6 | | |
| | | 7 | | |
| | MC0 single channel 512M | 4 | The BASE | 0 x00000000_20000000 |
| | | | MASK | 0 xffffffff_f0000000 |
| | | | MMAP | 0 x00000000_100000f0 |
| | | 5 | | |
| | | 6 | | |
| | | 7 | | |
| | MC0 single channel 1G | 4 | The BASE | 0 x00000000_40000000 |
| | | | MASK | 0 xffffffff_c0000000 |
| | | | MMAP | 0 x00000000_000000f0 |
| | | 5 | | |
| | | 6 | | |
| | | 7 | | |

| | | | | |
|---|---|---|---|---|
| | MC0 single channel 2G | 4 | The BASE | 0 x00000000_80000000 |
| | | | MASK | 0 xffffffff_80000000 |
| | | | MMAP | 0 x00000000_000000f0 |
| | | 5 | | |
| | | 6 | | |
| | | 7 | | |
| | Double channel 256M x 2<br>(use address [10] interleaving) | 4 | The BASE | 0 x00000000_20000000 |
| | | | MASK | 0 xffffffff_f0000400 |
| | | | MMAP | 0 x00000000_000004f0 |
| | | 5 | The BASE | 0 x00000000_20000400 |
| | | | MASK | 0 xffffffff_f0000400 |
| | | | MMAP | 0 x00000000_000004f1 |
| | | 6 | | |
| | | 7 | | |
| | Dual channel 512M x 2<br>(use address [10] interleaving) | 4 | The BASE | 0 x00000000_40000000 |
| | | | MASK | 0 xffffffff_e0000400 |
| | | | MMAP | 0 x00000000_000000f0 |
| | | 5 | The BASE | 0 x00000000_40000400 |
| | | | MASK | 0 xffffffff_e0000400 |
| | | | MMAP | 0 x00000000_000000f1 |
| | | 6 | The BASE | 0 x00000000_60000000 |
| | | | MASK | 0 xffffffff_e0000400 |
| | | | MMAP | 0 x00000000_000004f0 |
| | | 7 | The BASE | 0 x00000000_60000400 |
| | | | MASK | 0 xffffffff_e0000400 |
| | | | MMAP | 0 x00000000_000004f1 |
| | Dual channel 1G x 2<br>(use address [10] interleaving) | 4 | The BASE | 0 x00000000_80000000 |
| | | | MASK | 0 xffffffff_c0000400 |
| | | | MMAP | 0 x00000000_000000f0 |
| | | 5 | The BASE | 0 x00000000_80000400 |
| | | | MASK | 0 xffffffff_c0000400 |
| | | | MMAP | 0 x00000000_000000f1 |
| | | 6 | The BASE | 0 x00000000_c0000000 |
| | | | MASK | 0 xffffffff_c0000400 |
| | | | MMAP | 0 x00000000_000004f0 |
| | | 7 | The BASE | 0 x00000000_c0000400 |
| | | | MASK | 0 xffffffff_c0000400 |
| | | | MMAP | 0 x00000000_000004f1 |
| | Dual channel 2G x 2<br>(use address [10] interleaving) | 4 | The BASE | 0 x00000001_00000000 |
| | | | MASK | 0 xffffffff_80000400 |

| | | | | |
|---|---|---|---|---|
| | | | MMAP | 0 x00000000_000000f0 |
| | | 5 | The BASE | 0 x00000001_00000400 |
| | | | MASK | 0 xffffffff_80000400 |
| | | | MMAP | 0 x00000000_000000f1 |
| | | 6 | The BASE | 0 x00000001_80000000 |
| | | | MASK | 0 xffffffff_80000400 |
| | | | MMAP | 0 x00000000_000004f0 |
| | | 7 | The BASE | 0 x00000001_80000400 |
| | | | MASK | 0 xffffffff_80000400 |
| | | | MMAP | 0 x00000000_000004f1 |

## 15. 2 The mapping relationship between the system memory space and the peripheral DMA space

With this configuration in place, let's move on to the use of memory addresses for DMA.The traditional PCI DMA space exists at the address above 0x8000_0000. When the device conducts DMA operations, the access of 0x8000_0000 is mapped to the space of 0x0000_0000, which is then mapped to system memory one by one.

In loongson 3A1000, in order to solve the problem of using DMA space conversion for large memory, the following provisions are made.

（1） When the system memory space 0x0000_0000 -- 0x0FFF_FFFF is used as a DMA space, the peripheral USES 0x8000_0000 -- 0x8FFF_FFFF for access;

（2） Other system memory Spaces can be used as DMA Spaces without the need for address translation.Therefore, taking 2GB of memory space as an example, the following address translation table can be obtained:

| | instructions | | The starting address | End address |
|---|---|---|---|---|
| Address 0 | 0-256 MB | System space | 0 x0000_0000_0000_0000 | 0 x0000_0000_0fff_ffff |
| | | The DMA space | 0 x0000_0000_8000_0000 | 0 x0000_0000_8fff_ffff |
| Address 1 | 256 MB - 2 gb | System space | 0 x0000_0000_9000_0000 | 0 x0000_0000_ffff_ffff |
| | | The DMA space | 0 x0000_0000_9000_0000 | 0 x0000_0000_ffff_ffff |

When using the HyperTransport interface, the above address translation method can be

configured using the HyperTransport "receive address window" configuration.See section 1.6.2.Use two sets of address Windows to accomplish this conversion.

| | instructions | Window enable register | Window base register |
|---|---|---|---|
| Window 0 | 0-256 MB | 0 xc000_0000 | 0 x0080_fff0 |
| Window 1 | 256 MB to 2 gb | 0 xc000_0080 | 0 x0080_ff80 |

Window 0 converts the address 0x8000_0000 -- 0x8FFF_FFFF to 0x0000_0000 -- 0x0FFF_FFFF access.

Window 1 converts the address of 0x8000_0000 -- 0xFFFF_FFFF to 0x8000_0000 -- 0xFFFF_FFFF

Access. From the priority rule hit by the window, you can tell the actual address of 0x8000_0000 -- 0x8FFF_FFFF

On is only mapped by window 0, so the address that window 1 handles is actually 0x9000_0000 -- 0xFFFF_FFFF.

## 15.3 Other mapping methods for system memory space

The system memory space mapping method described in the previous two sections is only an effective reference method, and the system designer can also redefine the memory mapping rules according to his own needs.

For example, the memory space is all concentrated in the low address, while the IO address and system configuration space are mapped to the higher space.

# 16 Memory allocation for system X

X system memory allocation problem, describes the video card memory allocation problem.In the 3a-690e, the graphics card is integrated inside the northbridge 690e and is a PCIE device.The display core of the card is the ATI X1250, which integrates 128M of video memory internally, and also supports the way of sharing video memory, which can be up to 128M.The display process of the graphics card is as follows: the CPU transmits the instructions and data related to drawing to the graphics card through the PCIE bus.The GPU then completes the image processing process according to the requirements of the CPU and saves the final image data in the video memory.

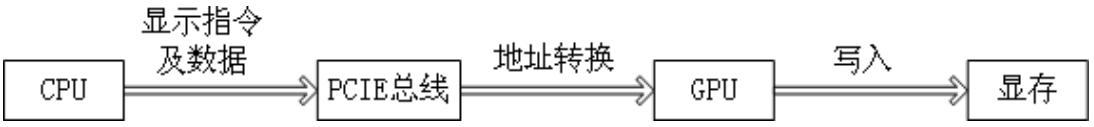



Figure 16-1 graphics card processing image display process

In the case of using separate video memory, the process is simpler because the video memory is inside the video card, which can directly write content to the video memory.For Shared video memory, the process is a little more complicated. After the GPU tells the CPU the video memory address to be written, there will be two situations: first, the video memory address is a PCI spatial address. The CPU will write the content directly to the PCIE bus, and the PCIE will do the address translation again, which will be converted to the actual video memory address, that is, the physical address. Two, the video memory address is the memory address. At this point, the CPU will write the desired content directly to the video memory location.Obviously, the second option is more efficient for the Shared memory approach.

Let's talk more about how the second way of sharing video memory is implemented in PMON Chinese.To extend our PCI

Space, the TLB mapping we used.In PMON's bonito.h, we define it this way:

```
# define BONITO_PCILO_BASE0x10000000
# define BONITO_PCILO_BASE_VA0xd0000000
# define BONITO_PCIIO_BASE0x18000000
# define BONITO_PCIIO_BASE_VA0xb8000000
```

It means to divide the 256M PCI space (0x10000000~0x20000000) into two parts: 0x10000000~0x17ffffff for mem space and 0x18000000~0x20000000 for IO space.The virtual

address 0xd0000000 to the physical address 0x10000000 of the mem space is transformed by manually populating TLB.In 3a-690e, if the memory is 2G, the PCI address of the video memory is actually 0x10000000, and the corresponding memory address is

0xf8000000, if memory is 1G, the PCI address of video memory is actually 0x10000000, the corresponding memory address is

0x78000000, which is also implemented by TLB mapping. The code for TLB mapping is as follows:

```
Lit0, 15
Lit3,0xf0000000# entry_hi, the starting address for the virtual address that
needs to be mapped, lia0, 0x3f000000
Bleumsize, a0, 1f// to determine whether the memory is 1gb or 2gb, if it is 1gb, skip to 1 to execute

The nop
Lit4, 0x0000f000//2G case, 0xf0000000 to 0xf0000000 b2f/ / jump to 2

The nop
1:
Lit4, 0x00007000//1G case, 0xf0000000 to 0x70000000
2:
. Set mips64
Dsllt4, t4, 10
. Set mips3
ori        T4, t4, 0 x1f
li         T5, (0 x1000000 > > 6)          # 16M stride, 16M for one
                                           page
li         T6, 0 x2000000                  # 32 m VPN2 stride
. Set mips64
1:
Dmtc0t3, COP_0_TLB_HI// fill in the TLB table entry
Daddut3, t3, t6
Dmtc0t4, COP_0_TLB_LO0
daddut4, t4, t5
Dmtc0t4, COP_0_TLB_LO1
daddut4, t4, t5
. Set mips3
Addiut1 t0, 16
Mtc0t1, COP_0_TLB_INDEX# 16MB page nop
The nop
```

The

nop

nop

nop

tlbw

i

Bnez t0, 1b// total mapping size 16*16=256M addiu t0, t0, -

1

In this way, when the video card accesses the memory, we can use the address 0xf8000000 as the starting address of the memory, so that it actually USES the high end of the memory.This is set in the structure body of ati_nb_cfg in rs690_struct.c. Set system_memory_tom_lo to 0x1000M, which is 0x100000000. If the video memory is 128M, then the starting address is 0x1000m-128m =0xf8000000, which is the virtual address of the video memory mentioned above.So far, the video card directly access video memory is achieved.