Godson 3A1000 Processor User Manual

volume One

Multi-core processor architecture, register description and system software programming guide

V1.15

2015 Nian ***09*** Yue

Loongson Zhongke Technology Co., Ltd.

Copyright Notice

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 notice. Ruin
The company does not assume any responsibility for direct or indirect losses caused by improper use of documents.

Loongson Zhongke Technology Co., Ltd.

Loongson Technology Corporation Limited
Address: Building 2, Longxin Industrial Park, Zhongguancun Environmental Protection Technology Demonstration Park, Haidian District, Beijing
Building No. 2, Loongson Industrial Park,
Zhongguancun Environmental Protection Park, Haidian District, Beijing
Telephone (Tel): 010-62546668
Fax: 010-62600826

Reading guide

"Godson 3A1000 Processor User Manual" is divided into the first and second volumes.

"Loongson 3A1000 Processor User Manual" is divided into two parts, the first part (Chapter 1 ~ Chapter 10) introduces Loongson

3A1000 multi-core processor architecture and register description, on chip system architecture, function and configuration of main modules, registers

Lists and bit fields are explained in detail; the second part (Chapter 11 ~ Chapter 16) is the system software programming guide

Special presentations on common problems in the operating system development process.

The second volume of the "Loongson 3A1000 Processor User Manual" introduces in detail the adoption of Loongson 3A1000 from the perspective of system software developers

GS464 high-performance processor core.

Page 4

revise history

Document update record

Document name:	Godson 3A1000 Processor User Manual --volume One
version number	V1.15
founder:	R & D Center
Creation Date:	2015-09-11

Update history

Serial number	Updated	version number	update content
1	2009-10-30	V1.0	Increase the definition of DDR related parameters; modify the base address of UART and SPI
2	2009-11-13	V1.1	Add definition of PCI_CONFIG to configuration pins
3	2010-06-25	V1.2	Add the second part of the manual, including the configuration and use of interrupts, serial port configuration and Use, EJTAG debugging instructions, address window configuration conversion, system memory space Distributed design and memory allocation of X system
4	2010-06-29	V1.3	Revised Chapter 1 Overview, Chapter 2 Address Distribution
5	2010-07-20	V1.4	Corrected some text errors in the HT configuration register
6	2010-07-28	V1.5	Added Chip Config and Chip Sample register definitions in Section 10.5
7	2010-12-17	V1.6	Revise the definition of DDR related parameters

8	2011-11-24	V1.7	Edit cover
9	2012-02-14	V1.8	Increase CLKSEL setting limit
10	2012-02-23	V1.9	Add DDR configuration register interrupt vector description
11	2012-04-25	V1.10	Add detailed description of chip configuration register Add detailed description of HT diagnostic register
12	2012-08-23	V1.11	revised DDR parameter definition Add the definition of HT register 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, Loongson 3A processor was renamed Loongson 3A1000 Organizer
15	2014-07-24	V1.14	Add industrial-grade chip content
16	2015-09-11	V1.15	Revise the description of GPIO configuration register

Godson **3A1000** processor user manual directory

Manual feedback: service@loongson.cn

Page 6

table of Contents

Page 7

II

Page 8

Godson **3A1000** processor user manual directory

Figure catalog

Table directory

Godson **3A1000** processor user manual table entry

first part

Multi-core processor architecture, register description

1

Page 14

Godson **3A1000** Processor User Manual Part 1

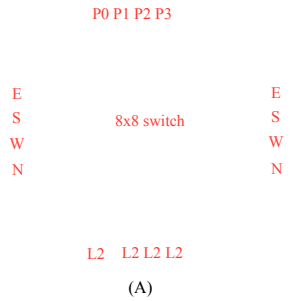
1 Overview

1.1 Introduction to Loongson series processors

Loongson processor mainly includes three series. Loongson No. 1 processor and its IP series are mainly for embedded applications. Core 2 superscalar processor and its IP series are mainly for desktop applications, and Godson 3 multi-core processor series is mainly for service Server and high-performance machine applications. According to the needs of the application, some of Loongson 2 can also face some high-end embedded Yes, some low-end Loongson 3 can also be used for some desktop applications. The above three series will be developed in parallel.

Loongson No. 3 multi-core series processor is based on a scalable multi-core interconnect architecture design, integrating multiple high-end on a single chip Performance processor core and a large number of level 2 caches, and also realize the interconnection of multiple chips through high-speed I / O interface to form a larger Modular system.

The scalable interconnection structure adopted by Loongson 3 is as follows Picture 1-1 As shown. Both the on-chip and multi-chip systems of Godson No. 3 adopt two Dimension mesh interconnection structure, where each node is composed of 8 * 8 crossbars, each crossbar is connected to four processor cores And four secondary caches, and interconnect with other nodes in the four directions of east (E) south (N) west (W) north (N). therefore, 2 * 2 meshes can be connected to 16 processor cores, and 4 * 4 meshes can be connected to 64 processor cores.



Loongson No. 3 node and two-dimensional interconnection structure, (a) node structure, (b) 2 * 2 mesh network connected to 16 processors, (c) The 4 * 4 mesh network connects 64 processors.

Figure 1-1 Loongson No. 3 system structure

The structure of Loongson No. 3 node is shown in Figure 1-2 below. Each node has two levels of AXI crossbars connected to the processor and two levels Cache, memory controller and IO controller. Among them, the first level AXI crossbar switch (called X1 Switch, referred to as X1) Connect the processor and secondary cache. The second-level crossbar switch (called X2 Switch, referred to as X2 for short) connects the second-level cache and Memory controller.

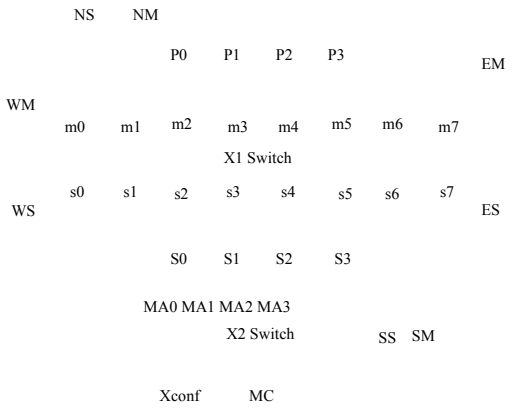


Figure 1-2 Loongson No. 3 node structure

In each node, up to 8 * 8 X1 crossbars are connected to four GS464 processor cores through four Master ports (P0, P1, P2, P3 in the figure), connect four interleave secondary caches that are addressed by four slave ports Block (S0, S1, S2, S3 in the figure), connected to the four directions of east, south, west and north through four pairs of Master / Slave Other nodes or IO nodes (EM / ES, SM / SS, WM / WS, NM / NS in the figure).

The X2 crossbar is connected to four secondary caches through four Master ports, and one is connected to at least one Slave port Memory controller, at least one Slave port connected to a crossbar configuration module (Xconf) is used to configure this node The X1 and X2 address windows, etc. You can also connect more memory controllers and IO ports as needed.

The interconnection system of Loongson 3 only defines the upper layer protocol, and will not make specific provisions on the implementation of the transmission protocol, Therefore, the interconnection between the nodes can be implemented using an on-chip network, or multiple chips can be implemented through the I / O control link Interconnection. In a 4-node 16-core system as an example, it can be composed of 4 4-core chips or 2 8 Core chip, or based on a single chip 4 node 16 core chip. Since 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 Godson 3A1000

Loongson 3A1000 is the first product in Loongson No. 3 multi-core processor series. It is a single-node 4-core configuration.

The processor is manufactured with 65nm process and the highest working frequency is 1GHz. The main technical characteristics are as follows:

- Four 64-bit super-scalar GS464 high-performance processor cores are integrated on-chip;
- On-chip integrated 4 MB split shared secondary cache (composed of 4 individual modules, each with a capacity of 1MB);
- Maintain the cache consistency of multi-core and I / O DMA access through the directory protocol;

- Two 64-bit 400MHz DDR2 / 3 controllers are integrated on-chip;
- Two 16-bit 800MHz HyperTransport controllers are integrated on-chip;
- Each 16-bit HT port is split into two 8-way HT ports for use.
- On-chip integrated 32-bit 100MHz PCIX / 66MHz PCI;
- Integrate 1 LPC, 2 UART, 1 SPI, 16 GPIO interfaces on-chip;

The overall architecture of Loongson 3A1000 chip is based on two-level interconnection. The structure is shown in Figure 1-3 below.

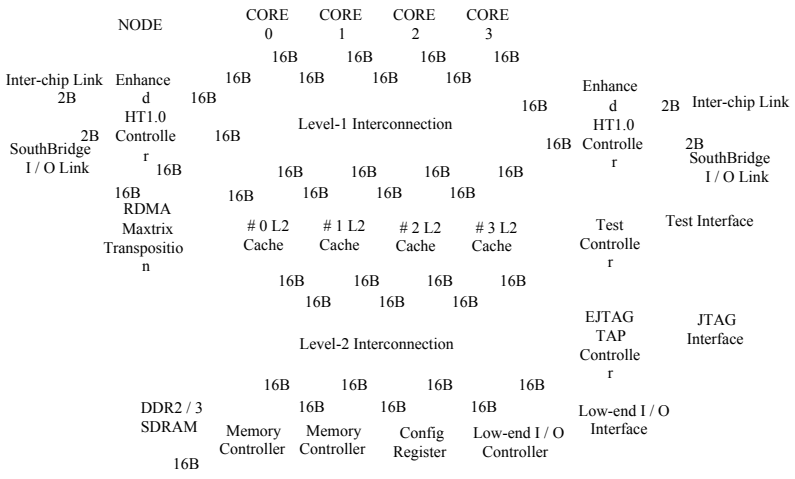


Figure 1-3 Godson 3A1000 chip structure

The first level interconnection uses a 6x6 crossbar switch, which is used to connect four CPUs (as the main device) and four second level caches Module (as a slave), and two IO ports (each port uses a Master and a Slave). First class Each IO port connected to the 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 are used. The HT controller is connected to the primary interconnection switch via a DMA controller. The DMA controller Responsible for DMA DMA control and responsible for maintaining consistency between slices. The DMA controller of Godson 3 can also be realized through configuration Prefetch and matrix transposition or relocation.

The second level interconnection uses a 5x4 crossbar switch, connecting 4 second level cache modules (as the main device), two DDR2 Memory controller, low-speed high-speed I / O (including PCI, LPC, SPI, etc.) and the control register module inside the chip.

The above two-level interconnect switches all use separate data channels for reading and writing. The width of the data channel is 128 bits. The processor core has the same frequency to provide high-speed on-chip data transmission.

Based on Loongson No. 3 scalable interconnection architecture, 4 quad-core Loongson 3A1000 can be connected through HT port to form 4 chips 16-core SMP structure.

1.3 Description of Loongson 3A1000 Commercial and Industrial Chips

Loongson 3A1000 chips are available in both industrial and commercial grades. Their main features are as follows:

Configuration	Commercial grade	Industrial grade
Operating temperature	0 ℃ ～ 70 ℃	-40 ℃ ～ 85 ℃
Whether to filter	—	√
Whether the quality consistency test	—	√
Quality consistency test standard	—	GB 4937-1995

The Loongson 3A chip, like most semiconductor devices, has a failure rate that conforms to the bathtub curve model. Loongson 3A industrial grade chip In order to ensure longer-term, stable, and reliable operation, and to be able to adapt to more demanding environmental temperature requirements, the chip Reliability screening was conducted to eliminate early failure chips. This reliability screening is a 100% test, passed the screening To meet the requirements of industrial grade chips

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

Filter items	Methods and conditions (Summary)	Claim
1. Visual inspection	The logo is clear, no contamination, no solder ball oxidation, and the chip is intact	100%
2. Stability baking	125 ℃, 24h	100%
3. Rapid temperature changes	10 cycles at maximum and minimum storage temperature	100%
4. Serial number		100%
5. Intermediate (before aging) electrical testing		100%
6, veteran	TC = 85 ℃, 160h	100%
7. Intermediate (after aging) electrical test at room temperature		100%
8. Permitted non-conforming product rate (PDA) Calculation	PDA≤5%, normal temperature, when 5% ＜ PDA≤10%, it can be Newly submitted and refined, but only allowed once	All batches
9. End point electrical test	Three temperature, record all test data	100%
10. External visual inspection	The logo is clear, no contamination, no solder ball oxidation, and the chip is intact	100%

5

2 System configuration and control

2.1 Chip working mode

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

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

● Multi-chip interconnect mode. The system contains 2 pieces or 4 pieces Godson 3A1000, through the HT end of Godson 3A1000

It is a non-uniform memory access multiprocessor system (CC-NUMA).

2.2 Description of control pins

The control pins of Loongson 3A1000 include DO_TEST, ICCC_EN, NODE_ID [1: 0], CLKSEL [15: 0], PCI_CONFIG.

Table 2-1 Control pin description

signal	Up and down	effect
DO_TEST	pull up	1'b1 means function mode 1'b0 means test mode
ICCC_EN	drop down	1'b1 means multi-chip consistent interconnect mode 1'b0 means single chip mode
NODE_ID [1: 0]		Indicates the processor number in multi-chip consistent interconnect mode
CLKSEL [15: 0]		Power-on clock control

HT clock control

signal	effect
CLKSEL [15]	1'b1 means use internal reference voltage 1'b0 means use external reference voltage
CLKSEL [14]	1'b1 means HT PLL uses differential clock input 1'b0 means HT PLL uses normal clock input
CLKSEL [13:12]	2'b00 means the PHY clock is 1.6GHZ / 1 2'b01 means the PHY clock is 3.2GHZ / 2 2'b10 indicates that the PHY clock is a normal input clock 2'b11 indicates that the PHY clock is a differential input clock
CLKSEL [11:10]	2'b00 means HT controller clock 200MHz 2'b01 means HT controller clock 400MHz 2'b1x means that the HT controller clock is a normal input clock

MEM clock control

signal	effect
CLKSEL [9: 5]	5'b11111 means MEM clock directly uses memclk In other cases, the MEM clock is memclk * (clksel [8: 5] +30) / (clksel [9] +3) Note: memclk * (clksel [8: 5] +30) must be 600MHz ~ 1.36GHz memclk must be 10 ~ 40MHz

CORE clock control

signal	effect
CLKSEL [4: 0]	5'b11111 indicates that the CORE clock directly uses sysclk In other cases, the CORE clock is sysclk * (clksel [3: 0] +30) / (clksel [4] +1) Note: sysclk * (clksel [3: 0] +30) must be 600MHz ~ 1.36GHz sysclk must be 10 ~ 40MHz

IO configuration control

7 HT control signal pin voltage control bit 1 *

6: 5 PCIX bus speed selection *

4 PCIX bus mode selection

3 PCI master mode

2 The system starts from the PCI space

1 Use external PCI arbitration

0 HT control signal pin voltage control bit 0 *

PCI_CONFIG [7: 0]

Note*:

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

Note*:

7	0	HT control signal pin voltage, these signals include HT_8x2, HT_Mode, HT_Powerok, HT_Rstn, HT_Ldt_Stopn, HT_Ldt_Reqn
0	0	1.8v
0	1	Reserved
1	0	2.5v
1	1	3.3v

7

2.3 Cache consistency

Loongson 3A1000 maintains the cache consistency between the processor and the I / O accessed through the HT port by hardware, but The hardware does not maintain the cache consistency of I / O devices connected to the system through PCI. During driver development, When PCI access devices perform DMA (Direct Memory Access) transmission, the software needs to perform Cache consistency maintain.

2.4 Distribution of physical address space at the node level of the system

The system physical address distribution of Loongson No. 3 series processors adopts a globally accessible hierarchical addressing design to System development is compatible with expansion. The physical address width of the entire system is 48 bits. According to the upper 4 bits of the address, the entire address is empt Time is evenly distributed to 16 nodes, that is, each node is allocated 44-bit address space.

Loongson 3A1000 uses a single node 4 core configuration, so Loongson 3A1000 chip integrated DDR memory controller, HT The corresponding addresses of the bus and PCI bus are contained in the 44-bit field from 0x0 (inclusive) to 0x1000_0000_0000 (not included) In the address space, please refer to the subsequent chapters for the specific address distribution of each device.

Table 2-2 Node-level system global address distribution

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

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

8

Page 21

Within each node, the 44-bit address space is further evenly distributed to a maximum of 8 devices that may be connected within the node
Prepare. Among them, the lower 43 bits of addresses are owned by 4 level 2 cache modules, and the higher 43 bits of addresses are further according to the address [43:42]
Bits are distributed to devices connected to the 4 directional ports. According to the different configuration of chip and system structure, if a port
If no slave device is connected, the corresponding address space is reserved address space, and access is not allowed.

Table 2-3 Address distribution in nodes

device	Address [43:41]	Start address within the node	Node end address
Level 2 Cache	0,1,2,3	0x000_0000_0000	0x800_0000_0000
east	4	0x800_0000_0000	0xa00_0000_0000
south	5	0xa00_0000_0000	0xc00_0000_0000
oo	6	0xc00_0000_0000	0xe00_0000_0000
north	7	0xe00_0000_0000	0x1000_0000_0000

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

Unlike the mapping relationship of direction ports, Loongson 3A1000 can determine the second level according to the actual application access behavior
Cache cross-addressing mode. The four Level 2 Cache modules in the node correspond to a total of 43 bits of address space, and each 2
The address space corresponding to the level module is determined according to one of the two selection bits of the address bit, and can be dynamically configured by software
modify. The configuration register named SCID_SEL is set in the system to determine the address selection bits, as shown in the following table. In default
In this case, it is distributed by means of [6: 5] status hash, that is, two bits of address [6: 5] determine the corresponding level 2 cache number.
The register address is 0x3FF00400.

Table 2-4 Address distribution in 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
4'h7	21:20	4'hf	37:36

9

Page 22

2.5 Address Routing Distribution and Configuration

The routing of Loongson 3A1000 is mainly realized through the two-stage crossbar of the system. One-level crossbar can

The master port receives requests for routing configuration. Each master port has 8 address windows, which can be completed

Target routing in 8 address windows. Each address window consists of three 64-bit registers BASE, MASK and MMAP,

BASE is aligned in K bytes; MASK adopts a format similar to the high bit of the netmask; the lower three bits of MMAP indicate the corresponding target

Slave port number, MMAP [4] means to allow instruction fetch, MMAP [5] means to allow block read, MMAP [7] means window enable

can.

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

Since Loongson 3 uses fixed routing by default, the configuration window is closed when the power is turned on.

System software is required to enable and configure it.

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

Table 2-5 Primary Crossbar Address Window Register Table

address	register	address	register
0x3ff0_2000	CORE0_WIN0_BASE	0x3ff0_2100	CORE1_WIN0_BASE
0x3ff0_2008	CORE0_WIN1_BASE	0x3ff0_2108	CORE1_WIN1_BASE
0x3ff0_2010	CORE0_WIN2_BASE	0x3ff0_2110	CORE1_WIN2_BASE
0x3ff0_2018	CORE0_WIN3_BASE	0x3ff0_2118	CORE1_WIN3_BASE
0x3ff0_2020	CORE0_WIN4_BASE	0x3ff0_2120	CORE1_WIN4_BASE
0x3ff0_2028	CORE0_WIN5_BASE	0x3ff0_2128	CORE1_WIN5_BASE
0x3ff0_2030	CORE0_WIN6_BASE	0x3ff0_2130	CORE1_WIN6_BASE
0x3ff0_2038	CORE0_WIN7_BASE	0x3ff0_2138	CORE1_WIN7_BASE
0x3ff0_2040	CORE0_WIN0_MASK	0x3ff0_2140	CORE1_WIN0_MASK
0x3ff0_2048	CORE0_WIN1_MASK	0x3ff0_2148	CORE1_WIN1_MASK
0x3ff0_2050	CORE0_WIN2_MASK	0x3ff0_2150	CORE1_WIN2_MASK
0x3ff0_2058	CORE0_WIN3_MASK	0x3ff0_2158	CORE1_WIN3_MASK
0x3ff0_2060	CORE0_WIN4_MASK	0x3ff0_2160	CORE1_WIN4_MASK
0x3ff0_2068	CORE0_WIN5_MASK	0x3ff0_2168	CORE1_WIN5_MASK
0x3ff0_2070	CORE0_WIN6_MASK	0x3ff0_2170	CORE1_WIN6_MASK
0x3ff0_2078	CORE0_WIN7_MASK	0x3ff0_2178	CORE1_WIN7_MASK
0x3ff0_2080	CORE0_WIN0_MMAP	0x3ff0_2180	CORE1_WIN0_MMAP
0x3ff0_2088	CORE0_WIN1_MMAP	0x3ff0_2188	CORE1_WIN1_MMAP
0x3ff0_2090	CORE0_WIN2_MMAP	0x3ff0_2190	CORE1_WIN2_MMAP
0x3ff0_2098	CORE0_WIN3_MMAP	0x3ff0_2198	CORE1_WIN3_MMAP

10

Page 23

0x3ff0_20a0	CORE0_WIN4_MMAP	0x3ff0_21a0	CORE1_WIN4_MMAP
0x3ff0_20a8	CORE0_WIN5_MMAP	0x3ff0_21a8	CORE1_WIN5_MMAP
0x3ff0_20b0	CORE0_WIN6_MMAP	0x3ff0_21b0	CORE1_WIN6_MMAP
0x3ff0_20b8	CORE0_WIN7_MMAP	0x3ff0_21b8	CORE1_WIN7_MMAP

0x3ff0_2200 CORE2_WIN0_BASE 0x3ff0_2300 CORE3_WIN0_BASE

0x3ff0_2208 CORE2_WIN1_BASE 0x3ff0_2308 CORE3_WIN1_BASE

0x3ff0_2210 CORE2_WIN2_BASE 0x3ff0_2310 CORE3_WIN2_BASE

0x3ff0_2218 CORE2_WIN3_BASE 0x3ff0_2318 CORE3_WIN3_BASE

0x3ff0_2220 CORE2_WIN4_BASE 0x3ff0_2320 CORE3_WIN4_BASE

0x3ff0_2228 CORE2_WIN5_BASE 0x3ff0_2328 CORE3_WIN5_BASE

0x3ff0_2230 CORE2_WIN6_BASE 0x3ff0_2330 CORE3_WIN6_BASE

0x3ff0_2238 CORE2_WIN7_BASE 0x3ff0_2338 CORE3_WIN7_BASE

0x3ff0_2240 CORE2_WIN0_MASK 0x3ff0_2340 CORE3_WIN0_MASK

0x3ff0_2248 CORE2_WIN1_MASK 0x3ff0_2348 CORE3_WIN1_MASK

0x3ff0_2250 CORE2_WIN2_MASK 0x3ff0_2350 CORE3_WIN2_MASK

0x3ff0_2258 CORE2_WIN3_MASK 0x3ff0_2358 CORE3_WIN3_MASK

0x3ff0_2260 CORE2_WIN4_MASK 0x3ff0_2360 CORE3_WIN4_MASK

0x3ff0_2268 CORE2_WIN5_MASK 0x3ff0_2368 CORE3_WIN5_MASK

0x3ff0_2270 CORE2_WIN6_MASK 0x3ff0_2370 CORE3_WIN6_MASK

0x3ff0_2278 CORE2_WIN7_MASK 0x3ff0_2378 CORE3_WIN7_MASK

0x3ff0_2280 CORE2_WIN0_MMAP 0x3ff0_2380 CORE3_WIN0_MMAP

0x3ff0_2288 CORE2_WIN1_MMAP 0x3ff0_2388 CORE3_WIN1_MMAP

0x3ff0_2290 CORE2_WIN2_MMAP 0x3ff0_2390 CORE3_WIN2_MMAP

0x3ff0_2298 CORE2_WIN3_MMAP 0x3ff0_2398 CORE3_WIN3_MMAP

0x3ff0_22a0 CORE2_WIN4_MMAP 0x3ff0_23a0 CORE3_WIN4_MMAP

0x3ff0_22a8 CORE2_WIN5_MMAP 0x3ff0_23a8 CORE3_WIN5_MMAP

0x3ff0_22b0 CORE2_WIN6_MMAP 0x3ff0_23b0 CORE3_WIN6_MMAP

0x3ff0_22b8 CORE2_WIN7_MMAP 0x3ff0_23b8 CORE3_WIN7_MMAP

0x3ff0_2400 EAST_WIN0_BASE 0x3ff0_2500 SOUTH_WIN0_BASE

0x3ff0_2408 EAST_WIN1_BASE 0x3ff0_2508 SOUTH_WIN1_BASE

0x3ff0_2410 EAST_WIN2_BASE 0x3ff0_2510 SOUTH_WIN2_BASE

0x3ff0_2418 EAST_WIN3_BASE 0x3ff0_2518 SOUTH_WIN3_BASE

11

Page 24

0x3ff0_2420 EAST_WIN4_BASE 0x3ff0_2520 SOUTH_WIN4_BASE

0x3ff0_2428 EAST_WIN5_BASE 0x3ff0_2528 SOUTH_WIN5_BASE

0x3ff0_2430 EAST_WIN6_BASE 0x3ff0_2530 SOUTH_WIN6_BASE

0x3ff0_2438 EAST_WIN7_BASE 0x3ff0_2538 SOUTH_WIN7_BASE

0x3ff0_2440 EAST_WIN0_MASK 0x3ff0_2540 SOUTH_WIN0_MASK

0x3ff0_2448 EAST_WIN1_MASK 0x3ff0_2548 SOUTH_WIN1_MASK

0x3ff0_2450 EAST_WIN2_MASK 0x3ff0_2550 SOUTH_WIN2_MASK

0x3ff0_2458 EAST_WIN3_MASK 0x3ff0_2558 SOUTH_WIN3_MASK

0x3ff0_2460 EAST_WIN4_MASK 0x3ff0_2560 SOUTH_WIN4_MASK

0x3ff0_2468 EAST_WIN5_MASK 0x3ff0_2568 SOUTH_WIN5_MASK

0x3ff0_2470 EAST_WIN6_MASK 0x3ff0_2570 SOUTH_WIN6_MASK

0x3ff0_2478 EAST_WIN7_MASK 0x3ff0_2578 SOUTH_WIN7_MASK

0x3ff0_2480 EAST_WIN0_MMAP 0x3ff0_2580 SOUTH_WIN0_MMAP

0x3ff0_2488 EAST_WIN1_MMAP 0x3ff0_2588 SOUTH_WIN1_MMAP

0x3ff0_2490 EAST_WIN2_MMAP 0x3ff0_2590 SOUTH_WIN2_MMAP

0x3ff0_2498 EAST_WIN3_MMAP 0x3ff0_2598 SOUTH_WIN3_MMAP

0x3ff0_24a0 EAST_WIN4_MMAP 0x3ff0_25a0 SOUTH_WIN4_MMAP

0x3ff0_24a8 EAST_WIN5_MMAP 0x3ff0_25a8 SOUTH_WIN5_MMAP

0x3ff0_24b0 EAST_WIN6_MMAP 0x3ff0_25b0 SOUTH_WIN6_MMAP

0x3ff0_24b8 EAST_WIN7_MMAP 0x3ff0_25b8 SOUTH_WIN7_MMAP

0x3ff0_2600 WEST_WIN0_BASE 0x3ff0_2700 NORTH_WIN0_BASE

0x3ff0_2608 WEST_WIN1_BASE 0x3ff0_2708 NORTH_WIN1_BASE

0x3ff0_2610 WEST_WIN2_BASE 0x3ff0_2710 NORTH_WIN2_BASE

0x3ff0_2618 WEST_WIN3_BASE 0x3ff0_2718 NORTH_WIN3_BASE

0x3ff0_2620 WEST_WIN4_BASE 0x3ff0_2720 NORTH_WIN4_BASE

0x3ff0_2628 WEST_WIN5_BASE 0x3ff0_2728 NORTH_WIN5_BASE

0x3ff0_2630 WEST_WIN6_BASE 0x3ff0_2730 NORTH_WIN6_BASE

0x3ff0_2638 WEST_WIN7_BASE 0x3ff0_2738 NORTH_WIN7_BASE

0x3ff0_2640 WEST_WIN0_MASK 0x3ff0_2740 NORTH_WIN0_MASK

0x3ff0_2648 WEST_WIN1_MASK 0x3ff0_2748 NORTH_WIN1_MASK

0x3ff0_2650 WEST_WIN2_MASK 0x3ff0_2750 NORTH_WIN2_MASK

0x3ff0_2658 WEST_WIN3_MASK 0x3ff0_2758 NORTH_WIN3_MASK

0x3ff0_2660 WEST_WIN4_MASK 0x3ff0_2760 NORTH_WIN4_MASK

12

Page 25

Godson **3A1000** Processor User Manual Part 1

0x3ff0_2668 WEST_WIN5_MASK 0x3ff0_2768 NORTH_WIN5_MASK

0x3ff0_2670 WEST_WIN6_MASK 0x3ff0_2770 NORTH_WIN6_MASK

0x3ff0_2678 WEST_WIN7_MASK 0x3ff0_2778 NORTH_WIN7_MASK

0x3ff0_2680 WEST_WIN0_MMAP 0x3ff0_2780 NORTH_WIN0_MMAP

0x3ff0_2688 WEST_WIN1_MMAP 0x3ff0_2788 NORTH_WIN1_MMAP

0x3ff0_2690 WEST_WIN2_MMAP 0x3ff0_2790 NORTH_WIN2_MMAP

0x3ff0_2698 WEST_WIN3_MMAP 0x3ff0_2798 NORTH_WIN3_MMAP

0x3ff0_26a0 WEST_WIN4_MMAP 0x3ff0_27a0 NORTH_WIN4_MMAP

0x3ff0_26a8 WEST_WIN5_MMAP 0x3ff0_27a8 NORTH_WIN5_MMAP

0x3ff0_26b0 WEST_WIN6_MMAP 0x3ff0_27b0 NORTH_WIN6_MMAP

0x3ff0_26b8 WEST_WIN7_MMAP 0x3ff0_27b8 NORTH_WIN7_MMAP

In the second-level XBAR of Godson 3, there are CPU address space (including HT space), DDR2 address space, and PCI

The address space has three IP-related address spaces. The address window is for CPU and PCI-DMA with two Master functions

IP is set for routing and address translation. Both CPU and PCI-DMA have 8 address windows, you can finish

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 BASE, MASK and MMAP are composed of three 64-bit registers, BASE is aligned with K bytes, and MASK adopts a similar netmask high bit as 1.

, The lower three digits of MMAP are routing.

At level 2 XBAR, the correspondence between the label and the module is as follows: the number corresponding to the new address space (two of which

The number of each DDR2 is 0 and 1, the PCI / Local IO number is 2, and the configuration register module is connected to port 3).

Table 2-6 Correspondence between the labels and the modules at level 2 XBAR

Label	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

As shown in the table below. MMAP [4] means to allow instruction fetch, MMAP [5] means to allow block read, MMAP [7] means to use window can.

Table 2-7 The space access attributes corresponding to the MMAP field

[4]	[5]	[7]
Allow fetching	Block read	Window enable

Compared with the address configuration of the first-level XBAR, the address configuration of the second-level XBAR adds the function of address translation. In contrast Next, the window configuration of the first-level XBAR cannot perform address translation for Cache consistency requests, otherwise it is in the second-level cache Will be inconsistent with the address of the first-level cache of the processor, resulting in incorrect maintenance of Cache consistency.

13

Page 26

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

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

The address window conversion register is as follows.

Table 2-8 Secondary XBAR address window conversion register table

address	register	description	Default value
3ff0 0000	CPU_WIN0_BASE	CPU window 0 base address	0x0
3ff0 0008	CPU_WIN1_BASE	CPU window 1 base address	0x1000_0000
3ff0 0010	CPU_WIN2_BASE	CPU window 2 base address	0x0
3ff0 0018	CPU_WIN3_BASE	CPU window 3 base address	0x0
3ff0 0020	CPU_WIN4_BASE	CPU window 4 base address	0x0
3ff0 0028	CPU_WIN5_BASE	CPU window 5 base address	0x0
3ff0 0030	CPU_WIN6_BASE	CPU window 6 base address	0x0
3ff0 0038	CPU_WIN7_BASE	CPU window 7 base address	0x0
3ff0 0040	CPU_WIN0_MASK	CPU window 0 mask	0xffff_ffff_f000_0000
3ff0 0048	CPU_WIN1_MASK	CPU window 1 mask	0xffff_ffff_f000_0000
3ff0 0050	CPU_WIN2_MASK	CPU window 2 mask	0x0
3ff0 0058	CPU_WIN3_MASK	CPU window 3 mask	0x0
3ff0 0060	CPU_WIN4_MASK	CPU window 4 mask	0x0
3ff0 0068	CPU_WIN5_MASK	Mask of CPU window 5	0x0
3ff0 0070	CPU_WIN6_MASK	CPU window 6 mask	0x0
3ff0 0078	CPU_WIN7_MASK	CPU window 7 mask	0x0
3ff0 0080	CPU_WIN0_MMAP	CPU window 0 new base address	0xf0
3ff0 0088	CPU_WIN1_MMAP	CPU window 1 new base address	0x1000_00f2
3ff0 0090	CPU_WIN2_MMAP	CPU window 2 new base address	0
3ff0 0098	CPU_WIN3_MMAP	CPU window 3 new base address	0
3ff0 00a0	CPU_WIN4_MMAP	CPU window 4 new base address	0x0
3ff0 00a8	CPU_WIN5_MMAP	CPU window 5 new base address	0x0
3ff0 00b0	CPU_WIN6_MMAP	CPU window 6 new base address	0
3ff0 00b8	CPU_WIN7_MMAP	CPU window 7 new base address	0

14

Page 27

3ff0 0100 PCI_WIN0_BASE PCI window 0 base address 0x8000_0000

3ff0 0108 PCI_WIN1_BASE PCI window 1 base address 0x0

3ff0 0110 PCI_WIN2_BASE PCI window 2 base address 0x0

3ff0 0118 PCI_WIN3_BASE PCI window 3 base address 0x0

3ff0 0120 PCI_WIN4_BASE PCI window 4 base address 0x0

3ff0 0128 PCI_WIN5_BASE PCI window 5 base address 0x0

3ff0 0130 PCI_WIN6_BASE PCI window 6 base address 0x0

3ff0 0138 PCI_WIN7_BASE PCI window 7 base address 0x0

3ff0 0140 PCI_WIN0_MASK PCI window 0 mask 0xffff_ffff_8000_0000

3ff0 0148 PCI_WIN1_MASK Mask of PCI window 1 0x0

3ff0 0150 PCI_WIN2_MASK PCI window 2 mask 0x0

3ff0 0158 PCI_WIN3_MASK PCI window 3 mask 0x0

3ff0 0160 PCI_WIN4_MASK PCI window 4 mask 0x0

3ff0 0168 PCI_WIN5_MASK PCI window 5 mask 0x0

3ff0 0170 PCI_WIN6_MASK Mask of PCI window 6 0x0

3ff0 0178 PCI_WIN7_MASK Mask of PCI window 7 0x0

3ff0 0180 PCI_WIN0_MMAP PCI window 0 new base address 0xf0

3ff0 0188 PCI_WIN1_MMAP PCI window 1 new base address 0x0

3ff0 0190 PCI_WIN2_MMAP New base address of PCI window 2 0

3ff0 0198 PCI_WIN3_MMAP PCI window 3 new base address 0

3ff0 01a0 PCI_WIN4_MMAP PCI window 4 new base address 0x0

3ff0 01a8 PCI_WIN5_MMAP PCI window 5 new base address 0x0

3ff0 01b0 PCI_WIN6_MMAP New base address of PCI window 6 0

3ff0 01b8 PCI_WIN7_MMAP PCI window 7 new base address 0

According to the default register configuration, after the chip is started, the address range of 0x00000000-0x0fffffff of the CPU (256M) mapped to the address range of 0x00000000-0x0fffffff of DDR2, 0x10000000 of CPU-

The 0x1fffffff interval (256M) is mapped to PCI 0x10000000-0x1fffffff interval, PCIDMA 0x80000000

-The address range (256M) of 0x8fffffff is mapped to the address range of 0x00000000-0x0fffffff of DDR2.

The software can implement new address space routing and conversion by modifying the corresponding configuration registers.

In addition, when there is a read access to an illegal address due to CPU speculative execution, none of the eight address windows hit.

15

Page 28

The configuration register module returns all 0 data to the CPU to prevent the CPU from dying.

Table 2-9 Secondary XBAR default address configuration

Base address	High position	owner
0x0000_0000_0000_0000	0x0000_0000_1000_0000	No. 0 DDR controller

0x0000_0000_1000_0000	0x0000_0000_2000_0000	Low-speed I / O (PCI, etc.)

2.6 Chip configuration and sampling register

The chip configuration register (Chip_config) and chip sampling register (chip_sample) in Godson 3 provide A mechanism to read and write the configuration of the chip.

Table 2-10 Chip Configuration Register (Physical Address 0x1fe00180)

Bit field	Field name	access	Reset value	description
2: 0	Freq_scale_ctrl	RW	3'b111	Processor core frequency division The actual frequency of the processor core is PLL frequency * (Freq_scal_ctrl + 1) / 8
3	DDR_Clksel_en	RW	1'b0	Whether to use software to configure DDR frequency multiplication 1: Use software configuration 0: use pin CLKSEL configuration
8	Disable_ddr2_confspace	RW	1'b0	Whether to disable the DDR configuration space 1: Disabled 0: Do not disable
9	DDR_buffer_cpu	RW	1'b0	Whether to open DDR read access buffer 1: open 0: disabled
12	Core0_en	RW	1'b1	Whether to enable processor core 0 1: open 0: disabled
13	Core1_en	RW	1'b1	Whether to enable processor core 1 1: open 0: disabled
14	Core2_en	RW	1'b1	Whether to enable processor core 2 1: open 0: disabled
15	Core3_en	RW	1'b1	Whether to enable processor core 3 1: open 0: disabled
16	Mc0_en	RW	1'b1	Whether to enable DDR controller 0 1: open 0: disabled
17	Mc1_en	RW	1'b1	Whether to enable DDR controller 1 1: open 0: disabled
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 the HT controller 0 1: open 0: disabled
23	HT1_en	RW	1'b1	Whether to enable the HT controller 1 1: open 0: disabled
28:24	DDR_Clksel	RW	5'b11111	Software configuration DDR clock multiplier relationship (when (Valid when DDR_Clksel_en is 1)

16

	HT_freq_scale_ctrl0	RW	3'b111	HT controller divide by 0 The actual frequency of the controller is
31:29				HT controller frequency * (HTFreq_scal_ctrl + 1) / 8
	HT_freq_scale_ctrl0	RW	3'b111	HT controller divided by 1 The actual frequency of the controller is
34:32				HT controller frequency * (HTFreq_scal_ctrl + 1) / 8
	Mc0_prefetch_disable	RW	1'b0	Whether to disable MC0 prefetch function (for different Program behavior will produce different performance effects
35				ring) 1: Disabled 0: Do not disable
	Mc1_prefetch_disable	RW	1'b0	Whether to disable MC1 prefetch function (for different Program behavior will produce different performance effects
36				ring) 1: Disabled 0: Do not disable
other		R		Keep

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

Bit field	Field name	access	Reset value	description
15: 0	Pad2v5_ctrl	RW	16'h780	2v5pad control
31:16	Pad3v3_ctrl	RW	16'h780	3v3pad control

17

47:32	Sys_clksel	R	Onboard frequency setting
51:48	Bad_ip_core	R	Indicates whether the processor core is unavailable, each bit Do not correspond to processor core 3-processor core 0 0-available 1-not available
53:52	Bad_ip_ddr	R	Whether 2 DDR controllers are bad
57:56	Bad_ip_ht	R	Whether 2 HT controllers are bad
102: 96	Thsens0_out	R	Temperature sensor 0 temperature, used to monitor secondary Temperature near the buffer, accuracy is +/- 6 Fahrenheit (Note: The sensor sometimes has abnormalities temperature)
103	Thsens0_overflow	R	Temperature sensor 0 temperature overflow (over 128 degree)
110: 104	Thsens1_out	R	Temperature sensor 1 temperature for monitoring processing The temperature near the core, the accuracy is +/- 6 Degree
111	Thsens1_overflow	R	Temperature sensor 1 temperature overflow (over 128 degree)
other		R	Keep

18

3 GS464 processor core

GS464 is a four-launch 64-bit high-performance processor core. The processor core can be used as a single core for high-end embedded Applications and desktop applications can also be used as basic processor cores to form on-chip multi-core systems for server and high-performance applications use. Multiple GS464 cores in Loongson 3A1000 and the secondary cache module form one through the AXI interconnection network Multi-core structure of distributed shared secondary cache. The main features of GS464 are as follows:

- MIPS64 compatible, support Godson extended instruction set;
- Four-shot superscalar structure, two fixed-point, two floating-point, and one memory access component;
- Each floating-point component supports full-pipe 64-bit / dual 32-bit floating-point multiply-add operations;
- The memory access component supports 128-bit memory access, and the virtual address and physical address are 48 bits each;
- Support register renaming, dynamic scheduling, branch prediction and other out-of-order execution technologies;
- 64 fully linked TLBs, independent 16 instruction TLBs, variable page size;
- The size of the first-level instruction cache and data cache are 64KB, and the 4-way group is connected;
- Support Non-blocking access and Load-Speculation and other access optimization technologies;
- Support Cache consistency protocol, can be used for on-chip multi-core processor;
- Instruction Cache implements parity check, and Data Cache implements ECC check;
- Support the standard EJTAG debugging standard, which is convenient for hardware and software debugging;
- Standard 128-bit AXI interface.

The structure of GS464 is shown in the figure below. For more detailed introduction, please refer to GS464 user manual and MIPS64 User manual.

19

Page 32

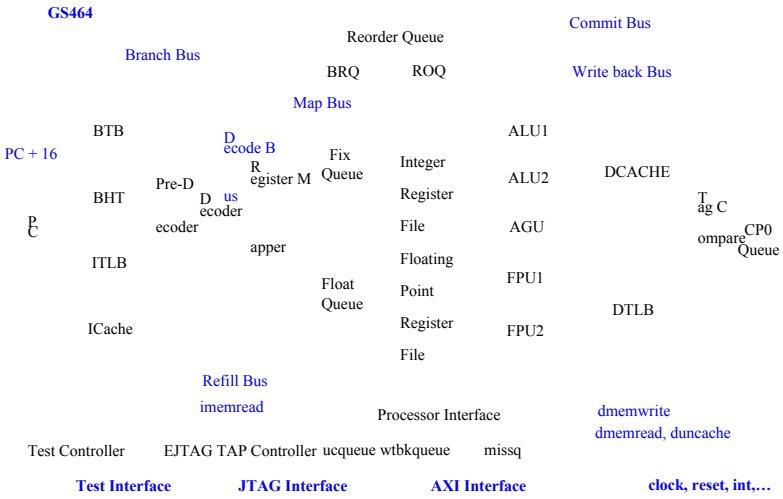


Figure 3-1 GS464 structure diagram

20

Page 33

4 Secondary Cache

The secondary cache module is a module designed to match the IP of the GS464 processor. This module can be docked with GS464,

Make GS464 the processor IP including secondary cache; you can also connect multiple GS464 through AXI network and

Multiple secondary cache modules form an on-chip multi-processor CMP structure. The main features of the secondary cache module include:

- Using 128-bit AXI interface.
- 8 items Cache access queue.
- Keywords first.
- The fastest read is 8 beats from receiving a read invalid request to returning data.
- Support Cache consistency protocol through the directory.
- It can be used for on-chip multi-core structure, and can also be directly connected with single processor IP.
- The soft IP level can be configured with the size of the secondary cache (512KB / 1MB).
- The four-way group connection structure is adopted.
- It can be closed dynamically during operation.
- Support ECC check.
- Support DMA consistent read and write and prefetch reading.
- Support 16 kinds of second-level cache hashes.
- Support to lock secondary cache by window.
- Ensure that read data returns atomicity.

The secondary cache module includes the secondary cache management module scachemanage and the secondary cache access module scacheaccess. The Scachemanage module is responsible for processor access requests from the processor and DMA, while the secondary cache The TAG, directory and data are stored in the scacheaccess module. In order to reduce power consumption, the TAG of the secondary cache, The directory and data can be accessed separately. The secondary cache status bit and w bit are stored with TAG, and TAG is stored in TAG RAM In, the directory is stored in DIR RAM, and the data is stored in DATA RAM. Invalid request to access secondary cache while reading The TAG, directory and data of all channels are output, and the data and directory are selected according to the TAG. Replace request, refill request and write back Request to operate only TAG, directory and data of all the way.

In order to improve the performance of some specific computing tasks, the secondary cache adds a locking mechanism. Level 2 in the locked area The Cache block will be locked, so it will not be replaced by the secondary cache (unless the four-way secondary cache is locked Piece). Four groups of lock window registers inside the secondary cache module can be dynamically configured through confbus, but must be Ensure that one of the four secondary caches is locked. The size of each group of windows can be adjusted according to the mask, but not Can exceed 3/4 of the size of the entire secondary cache. In addition, when the secondary cache receives the DMA write request, if the written area

If it is hit and locked in the secondary cache, the DMA write will be directly written to the secondary cache instead of memory.

Table 4-1 Secondary Cache Lock Window Register Configuration

name	address	Bit field	description
Slock0_valid	0x3ff00200	[63:63]	Lock window 0 valid bits
Slock0_addr	0x3ff00200	[47: 0]	No. 0 lock window lock address
Slock0_mask	0x3ff00240	[47: 0]	No. 0 lock window mask
Slock1_valid	0x3ff00208	[63:63]	Lock window 1 valid bit
Slock1_addr	0x3ff00208	[47: 0]	No. 1 lock window lock address
Slock1_mask	0x3ff00248	[47: 0]	No. 1 lock window mask
Slock2_valid	0x3ff00210	[63:63]	Lock window 2 valid bits
Slock2_addr	0x3ff00210	[47: 0]	No. 2 lock window lock address
Slock2_mask	0x3ff00250	[47: 0]	No. 2 lock window mask
Slock3_valid	0x3ff00218	[63:63]	Lock window 3 valid bits
Slock3_addr	0x3ff00218	[47: 0]	No. 3 lock window lock address
Slock3_mask	0x3ff00258	[47: 0]	No. 3 lock window mask

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

5 Matrix processing accelerator

There are two matrix processing accelerators independent of the processor core built into Loongson 3A1000, the basic function of which is through software
The configuration of the matrix can be used to transpose or move the matrix stored in the memory from the source matrix to the target matrix
(The previous version of LS3A1000E only supports the transpose function). Two accelerators are integrated in the two of Godson 3A1000
Inside the HyperTransport controller, read and write to the second-level cache and memory is achieved through a level 1 crossbar switch.

Since the order of elements in the same cache line before transposition is scattered in the matrix after transposition, in order to improve the read and write efficiency
Rate, you need to read in multiple rows of data, so that these data can be written in Cache rows in the transposed matrix
Input, so a buffer area with a size of 32 lines is set in the module to achieve horizontal writing (reading from the source matrix to
(Buffer), vertical readout (written from the buffer to the target matrix).

The working process of matrix processing is to first read 32 rows of source matrix data, and then write the 32 rows of data to the target matrix
Go on again until the entire matrix is transposed or moved. The matrix processing accelerator can also only perform prefetching as needed
The source matrix does not write the target matrix. In this way, the data is pre-fetched to the level 2 cache.

The source matrix involved in transposing or moving may be a small matrix located in a large matrix, so its matrix address
It may not be completely continuous. There will be gaps between the addresses of adjacent rows, and more programming control interfaces need to be implemented. The table below
5-1 to 5-4 illustrate the programming interfaces involved in matrix processing.

Table 5-1 Matrix processing programming interface description

address	name	Attributes	Explanation
0x3ff00600	src_start_addr	RW	Source matrix start address
0x3ff00608	dst_start_addr	RW	Target matrix start address

0x3ff00610	row	RW	Number of elements in a row of the source matrix
0x3ff00618	col	RW	Number of elements in a column of the source matrix
0x3ff00620	length	RW	Row span of the large matrix where the source matrix is located (bytes)
0x3ff00628	width	RW	Row span of the large matrix where the target matrix is located (bytes)
0x3ff00630	trans_ctrl	RW	Transpose control register
0x3ff00638	trans_status	RO	Transpose Status Register

Table 5-2 Matrix processing register address description

address	name
0x3ff00600	Src_start_addr of transpose module 0
0x3ff00608	Dst_start_addr of transpose module 0

twenty three

0x3ff00610	Row 0 transpose module
0x3ff00618	Col of transpose module 0
0x3ff00620	Length of transposed module 0
0x3ff00628	Width of transposed module 0
0x3ff00630	Trans_ctrl of transpose module 0
0x3ff00638	Trans_status of transpose module 0
0x3ff00700	Src_start_addr of transpose module 1
0x3ff00708	Dst_start_addr of transpose module 1
0x3ff00710	Src_row_stride of transpose module 1
0x3ff00718	Src_last_row_addr of transpose module 1
0x3ff00720	The length of transpose module 1
0x3ff00728	Width of transpose module 1
0x3ff00730	Trans_ctrl of transpose module 1
0x3ff00738	Trans_status of transpose module 1

Table 5-3 Explanation of the trans_ctrl register

Field	Explanation
0	Enable bit
1	Whether to write the target matrix. When it is 0, only the source matrix is prefetched, but the target matrix is not written.
2	After the source matrix is read, whether it is effectively interrupted.
3	After the target matrix is written, whether it is effectively interrupted,
7..4	Arcmd, read command internal control bit. When arcache is 4'hf, it must be set to 4'hc. It is meaningless when arcache is other value.
11..8	Arcache, read command internal control bit. When it is 4'hf, the cache path is used, and when it is 4'h0, the uncache path is used. its It is meaningless.
15..12	Awcmd, write command internal control bit. When awcache is 4'hf, it must be set to 4'hb. Unintentional when awcache is other values Righteousness.
19..16	Awcache, write command internal control bit. When it is 4'hf, the cache path is used, and when it is 4'h0, the uncache path is used. its It is meaningless.
21..20	Element size of matrix, 00 means 1 byte, 01 means 2 bytes, 10 means 4 bytes, 11 means 8 bytes
twenty two	trans_yes, is a transposed representation; 0 to not transpose (LS3A1000E earlier versions of the bit is read-only support Transpose function)

twenty four

Page 37

Table 5-4 Explanations of the trans_status registers

Field	Explanation
0	Source matrix read
1	The target matrix is written

25

Page 38

6 Inter-processor interrupt and communication

Loongson 3A1000 implements 8 inter-core interrupt registers (IPI) for each processor core to support multi-core BIOS boot

Interrupt and communication between the processor cores when the mobile and operating system are running, the description and addresses are shown in Table 6-1 to Table 6-5.

Table 6-1 Inter-processor interrupt related registers and their functional description

name	Read and write permissions	description
IPI_Status	R	32-bit status register, if any bit is set and the corresponding bit is enabled, the The processor core INT4 interrupt line is set.
IPI_Enable	RW	32-bit enable register to control whether the corresponding interrupt bit is valid
IPI_Set	W	32 position register, write 1 to the corresponding bit, the corresponding STATUS register Bit is set
IPI_Clear	W	32-bit clear register, write 1 to the corresponding bit, the corresponding STATUS register Bit cleared 0
MailBox0	RW	Cache register, used to transfer parameters at startup, according to 64 or 32 bit Uncache access.
MailBox01	RW	Cache register, used to transfer parameters at startup, according to 64 or 32 bit Uncache access.
MailBox02	RW	Cache register, used to transfer parameters at startup, according to 64 or 32 bit Uncache access.
MailBox03	RW	Cache register, used to transfer parameters at startup, according to 64 or 32 bit Uncache access.

The registers and functions of the interrupts between Loongson 3A1000 and processor cores are described as follows:

Table 6-2 Interrupt and communication register list of processor core 0

name	address	Authority	description
Core0_IPI_Status	0x3ff01000	R	IPI_Status register of processor core 0
Core0_IPI_Enalbe	0x3ff01004	RW	IPI_Enalbe register of processor core 0
Core0_IPI_Set	0x3ff01008	W	IPI_Set register of processor core 0
Core0 _IPI_Clear	0x3ff0100c	W	IPI_Clear register of processor core 0
Core0_MailBox0	0x3ff01020	RW	IPI_MailBox0 register of processor core 0
Core0_ MailBox1	0x3ff01028	RW	IPI_MailBox1 register of processor core 0
Core0_ MailBox2	0x3ff01030	RW	IPI_MailBox2 register of processor core 0
Core0_ MailBox3	0x3ff01038	RW	IPI_MailBox3 register of processor core 0

Table 6-3 Internuclear Interrupt and Communication Register List of No. 1 Processor Core

name	address	Authority	description
Core1_IPI_Status	0x3ff01100	R	IPI_Status register of processor core 1
Core1_IPI_Enalbe	0x3ff01104	RW	IPI_Enalbe register of processor core 1
Core1_IPI_Set	0x3ff01108	W	IPI_Set register of processor core 1

Core1 _IPI_Clear	0x3ff0110c	W	IPI_Clear register of processor core 1
Core1_MailBox0	0x3ff01120	R	IPI_MailBox0 register of processor core 1
Core1_ MailBox1	0x3ff01128	RW	IPI_MailBox1 register of processor core 1
Core1_ MailBox2	0x3ff01130	W	IPI_MailBox2 register of processor core 1
Core1_ MailBox3	0x3ff01138	W	IPI_MailBox3 register of processor core 1

Table 6-4 Internuclear Interrupt and Communication Register List of No. 2 Processor Core

name	address	Authority	description
Core2_IPI_Status	0x3ff01200	R	IPI_Status register of processor core 2
Core2_IPI_Enalbe	0x3ff01204	RW	IPI_Enalbe register of processor core 2
Core2_IPI_Set	0x3ff01208	W	IPI_Set register of processor core 2
Core2 _IPI_Clear	0x3ff0120c	W	IPI_Clear register of processor core 2
Core2_MailBox0	0x3ff01220	R	IPI_MailBox0 register of processor core 2
Core2_ MailBox1	0x3ff01228	RW	IPI_MailBox1 register of processor core 2

Core2_ MailBox2	0x3ff01230	W	IPI_MailBox2 register of processor core 2
Core2_ MailBox3	0x3ff01238	W	IPI_MailBox3 register of processor core 2

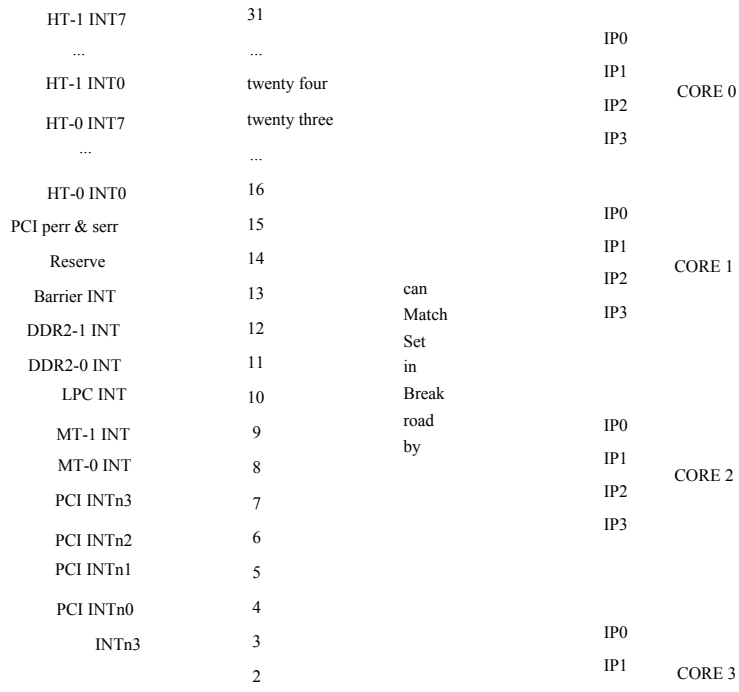
Table 6-5 List of Internuclear Interrupts and Communication Registers of Processor Core

name	address	Authority	description
Core3_IPI_Status	0x3ff01300	R	IPI_Status register of processor core 3
Core3_IPI_Enalbe	0x3ff01304	RW	IPI_Enalbe register of processor core 3
Core3_IPI_Set	0x3ff01308	W	IPI_Set register of processor core 3
Core3 _IPI_Clear	0x3ff0130c	W	IPI_Clear register of processor core 3
Core3_MailBox0	0x3ff01320	R	IPI_MailBox0 register of processor core 3
Core3_ MailBox1	0x3ff01328	RW	IPI_MailBox1 register of processor core 3
Core3_ MailBox2	0x3ff01330	W	IPI_MailBox2 register of processor core 3
Core3_ MailBox3	0x3ff01338	W	IPI_MailBox3 register of processor core 3

Listed above are the inter-core interrupt related messages for a single-node multiprocessor system composed of a single Loongson 3A1000 chip Memory list. When using multiple Loongson 3A1000 interconnects to form a multi-node CC-NUMA system, the node pairs in each chip Should be a system global node number, the IPI register address of the processor core in the node is based on the above table and the base of the node The addresses are in a fixed offset relationship. For example, the IPI_Status address of processor core 0 in node 0 is 0x3ff01000, and 1 The address of the No. 0 processor of the No. node is 0x10003ff01000, and so on.

7 I / O interrupt

Loongson 3A1000 chip supports up to 32 interrupt sources, which are managed in a unified manner, as shown in Figure 7-1 below, An IO interrupt source can be configured as enabled, triggered, and routed to the processor core interrupt pin.



INTn2 IP2
INTn1 1 IP3
INTn0 0

Figure 7-1 Loongson 3A1000 processor interrupt routing diagram

Interrupt related configuration registers are used to control the corresponding interrupt lines in the form of bits. See Table 7-1 for sexual configuration. The interrupt enable (Enable) configuration has three registers: Intenset, Intenclr and Inten. Intenset sets the interrupt enable, and the interrupt corresponding to the bit written to 1 in the Intenset register is enabled. Intenclr The clear interrupt is enabled, and the interrupt corresponding to the bit written in the Intenclr register is cleared. Inten register reads the current interrupt Enabled situation. The interrupt signal in the form of pulse (such as PCI_SERR) is selected by the Intedge configuration register, write 1 Display pulse trigger, write 0 to indicate level trigger. The interrupt handler can clear the pulse record through the corresponding bit of Intenclr record.

28

Page 41

Godson **3A1000** Processor User Manual Part 1

Table 7-1 Interrupt Control Register

Bit field	Access properties / default				
	Intedge	Inten	Intenset	Intenclr	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	Keep	Keep	Keep	Mc0-1
13	RW / 0	R / 0	RW / 0	RW / 0	Barrier
14	RW / 0	R / 0	RW / 0	RW / 0	Keep
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 Address

name	Address offset	description
Intisr	0x3ff01420	32-bit interrupt status register
Inten	0x3ff01424	32-bit interrupt enable status register
Intenset	0x3ff01428	32-bit setting enable register
Intenclr	0x3ff0142c	32-bit clear enable register
Intedge	0x3ff01438	32-bit trigger mode register
CORE0_INTISR	0x3ff01440	32-bit interrupt status routed to CORE0
CORE1_INTISR	0x3ff01448	32-bit interrupt status routed to CORE1
CORE2_INTISR	0x3ff01450	32-bit interrupt status routed to CORE2
CORE3_INTISR	0x3ff01458	32-bit interrupt status routed to CORE3

Four processor cores are integrated in Loongson 3A1000. The above 32-bit interrupt sources can be selected through software configuration. The target processor core is expected to be interrupted. Further, the interrupt source can be selected to route to any of the processor core interrupts Meaning one, that is, IP2 to IP5 corresponding to CP0_Status. Each of the 32 I / O interrupt sources corresponds to an 8-bit path By the controller, its format and address are shown in Tables 7-3 and 7-4 below. The routing register is routed in a vector way Select, such as 0x48 to route to INT2 of processor 3.

Table 7-3 Interrupt Routing Register Description

Bit field	Explanation

3: 0	Routed processor core vector number
7: 4	Routed processor core interrupt pin vector number

29

Page 42

Table 7-4 Interrupt Routing Register Address

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

30

Page 43

8 DDR2 / 3 SDRAM controller configuration

The design of the integrated memory controller inside Loongson No. 3 processor complies with the industry standard of DDR2 / 3 SDRAM (JESD79-2 And JESD79-3). In the Godson 3 processor, all memory read / write operations are implemented in compliance with JESD79-2B and The provisions of JESD79-3.

8.1 Overview of DDR2 / 3 SDRAM controller functions

Loongson No. 3 processor supports a maximum of 4 CS (implemented by 4 DDR2 SDRAM chip select signals, that is, two double-sided memory Article), contains a total of 18-bit address bus (ie: 15-bit row and column address bus and 3-bit logical Bank bus).

When Loongson No. 3 processor chooses to use different memory chip types, it can adjust the DDR2 / 3 controller parameter settings To support. Among them, the maximum number of chip selects (CS_n) supported is 4, the number of row addresses (RAS_n) is 15, and the column addresses The number of (CAS_n) is 14, and the number of logical body selection (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 according to the method shown in the figure below:

Taking the 4GB address space as an example, follow the configuration below:

Chip select = 4 Bank number = 8

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

Figure 8-1 Conversion of DDR2 SDRAM row and column addresses and CPU physical addresses

The memory control circuit integrated in the Loongson 3 processor only accepts memory read / write requests from the processor or external devices Demand, in all memory read / write operations, the memory control circuit is in the slave state.

The memory controller in Loongson No. 3 processor has the following characteristics:

- Full pipeline operation of commands and read and write data on the interface
- Memory commands are combined and sorted to improve overall bandwidth
- Configure register read and write ports, you can modify the basic parameters of the memory device
- Built-in dynamic delay compensation circuit (DCC) for reliable transmission and reception of data
- The ECC function can detect 1-bit and 2-bit errors on the data path, and can automatically detect 1-bit errors.

  Error correction
- Support 133-400MHZ working frequency

8.2 DDR2 / 3 SDRAM read operation protocol

The protocol of DDR2 / 3 SDRAM read operation is shown in Figure 11-2. In the figure, the command (Command, CMD for short) consists of

31

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 of DDR2 / 3 SDRAM write operation is shown in Figure 11-3. The command CMD in the figure is composed of RAS_n, CAS_n and WE_n is composed of three signals. For write operations, RAS_n = 1, CAS_n = 0, and WE_n = 0. In addition, with the read operation Differently, write operations require DQM to identify the mask of the write operation, that is, the number of bytes to be written. DQM is the same as the DQs signal in the figure step.

Figure 8-3 DDR2 SDRAM write operation protocol

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

8.4 DDR2 / 3 SDRAM parameter configuration format

Since different types of DDR2 / 3 SDRAM may be used in the system, after power-on reset DDR2 / 3 SDRAM configuration. The detailed configuration operation and configuration process are specified in JESD79-2B and JESD79-3,

DDR2 / 3 is not available until the DDR2 / 3 memory initialization operation is completed. The sequence of memory initialization operations is as follows under:

(1) When the system is reset, all the registers in the controller will be cleared to their initial values.

(2) The system is reset.

(3) Send a 64-bit write command to the configuration register address to configure all 180 configuration registers. If you write CTRL_03, the parameter START should be set to 0. All registers must be properly configured to work properly.

(4) Send a 64-bit write instruction to the configuration register CTRL_03. In this case, the parameter START should be set to 1. After the end The memory controller will automatically initiate initialization commands to the memory.

In the design of Loongson 3 processor, the configuration of DDR2 / 3 SDRAM needs to be Before using memory, configure the memory type. The specific configuration operation is to physical address 0x0000 0000 0FF0 0000 The corresponding 180 64-bit registers write the corresponding configuration parameters. A register may include multiple, one, and Subparameter data. The meanings of these configuration registers and the parameters they contain are as follows (unused bits in the registers are reserved Bit), a register configuration method based on DDR2 667 is also given in the table, the specific configuration can be based on the actual situation Then decide:

Table 8-1 DDR2 SDRAM configuration parameter register format

parameter name	Bit	Default value	range	description
CONF_CTL_00 [63: 0] Offset: 0x00			DDR2 667: 0x0000010000000101	
CONCURRENTAP	48:48	0x0	0x0-0x1	Whether the controller is allowed to automate a bank During precharge, issue a command to another bank. Note:

				Some memory modules are not supported
BANK_SPLIT_EN	40:40	0x0	0x0-0x1	Whether to allow the command queue reordering logic to split the bank (Split)
AUTO_REFRESH_MODE	32:32	0x0	0x0-0x1	Set whether auto-refresh is at the next burst or next Command boundary issue
AREFRESH	24:24	0x0	0x0-0x1	According to the setting of the auto_refresh_mode parameter, send to the memory Automatic refresh command (write only)
AP	16:16	0x0	0x0-0x1	Whether to enable the automatic refresh function of the memory controller, set to 1, indicating Memory access is CLOSE PAGE mode.
ADDR_CMP_EN	8: 8	0x0	0x0-0x1	Whether to allow command queue reordering logic to address conflicts Testing
CONF_CTL_01 [63: 0] Offset: 0x10			DDR2 667: 0x0000010100010000	
FWC	56:56	0x0	0x0-0x1	Whether write check is mandatory, when this parameter is set, memory The controller will use the number and data specified by the xor_check_bits parameter

Page 46

				Write XOR to memory (write only)
FAST_WRITE	48:48	0x0	0x0-0x1	Whether to allow the controller to open the fast write function. Open Quick Write After the function is enabled, the controller sends the memory mode after receiving all the written data. The block issues a write command.
ENABLE_QUICK_SREFRESH	40:40	0x0	0x0-0x1	Whether to enable fast self-refresh. When this parameter is enabled, the memory The self-refresh state is entered before the initialization of
EIGHT_BANK_MODE 32:32		0x0	0x0-0x1 indicates whether the memory module has 8 banks	
ECC_DISBALE_W_UC_ERR	24:24	0x0	0x0-0x1	When the write operation detects an unrecoverable error, is it forbidden to Memory verification code is set to error
DQS_N_EN	16:16	0x0	0x0-0x1 Set whether the DQS signal is single-ended or differential.	
DLLLOCKREG	0: 0	0x0	0x0-0x1	Indicates whether the DLL is locked (read-only), only if the DLL is locked After that, the read and write operations initiated to the memory can effectively reach the internal Memory, so you can use this bit to determine when to write memory for the first time.
CONF_CTL_02 [63: 0] Offset: 0x20			DDR2 667: 0x0100010100000000	
PRIORITY_EN	56:56	0x0	0x0-0x1 Whether to enable command queue reordering logic to use priority	
POWER_DOWN	48:48	0x0	0x0-0x1	When this parameter is enabled, the memory controller will use pre-charge Command to close all pages of the memory module and enable the clock enable signal Is low, do not send all commands received until this parameter is reset New setting is 0
PLACEMENT_EN	40:40	0x0	0x0-0x1 Whether to enable command reordering logic	
ODT_ADD_TURN_CLK_EN	32:32	0x0	0x0-0x1	In the middle of fast back-to-back read or write commands for different chip selections Whether to insert a turn-around clock. usually, Inserting such a cycle is needed for memory.
NO_CMD_INIT	24:24	0x0	0x0-0x1	In the memory initialization process, whether to prohibit the Issue other commands within tDLL time
INTRPTWRITENA	16:16	0x0	0x0-0x1	Whether to use autoprecharge command to add the same Other write commands of bank interrupt the previous write command
INTRPTREADA	8: 8	0x0	0x0-0x1	Whether to use autoprecharge command to add the same The other read commands of the bank interrupt the previous read command
INTRPTAPBURST	0: 0	0x0	0x0-0x1	Whether to allow the other bank's other commands to interrupt the current auto-precharge command

CONF_CTL_03 [63: 0] Offset: 0x30		DDR2 667: 0x0101010001000000	
SWAP_PORT_RW_SAME_EN	56:56	0x0	0x0-0x1 When swap_en is enabled, this parameter determines whether the same end Exchange similar commands on the mouth
SWAP_EN	48:48	0x0	0x0-0x1 When the command queue reordering logic is enabled, when the high priority command

34

START	40:40	0x0	0x0-0x1 When arriving, whether to exchange the command being executed with the new command Whether to initialize the memory. All parameters are required After the configuration is complete, set this bit to allow the memory to enter initialization Configuration. Configure this before completing the configuration of other bits Bit, it is likely to cause memory access errors.
SREFRESH	32:32	0x0	0x0-0x1 Whether the memory module enters the self-refresh working mode
RW_SAME_EN	24:24	0x0	0x0-0x1 Whether to consider the same bank in the command queue reordering logic Reorganization of read and write commands
REG_DIMM_EN	16:16	0x0	0x0-0x1 Whether to enable registered DIMM memory module
REDUC	8: 8	0x0	0x0-0x1 Whether to use only 32-bit wide memory data channels, usually In this case, the bit should not be set
PWRUP_SREFRESH_EXIT	0: 0	0x0	0x0-0x1 Use self-refresh command instead of normal memory initialization Command to exit power-down mode
CONF_CTL_04 [63: 0] Offset: 0x40		DDR2 667: 0x0102010100010101	
RTT_0	57:56	0x0	0x0-0x3 Enable the on-chip terminating resistor of the memory module. 00 –disable Other-enable, the size of the resistor is determined by the value in mrs_data
CTRL_RAW	49:48	0x0	0x0-0x3 Set the error detection and correction mode of ECC 2'b00 – without ECC 2'b01-only report errors, not correct them 2'b10-No ECC device is used 2'b11-Error correction using ECC
AXI0_W_PRIORITY	41:40	0x0	0x0-0x3 Set AXI0 port write command priority
AXI0_R_PRIORITY	33:32	0x0	0x0-0x3 Set the priority of AXI0 port read command
WRITE_MODEREG	24:24	0x0	0x0-0x1 Whether to write the EMRS register of the memory module (write only), each time When writing 1, the controller will set emrs_data and mrs_data is sent to memory.
WRITEINTERP	16:16	0x0	0x0-0x1 defines whether a read command can be used to interrupt a write burst
TREF_ENABLE	8: 8	0x0	0x0-0x1 Whether to enable the auto refresh function inside the controller, the usual situation In this case, the bit should be set to 1
TRAS_LOCKOUT	0: 0	0x0	0x0-0x1 Whether to issue auto-prechareg before the tRAS time expires command
CONF_CTL_05 [63: 0] Offset: 0x50		DDR2 667: 0x0700000404050100	
Q_FULLNESS	58:56	0x0	0x0-0x7 defines how many commands are in the memory controller command queue

35

			Command queue is full
PORT_DATA_ERROR_TYPE	50:48	0x0	0x0-0x7 Define the data error type on the memory controller port (read only) Bit 0-The number of burst data is greater than 16 Bit 1-Write data interleaving Bit 2-ECC 2 bit error
OUT_OF_RANGE_TYPE	42:40	0x0	0x0-0x7 defines the type of error when an out-of-bounds access occurs (read only)
MAX_CS_REG	34:32	0x4	0x0-0x4 defines the number of chip selects used by the controller (read only)
COLUMN_SIZE	26:24	0x0	0x0-0x7 Set the difference between the actual number of column addresses and the maximum number of column addresses (14) The value should be configured according to the specific memory particles. Number of column addresses used in memory = 14-COLUMN_SIZE
CASLAT	18:16	0x0	0x0-0x7 Set the CAS latency value. Should be based on specific memory particles Configure at different operating frequencies.
ADDR_PINS	10: 8	0x0	0x0-0x7 Set the difference between the number of actual address pins and the maximum number of addresses (15) value Number of address lines used in memory = 15 – ADDR_PINS
CONF_CTL_06 [63: 0] Offset: 0x60			DDR2 667: 0x0a04040603040003
APREBIT	59:56	0x0	0x0-0xf Define which address line is used to issue autoprecharge to memory Order, generally bit 10.
WRLAT	50:48	0x0	0x0-0x7 When the write command is issued until the first data is received (According to the number of clock cycles), at the same time decide when to make the corresponding ODT The signal is valid. Note: When WRLAT = (CASLAT_LIN / 2), it will not Add an extra delay between CS reading and writing.
TWTR	42:40	0x0	0x0-0x7 Define the clock period required to switch from write command to read command Number, need to be configured according to specific memory particles and operating frequency Set.
TWR_INT	34:32	0x0	0x0-0x7 Define the write recovery time of the memory module, according to the specific memory Particles and operating frequency are configured.
TRTP	26:24	0x0	0x0-0x7 Define the number of memory module read commands to precharge cycles It should be configured according to specific memory particles and operating frequency.
TRRD	18:16	0x0	0x0-0x7 Define the active command interval to different banks, need Configure according to specific memory particles and operating frequency.
TCKE	2: 0	0x0	0x0-0x7 defines the minimum pulse width of CKE signal

CONF_CTL_07 [63: 0] Offset: 0x70			DDR2 667: 0x0f0e0200000f0a0a
MAX_ROW_REG	59:56	0xf	0x0-0xf system maximum number of line addresses (read only)
MAX_COL_REG	51:48	0xe	0x0-0xe system maximum number of column addresses (read only)
INITAREF	43:40	0x0	0x0-0xf Define the autorefresh command to be executed during system initialization Order number. Set to 2 for DDR2 and 0 for DDR3.
CS_MAP	19:16	0x0	0x0-0xf Define the available chip select signals, this parameter should be based on the actual use The number of chip selects is used for correct configuration. Incorrect configuration will guide Memory access that caused the error. The four digits of this parameter correspond to CS0- CS3
			When the board trace delay is DDR2 clock cycle

CASLAT_LIN	3: 0	0x0	0x0-0xf	0.5 ~ 1.5 times: CASLAT_LIN = CASLAT × 2 Less than 0.5 times: CASLAT_LIN = CASLAT × 2-1 Greater than 1.5 times: CASLAT_LIN = CASLAT × 2 + 1 (In half clock cycles)
CONF_CTL_08 [63: 0] Offset: 0x80			DDR2 667: 0x0804020108040201	
ODT_WR_MAP_CS3 59:56		0x0	0x0-0xf	When CS3 has a write command, the specified CS ODT will be terminated. The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
ODT_WR_MAP_CS2 51:48		0x0	0x0-0xf	Define the CS2 ODT terminal when CS2 has a write command The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
ODT_WR_MAP_CS1 43:40		0x0	0x0-0xf	When CS1 has a write command, the ODT of the specified CS will be terminated. The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
ODT_WR_MAP_CS0 35:32		0x0	0x0-0xf	When CS0 has a write command, the ODT of the specified CS will be terminated. The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
ODT_RD_MAP_CS3 27:24		0x0	0x0-0xf	Define the CS3 ODT terminal when CS3 has a read command The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3

ODT_RD_MAP_CS2 19:16		0x0	0x0-0xf	Define the CS2 ODT terminal when CS2 has a read command The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
ODT_RD_MAP_CS1	11: 8	0x0	0x0-0xf	Define the CS1 ODT terminal when CS1 has a read command The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
ODT_RD_MAP_CS0	3: 0	0x0	0x0-0xf	When CS0 has a read command, the ODT of the specified CS will be terminated. The terminal resistance is valid, the specific configuration should refer to the corresponding memory chip The requirements for ODT configuration in the manual. The four digits of the parameter Do not correspond to CS0-CS3
CONF_CTL_09 [63: 0] Offset: 0x90 DDR2 667: 0x0000070d00000000				
OCD_ADJUST_PUP_CS0	60:56	0x0	0x0-0x1f	Set the memory module chip select 0 OCD pull-up adjustment value. Memory control The controller will initialize the memory module Group issues OCD adjustment commands
OCD_ADJUST_PDN_CS0	52:48	0x0	0x0-0x1f	Set the memory module chip select 0 OCD pull-down adjustment value. Memory control The controller will initialize the memory module Group issues OCD adjustment commands
TRP	43:40	0x0	0x0-0xf	Define the clock cycle required for the memory module to execute pre-charge Number of periods, need to be allocated according to specific memory particles and operating frequency

				Set.
TDAL	35:32	0x0	0x0-0xf	When the auto-precharge parameter is set, this parameter defines The number of auto-precharge and write recovery clock cycles. TDAL = auto-precharge + write recovery This parameter takes effect only after the AP is set.
PORT_CMD_ERROR_TYPE	19:16	0x0	0x0-0xf	Type of command error on the port (read only) Bit 0-Data bit width is too large Bit 1 – Keyword priority address misalignment Bit 2 – Keyword priority word count is not a power of 2 Bit 3-Error in narrow transform

CONF_CTL_10 [63: 0] Offset: 0xa0 DDR2 667: 0x0000003f3f140612

COMMAND_AGE_COUNT	37:32	0x0	0x0-0x3f	Define the command queue reordering logic when using the aging algorithm The initial aging value of the command
AGE_COUNT	29:24	0x0	0x0-0x3f	defines the command queue reordering logic when using the aging algorithm

				The initial aging value of the command
TRC	20:16	0x0	0x0-0x1f	Define the active commands between the same bank of the memory module The number of clock cycles, according to the specific memory particles and operating frequency To configure.
TMRD	12: 8	0x0	0x0-0x1f	Define the clock cycle required to configure the memory module mode register Number, usually 2 cycles
TFAW	4: 0	0x0	0x0-0x1f	defines the tFAW parameter of the memory module, used when there are 8 logical banks

CONF_CTL_12 [63: 0] Offset: 0xc0 DDR2 667: 0x00002c0511000000

TRFC	47:40	0x0	0x0-0xff	Define the number of clock cycles required for the refresh operation of the memory module. Configure according to specific memory particles and operating frequency.
TRCD_INT	39:32	0x0	0x0-0xff	Define the number of clock cycles between the memory module RAS and CAS, Need to be configured according to specific memory particles and operating frequency.
TRAS_MIN	31:24	0x0	0x0-0xff	Define the minimum clock cycle of the effective command of the memory module row address number
OUT_OF_RANGE_LENGTH	23:16	0x0	0x0-0xff	Command length when out-of-bounds access occurs (read only)
ECC_U_SYND	15: 8	0x0	0x0-0xff	Cause of 2bit uncorrectable error (read only)
ECC_C_SYND	7: 0	0x0	0x0-0xff	Cause of 1-bit error correction error (read only)

CONF_CTL_17 [63: 0] Offset: 0x110 DDR2 667: 0x0000000000000c2d

TREF	13: 0	0x0	0x0-0x3ff	Define the clock interval between two refresh commands of the memory module. Configure according to specific memory particles and operating frequency.

CONF_CTL_18 [63: 0] Offset: 0x120 DDR2 667: 0x001c000000000000

AXI0_EN_LT_WIDTH_INSTR	63:48	0x0000	0x0-0xffff	Defines whether the AXI0 port receives memory accesses that are less than 64 bits wide ask

CONF_CTL_19 [63: 0] Offset: 0x130 DDR2 667: 0x6d56000302000000

TRAS_MAX	63:48	0x0000	0x0-0xffff	Define the maximum number of clock cycles for the effective command of the memory module It should be configured according to specific memory particles and operating frequency.

TPDEX	47:32	0x0000	0x0-0xffff	defines the number of clock cycles for the memory module power-down exit command
TDLL	31:16	0x0000	0x0-0xffff	defines the number of clock cycles required to lock the memory module DLL
TCPD	15: 0	0x0000	0x0-0xffff	Define the clock between the memory module clock and the precharge Number of cycles, need to be based on specific memory particles and operating frequency Configuration.

CONF_CTL_20 [63: 0] Offset: 0x140 DDR2 667: 0x0000204002000030

XOR_CHECK_BITS	63:48	0x0000	0x0-0xffff	When the fwc parameter is set, the check bit of the next write operation will be Write to memory after XOR with this parameter
VERSION	47:32	0x2040	0x2040	Define the memory controller version number (read only)
TXSR	31:16	0x0000	0x0-0xffff	defines the number of clock cycles required for memory module self-refresh exit
TXSNR	15: 0	0x0000	0x0-0xffff	defines the memory module tXSNR parameters

CONF_CTL_21 [63: 0] Offset: 0x150 DDR2 667: 0x0000000000000000

ECC_C_ADDR [36: 8] 60:32		0x0	0x0-0x1fffffff	Record address information when a 1bit ECC error occurs (read only)
ECC_C_ADDR [7: 0]	31:24	0x0000	0x0-0x1fffffff	Record address information when a 1bit ECC error occurs (read only)
TINIT	23: 0	0x0000	0x0-0xfffff	Define the number of clock cycles required for memory module initialization Configure according to specific memory particles and operating frequency. Generally for 200us.

CONF_CTL_22 [63: 0] Offset: 0x160 DDR2 667: 0x0000000000000000

ECC_U_ADDR [36:32] 36:32		0x0	0x0-0x1fffffff	Record address information when a 2bit ECC error occurs (read only)
ECC_U_ADDR [31: 0]	31: 0	0x0	0x0-0x1fffffff	Record address information when a 2bit ECC error occurs (read only)

CONF_CTL_23 [63: 0] Offset: 0x170 DDR2 667: 0x0000000000000000

OUT_OF_RANGE_ADDR [36:32]	36:32	0x0	0x0-0x1fffffff	Record address information when cross-border access occurs (read only)
OUT_OF_RANGE_ADDR [31: 0]	31: 0	0x0	0x0-0x1fffffff	Record address information when cross-border access occurs (read only)

CONF_CTL_24 [63: 0] Offset: 0x180 DDR2 667: 0x0000000000000000

PORT_CMD_ERROR_ADDR [36:32]	36:32	0x0	0x0-0x1fffffff	Record address information when a command error occurs on the port (read only)
PORT_CMD_ERROR_ADDR [31: 0]	31: 0	0x0	0x0-0x1fffffff	Record address information when a command error occurs on the port (read only)

CONF_CTL_25 [63: 0] Offset: 0x190 DDR2 667: 0x0000000000000000

ECC_C_DATA [63:32] 63:32		0x0	0x0-0x1fffffff	Record data information when 1bit ECC error occurs (read only)
ECC_C_DATA [31: 0]	31: 0	0x0	0x0-0x1fffffff	Record data information when 1bit ECC error occurs (read only)

CONF_CTL_26 [63: 0] Offset: 0x1a0 DDR2 667: 0x0000000000000000

ECC_U_DATA [63:32]	63:32	0x0	0x0-0x1fffffff	Record data information when 2bit ECC error occurs (read only)
ECC_U_DATA [31: 0]	31: 0	0x0	0x0-0x1fffffff	Record data information when 2bit ECC error occurs (read only)

CONF_CTL_27 [63: 0] Offset: 0x1b0 DDR2 667: 0x0000000000000000

CKE_DELAY	2: 0	0x0	0x0-0x7	CKE effective delay. Note: Used to control the response of the internal srefresh_enter command Time, invalid for Godson No. 3.

CONF_CTL_29 [63: 0] Offset: 0x1d0 DDR2 667: 0x0103070400000101

TDFI_PHY_WRLAT_BASE	59:56	0x0	0x0-0xf	Set the delay required to write data in DDR PHY. For the dragon The value of core 3 should be 2
TDFI_PHY_WRLAT	51:48	0x0	0x0-0xf	Used to display the actual interval from write command to write data issue Number of cycles (read only)
TDFI_PHY_RDLAT	44:40	0x0	0x0-0xf	sets the number of cycles between the read command and the read data return interval
TDFI_CTRLUPD_MIN	35:32	0x4	0x0-0xf	Save DFI Tctrlup_min time parameter (read only)
DRAM_CLK_DISABLE	19:16	0x0	0x0-0xf	Set whether to output DRAM clock signal, each bit corresponds to a chip 选 信号。 Choice signal. 0: output clock signal; 1: disable output clock signal number.
ODT_ALT_EN	8: 8	0x0	0x0-0x1	Whether to support the ODT signal when CAS = 3. Note: For Godson No. 3, invalid
DRIVE_DQ_DQS	0: 0	0x0	0x0-0x1	Set whether to drive the data bus when the controller is idle

CONF_CTL_30 [63: 0] Offset: 0x1e0 DDR2 667: 0x0c2d0c2d0c2d0205

TDFI_PHYUPD_TYPE0	61:48	0x0000	0x0-0x3fff	This value is equal to TREF (read only)
TDFI_PHYUPD_RESP	45:32	0x0000	0x0-0x3fff	This value is equal to TREF (read only)
TDFI_CTRLUPD_MAX	29:16	0x0000	0x0-0x3fff	This value is equal to TREF (read only)
TDFI_RDDATA_EN_BASE	12: 8	0x00	0x0-0x1f	Basic time from DDR PHY internal read command to read return between. For Godson 3 this value is 2
TDFI_RDDATA_EN	4: 0	0x00	0x0-0x1f	Used to display the actual week from when the read command is issued to when the read data is returned Period

CONF_CTL_31 [63: 0] Offset: 0x1f0 DDR2 667: 0x0020008000000000

DLL_CTRL_REG_0_0	63:32	0x00000	0x0-0xffffffff	0th data group (DQ7-DQ0) DLL control signal

41

Page 54

				24: Control the enable signal of internal DLL, when it is 0, the DLL has effect 23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20 7: 0: Control the accuracy of the internal DLL. In Godson 3, this The value is generally 8'h80
DFT_CTRL_REG	7: 0	0x00	0x0-0xff	test enable signal, 0x0 is normal working mode

CONF_CTL_32 [63: 0] Offset: 0x200 DDR2 667: 0x0020008000200080

DLL_CTRL_REG_0_2	63:32	0x000000	0x0-0xffffffff	The second data group (DQ23-DQ16) DLL control signal 24: Control the enable signal of internal DLL, when it is 0, the DLL has effect 23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20 7: 0: Control the accuracy of the internal DLL. In Godson 3, this The value is generally 8'h80
DLL_CTRL_REG_0_1	31: 0	0x0000	0x0-0xffffffff	The first data group (DQ15-DQ8) DLL control signal 24: Control the enable signal of internal DLL, when it is 0 DLL works 23:16: Between control write data (DQ) and DQS Phase relationship, each value is expressed as (1 / precision) * 360. In Godson 3, this value is generally 1/4, Ie 8'h20 7: 0: Control the accuracy of the internal DLL. In, this value is generally 8'h80

CONF_CTL_33 [63: 0] Offset: 0x210 DDR2 667: 0x0020008000200080

DLL_CTRL_REG_0_4	63:32	0x000000	0x0-0xffffffff	The fourth data group (DQ39-DQ32) DLL control letter number 24: Control the enable signal of internal DLL, when it is 0 DLL works 23:16: Between control write data (DQ) and DQS Phase relationship, each value is expressed as (1 / precision) * 360. In Godson 3, this value is generally 1/4, Ie 8'h20

DLL_CTRL_REG_0_3	31: 0	0x00000	0x0-0xffffffff	7: 0: Control the accuracy of the internal DLL. In, this value is generally 8'h80 The third data group (DQ31-DQ24) DLL control signal 24: Control the enable signal of internal DLL, when it is 0, the DLL has effect 23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20 7: 0: Control the accuracy of the internal DLL. In Godson 3, this The value is generally 8'h80

CONF_CTL_34 [63: 0] Offset: 0x220 DDR2 667: 0x0020008000200080

DLL_CTRL_REG_0_6	63:32	0x00000	0x0-0xffffffff	Data group 6 (DQ55-DQ48) DLL control signal 24: Control the enable signal of internal DLL, when it is 0, the DLL has effect 23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20 7: 0: Control the accuracy of the internal DLL. In Godson 3, this The value is generally 8'h80

		The fifth data group (DQ47-DQ40) DLL control signal
		24: Control the enable signal of internal DLL, when it is 0, the DLL has effect
DLL_CTRL_REG_0_5 31: 0	0x00000 0x0-0xffffffff	23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20
		7: 0: Control the accuracy of the internal DLL. In Godson 3, this The value is generally 8'h80

CONF_CTL_35 [63: 0] Offset: 0x230 DDR2 667: 0x0020008000200080

		8th data group (DQ71-DQ64) DLL control signal
		24: Control the enable signal of internal DLL, when it is 0, the DLL has effect
DLL_CTRL_REG_0_8 63:32	0x00000 0x0-0xffffffff	23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20
		7: 0: Control the accuracy of the internal DLL. In Godson 3, this

		The value is generally 8'h80
		7th data group (DQ63-DQ56) DLL control signal
		24: Control the enable signal of internal DLL, when it is 0, the DLL has effect
DLL_CTRL_REG_0_7 31: 0	0x0000 0x0-0xffffffff	23:16: Control the phase between write data (DQ) and DQS Relationship, each value is expressed as (1 / precision) * 360. In Godson 3 In the number, this value is generally 1/4, which is 8'h20
		7: 0: Control the accuracy of the internal DLL. In Godson 3, this The value is generally 8'h80

CONF_CTL_36 [63: 0] Offset: 0x240 DDR2 667: 0x00001e0000001e00

		1st data group DLL control signal
DLL_CTRL_REG_1_1 63:32	0x0000 0x0-0xffffffff	15: 8: When the read data returns, the phase of DQSn is delayed.
		5: 0: DLL test control signal, normally 8'h0
		0th data group DLL control signal
DLL_CTRL_REG_1_0 31: 0	0x00000 0x0-0xffffffff	15: 8: When the read data returns, the phase of DQSn is delayed.
		5: 0: DLL test control signal, normally 8'h0

CONF_CTL_37 [63: 0] Offset: 0x250 DDR2 667: 0x00001e0000001e00

		The third data group DLL control signal
DLL_CTRL_REG_1_3 63:32	0x0000 0x0-0xffffffff	15: 8: When the read data returns, the phase of DQSn is delayed.
		5: 0: DLL test control signal, normally 8'h0
		The second data group DLL control signal
DLL_CTRL_REG_1_2 31: 0	0x000000 0x0-0xffffffff	15: 8: When the read data returns, the phase of DQSn is delayed.
		5: 0: DLL test control signal, normally 8'h0

CONF_CTL_38 [63: 0] Offset: 0x260 DDR2 667: 0x00001e0000001e00

		5th data group DLL control signal
DLL_CTRL_REG_1_5 63:32	0x00000 0x0-0xffffffff	15: 8: When the read data returns, the phase of DQSn is delayed.
		5: 0: DLL test control signal, normally 8'h0
		4th data group DLL control signal

DLL_CTRL_REG_1_4 31: 0 0x0000 0x0-0xffffffff 15: 8: When the read data returns, the phase of DQSn is delayed. 5: 0:

DLL test control signal, normally 8'h0

CONF_CTL_39 [63: 0] Offset: 0x270 DDR2 667: 0x00001e0000001e00

7th data group DLL control signal

DLL_CTRL_REG_1_7 63:32 0x0000 0x0-0xffffffff

15: 8: When the read data returns, the phase of DQSn is delayed. 5: 0:

Page 57

DLL test control signal, normally 8'h0.

6th data group DLL control signal

DLL_CTRL_REG_1_6 31: 0 0x0000 0x0-0xffffffff 15: 8: When the read data returns, the phase of DQSn is delayed.

5: 0: DLL test control signal, normally 8'h0

CONF_CTL_40 [63: 0] Offset: 0x280 DDR2 667: 0x0000000000001e00

DLL_OBS_REG_0_0 33:32 0x0 0x0-0x3 DLL output of data group 0 in test mode (read only)

8th data group DLL control signal

DLL_CTRL_REG_1_8 31: 0 0x00000 0x0-0xffffffff 15: 8: When the read data returns, the phase of DQSn is delayed.

5: 0: DLL test control signal, normally 8'h0

CONF_CTL_41 [63: 0] Offset: 0x290 DDR2 667: 0x0000000000000000

DLL_OBS_REG_0_2 33:32 0x0 0x0-0x3 DLL output of the second data group in test mode (read only)

DLL_OBS_REG_0_1 1: 0 0x0 0x0-0x3 DLL output of the first data group in test mode (read only)

CONF_CTL_42 [63: 0] Offset: 0x2a0 DDR2 667: 0x0x0000000000000000

DLL_OBS_REG_0_4 33:32 0x0 0x0-0x3 DLL output of the 4th data group in test mode (read only)

DLL_OBS_REG_0_3 1: 0 0x0 0x0-0x3 DLL output of the 3rd data group in test mode (read only)

CONF_CTL_43 [63: 0] Offset: 0x2b0 DDR2 667: 0x0x0000000000000000

DLL_OBS_REG_0_6 33:32 0x0 0x0-0x3 DLL output of the 6th data group in test mode (read only)

DLL_OBS_REG_0_5 1: 0 0x0 0x0-0x3 DLL output of the 5th data group in test mode (read only)

CONF_CTL_44 [63: 0] Offset: 0x2c0 DDR2 667: 0x0000000000000000

DLL_OBS_REG_0_8 33:32 0x0 0x0-0x3 8th data group DLL output in test mode (read only)

DLL_OBS_REG_0_7 1: 0 0x0 0x0-0x3 DLL output of the 7th data group in test mode (read only)

CONF_CTL_45 [63: 0] Offset: 0x2d0 DDR2 667: 0xf30029470000019d

Data group 0 delay control.

28: Whether to use deburring circuit for reading DQS, refer to gate

Whether the signal is delayed by PAD_feedback

27: Use read FIFO effective signal to automatically control read data return

Sampling (1), or use fixed time sampling in 26:24 (0)

PHY_CTRL_REG_0_0 63:32 0x00000 0x0-0xffffffff 26:24: Reading data returns to the timing of sampling completion, from the internal clock

The sampling delay of the domain.

21: In Read Leveling mode, sample the data bus

Level

20: The level of the effective data control signal, which is 0 in Godson 3

19: Whether to delay writing data by another cycle

18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize)

17: Does the write data / DQS delay increase the half-cycle delay

16: Whether CAS delay is half cycle

15:12: Effective start time for writing DQS, for DDR3
It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS

11: 8: The effective end time for writing DQS

6: 4: Start time when writing data is valid

2: 0: End time when write data is valid

Pin control signal

25:22: Corresponding to COMPZCP_dig

21:18: Corresponding to COMPZCN_dig

17: TQ1v8 corresponding to the pin

16: Corresponding to the enable signal of the internal feedback pin, active low

15: Output enable signal corresponding to internal feedback pin, active low

14: Output enable signal corresponding to the data strobe pin, active low

13: Output enable signal corresponding to the data shield pin, active low

12: Output enable signal corresponding to the data pin, active low

11: USEPAD of the corresponding pin

0: Use internal reference voltage;

PAD_CTRL_REG_0 25: 0 0x0000 0x0-0x3ffffff 1: Use external reference voltage.

8: The enable signal corresponding to the clock pins {1, 3, 5}, high effective

7: Enable signal corresponding to clock pins {0, 2, 4}, high effective

6: Enable signal corresponding to address pin, active low

5: PROGB1v8 corresponding to the pin

4: PROGA1v8 corresponding to the pin

Used to control pin drive capability

3: ODTB of the corresponding pin

2: ODTA of the corresponding pin

Used to control the pin ODT resistance

ODTA	ODTB	DDRII	DDRIII
1	0	150	120

1	1	75	60
0	0	Disable	Disable

1: MODEZI1v8 corresponding to the pin

For Loongson 3, it should be set to 0.

0: DDR1v8 corresponding to the pin

0: Corresponding to the 1.8v mode of DDRII

1: 1.5v mode corresponding to DDRIII

CONF_CTL_46 [63: 0] Offset: 0x2e0 DDR2 667: 0xf3002947f3002947

		Data group 2 delay control. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus Level 20: The level of the effective data control signal, which is 0 in Godson 3
PHY_CTRL_REG_0_2 63:32	0x00000 0x0-0xffffffff	19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid
PHY_CTRL_REG_0_1 31: 0	0x00000 0x0-0xffffffff	Delay control of the first data group. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0)

47

Page 60

		26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid

CONF_CTL_47 [63: 0] Offset: 0x2f0 DDR2 667: 0xf3002947f3002947

		Data group 4 delay control. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus
PHY_CTRL_REG_0_4 63:32	0x00000 0x0-0xffffffff	Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements

48

Page 61

		Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid Data group 3 delay control. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus
PHY_CTRL_REG_0_3 31: 0	0x0000 0x0-0xffffffff	Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid

CONF_CTL_48 [63: 0] Offset: 0x300 DDR2 667: 0xf3002947f3002947

Data group 6 delay control.

			28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus
PHY_CTRL_REG_0_6	63:32	0x00000000	0x0-0xffffffff
49			

			Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid Data group 5 delay control. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid
PHY_CTRL_REG_0_5	31: 0	0x00000000	0x0-0xffffffff
50			

Page 63

CONF_CTL_49 [63: 0] Offset: 0x310 DDR2 667: 0xf3002947f3002947

PHY_CTRL_REG_0_8 63:32	0x00000 0x0-0xffffffff	Data group 8 delay control. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr Synchronize) 17: Does the write data / DQS delay increase the half-cycle delay 16: Whether CAS delay is half cycle 15:12: Effective start time for writing DQS, for DDR3 It should be opened one cycle earlier than DDR2 to provide particle requirements Preamble DQS 11: 8: The effective end time for writing DQS 6: 4: Start time when writing data is valid 2: 0: End time when writing data is valid
PHY_CTRL_REG_0_7 31: 0	0x0000 0x0-0xffffffff	7th data group delay control. 28: Whether to use deburring circuit for reading DQS, refer to gate Whether the signal is delayed by PAD_feedback 27: Use read FIFO effective signal to automatically control read data return Sampling (1), or use fixed time sampling in 26:24 (0) 26:24: Reading data returns to the timing of sampling completion, from the internal clock The sampling delay of the domain. 21: In Read Leveling mode, sample the data bus Level 20: The level of the effective data control signal, which is 0 in Godson 3 19: Whether to delay writing data by another cycle 18: Whether reading DQS sampling is 1/4 cycle ahead (with clk_wr

51

Page 64

Synchronize)

17: Does the write data / DQS delay increase the half-cycle delay

16: Whether CAS delay is half cycle

15:12: Effective start time for writing DQS, for DDR3

It should be opened one cycle earlier than DDR2 to provide particle requirements

Preamble DQS

		11: 8: The effective end time for writing DQS
		6: 4: Start time when writing data is valid
		2: 0: End time when writing data is valid

CONF_CTL_50 [63: 0] Offset: 0x320 DDR2 667: 0x07c0000007c00000

		Terminal resistance control of PAD in the first data group, initiate read operation
		Will only be enabled when
		31:28: Timing control of terminal resistance opening, read command sent from
		The calculation starts after the last 4 beats, each value represents a half cycle
		27:24: Timing control of terminal resistance off
		23: Effective level control of termination resistance
		Is 1
		22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used
PHY_CTRL_REG_1_1 63:32	0x00000 0x0-0xffffffff	The control of the termination resistance is enabled; when it is 0, it can be
		The terminating resistor on bit PAD is always valid (set to 0) or never
		Effectiveness (set to 1)
		21: Test signal, normal should be 0
		20:16: Test signal, normally 0
		14:12: Test signal, normally 0
		11: 8: read sampling delay 1, of which only 1 bit is valid, used for
		Control when DQS sampling window is closed
		7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
		Opening timing of the DQS sampling window
		Terminal resistance control of PAD in data group 0, initiate read operation
		Will only be enabled when
		31:28: Timing control of terminal resistance opening, read command sent from
PHY_CTRL_REG_1_0 31: 0	0x00000 0x0-0xffffffff	The calculation starts after the last 4 beats, each value represents a half cycle
		27:24: Timing control of terminal resistance off
		23: Effective level control of termination resistance

		Is 1
		22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used
		The control of the termination resistance is enabled; when it is 0, it can be
		The terminating resistor on bit PAD is always valid (set to 0) or never
		Effectiveness (set to 1)
		21: Test signal, normal should be 0
		20:16: Test signal, normally 0
		14:12: Test signal, normally 0
		11: 8: read sampling delay 1, of which only 1 bit is valid, used for
		Control when DQS sampling window is closed
		7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
		Opening timing of the DQS sampling window

CONF_CTL_51 [63: 0] Offset: 0x330 DDR2 667: 0x07c0000007c00000

		Terminal resistance control of PAD in data group 3, initiate read operation
		Will only be enabled when
		31:28: Timing control of terminal resistance opening, read command sent from

		The calculation starts after the last 4 beats, each value represents a half cycle 27:24: Timing control of terminal resistance off 23: Effective level control of termination resistance Is 1 22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used The control of the termination resistance is enabled; when it is 0, it can be The terminating resistor on bit PAD is always valid (set to 0) or never Effectiveness (set to 1) 21: Test signal, normal should be 0 20:16: Test signal, normally 0 14:12: Test signal, normally 0 11: 8: read sampling delay 1, of which only 1 bit is valid, used for Control when DQS sampling window is closed 7: 0: read sampling delay 0, of which only 1 bit is valid for controlling Opening timing of the DQS sampling window
PHY_CTRL_REG_1_3 63:32	0x00000 0x0-0xffffffff	
PHY_CTRL_REG_1_2 31: 0	0x00000 0x0-0xffffffff	Terminal resistance control of PAD in the second data group, initiate read operation Will only be enabled when 31:28: Timing control of terminal resistance opening, read command sent from The calculation starts after the last 4 beats, each value represents a half cycle

Page 66

CONF_CTL_52 [63: 0] Offset: 0x340 DDR2 667: 0x07c0000007c00000

		27:24: Timing control of terminal resistance off 23: Effective level control of termination resistance Is 1 22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used The control of the termination resistance is enabled; when it is 0, it can be The terminating resistor on bit PAD is always valid (set to 0) or never Effectiveness (set to 1) 21: Test signal, normal should be 0 20:16: Test signal, normally 0 14:12: Test signal, normally 0 11: 8: read sampling delay 1, of which only 1 bit is valid, used for Control when DQS sampling window is closed 7: 0: read sampling delay 0, of which only 1 bit is valid for controlling Opening timing of the DQS sampling window
PHY_CTRL_REG_1_5 63:32	0x00000 0x0-0xffffffff	Terminal resistance control of PAD in the 5th data group, initiate read operation Will only be enabled when 31:28: Timing control of terminal resistance opening, read command sent from The calculation starts after the last 4 beats, each value represents a half cycle 27:24: Control of the timing of opening the terminal resistance 23: Effective level control of termination resistance Is 1 22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used The control of the termination resistance is enabled; when it is 0, it can be The terminating resistor on bit PAD is always valid (set to 0) or never Effectiveness (set to 1)

21: Test signal, normal should be 0
20:16: Test signal, normally 0
14:12: Test signal, normally 0
11: 8: read sampling delay 1, of which only 1 bit is valid, used for
Control when DQS sampling window is closed
7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
Opening timing of the DQS sampling window

PHY_CTRL_REG_1_4 31: 0 0x00000 0x0-0xffffffff Terminal resistance control of PAD in the 4th data group, initiate read operation
Will only be enabled when

54

Page 67

31:28: Timing control of terminal resistance opening, read command sent from
The calculation starts after the last 4 beats, each value represents a half cycle
27:24: Timing control of terminal resistance off
23: Effective level control of termination resistance
Is 1
22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used
The control of the termination resistance is enabled; when it is 0, it can be
The terminating resistor on bit PAD is always valid (set to 0) or never
Effectiveness (set to 1)
21: Test signal, normal should be 0
20:16: Test signal, normally 0
14:12: Test signal, normally 0
11: 8: read sampling delay 1, of which only 1 bit is valid, used for
Control when DQS sampling window is closed
7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
Opening timing of the DQS sampling window

CONF_CTL_53 [63: 0] Offset: 0x350 DDR2 667: 0x07c0000007c00000

Terminal resistance control of PAD in the 7th data group, initiate read operation
Will only be enabled when
31:28: Timing control of terminal resistance opening, read command sent from
The calculation starts after the last 4 beats, each value represents a half cycle
27:24: Timing control of terminal resistance off
23: Effective level control of termination resistance
Is 1
22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used
The control of the termination resistance is enabled; when it is 0, it can be
PHY_CTRL_REG_1_7 63:32 0x00000 0x0-0xffffffff The terminating resistor on bit PAD is always valid (set to 0) or never
Effectiveness (set to 1)
21: Test signal, normal should be 0
20:16: Test signal, normally 0
14:12: Test signal, normally 0
11: 8: read sampling delay 1, of which only 1 bit is valid, used for
Control when DQS sampling window is closed
7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
Opening timing of the DQS sampling window

55

Page 68

		Terminal resistance control of PAD in the 6th data group, initiate read operation
		Will only be enabled when
		31:28: Timing control of terminal resistance opening, read command sent from
		The calculation starts after the last 4 beats, each value represents a half cycle
		27:24: Timing control of terminal resistance off
		23: Effective level control of termination resistance
		Is 1
		22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used
PHY_CTRL_REG_1_6 31: 0	0x00000 0x0-0xffffffff	The control of the termination resistance is enabled; when it is 0, it can be
		The terminating resistor on bit PAD is always valid (set to 0) or never
		Effectiveness (set to 1)
		21: Test signal, normal should be 0
		20:16: Test signal, normally 0
		14:12: Test signal, normally 0
		11: 8: read sampling delay 1, of which only 1 bit is valid, used for
		Control when DQS sampling window is closed
		7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
		Opening timing of the DQS sampling window

CONF_CTL_54 [63: 0] Offset: 0x360 DDR2 667: 0x0800c00507c00000

		Read and write data delay control
		27: Select the read data buffer type, the default is 0
		26: Used to clear the data returned from the read buffer, normally 0
		25: High-speed pin enable, when it is 1, all signals pass through the pin
		Outbound transmission delay is reduced by 1 cycle
PHY_CTRL_REG_2 63:32	0x00000 0x0-0xffffffff	16:13: Set the valid time for reading data and collect data from FIFO
		According to the delay back to the controller. If the delay from the pin to the FIFO
		Increase late, this value must also increase
		8: Set whether the DQS signal output is DDR3 mode, DDR3
		In mode, the Preamble to write DQS will contain a pulse
		5: Test mode signal, normally 0
		4: Test mode signal, normally 0
		Termination resistance control in the 8th data group
PHY_CTRL_REG_1_8 31: 0	0x00000 0x0-0xffffffff	Terminal resistance control of PAD, it will be enabled only when the read operation is initiated
		31:28: Timing control of terminal resistance opening, read command sent from

56

Page 69

		The calculation starts after the last 4 beats, each value represents a half cycle
		27:24: Timing control of terminal resistance off
		23: Effective level control of termination resistance
		Is 1

22: Enable signal of the terminating resistor, when it is 1, the dynamic square is used
The control of the termination resistance is enabled; when it is 0, it can be
The terminating resistor on bit PAD is always valid (set to 0) or never
Effectiveness (set to 1)
21: Test signal, normal should be 0
20:16: Test signal, normally 0
14:12: Test signal, normally 0
11: 8: read sampling delay 1, of which only 1 bit is valid, used for
Control when DQS sampling window is closed
7: 0: read sampling delay 0, of which only 1 bit is valid for controlling
Opening timing of the DQS sampling window

CONF_CTL_55 [63: 0] Offset: 0x370 DDR2 667: 0x0000000000000000

PHY_OBS_REG_0_1 63:32 0x00000 0x0-0xffffffff Observation signal for the 1st data set test (read only)

PHY_OBS_REG_0_0 31: 0 0x00000 0x0-0xffffffff Observation signal for the 0th data set test (read only)

CONF_CTL_56 [63: 0] Offset: 0x380 DDR2 667: 0x0000000000000000

PHY_OBS_REG_0_3 63:32 0x00000 0x0-0xffffffff Observation signal for the 3rd data set test (read only)

PHY_OBS_REG_0_2 31: 0 0x00000 0x0-0xffffffff Observation signal for the 2nd data set test (read only)

CONF_CTL_57 [63: 0] Offset: 0x390 DDR2 667: 0x0000000000000000

PHY_OBS_REG_0_5 63:32 0x00000 0x0-0xffffffff Observation signal for the 5th data set test (read only)

PHY_OBS_REG_0_4 31: 0 0x0000 0x0-0xffffffff 4th data set test observation signal (read only)

CONF_CTL_58 [63: 0] Offset: 0x3a0 DDR2 667: 0x0000000000000000

PHY_OBS_REG_0_7 63:32 0x00000 0x0-0xffffffff 7th data set test observation signal (read only)

PHY_OBS_REG_0_6 31: 0 0x00000 0x0-0xffffffff Observation signal for the 6th data set test (read only)

CONF_CTL_59 [63: 0] Offset: 0x3b0 DDR2 667: 0x0000000000000000

PHY_OBS_REG_0_8 31: 0 0x00000 0x0-0xffffffff 8th data group test observation signal (read only)

CONF_CTL_114 [63: 0] Offset: 0x720 DDR2 667: 0x0000000000000000

RDLVL_GATE_REQ 56 0x0 0x0-0x1 User request read strobe sampling training function. (Write only)

RDLVL_GATE_PREA 48 0x0 0x0-0x1 Enable pre-sampling check when reading strobe sampling training

57

Page 70

MBLE_CHECK_EN

RDLVL_GATE_EN 40 0x0 0x0-0x1 Read strobe sampling training when Read Leveling is enabled, in the initial
After the completion of the conversion, it will send a command to the particle to read the DQS sampling
Window training

RDLVL_EN 32 0x0 0x0-0x1 enable Read Leveling function

RDLVL_BEGIN_DELA Y_EN twenty four 0x0 0x0-0x1 Enable Read Leveling to find data sampling point function

SWLVL_OP_DONE 8 0x0 0x0-0x1 is used to indicate whether the software Leveling is completed (read only)

CONF_CTL_115 [63: 0] Offset: 0x730 DDR2 667: 0x0000000000000000

RDLVL_OFFSET_DIR _7 56 0x0 0x0-0x1 7th data group Read Leveling adjustment direction of midpoint.
When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is
1 is added.

RDLVL_OFFSET_DIR _6 48 0x0 0x0-0x1 6th data set Read Leveling adjustment direction of midpoint.
When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is
1 is added.

RDLVL_OFFSET_DIR_5	40	0x0	0x0-0x1	5th data set Read Leveling adjustment direction of midpoint. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.
RDLVL_OFFSET_DIR_4	32	0x0	0x0-0x1	The adjustment direction of the midpoint during the fourth data set Read Leveling. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.
RDLVL_OFFSET_DIR_3	twenty four	0x0	0x0-0x1	The adjustment direction of the midpoint during the third data set Read Leveling. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.
RDLVL_OFFSET_DIR_2	16	0x0	0x0-0x1	The adjustment direction of the midpoint during the second data set Read Leveling. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.
RDLVL_OFFSET_DIR_1	8	0x0	0x0-0x1	The adjustment direction of the midpoint during the first data set Read Leveling. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.
RDLVL_OFFSET_DIR_0	0	0x0	0x0-0x1	The adjustment direction of the midpoint during the 0th data set Read Leveling. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.

CONF_CTL_116 [63: 0] Offset: 0x740 DDR2 667: 0x0100000000000000

AXI1_PORT_ORDERING	57:56	0x0	0x0-0x3	Whether internal port 1 can be executed out of order, invalid for Godson No. 3
	58			

NG				
AXI0_PORT_ORDERING	49:48	0x0	0x0-0x3	Whether internal port 0 can be executed out of order
WRLVL_REQ	40	0x0	0x0-0x1	User request to start Write Leveling training function. (Write only)
WRLVL_INTERVAL_CT_EN	32	0x0	0x0-0x1	Enable Write Leveling time interval function
WEIGHTED_ROUND_ROBIN_WEIGHT_SHARING	twenty four	0x0	0x0-0x1	Per-port pair shared arbitration for WRR
WEIGHTED_ROUND_ROBIN_LATENCY_CONTROL	16	0x0	0x0-0x1	Free-running or limited WRR latency counters.
RDLVL_REQ	8	0x0	0x0-0x1	User request to start the Read Leveling training function. (Write only)
RDLVL_OFFSET_DIR_8	0	0x0	0x0-0x1	The 8th data set Read Leveling adjustment direction of the midpoint. When 0, the midpoint is calculated by subtracting rdlvl_offset_delay, which is 1 is added.

CONF_CTL_117 [63: 0] Offset: 0x750 DDR2 667: 0x0100000101020101

WRLVL_CS	57:56	0x0	0x0-0x3	indicates the chip select signal of the current Write Leveling operation
SW_LEVELING_MODE	49:48	0x0	0x0-0x3	Define the mode of software leveling operation
RDLVL_CS	41:40	0x0	0x0-0x3	indicates the chip select signal of the current Read Leveling operation
AXI2_W_PRIORITY	33:32	0x0	0x0-0x3	The write priority of internal port 2 is invalid for Godson No. 3
AXI2_R_PRIORITY	25:24	0x0	0x0-0x3	The read priority of internal port 2 is invalid for Godson No. 3
AXI2_PORT_ORDERING	17:16	0x0	0x0-0x3	Whether internal port 2 can be executed out of order, invalid for Godson 3

AXI1_W_PRIORITY	9: 8	0x0	0x0-0x3 Internal port 1 write priority, invalid for Godson No. 3
AXI1_R_PRIORITY	1: 0	0x0	0x0-0x3 The read priority of internal port 1 is invalid for Godson No. 3

CONF_CTL_118 [63: 0] Offset: 0x760 DDR2 667: 0x0303030000020002

AXI0_PRIORITY2_RELATIVE_PRIORITY	59:56	0x0	0x0-0xf Relative priority of commands for internal port 0 priority 2
AXI0_PRIORITY1_RELATIVE_PRIORITY	51:48	0x0	0x0-0xf The relative priority of the internal port 0 priority 1 command
AXI0_PRIORITY0_RELATIVE_PRIORITY	43:40	0x0	0x0-0xf Relative priority of commands of internal port 0 priority 0

ADDRESS_MIRRORING	35:32	0x0	0x0-0xf indicates which chip select supports Address mirroring
TDFI_DRAM_CLK_DISABLE	26:24	0x0	0x0-0x7 Delay setting from internal clock off to external clock off
BSTLEN	18:16	0x0	0x0-0x7 Set the Burst length value sent to the memory module on the controller
ZQ_REQ	9: 8	0x0	0x0-0x3 User request to start ZQ adjustment function
ZQ_ON_SREF_EXIT	1: 0	0x0	0x0-0x3 define the mode of ZQ adjustment function when exiting self-refresh mode

CONF_CTL_119 [63: 0] Offset: 0x770 DDR2 667: 0x0101010202020203

AXI2_PRIORITY2_RELATIVE_PRIORITY	59:56	0x0	0x0-0xf The relative priority of the internal port 2 priority 2 commands, for Godson 3 is invalid
AXI2_PRIORITY1_RELATIVE_PRIORITY	51:48	0x0	0x0-0xf The relative priority of the internal port 2 priority 1 command, for Godson 3 is invalid
AXI2_PRIORITY0_RELATIVE_PRIORITY	43:40	0x0	0x0-0xf The relative priority of the internal port 2 priority 0 command, for Godson 3 is invalid
AXI1_PRIORITY3_RELATIVE_PRIORITY	35:32	0x0	0x0-0xf The relative priority of the internal port 1 priority 3 commands, for Godson 3 is invalid
AXI1_PRIORITY2_RELATIVE_PRIORITY	27:24	0x0	0x0-0xf The relative priority of the internal port 1 priority 2 commands, for Godson 3 is invalid
AXI1_PRIORITY1_RELATIVE_PRIORITY	19:16	0x0	0x0-0xf The relative priority of the internal port 1 priority 1 command, for Godson 3 is invalid
AXI1_PRIORITY0_RELATIVE_PRIORITY	11: 8	0x0	0x0-0xf The relative priority of the internal port 1 priority 0 command, for Godson 3 is invalid
AXI0_PRIORITY3_RELATIVE_PRIORITY	3: 0	0x0	0x0-0xf Relative priority for commands with internal port 0 priority 3

CONF_CTL_120 [63: 0] Offset: 0x780 DDR2 667: 0x0102020400040c01

TDFI_DRAM_CLK_ENABLE	59:56	0x0	0x0-0xf Delay from internal clock valid to output clock valid
TDFI_CTRL_DELAY	51:48	0x0	0x0-0xf Delay from clock valid to output command
RDLVL_GATE_DQ_ZERO_COUNT	43:40	0x0	0x0-0xf When setting the read gate sampling training, it means the number of 0 from 1 to 0 number
RDLVL_DQ_ZERO_COUNT	35:32	0x0	0x0-0xf When setting Read Leveling, it means 0 from 1 to 0. Number
LOWPOWER_REFRESH_ENABLE	27:24	0x0	0x0-0xf enable refresh function in low power mode

60

Page 73

DRAM_CLASS	19:16	0x0	0x0-0xf	Define the external memory type of the controller 110: DDR3 100: DDR2
BURST_ON_FLY_BIT	11: 8	0x0	0x0-0xf	burst-on-fly bit in the mode configuration issued to DRAM
AXI2_PRIORITY3_RELATIVE_PRIORITY	3: 0	0x0	0x0-0xf	The relative priority of the internal port 2 priority 3 commands, for Godson 3 is invalid

CONF_CTL_121 [63: 0] Offset: 0x790 DDR2 667: 0x281900000f000303

WLMRD	61:56	0x00	0x0-0x3f	Delay from DRAM transmission mode configuration to Write Leveling late
WLDQSEN	53:48	0x00	0x0-0x3f	From the configuration of DRAM transmission mode to Write Leveling Gating data sampling delay
LOWPOWER_CONTROL	44:40	0x00	0x0-0x1f	Low power mode enable Bit 4: power down Bit 3: power down external Bit 2: self refresh Bit 1: external Bit 0: internal
LOWPOWER_AUTO_ENABLE	36:32	0x00	0x0-0x1f	Enable to automatically enter low power mode when the controller is idle The control bit is the same as LOWERPOWER_CONTROL
ZQCS_CHIP	27:24	0x0	0x0-0xf	defines the valid chip selection for the next ZQ
WRR_PARAM_VALUE_ERR	19:16	0x0	0x0-0xf	Errors / warnings related to the WRR parameters. (read only)
TDFI_WRLVL_DLL	15: 8	0x00	0x0-0xff	Read operation to Write Leveling Update the minimum number of delay lines cycle
TDFI_RDLVL_DLL	7: 0	0x00	0x0-0xff	Read operation to Read Leveling Update the minimum number of delay lines cycle

CONF_CTL_122 [63: 0] Offset: 0x7a0 DDR2 667: 0x0000000000000000

SWLVL_RESP_6	63:56	0x00	0x0-0xff	Leveling response of the 6th data group
SWLVL_RESP_5	55:48	0x00	0x0-0xff	Leveling response of the 5th data group
SWLVL_RESP_4	47:40	0x00	0x0-0xff	Leveling response of the 4th data group
SWLVL_RESP_3	39:32	0x00	0x0-0xff	Leveling response of the 3rd data group
SWLVL_RESP_2	31:24	0x00	0x0-0xff	Leveling response of the 2nd data group
SWLVL_RESP_1	23:16	0x00	0x0-0xff	Leveling response of the 1st data group

61

Page 74

SWLVL_RESP_0	15: 8	0x00	0x0-0xff	Leveling response of data group 0

CONF_CTL_123 [63: 0] Offset: 0x7b0 DDR2 667: 0x0000000000000000

OBSOLETE	63:16

SWLVL_RESP_8	15: 8	0x00	0x0-0xff Leveling response of the 8th data group
SWLVL_RESP_7	7: 0	0x00	0x0-0xff Leveling response of the 7th data group

CONF_CTL_124 [63: 0] Offset: 0x7c0 DDR2 667: 0x0000000000000000

OBSOLETE

CONF_CTL_125 [63: 0] Offset: 0x7d0 DDR2 667: 0x0000000000000000

RDLVL_GATE_CLK_ADJUST_3	63:56	0x00	0x0-0xff In the third data group, read the start value of sampling training
RDLVL_GATE_CLK_ADJUST_2	55:48	0x00	0x0-0xff In the second data set, read the start value of sampling training
RDLVL_GATE_CLK_ADJUST_1	47:40	0x00	0x0-0xff In the first data set, read the start value of sampling training
RDLVL_GATE_CLK_ADJUST_0	39:32	0x00	0x0-0xff In the 0th data set, read the start value of sampling training

CONF_CTL_126 [63: 0] Offset: 0x7e0 DDR2 667: 0x0000000000000000

RDLVL_GATE_CLK_ADJUST_8	39:32	0x00	0x0-0xff In the 8th data set, read the start value of sampling training
RDLVL_GATE_CLK_ADJUST_7	31:24	0x00	0x0-0xff In the 7th data set, read the start value of sampling training
RDLVL_GATE_CLK_ADJUST_6	23:16	0x00	0x0-0xff In the 6th data set, read the start value of sampling training
RDLVL_GATE_CLK_ADJUST_5	15: 8	0x00	0x0-0xff In the fifth data set, read the start value of the sample training
RDLVL_GATE_CLK_ADJUST_4	7: 0	0x00	0x0-0xff In the fourth data set, read the start value of the sample training

CONF_CTL_127 [63: 0] Offset: 0x7f0 DDR2 667: 0x0000000000000000

OBSOLETE

CONF_CTL_128 [63: 0] Offset: 0x800 DDR2 667: 0x0000000000000000

OBSOLETE

CONF_CTL_129 [63: 0] Offset: 0x810 DDR2 667: 0x0000000000000000

OBSOLETE

62

Page 75

CONF_CTL_130 [63: 0] Offset: 0x820 DDR2 667: 0x0420000c20400000

TDFI_WRLVL_RESPLAT	63:56	0x00	0x0-0xff Write Leveling Strobe to valid number of cycles
TDFI_RDLVL_RESPLAT	39:32	0x00	0x0-0xff Read Leveling The number of cycles from strobe to response valid
REFRESH_PER_ZQ 23:16		0x00	0x0-0xff Number of refresh commands between automatic ZQCS commands

CONF_CTL_131 [63: 0] Offset: 0x830 DDR2 667: 0x0000000000000c0a

TMOD	15: 8	0x00	0x0-0xff Number of cycles to be idle after DRAM mode configuration

CONF_CTL_132 [63: 0] Offset: 0x840 DDR2 667: 0x0000640064000000

AXI1_PRIORITY_RELAX	49:40	0x000	0x0-0x3ff Counter value that triggers priority control relaxation on internal port 1 Yulongxin No. 3 is invalid
AXI0_PRIORITY_RELAX [9: 8]	33:32	0x0	0x0-0x3 Counter value that triggers priority control relaxation on internal port 0

AXI0_PRIORITY_RELAX [7: 0]	31:24	0x00	0x0-0xff	Counter value that triggers priority control relaxation on internal port 0

CONF_CTL_133 [63: 0] Offset: 0x850 DDR2 667: 0x0000000000000064

OUT_OF_RANGE_SOURCE_ID	57:48	0x000	0x0-0x3ff	ID number of access address overflow request (read only)
ECC_U_ID	41:32	0x000	0x0-0x3ff	access ID number request with 2 bit error (read only)
ECC_C_ID	25:16	0x000	0x0-0x3ff	access ID number request with 1 bit error (read only)
AXI2_PRIORITY_RELAX	9: 0	0x000	0x0-0x3ff	Counter value that triggers priority control relaxation on internal port 2 Yulongxin No. 3 is invalid

CONF_CTL_134 [63: 0] Offset: 0x860 DDR2 667: 0x0000004000000000

ZQCS	43:32	0x000	0x0-0xfff	Number of cycles required by ZQCS command
PORT_DATA_ERROR_ID	25:16	0x000	0x0-0x3ff	ID number of internal port data error request (read only)
PORT_CMD_ERROR_ID	9: 0	0x000	0x0-0x3ff	Internal port command error request ID number (read only)

CONF_CTL_135 [63: 0] Offset: 0x870 DDR2 667: 0x0000000000000000

OBSOLETE

CONF_CTL_136 [63: 0] Offset: 0x880 DDR2 667: 0x0000000000000000

OBSOLETE

CONF_CTL_137 [63: 0] Offset: 0x890 DDR2 667: 0x0000000000000000

63

Page 76

OBSOLETE

CONF_CTL_138 [63: 0] Offset: 0x8a0 DDR2 667: 0x00000000001c001c

LOWPOWER_INTERNAL_CNT	63:48	0x0000	0x0-0xffff	Counts idle cycles to self-refresh with memory and controller clk gating.
LOWPOWER_EXTERNAL_CNT	47:32	0x0000	0x0-0xffff	Counts idle cycles to self-refresh with memory clock gating.
AXI2_EN_SIZE_LT_WIDTH_INSTR	31:16	0x0000	0x0-0xffff	Enable various narrow accesses on internal port 2 for Loongson 3 invalid
AXI1_EN_SIZE_LT_WIDTH_INSTR	15: 0	0x0000	0x0-0xffff	Enable various narrow accesses on internal port 1, for Godson 3 invalid

CONF_CTL_139 [63: 0] Offset: 0x8b0 DDR2 667: 0x0000000000000000

LOWPOWER_POWER_DOWN_CNT	15: 0	0x0000	0x0-0xffff	Number of idle cycles before entering Power Down mode
LOWPOWER_REFRESH_HOLD	31:16	0x0000	0x0-0xffff	In clock gating mode, before the memory controller re-locks the DLL Number of idle cycles
LOWPOWER_SELF_REFRESH_CNT	47:32	0x0000	0x0-0xffff	Number of idle cycles before entering memory self-refresh mode

CONF_CTL_140 [63: 0] Offset: 0x8c0 DDR2 667: 0x0004000000000000

OBSOLETE

CONF_CTL_141 [63: 0] Offset: 0x8d0 DDR2 667: 0x00000000c8000000

CKE_INACTIVE [31: 8]	55:32	0x0000000	0x0-0xffffffff	The time from output DDR_RESET valid to CKE valid High
CKE_INACTIVE [7: 0]	31:24	0x00	0x0-0xff	The time from output DDR_RESET valid to CKE valid

				Low
WRLVL_STATUS	17: 0	0x00000	0x0-0x3ffff	Last Write Leveling operation status (read only)

CONF_CTL_142 [63: 0] Offset: 0x8e0 DDR2 667: 0x0000000000000050

TRST_PWRON	31: 0	0x0000000	0x0-0xffffffff	From 500 shots after start is valid to DDR_RESET is valid Delay

CONF_CTL_143 [63: 0] Offset: 0x8f0 DDR2 667: 0x0000000020202080

DLL_CTRL_REG_2 [32]	32:32	0x0	0x0-0x1	output clock DLL enable signal, active high
DLL_CTRL_REG_2 [31: 0]	31: 0	0x00000000	0x0-0xffffffff	Output clock DLL control 31:24: Delay of CLK4 and CLK5 on the output clock DLL 23:16: Delay of CLK2 and CLK3 on the output clock DLL 15: 8: Delay of CLK0 and CLK1 on the output clock DLL 7: 0: Accuracy value on output clock DLL

64

Page 77

CONF_CTL_144 [63: 0] Offset: 0x900 DDR2 667: 0x0000000000000000

RDLVL_ERROR_STATUS [37:32]	37:32	0x00	0x0-0x3f	indicates the status of RDLVL when an error occurs
RDLVL_ERROR_STATUS [31: 0]	31: 0	0x00000000	0x0-0xffffffff	indicates the status of RDLVL when an error occurs

CONF_CTL_145 [63: 0] Offset: 0x910 DDR2 667: 0x0000000000000000

RDLVL_GATE_RESP_MASK [63:32]	63:32	0x00000000	0x0-0xffffffff	read back mask during sampling training
RDLVL_GATE_RESP_MASK [31: 0]	31: 0	0x00000000	0x0-0xffffffff	read back mask during sampling training

CONF_CTL_146 [63: 0] Offset: 0x920 DDR2 667: 0x0000000000000000

RDLVL_GATE_RESP_MASK [71:64]	7: 0	0x00	0x0-0xff	read back masking during sampling training

CONF_CTL_147 [63: 0] Offset: 0x930 DDR2 667: 0x0000000000000000

RDLVL_RESP_MASK [63:32]	63:32	0x00000000	0x0-0xffffffff	Read Leveling read back mask
RDLVL_RESP_MASK [31: 0]	31: 0	0x00000000	0x0-0xffffffff	Read Leveling read back mask

CONF_CTL_148 [63: 0] Offset: 0x940 DDR2 667: 0x0301010000050500

TDFI_RDLVL_EN	59:56	0x0	0x0-0xf	Read Leveling Enable to Read Leveling Minimum number of cycles
W2R_SAMECS_DLY	50:48	0x0	0x0-0x7	Additional delay between write and read for the same chip select signal
W2R_DIFFCS_DLY	42:40	0x0	0x0-0x7	Additional delay between write and read for different chip select signals
LVL_STATUS	34:32	0x0	0x0-0x7	Write Leveling, Read Leveling and sampling training request Status for LVL_REQ interrupt (read only)
RDLVL_EDGE	twenty four	0x0	0x0-0x1	In Read Leveling operation, it indicates that the rising edge of DQS is valid or Valid on falling edge
CKSRX	19:16	0x0	0x0-0x0	Clock cycle delay to exit self refresh
CKSRE	11: 8	0x0	0x0-0x0	clock cycle delay to enter self-refresh mode
RDLVL_RESP_MASK [71:64]	7: 0	0x00	0x0-0xff	Read Leveling read back mask

CONF_CTL_149 [63: 0] Offset: 0x950 DDR2 667: 0x0000000000000a03

65

Page 78

INT_MASK	42:24	0x00	0x0-0x7ffff	Interrupt mask [18] = OR of all interrupt bits; [17] = User initiated DLL synchronization end flag; [16] = DLL lock signal changed (locked and unlocked) Switch); [15] = Error reading the sampling clock; [14] = A read-write training operation is completed; [13] = A read-write training request has been initiated; [12] = An error in writing the training result; [11] = An error in reading the sample training results; [10] = An error reading the training results; [9] = ODT is enabled, and CAS Latency is 3; [8] = DRAM initialization completed; [7] = Internal port data error; [6] = Internal port command error; [5] = Two errors in ECC are found many times; [4] = Two errors of ECC are found at a time; [3] = One bit error in multiple ECCs is found; [2] = One bit error in ECC is found at a time; [1] = Multiple accesses exceeding the physical space of memory are found; [0] = An access is found that exceeds the physical space of memory
TXPDLL	23: 8	0x0000	0x0-0xffff	DRAM TXPDLL parameter in cycles.
TDFI_WRLVL_EN	3: 0	0x0	0x0-0xf	Write Leveling is enabled until Write Leveling read operation is minimal Number of cycles

CONF_CTL_150 [63: 0] Offset: 0x960 DDR2 667: 0x0604000000000000

RDLAT_ADJ	60:56	0x00	0x0-0x1f	PHY read delay period
WRLAT_ADJ	51:48	0x0	0x0-0xf	PHY write delay period
SWLVL_START	40	0x0	0x0-0x1	Start operation in software leveling mode (write only)
SWLVL_LOAD	32	0x0	0x0-0x1	Load operation in software Leveling mode (write only)
SWLVL_EXIT	twenty four	0x0	0x0-0x1	Exit operation in software Leveling mode (write only)
INT_STATUS	18: 0	0x00	0x0-0x7ffff	Interrupt status (read only) [18] = OR of all interrupt bits; [17] = User initiated DLL synchronization end flag;

66

Page 79

[16] = DLL lock signal changed (locked and unlocked)

Switch);

				[15] = Error reading the sampling clock; [14] = A read-write training operation is completed; [13] = A read-write training request has been initiated; [12] = An error in writing the training result; [11] = An error in reading the sample training results; [10] = An error reading the training results; [9] = ODT is enabled, and CAS Latency is 3; [8] = DRAM initialization completed; [7] = Internal port data error; [6] = Internal port command error; [5] = Two errors in ECC are found many times; [4] = Two errors of ECC are found at a time; [3] = One bit error in multiple ECCs is found; [2] = One bit error in ECC is found at a time; [1] = Multiple accesses exceeding the physical space of memory are found; [0] = An access is found that exceeds the physical space of memory

CONF_CTL_151 [63: 0] Offset: 0x970 DDR2 667: 0x000000000003e805

CONCURRENTAP_WR_ONLY	56	0x0	0x0-0x1	After the write operation, whether to wait for the write to resume before the read operation To prevent concurrent auto-precharge operations
CKE_STATUS	48	0x0	0x0-0x1	indicates CKE_STATUS (read only)
INT_ACK	41:24	0x00	0x0-0x3ffff	Interrupt clear (write only) [17] = User initiated DLL synchronization end flag; [16] = DLL lock signal changed (locked and unlocked) Switch); [15] = Error reading the sampling clock; [14] = A read-write training operation is completed; [13] = A read-write training request has been initiated; [12] = An error in writing the training result; [11] = An error in reading the sample training results; [10] = An error reading the training results;

67

Page 80

				[9] = ODT is enabled, and CAS Latency is 3; [8] = DRAM initialization completed; [7] = Internal port data error; [6] = Internal port command error; [5] = Two errors in ECC are found many times; [4] = Two errors of ECC are found at a time; [3] = One bit error in multiple ECCs is found; [2] = One bit error in ECC is found at a time; [1] = Multiple accesses exceeding the physical space of memory are found; [0] = An access is found that exceeds the physical space of memory
DLL_RST_DELAY	23: 8	0x0000	0x0-0xffff	DLL reset minimum number of cycles

DLL_RST_ADJ_DLY	7: 0	0x00	0x0-0xff Configure the minimum number of cycles from DLL precision to DLL reset end

CONF_CTL_152 [63: 0] Offset: 0x980 DDR2 667: 0x0001010001000101

ZQ_IN_PROGRESS	56	0x0	0x0-0x1 indicates that ZQ operation is in progress (read only)
ZQCS_ROTATE	48	0x0	0x0-0x1 Enable ZQCS (short ZQ) rotation correction. When the bit is 0 , Every ZQCS request command will select all chips in the system Carry out correction, when the position is 1, the system is in each ZQCS Only one chip selection is corrected when the command comes, the system will take turns Correct all chip selections. ZQCS and REFRESH_PER_ZQ The parameter setting should be consistent with this bit
WRLVL_REG_EN	40	0x0	0x0-0x1 enable write wrlvl_delay register
WRLVL_EN	32	0x0	0x0-0x1 Enable the Write Leveling function of the controller
RESYNC_DLL_PER_ AREF_EN	twenty four	0x0	0x0-0x1 enable automatic DLL synchronization after each refresh command
RESYNC_DLL	16	0x0	0x0-0x1 initiate a DLL synchronization command (write only)
RDLVL_REG_EN	8	0x0	0x0-0x1 enable write rdlvl_delay register
RDLVL_GATE_REG_ EN	0	0x0	0x0-0x1 enable write rdlvl_gate_delay register

CONF_CTL_153 [63: 0] Offset: 0x990 DDR2 667: 0x0101020202010100

W2W_SAMECS_DLY	58:56	0x0	0x0-0x7 When additional delay from write command to write command for the same chip Clock cycles
W2W_DIFFCS_DLY	50:48	0x0	0x0-0x7 Additional delay clock from write command to write command for different chip select Number of cycles
TBST_INT_INTERVAL	42:40	0x0	0x0-0x7 DRAM burst interrupt interval period

68

Page 81

R2W_SAMECS_DLY	34:32	0x0	0x0-0x7 When the additional delay from the read command to the write command of the same chip select Clock cycles
R2W_DIFFCS_DLY	26:24	0x0	0x0-0x7 Additional delay clock for read commands to write commands for different chip selects Number of cycles
R2R_SAMECS_DLY	18:16	0x0	0x0-0x7 When additional delay from the read command of the same chip selection to the read command Clock cycles
R2R_DIFFCS_DLY	10: 8	0x0	0x0-0x7 Additional delay clock for read commands from different chip selects to read commands Number of cycles
AXI_ALIGNED_STRO BE_DISABLE	2: 0	0x0	0x0-0x7 When the transaction of the AXI port has one of the following characteristics: 1. The start address and end address of the transaction are aligned according to the user word The service length is one user word (128 bits), AXI Strobe is prohibited, Each bit corresponds to an AXI port. When set to 0, the write operation will be performed in the order of read-modify-write Row; When set to 1, the write operation as a standard write operation (not Read-modify-write sequence)

CONF_CTL_154 [63: 0] Offset: 0x9a0 DDR2 667: 0x0707040200060100

TDFI_WRLVL_LOAD	63:56	0x0	0x0-0xff Write Leveling delay number is valid until the first write Leveling Load command minimum clock cycles
TDFI_RDLVL_LOAD	55:48	0x0	0x0-0xff Read Leveling delay number is valid until the first read Leveling Load command minimum clock cycles
TCKESR	44:40	0x0	0x0-0x1f CKE is kept at the minimum level from the refresh to the exit Number of clock cycles

TCCD	36:32	0x0	0x0-0x1f CAS # to CAS # command delay
ADD_ODT_CLK_DIFFTYPE_DIFFCS	28:24	0x0	0x0-0x1f In order to meet the ODT timing, different commands for different chip selects Number of additional clock cycles inserted between
TRP_AB	19:16	0x0	0x0-0xf trp time for all banks
ADD_ODT_CLK_SAMETYPE_DIFFCS	11: 8	0x0	0x0-0xf In order to meet the ODT timing, the same type of commands for different chip selects Number of additional clock cycles inserted between
ADD_ODT_CLK_DIFFTYPE_SAMECS	3: 0	0x0	0x0-0xf In order to meet the ODT timing, the different commands of the same chip select Number of additional clock cycles inserted between

CONF_CTL_155 [63: 0] Offset: 0x9b0 DDR2 667: 0x0200010000000000

ZQINIT	59:48	0x0	0x0-0xfff The number of clock cycles required by the ZQ command during DRAM initialization
ZQCL	43:32	0x0	0x0-0xfff The number of clock cycles required for a normal ZQCL command, it should be equal to Half of ZQINIT

TDFI_WRLVL_WW	25:16	0x0	0x0-0x3ff Minimum number of clock cycles between two consecutive leveling commands
TDFI_RDLVL_RR	9: 0	0x0	0x0-0x3ff The minimum number of clock cycles between two consecutive read leveling commands

CONF_CTL_156 [63: 0] Offset: 0x9c0 DDR2 667: 0x0a52000000000000

MR0_DATA_0	62:48	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 0 of chip select 0
TDFI_PHYUPD_TYPE3	45:32	0x0	0x0-0x3fff Save DFI Tphyupd_type3 parameters (read only)
TDFI_PHYUPD_TYPE2	29:16	0x0	0x0-0x3fff Save DFI Tphyupd_type2 parameters (read only)
TDFI_PHYUPD_TYPE1	13: 0	0x0	0x0-0x3fff Save DFI Tphyupd_type1 parameters (read only)

CONF_CTL_157 [63: 0] Offset: 0x9d0 DDR2 667: 0x00440a520a520a52

MR1_DATA_0	62:48	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 1 of chip select 0
MR0_DATA_3	46:32	0x0	0x0-0x7fff corresponds to the configuration value of mode register 0 of chip select 3
MR0_DATA_2	30:16	0x0	0x0-0x7fff corresponds to the configuration value of mode register 0 of chip select 2
MR0_DATA_1	14: 0	0x0	0x0-0x7fff corresponds to the configuration value of mode register 0 of chip select 1

CONF_CTL_158 [63: 0] Offset: 0x9e0 DDR2 667: 0x0000004400440044

MR2_DATA_0	62:48	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 2 of chip select 0
MR1_DATA_3	46:32	0x0	0x0-0x7fff corresponds to the configuration value of mode register 1 of chip select 3
MR1_DATA_2	30:16	0x0	0x0-0x7fff corresponds to the configuration value of mode register 1 of chip select 2
MR1_DATA_1	14: 0	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 1 of chip select 1

CONF_CTL_159 [63: 0] Offset: 0x9f0 DDR2 667: 0x0000000000000000

MR3_DATA_0	62:48	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 3 of chip select 0
MR2_DATA_3	46:32	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 2 of chip select 3
MR2_DATA_2	30:16	0x0	0x0-0x7fff corresponds to the configuration value of mode register 2 of chip select 2
MR2_DATA_1	14: 0	0x0	0x0-0x7fff corresponds to the configuration value of mode register 2 of chip select 1

CONF_CTL_160 [63: 0] Offset: 0xa00 DDR2 667: 0x00ff000000000000

DFI_WRLVL_MAX_DELAY	63:48	0x0	0x0-0xffff The maximum delay line that Hareware Write leveling will use series
MR3_DATA_3	46:32	0x0	0x0-0x7fff corresponds to the configuration value of the mode register 3 of chip select 3
MR3_DATA_2	30:16	0x0	0x0-0x7fff corresponds to the configuration value of mode register 3 of chip select 2
MR3_DATA_1	14: 0	0x0	0x0-0x7fff corresponds to the configuration value of mode register 3 of chip select 1

CONF_CTL_161 [63: 0] Offset: 0xa10 DDR2 667: 0x0000000000000000

RDLVL_BEGIN_DELA Y_3	63:48	0x0	0x0-0xffff	In the third data group, from the first 1 to the Read Leveling 0 number of delay units
RDLVL_BEGIN_DELA Y_2	47:32	0x0	0x0-0xffff	In the second data set, from the first 1 to the Read Leveling 0 number of delay units
RDLVL_BEGIN_DELA Y_1	31:16	0x0	0x0-0xffff	In the first data group, from the first 1 to the Read Leveling 0 number of delay units
RDLVL_BEGIN_DELA Y_0	15: 0	0x0	0x0-0xffff	In the 0th data group, from the first 1 to the Read Leveling 0 number of delay units

CONF_CTL_162 [63: 0] Offset: 0xa20 DDR2 667: 0x0000000000000000

RDLVL_BEGIN_DELA Y_7	63:48	0x0	0x0-0xffff	In the 7th data group, from the first 1 to the Read Leveling 0 number of delay units
RDLVL_BEGIN_DELA Y_6	47:32	0x0	0x0-0xffff	In the 6th data group, from the first 1 to the Read Leveling 0 number of delay units
RDLVL_BEGIN_DELA Y_5	31:16	0x0	0x0-0xffff	In the 5th data group, from the first 1 to the Read Leveling 0 number of delay units
RDLVL_BEGIN_DELA Y_4	15: 0	0x0	0x0-0xffff	In the 4th data group, from the first 1 to the Read Leveling 0 number of delay units

CONF_CTL_163 [63: 0] Offset: 0xa30 DDR2 667: 0x0000000000000000

RDLVL_DELAY_2	63:48	0x0	0x0-0xffff	In the second data group, the delay unit used by Read Leveling number
RDLVL_DELAY_1	47:32	0x0	0x0-0xffff	In the first data group, the delay unit used by Read Leveling number
RDLVL_DELAY_0	31:16	0x0	0x0-0xffff	Delay group used by Read Leveling in data group 0 number
RDLVL_BEGIN_DELA Y_8	15: 0	0x0	0x0-0xffff	In the 8th data group, from the first 1 to the Read Leveling 0 number of delay units

CONF_CTL_164 [63: 0] Offset: 0xa40 DDR2 667: 0x0000000000000000

RDLVL _DELAY_6	63:48	0x0	0x0-0xffff	In the sixth data group, the delay unit used by Read Leveling number
RDLVL_DELAY_5	47:32	0x0	0x0-0xffff	Delay group used by Read Leveling in data group 5 number
RDLVL_DELAY_4	31:16	0x0	0x0-0xffff	In the fourth data group, the delay unit used by Read Leveling number
RDLVL_DELAY_3	15: 0	0x0	0x0-0xffff	In the third data group, the delay unit used by Read Leveling number

CONF_CTL_165 [63: 0] Offset: 0xa50 DDR2 667: 0x0000000000000000

RDLVL_END_DELAY_1	63:48	0x0	0x0-0xffff	In the first data group, from the first 0 to the Read Leveling 1 number of delay units
RDLVL_END_DELAY_0	47:32	0x0	0x0-0xffff	In the 0th data group, the first 0 to 1 number of delay units
RDLVL_DELAY_8	31:16	0x0	0x0-0xffff	In the 8th data group, the delay unit used by Read Leveling number
RDLVL_DELAY_7	15: 0	0x0	0x0-0xffff	In the seventh data group, the delay unit used by Read Leveling number

CONF_CTL_166 [63: 0] Offset: 0xa60 DDR2 667: 0x0000000000000000

RDLVL_END_DELAY_5	63:48	0x0	0x0-0xffff	In the 5th data group, from the first 0 to the Read Leveling 1 number of delay units
RDLVL_END_DELAY_4	47:32	0x0	0x0-0xffff	In the 4th data group, from the first 0 to the Read Leveling 1 number of delay units
RDLVL_END_DELAY_3	31:16	0x0	0x0-0xffff	In the third data group, from the first 0 to the Read Leveling 1 number of delay units
RDLVL_END_DELAY_2	15: 0	0x0	0x0-0xffff	In the second data group, from the first 0 to the Read Leveling 1 number of delay units

CONF_CTL_167 [63: 0] Offset: 0xa70 DDR2 667: 0x0000000000000000

RDLVL_GATE_DELAY_0	63:48	0x0	0x0-0xffff	In the 0th data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units
RDLVL_END_DELAY_8	47:32	0x0	0x0-0xffff	In the 8th data group, from the first 0 to the Read Leveling 1 number of delay units
RDLVL_END_DELAY_7	31:16	0x0	0x0-0xffff	In the 7th data group, from the first 0 to the Read Leveling 1 number of delay units
RDLVL_END_DELAY_6	15: 0	0x0	0x0-0xffff	In the 6th data group, from the first 0 to the Read Leveling 1 number of delay units

CONF_CTL_168 [63: 0] Offset: 0xa80 DDR2 667: 0x0000000000000000

RDLVL_GATE_DELAY_4	63:48	0x0	0x0-0xffff	In the fourth data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units
RDLVL_GATE_DELAY_3	47:32	0x0	0x0-0xffff	In the third data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units
RDLVL_GATE_DELAY_2	31:16	0x0	0x0-0xffff	In the second data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units

72

Page 85

RDLVL_GATE_DELAY_1	15: 0	0x0	0x0-0xffff	In the first data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units

CONF_CTL_169 [63: 0] Offset: 0xa90 DDR2 667: 0x0000000000000000

RDLVL_GATE_DELAY_8	63:48	0x0	0x0-0xffff	In the 8th data group, the delay from sampling timing to the rising edge of strobe signal Number of late units
RDLVL_GATE_DELAY_7	47:32	0x0	0x0-0xffff	In the seventh data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units
RDLVL_GATE_DELAY_6	31:16	0x0	0x0-0xffff	In the sixth data group, the delay from the sampling timing to the rising edge of the strobe signal Number of late units
RDLVL_GATE_DELAY_5	15: 0	0x0	0x0-0xffff	In the fifth data group, the delay from sampling timing to the rising edge of the strobe signal Number of late units

CONF_CTL_170 [63: 0] Offset: 0xaa0 DDR2 667: 0x0000ffff00000010

RDLVL_MIDPOINT_DELAY_0	63:48	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_0 and rdlvl_end_delay_0 Interval, otherwise, equal to rdlvl_delay_0 (read only)
RDLVL_MAX_DELAY	47:32	0x0	0x0-0xffff	Read Leveling Maximum number of delay lines
RDLVL_GATE_REFRESH_INTERVAL	31:16	0x0	0x0-0xffff	Maximum number of refresh commands between two automatic Gate Training (Should be set to 0)
RDLVL_GATE_MAX_DELAY	15: 0	0x0	0x0-0xffff	Maximum number of sampling delay lines

CONF_CTL_171 [63: 0] Offset: 0xab0 DDR2 667: 0x0000000000000000

RDLVL_MIDPOINT_DELAY_4	63:48	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_4 and rdlvl_end_delay_4 Interval, otherwise, equal to rdlvl_delay_4 (read only)
RDLVL_MIDPOINT_DELAY_3	47:32	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_3 and rdlvl_end_delay_3 Interval, otherwise, equal to rdlvl_delay_3 (read only)
RDLVL_MIDPOINT_DELAY_2	31:16	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_2 and rdlvl_end_delay_2 Interval, otherwise, equal to rdlvl_delay_2 (read only)
RDLVL_MIDPOINT_DELAY_1	15: 0	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_1 and rdlvl_end_delay_1 Interval, otherwise, equal to rdlvl_delay_1 (read only)

CONF_CTL_172 [63: 0] Offset: 0xac0 DDR2 667: 0x0000000000000000

RDLVL_MIDPOINT_D	63:48	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, equal to

ELAY_8				of rdlvl_begin_delay_8 and rdlvl_end_delay_8 Interval, otherwise, equal to rdlvl_delay_8 (read only)
RDLVL_MIDPOINT_DELAY_7	47:32	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_7 and rdlvl_end_delay_7 Interval, otherwise, equal to rdlvl_delay_7 (read only)
RDLVL_MIDPOINT_DELAY_6	31:16	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_6 and rdlvl_end_delay_6 Interval, otherwise, equal to rdlvl_delay_6 (read only)
RDLVL_MIDPOINT_DELAY_5	15: 0	0x0	0x0-0xffff	When the Hardware read leveling module is enabled, it is equal to of rdlvl_begin_delay_5 and rdlvl_end_delay_5 Interval, otherwise, equal to rdlvl_delay_5 (read only)

CONF_CTL_173 [63: 0] Offset: 0xad0 DDR2 667: 0x0000000000000000

RDLVL_OFFSET_DELAY_3	63:48	0x0	0x0-0xffff	The offset to the midpoint of Read Leveling in the third data group
RDLVL_OFFSET_DELAY_2	47:32	0x0	0x0-0xffff	2nd data set, offset to the midpoint of Read Leveling
RDLVL_OFFSET_DELAY_1	31:16	0x0	0x0-0xffff	The offset to the midpoint of Read Leveling in the first data group
RDLVL_OFFSET_DELAY_0	15: 0	0x0	0x0-0xffff	Offset to the midpoint of Read Leveling in the 0th data group

CONF_CTL_174 [63: 0] Offset: 0xae0 DDR2 667: 0x0000000000000000

RDLVL_OFFSET_DELAY_7	63:48	0x0	0x0-0xffff 7th data set, offset to the midpoint of Read Leveling
RDLVL_OFFSET_DELAY_6	47:32	0x0	0x0-0xffff 6th data set, offset to the midpoint of Read Leveling
RDLVL_OFFSET_DELAY_5	31:16	0x0	0x0-0xffff 5th data set, offset to the midpoint of Read Leveling
RDLVL_OFFSET_DELAY_4	15: 0	0x0	0x0-0xffff 4th data set, offset to the midpoint of Read Leveling

CONF_CTL_175 [63: 0] Offset: 0xaf0 DDR2 667: 0x0000000000000000

WRLVL_DELAY_1	63:48	0x0	0x0-0xffff In the first data group, control the number of write DQS via DLL delay
WRLVL_DELAY_0	47:32	0x0	0x0-0xffff In the 0th data group, control the number of write DQS via DLL delay
RDLVL_REFRESH_INTERVAL	31:16	0x0	0x0-0xffff Maximum number of refresh commands between two automatic read levels (Should be set to 0)
RDLVL_OFFSET_DEL	15: 0	0x0	0x0-0xffff 8th data set, offset to the midpoint of Read Leveling

74

Page 87

AY_8

CONF_CTL_176 [63: 0] Offset: 0xb00 DDR2 667: 0x0000000000000000

WRLVL_DELAY_5	63:48	0x0	0x0-0xffff In the 5th data group, control the number of write DQS via DLL delay
WRLVL_DELAY_4	47:32	0x0	0x0-0xffff In the 4th data group, control the number of write DQS via DLL delay
WRLVL_DELAY_3	31:16	0x0	0x0-0xffff 3rd data group, control the number of write DQS via DLL delay
WRLVL_DELAY_2	15: 0	0x0	0x0-0xffff In the second data group, control the number of write DQS via DLL delay

CONF_CTL_177 [63: 0] Offset: 0xb10 DDR2 667: 0x0000000000000000

WRLVL_REFRESH_INTERVAL	63:48	0x0	0x0-0xffff Maximum number of refresh commands between two automatic Write Leveling (Should be set to 0)
WRLVL_DELAY_8	47:32	0x0	0x0-0xffff 8th data group, control the number of write DQS via DLL delay
WRLVL_DELAY_7	31:16	0x0	0x0-0xffff 7th data group, control write DQS via DLL delay number
WRLVL_DELAY_6	15: 0	0x0	0x0-0xffff In the 6th data group, control the number of write DQS via DLL delay

CONF_CTL_178 [63: 0] Offset: 0xb20 DDR2 667: 0x00000c2d00000c2d

TDFI_RDLVL_RESP	63:32	0x0	0x0-0xffff Save DFI Trdlvl_resp time parameter
TDFI_RDLVL_MAX	31: 0	0x0	0x0-0xffff Save DFI Trdlvl_max time parameter

CONF_CTL_179 [63: 0] Offset: 0xb30 DDR2 667: 0x00000c2d00000c2d

TDFI_WRLVL_RESP	63:32	0x0	0x0-0xffff Save DFI Twrlvl_resp time parameter
TDFI_WRLVL_MAX	31: 0	0x0	0x0-0xffff Save DFI Twrlvl_max time parameter

75

Page 88

9 HyperTransport controller

In Godson 3, the HyperTransport bus is used to connect external devices and interconnect multiple chips. When used for peripheral connection, The user program can freely choose whether to support IO Cache consistency (set by the address window Uncache window, see x.4.2 Section). In the Cache consistency support mode, the internal DMA access of the IO device is transparent to the Cache layer, that is, no communication is required. The consistency is maintained through the program Cache instruction, but the consistency is automatically maintained by the hardware. When used for multi-chip interconnection, H Controller (the initial address is 0x0C00_0000_0000 – 0x0DFF_FFFF_FFFF) hardware automatically Cache consistency, but HT1 controller (initial address is 0x0E00_0000_0000 – 0x0FFF_FFFF_FFFF) is not supported Cache consistency between slices. See section x.5 for details.

The HyperTransport controller supports up to two-way 16-bit width, and the maximum operating frequency is 800Mhz. Automatically in the system After the connection is initialized and established, the user can modify the required operating frequency and width by modifying the configuration register in the protocol Change, re-initialize, see section x.1 for details.

The main characteristics of Godson 3 HyperTransport controller are as follows:

- Support 200/400 / 800Mhz
- Support 8/16 bit width
- Each HT controller (HT0 / HT1) can be configured as two 8-bit HT controllers
- The direction of bus control signals (including PowerOK, Rstn, LDT_Stopn) can be configured
- Peripheral DMA space Cache / Uncache can be configured
- HT0 controller can be configured as Cache consistency mode when used for multi-chip interconnection

9.1 HyperTransport hardware setup and initialization

HyperTransport bus is composed of transmission signal bus and control signal pins, etc. The following table gives HyperTransport bus related pins and their functions.

Table 9-1 HyperTransport bus related pin signals

Pin	name	description
{PCI_Config [7], PCI_Config [0]}	HT surrounding level Voltage control	00: Use HyperTransport peripheral signals as 1.8v signals, these signals include HT_8x2, HT_Mode, HT_Powerok, HT_Rstn, HT_Ldt_Stopn, HT_Ldt_Reqn. 01: reserved. 10: Use HyperTransport peripheral signals as 2.5v signals. 11: Use HyperTransport peripheral signals as 3.3v signals.
HT0_8x2	Bus width Set	1: Configure the 16-bit HyperTransport bus as two independent 8-bit buses, Controlled by two independent controllers, the address space is divided into HT0_Lo: address [40] = 0; HT0_Hi: address [40] = 1; 0: Use the 16-bit HyperTransport bus as a 16-bit bus, by HT0_Lo control, the address space is the address of HT0_Lo, namely address [40] = 0; HT0_Hi all signals are invalid.

76

Page 89

HT0_Lo_mode	Master mode 1: Set HT0_Lo to master mode, in this mode, bus control signals Driven by HT0_Lo, these control signals include HT0_Lo_Powerok, HT0_Lo_Rstn, HT0_Lo_Ldt_Stopn. In this mode, these controls The control signal can also be bidirectionally driven. At the same time this pin determines (negative) registration The initial value of the device "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 HyperTransport bus If the address is not hit in the receiving window of the controller, it will be regarded as a P2P request. Newly sent back to the bus, if this register is 1, there is no hit, it is regarded as an error Respond to false requests. 0: Set HT0_Lo to slave mode, in this mode, bus control signals, etc. Driven by the opposite device, these control signals include HT0_Lo_Powerok, HT0_Lo_Rstn, HT0_Lo_Ldt_Stopn. In this mode, these controls The control signal is driven by the other device. If it is not driven correctly, the Does not work correctly.	
HT0_Lo_Powerok	total line Powerok	HyperTransport bus Powerok signal, When HT0_Lo_Mode is 1, it is controlled by HT0_Lo; When HT0_Lo_Mode is 0, it is controlled by the opposite device.
HT0_Lo_Rstn	Bus Rstn	HyperTransport bus Rstn signal, When HT0_Lo_Mode is 1, it is controlled by HT0_Lo; When HT0_Lo_Mode is 0, it is controlled by the opposite device.
HT0_Lo_Ldt_Stopn	total line Ldt_Stopn	HyperTransport bus Ldt_Stopn signal, When HT0_Lo_Mode is 1, it is controlled by HT0_Lo; When HT0_Lo_Mode is 0, it is controlled by the opposite device.
HT0_Lo_Ldt_Reqn	total line Ldt_Reqn	HyperTransport bus Ldt_Reqn signal,
HT0_Hi_mode	Master mode 1: Set HT0_Hi to master mode, in this mode, bus control signals, etc. Driven by HT0_Hi, these control signals include HT0_Hi_Powerok, HT0_Hi_Rstn, HT0_Hi_Ldt_Stopn. In this mode, these controls The signal can also be bidirectionally driven. At the same time this pin determines (inverts) the register The initial value of "Act as Slave", when this register is 0, HyperTransport The Bridge bit in the packet on the bus is 1, otherwise it is 0. In addition, this deposit When the device is 0, if the requested address on the HyperTransport bus is not in control When the receiving window of the controller hits, it will be sent back to the bus as a P2P request, such as If this register is 1, there is no hit, it will respond as an error request. 0: Set HT0_Hi to slave mode, in this mode, bus control signals, etc. Driven by the counterpart device, these control signals include HT0_Hi_Powerok, HT0_Hi_Rstn, HT0_Hi_Ldt_Stopn. In this mode, these controls The signal is driven by the other device. If it is not driven correctly, the HT bus does not Works correctly.	
HT0_Hi_Powerok	total line Powerok	HyperTransport bus Powerok signal, When HT0_Lo_Mode is 1, it is controlled by HT0_Hi; When HT0_Lo_Mode is 0, it is controlled by the opposite device. When HT0_8x2 is 1, control the upper 8-bit bus; When HT0_8x2 is 0, it is invalid.
HT0_Hi_Rstn	Bus Rstn	HyperTransport bus Rstn signal, When HT0_Lo_Mode is 1, it is controlled by HT0_Hi; When HT0_Lo_Mode is 0, it is controlled by the opposite device. When HT0_8x2 is 1, control the upper 8-bit bus; When HT0_8x2 is 0, it is invalid.
HT0_Hi_Ldt_Stopn	total line Ldt_Stopn	HyperTransport bus Ldt_Stopn signal, When HT0_Lo_Mode is 1, it is controlled by HT0_Hi; When HT0_Lo_Mode is 0, it is controlled by the opposite device. When HT0_8x2 is 1, control the upper 8-bit bus; When HT0_8x2 is 0, it is invalid.
HT0_Hi_Ldt_Reqn	total line Ldt_Reqn	HyperTransport bus Ldt_Reqn signal, When HT0_8x2 is 1, control the upper 8-bit bus; When HT0_8x2 is 0, it is invalid.
HT0_Rx_CLKp [1: 0]	CLK [1: 0]	HyperTransport bus CLK signal

77

Page 90

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

The initialization of HyperTransport starts automatically after each reset is completed, and the HyperTransport bus after a cold start

Will automatically work at the lowest frequency (200Mhz) and the smallest width (8bit), and try to initiate the bus handshake. initialization

The status of completion can be read from the register "Init Complete" (see Section 9.5.2). After initialization, the total

The width of the line can be read from the registers "Link Width Out" and "Link Width In" (see Section 9.5.2). in

After the initialization is completed, the user can reset the registers "Link Width Out", "Link Width In" and "Link

Freq "programming, at the same time need to program the corresponding register of the other device, after the programming is completed, you need to reheat

Bit bus or re-initialize the bus by HT_Ldt_Stopn signal, so that the programmed value

Effective after initialization. After reinitialization, the HyperTransport bus will work at the new frequency and width. Need attention

Yes, the configuration of the devices at both ends of HyperTransport needs to be one-to-one correspondence, otherwise it will make the HyperTransport interface not

Can work normally.

9.2 HyperTransport protocol support

The HyperTransport bus supports most of the commands in the 1.03 protocol, and the extensions supported in multi-chip interconnect are consistent

Some extended instructions have been added to the sex protocol. For the two modes, the commands that the HyperTransport receiver can receive are

As shown in the table below. It should be noted that the atomic operation commands of the HyperTransport bus are not supported.

Table 9-2 Commands that the HyperTransport receiver can receive

coding	Channel	command	Standard mode	Extension (consistency)
000000-		NOP	Empty package or flow control	
000001	NPC	FLUSH	No operation	
x01xxx	NPC	Write	bit 5: 0-Nonposted	bit 5: Must be 1, POSTED
	or		1-Posted	
	PC		bit 2: 0 – Byte	bit 2: 0 – Byte
			1 – Doubleword	1 – Doubleword
			bit 1: Don't Care	bit 1: Don't Care
			bit 0: Don't Care	bit 0: must be 1
01xxxx	NPC	Read	bit 3: Don't Care	bit 3: Don't Care
			bit 2: 0 – Byte	bit 2: 0 – Byte
			1 – Doubleword	1 – Doubleword
			bit 1: Don't Care	bit 1: Don't Care
			bit 0: Don't Care	bit 0: must be 1

110000	R	RdResponse	Read operation returns	
110011	R	TgtDone	Write operation returns	
110100	PC	WrCoherent	----	Write command extension
110101	PC	WrAddr	----	Write address extension
111000	R	RespCoherent	----	Read response extension
111001	NPC	RdCoherent	----	Read command extension
111010	PC	Broadcast	No operation	
111011	NPC	RdAddr	----	Read address extension
111100	PC	FENCE	Guaranteed order relationship	
111111-		Sync / Error	Sync / Error	

For the sending end, the commands sent out in the two modes are shown in the following table.

Table 9-3 Commands to be sent out in two modes

coding	aisle	command	Standard mode	Extension (consistency)
000000-		NOP	Empty package or flow control	
x01x0x	NPC	Write	bit 5: 0-Nonposted	bit 5: Must be 1, POSTED
	or		1-Posted	
	PC		bit 2: 0 – Byte	bit 2: 0 – Byte
			1 – Doubleword	1 – Doubleword
			bit 0: must be 0	bit 0: must be 1
010x0x	NPC	Read	bit 2: 0 – Byte	bit 2: 0 – Byte
			1 – Doubleword	1 – Doubleword
			bit 0: Don't Care	bit 0: must be 1
110000	R	RdResponse	Read operation returns	
110011	R	TgtDone	Write operation returns	
110100	PC	WrCoherent	----	Write command extension
110101	PC	WrAddr	----	Write address extension
111000	R	RespCoherent	----	Read response extension
111001	NPC	RdCoherent	----	Read command extension

111011 NPC		RdAddr	----	Read address extension
111111	-	Sync / Error	Will only forward	

9.3 HyperTransport interrupt support

The HyperTransport controller provides 256 interrupt vectors, which can support Fix, Arbitor and other types of interrupts.

However, there is no support for hardware automatic EOI. For these two types of interrupts, the controller will automatically write after receiving

Into the interrupt register, and according to the interrupt mask register settings for the system interrupt controller interrupt notification. Concrete

For interrupt control, see the interrupt control register set in Section 5.

In addition, the controller specifically supports PIC interrupts to speed up this type of interrupt processing.

A typical PIC interrupt is completed by the following steps: ① The PIC controller sends a PIC interrupt request to the system; ② The system

Send the interrupt vector query to the PIC controller; ③ The PIC controller sends the interrupt vector number to the system; ④ The system clears the PIC controller

The corresponding interrupt on the controller. Only after the above four steps are completed, the PIC controller will issue the next interrupt to the system. for

Godson No.3 HyperTransport controller will automatically perform the first 3 steps and write the PIC interrupt vector to 256

Corresponding position in an interrupt vector. After the software system has processed the interrupt, it needs to perform step 4 processing, which is to

The PIC controller issues a clear interrupt. After that, the process of the next interrupt is started.

9.4 HyperTransport address window

9.4.1 HyperTransport space

In the Godson 3 processor, the default four HyperTransport address windows are as follows:

Table 9-4 Addresses of the default 4 HyperTransport address windows

Base address	End address	size	definition
0x0C00_0000_0000	0x0CFF_FFFF_FFFF	1 Tbytes	HT0_LO window
0x0D00_0000_0000	0x0DFF_FFFF_FFFF	1 Tbytes	HT0_HI window
0x0E00_0000_0000	0x0EFF_FFFF_FFFF	1 Tbytes	HT1_LO window
0x0F00_0000_0000	0x0FFF_FFFF_FFFF	1 Tbytes	HT1_HI window

By default (the system address window is not configured separately), the software

HyperTransport interface to access, in addition, the software can be configured through the address window on the crossbar

Set to access it with other address space (see section 2.5). Each HyperTransport interface has an internal 40-bit address space

The distribution of the specific address windows between is described in the following table.

The address window of HyperTransport interface protocol of Godson 3 processor is as follows:

Table 9-5 Address window distribution of HyperTransport interface of Loongson 3 processor

Base address	End address	size	definition
0x00_0000_0000	0xFC_FFFF_FFFF	1012 Gbytes	MEM space
0xFD_0000_0000	0xFD_F7FF_FFFF	3968 Mbytes	Keep
0xFD_F800_0000	0xFD_F8FF_FFFF	16 Mbytes	Interrupt
0xFD_F900_0000	0xFD_F90F_FFFF	1 Mbyte	PIC interrupt response
0xFD_F910_0000	0xFD_F91F_FFFF	1 Mbyte	system message
0xFD_F920_0000	0xFD_FAFF_FFFF	30 Mbytes	Keep
0xFD_FB00_0000	0xFD_FBFF_FFFF	16 Mbytes	HT controller configuration space
0xFD_FC00_0000	0xFD_FDFF_FFFF	32 Mbytes	I / O space
0xFD_FE00_0000	0xFD_FFFF_FFFF	32 Mbytes	HT bus configuration space

0xFE_0000_0000	0xFF_FFFF_FFFF	8 Gbytes	Keep

9.4.2 Internal window configuration of HyperTransport controller

Godson 3 processor HyperTransport interface provides a variety of rich address windows for users to use, including

As follows:

Table 9-6 Address window provided in HyperTransport interface of Loongson 3 processor

Address window	Window number	accept bus	effect	Remarks
Receive window (See window configuration 9.5.4)	3	HyperTransport	Determine whether to receive HyperTransport Visits sent on the bus ask.	When in main bridge mode (ie configuration register Act_as_slave is 0), only falls in Access in these address windows will be internal The bus responds, other visits will be recognized Send back for P2P access HyperTransport bus; in the design When in standby mode (that is, in the configuration register act_as_slave is 1), only falls here Access in these address windows will be Received and processed by the line, other visits will be According to the agreement to return an error.
Post window (See window configuration 9.5.8)	2	Internal bus	Determine if it will be internal total line Correct HyperTransport Bus write access Post Write	Internal write visits that fall in these address spaces Question will be as Post Write. Post Write: HyperTransport protocol In this kind of write access does not need to wait for writing In response, that is, the controller sends to the bus After this write access will enter the processor Row write access complete response.
Prefetch window (See window configuration 9.5.9)	2	Internal bus	Determine whether to receive Department 's Cache access Fetch access.	When the processor cores are executed out of order, the total Issue some guess read access or fetch Access, this access for some IO space it is wrong. By default, this Access to the HT controller will return directly without Visit the HyperTransport bus ask. Through these windows you can enable This type of access to the HyperTransport bus ask.
Uncache window (See window configuration Section 9.5.10)	2	HyperTransport	Determine whether to HyperTransport Access operations on the bus For internal Uncache access	IO DMA inside Loongson 3 processor Access, in the case of Cache side Access is determined by the secondary cache In order to maintain its IO consistency information. And through the configuration of these windows, you can make Access hits in these windows Uncache way to directly access memory, Without maintaining its IO consistency letter through hardware interest.

9.5 Configuration Register

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

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

Table 9-7 All software visible registers in this module

Offset address	name	description
0x30		
0x34		
0x38		
0x3c	Bridge Control	Bus Reset Control
0x40		Command, Capabilities Pointer, Capability ID
0x44	Capability Registers	Link Config, Link Control
0x48		Revision ID, Link Freq, Link Error, Link Freq Cap
0x4c		Feature Capability
0x50	Custom register	MISC
0x54	Receive Diagnostic Register	The receiver samples the received cad and ctl values
0x58	Interrupt routing register	Interrupt routing control register (LS3A1000E and above)
0x5c	Receive buffer register	sets the initial value of the receive buffer (LS3A1000E and above)
0x60	Receive address window Configuration register	HT bus receive address window 0 enable (external access)
0x64		HT bus receive address window 0 base address (external access)
0x68		HT bus receive address window 1 enable (external access)
0x6c		HT bus receive address window 1 base address (external access)
0x70		HT bus receive address window 2 enable (external access)
0x74		HT bus receive address window 2 base address (external access)
0x78		
0x7c		
0x80	Interrupt vector register	HT bus interrupt vector register [31: 0]
0x84		HT Bus Interrupt Vector Register [63:32]
0x88		HT Bus Interrupt Vector Register [95:64]
0x8c		HT bus interrupt vector register [127: 96]
0x90		HT bus interrupt vector register [159: 128]
0x94		HT Bus Interrupt Vector Register [191: 160]
0x98		HT Bus Interrupt Vector Register [223: 192]
0x9C		HT Bus Interrupt Vector Register [255: 224]
0xA0	Interrupt enable register	HT bus interrupt enable register [31: 0]
0xA4		HT bus interrupt enable register [63:32]
0xA8		HT bus interrupt enable register [95:64]
0xAC		HT bus interrupt enable register [127: 96]
0xB0		HT bus interrupt enable register [159: 128]
0xB4		HT bus interrupt enable register [191: 160]
0xB8		HT bus interrupt enable register [223: 192]
0xBC		HT bus interrupt enable register [255: 224]
0xC0	Interrupt Discovery & Configuration	Interrupt Capability
0xC4		DataPort
0xC8		IntrInfo [31: 0]
0xCC		IntrInfo [63:32]
0xD0	POST address window Configuration register	HT bus POST address window 0 enable (internal access)
0xD4		HT bus POST address window 0 base address (internal access)
0xD8		HT bus POST address window 1 enable (internal access)
0xDC		HT bus POST address window 1 base address (internal access)
0xE0	Prefetchable address window Configuration register	HT bus can be prefetched address window 0 enabled (internal access)
0xE4		HT bus prefetchable address window 0 base address (internal access)

0xE8		HT bus prefetch address window 1 enabled (internal access)
0xEC		Ht bus prefetchable address window 1 base address (internal access)
0xF0	Uncache address window Configuration register	HT bus Uncache address window 0 enable (internal access)
0xF4		HT bus Uncache address window 0 base address (internal access)
0xF8		HT bus Uncache address window 1 is enabled (internal access)
0xFC		HT bus Uncache address window 1 base address (internal access)

The specific meaning of each register is as follows:

9.5.1 Bridge Control

Offset: 0x3C

Reset value: 0x00200000

Name: Bus Reset Control

Bit field	Bit field name	Bit width	reset value	Visit	description
31:23	Reserved	4	0x0		Keep
twenty two	Reset	12	0x0	R / W	bus reset control: 0-> 1: Set HT_RSTn to 0, reset the bus 1-> 0: HT_RSTn is set to 1, the bus is reset
21:18	Reserved	4	0x0		Keep
17	B_interleave	1	0x0		Whether the R / W write response channel allows out-of-order execution (LS3A1000E and Previous version) When using multi-chip interconnect mode, it must be set to 1
16	Nop_interleave	1	0x0		Whether the R / W flow control channel allows out-of-order execution (LS3A1000E and above version) When using multi-chip interconnect mode, it must be set to 1
15: 0	Reserved	16	0x0		Keep

9.5.2 Capability Registers

Offset: 0x40

Reset value: 0x20010008

Name: Command, Capabilities Pointer, Capability ID

Bit field	Bit field name	Bit width	reset value	Visit	description
31:29	HOST / Sec	3	0x1	R	Command format is HOST / Sec
28:27	Reserved	2	0x0	R	Keep
26	Act as Slave	1	0x0 / 0x1	R / W	HOST / SLAVE mode The initial value is determined by the pin HOSTMODE HOSTMODE pull-up: 0 HOSTMODE drop-down: 1
25	Reserved	1	0x0		Keep
twenty four	Host Hide	1	0x0		Whether R / W prohibits register access from HT bus
twenty three	Reserved	1	0x0		Keep
22:18	Unit ID	5	0x0		In R / W HOST mode: can be used to record the number of IDs used In SLAVE mode: record your own Unit ID
17	Double Ended	1	0x0	R	No dual HOST mode

16	Warm Reset	1	0x1	R	Bridge Control uses warm reset in reset
15: 8	Capabilities Pointer 8		0xa0	R	Next Cap register offset address
7: 0	Capability ID	8	0x08	R	HyperTransport capability ID

Offset: 0x44

Reset value: 0x00112000

Name: Link Config, Link Control

Bit field	Bit field name	Bit width	reset value	Visit	description
31	Reserved	1	0x0		Keep
30:28	Link Width Out	3	0x0	R / W	Transmitter width The value after cold reset is the maximum width of the current connection, write this post The value of the register will be the next warm reset or HT Effective after Disconnect 000: 8-bit mode 001: 16-bit mode
27	Reserved	1	0x0		Keep
26:24	Link Width In	3	0x0	R / W	receiver width The value after cold reset is the maximum width of the current connection, write this post The value of the register will be the next warm reset or HT Effective after Disconnect
twenty three	Dw Fc out	1	0x0	R	The sender does not support double-word flow control
22:20	Max Link Width out	3	0x1	R	The maximum width of the sending end of the HT bus: 16bits
19	Dw Fc In	1	0x0	R	The receiver does not support double-word flow control
18:16	Max Link Width In	3	0x1	R	Maximum width of HT bus receiving end: 16bits

15:14	Reserved	2	0x0		Keep
13	LDTSTOP # Tristate Enable	1	0x1	R / W	When the HT bus enters the HT Disconnect state, is it off? Close HT PHY 1: Close 0: do not close
12:10	Reserved	3	0x0		Keep
9	CRC Error (hi)	1	0x0	R / W	CRC error in the upper 8 bits
8	CRC Error (lo)	1	0x0		CRC error occurred in the lower 8 bits of R / W
7	Trans off	1	0x0	R / W	HT PHY shutdown control When in 16-bit bus operating mode 1: Turn off high / low 8-bit HT PHY 0: enable the low 8-bit HT PHY, The upper 8-bit HT PHY is controlled by bit 0
6	End of Chain	0	0x0	R	HT bus end
5	Init Complete	1	0x0	R	Whether the HT bus initialization is completed
4	Link Fail	1	0x0	R	Indicates connection failure
3: 2	Reserved	2	0x0		Keep
1	CRC Flood Enable	1	0x0	R / W	Whether to flood the HT bus when a CRC error occurs
0	Trans off (hi)	1	0x0		When R / W uses 16-bit HT bus to run 8-bit protocol, High 8-bit PHY shutdown control 1: Turn off the upper 8-bit HT PHY 0: enable high 8-bit HT PHY

Offset: 0x48

Reset value: 0x80250023

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31:16	Link Freq Cap	16	0x0025	R	Supported HT bus frequency 200Mhz, 400Mhz, 800Mhz,
15:14	Reserved	2	0x0		Keep
13	Over Flow Error	1	0x0	R	HT bus packet overflow
12	Protocol Error	1	0x0	R / W	protocol error, Refers to an unrecognized command received on the HT bus
11: 8	Link Freq	4	0x0	R / W	HT bus operating frequency The value written to this register will be the next warm reset or HT Effective after Disconnect 0000: 200M 0010: 400M 0101: 800M
7: 0	Revision ID	8	0x23	R / W	version number: 1.03

Offset: 0x4C

Reset value: 0x00000002

Name: Feature Capability

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 9	Reserved	25	0x0		Keep
8	Extended Register 1		0x0	R	No
7: 4	Reserved	3	0x0		Keep
3	Extended CTL Time 1		0x0	R	No need
2	CRC Test Mode	1	0x0	R	not support
1	LDTSTOP #	1	0x1	R	Support LDTSTOP #
0	Isochronous Mode 1		0x0	R	not support

9.5.3 Custom register

Offset: 0x50

Reset value: 0x00904321

Name: MISC

Bit field	Bit field name	Bit width	reset value	Visit	description
31	Reserved	1	0x0		Keep

30	Ldt Stop Gen	1	0x0	R / W puts the bus into LDT DISCONNECT mode The correct method is: 0-> 1
29	Ldt Req Gen	1	0x0	R / W wake up HT bus from LDT DISCONNECT, set LDT_REQ_n The correct way is to set 0 first and then set 0: 0-> 1 In addition, direct read and write requests to the bus can also be automatically Wake up bus

Bit field	Bit field name	Bit width	reset value	Visit description
28:24	Interrupt Index	5	0x0	R / W to which interrupts other than standard interrupts are redirected In the interrupt vector (including SMI, NMI, INIT, INTA, INTB, INTC, INTD) A total of 256 interrupt vectors, this register indicates the interrupt The upper 5 bits of the vector, the internal interrupt vector is as follows: 000: SMI 001: NMI 010: INIT 011: Reservered 100: INTA 101: INTB 110: INTC 111: INTD
twenty three	Dword Write	1	0x1	R / W For 32/64/128/256 bit write access, whether to use Dword Write command format 1: Use Dword Write 0: Use Byte Write (with MASK)
twenty two	Coherent Mode	1	0x0	R Whether it is processor consistency mode Determined by pin ICCC_EN
twenty one	Not Care Seqid	1	0x0	Does R / W don't care about HT order relationship
20	Not Axi2Seqid	1	0x1	R Whether to convert the commands on the Axi bus to different SeqIDs, If not converted, all read and write commands will be used Fixed ID number in Fixed Seqid 1: No conversion 0: conversion
19:16	Fixed Seqid	4	0x0	R / W When Not Axi2Seqid is valid, configure the Seqid
15:12	Priority Nop	4	0x4	R / W HT bus Nop flow control packet priority
11: 8	Priority NPC	4	0x3	R / W Non Post channel read and write priority
7: 4	Priority RC	4	0x2	R / W Response channel reading and writing priority
3: 0	Priority PC	4	0x1	R / W Post channel read and write priority 0x0: highest priority 0xF: lowest priority The priority of each channel is changed according to time. High priority strategy, the group register is used to configure each channel 'S initial priority

9.5.4 Receive diagnostic register

Offset: 0x54

Reset value: 0x00000000

Name: Receive diagnostic register

Bit field	Bit field name	Bit width	reset value	Visit description
0	Sample_en	1	0x0	R / W enables cad and ctl for sampling input 0x0: prohibited 0x1: enable
15: 8	rx_ctl_catch	twenty four	0x0	R / W saves the input ctl obtained by sampling (0, 2, 4, 6) Four phases corresponding to CTL0 sampling (1, 3, 5, 7) Four phases corresponding to CTL1 sampling
31:16	rx_cad_phase_0	twenty four	0x0	R / W save the input CAD [15: 0] value obtained by sampling

9.5.5 Interrupt routing mode selection register (LS3A1000E and above)

Offset: 0x58

Reset value: 0x00000000

Name: Interrupt routing mode selection register

Bit field	Bit field name	Bit width	reset value	Visit description
9: 8	ht_int_stripe	2	0x0	R / W corresponds to 3 interrupt routing methods, as described in 9.5.7 Break vector register 0x0: ht_int_stripe_1 0x1: ht_int_stripe_2 0x2: 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

Bit field	Bit field name	Bit width	reset value	Visit description
27:24	rx_buffer_r_data	4	0x7	R / W Receive buffer read data buffer initialization information
23:20	rx_buffer_npc_data	4	0x7	R / W receive buffer npc data buffer initialization information
19:16	rx_buffer_pc_data	4	0x7	R / W receive buffer pc data buffer initialization information
15:12	rx_buffer_b_cmd	4	0x8	R / W receive buffer initialization command of the receive command buffer interest
11: 8	rx_buffer_r_cmd	4	0x8	R / W receive buffer read command initialization information
7: 4	rx_buffer_npc_cmd	4	0x8	R / W receive buffer npc command buffer initialization information
3: 0	rx_buffer_pc_cmd	4	0x8	R / W receive buffer 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 | ADDR & ~ MASK: ADDR

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

The actual number of digits in 0 indicates the size of the address window.

The address in this window is the address received on the HT bus. The HT address falling in this window will be sent to the CPU,

Commands at 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)

Bit field	Bit field name	Bit width	reset value	Visit description

87

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_rx_image0_en	1	0x0	R / W HT bus receives address window 0, enable signal
30	ht_rx_image0_ trans_en	1	0x0	R / W HT bus receives address window 0, mapping enable signal
29: 0	ht_rx_image0_ trans [53:24]	16	0x0	R / W HT bus receives the address window 0, the mapped address [53:24], For LS3A1000D and below, [29:23] is fixed at 0

Offset: 0x64

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_rx_image0_ base [39:24]	16	0x0	R / W HT bus receive address window 0, address base address [39:24]
15: 0	ht_rx_image0_ mask [39:24]	16	0x0	R / W HT bus receive address window 0, address masked [39:24]

Offset: 0x68

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_rx_image1_en 1		0x0	R / W HT bus receives address window 1, enable signal
30	ht_rx_image1_ trans_en	1	0x0	R / W HT bus receives address window 1, map enable signal
29: 0	ht_rx_image1_ trans [53:24]	16	0x0	R / W HT bus receives address window 1, the mapped address [53:24], For LS3A1000D and below, [29:23] is fixed at 0

Offset: 0x6c

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_rx_image1_ base [39:24]	16	0x0	R / W HT bus receive address window 1, address base address [39:24]
15: 0	ht_rx_image1_ mask [39:24]	16	0x0	R / W HT bus receive address window 1, address masked [39:24]

Offset: 0x70

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_rx_image2_en 1		0x0	R / W HT bus receives address window 2, enable signal
30	ht_rx_image2_ trans_en	1	0x0	R / W HT bus receives address window 2, map enable signal
29: 0	ht_rx_image2_ trans [53:24]	16	0x0	R / W HT bus receive address window 2, the translated address [53:24], For LS3A1000D and below, [29:23] is fixed at 0

Offset: 0x74

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_rx_image2_ base [39:24]	16	0x0	R / W HT bus receive address window 2, address base address [39:24]
15: 0	ht_rx_image2_ mask [39:24]	16	0x0	R / W HT bus receive address window 2, address masked [39:24]

9.5.8 Interrupt Vector Register

There are 256 interrupt vector registers in total, of which Fixed and Arbitrated are removed on the HT bus, and the PIC interrupt is directly reflected Into the 256 interrupt vectors, other interrupts, such as SMI, NMI, INIT, INTA, INTB, INTC, INTD It can be mapped to any 8-bit interrupt vector through [28:24] of register 0x50, the order of mapping is {INTD INTC, INTB, INTA, 1'b0, INIT, NMI, SMI}. At this time, the corresponding value of the interrupt vector is {Interrupt Index, Partial vector [2: 0]}.

For LS3A1000E and above, 256 interrupt vectors are selected according to the interrupt routing method.

The same mapping to different interrupt lines, the specific mapping method is:

ht_int_stripe_1:

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

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

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

[192,193,194,195 ... 255] corresponds to interrupt line 3 / HT HI corresponds to interrupt line 7

ht_int_stripe_2:

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

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

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

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

ht_int_stripe_4:

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

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

[2,6,10,14 ... 254] corresponds to interrupt line 2 / HT HI corresponds to interrupt line 6

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

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

For LS3A1000D and below, only ht_int_stripe_1 can be used.

89

Page 102

Offset: 0x80

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [31: 0]	32	0x0	R / W HT bus interrupt vector register [31: 0], Corresponding to interrupt line 0 / HT HI Corresponding to interrupt line 4

Offset: 0x84

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [63:32]	32	0x0	R / W HT bus interrupt vector register [63:32], Corresponding to interrupt line 0 / HT HI Corresponding to interrupt line 4

Offset: 0x88

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [95:64]	32	0x0	R / W HT bus interrupt vector register [95:64], Corresponding to interrupt line 1 / HT HI Corresponding to interrupt line 5

Offset: 0x8c

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [127: 96]	32	0x0	R / W HT bus interrupt vector register [127: 96], Corresponding to interrupt line 1 / HT HI Corresponding to interrupt line 5

Offset: 0x90

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [159: 128]	32	0x0	R / W HT bus interrupt vector register [159: 128], Corresponding to interrupt line 2 / HT HI Corresponding to interrupt line 6

Offset: 0x94

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [191: 160]	32	0x0	R / W HT bus interrupt vector register [191: 160], Corresponding to interrupt line 2 / HT HI Corresponding to interrupt line 6

Offset: 0x98

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [223: 192]	32	0x0	R / W HT bus interrupt vector register [223: 192], Corresponding to interrupt line 3 / HT HI Corresponding to interrupt line 7

Offset: 0x9c

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_case [255: 224]	32	0x0	R / W HT bus interrupt vector register [255: 224], Corresponding to interrupt line 3 / HT HI Corresponding to interrupt line 7

9.5.9 Interrupt enable register

A total of 256 interrupt enable registers correspond to the interrupt vector registers. Set 1 to enable the corresponding interrupt, set 0 to Interrupt masking.

Offset: 0xa0

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_mask [31: 0]	32	0x0	R / W HT bus interrupt enable register [31: 0], Corresponding to interrupt line 0 / HT HI Corresponding to interrupt line 4

Offset: 0xa4

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_mask [63:32]	32	0x0	R / W HT bus interrupt enable register [63:32], Corresponding to interrupt line 0 / HT HI Corresponding to interrupt line 4

Offset: 0xa8

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_mask [95:64]	32	0x0	R / W HT bus interrupt enable register [95:64], Corresponding to interrupt line 1 / HT HI Corresponding to interrupt line 5

Offset: 0xac

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31: 0	Interrupt_mask [127: 96]	32	0x0	R / W HT bus interrupt enable register [127: 96], Corresponding to interrupt line 1 / HT HI Corresponding to interrupt line 5

Offset: 0xb0

Reset value: 0x00000000

Page 104

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 0	Interrupt_mask [159: 128]	32	0x0	R / W	HT bus interrupt enable register [159: 128], Corresponding to interrupt line 2 / HT HI Corresponding to interrupt line 6

Offset: 0xb4

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 0	Interrupt_mask [191: 160]	32	0x0	R / W	HT bus interrupt enable register [191: 160], Corresponding to interrupt line 2 / HT HI Corresponding to interrupt line 6

Offset: 0xb8

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 0	Interrupt_mask [223: 192]	32	0x0	R / W	HT bus interrupt enable register [223: 192], Corresponding to interrupt line 3 / HT HI Corresponding to interrupt line 7

Offset: 0xbc

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 0	Interrupt_mask [255: 224]	32	0x0	R / W	HT bus interrupt enable register [255: 224], Corresponding to interrupt line 3 / HT HI Corresponding to interrupt line 7

9.5.10 Interrupt Discovery & Configuration

Offset: 0xc0

Reset value: 0x80000008

Name: Interrupt Capability

Bit field	Bit field name	Bit width	reset value	Visit	description
31:24	Capabilities Pointer 8		0x80	R	Interrupt discovery and configuration block
23:16	Index	8	0x0	R / W	Read register offset address
15: 8	Capabilities Pointer 8		0x0	R	Capabilities Pointer
7: 0	Capability ID	8	0x08	R	Hypertransport Capablity ID

Offset: 0xc4

Reset value: 0x00000000

Name: Dataport

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 0	Dataport	32	0x0	R / W	When the previous register Index is 0x10, this register is read and written The result is the 0xa8 register, otherwise 0xac

Offset: 0xc8

Reset value: 0xF8000000

Page 105

Name: IntrInfo [31: 0]

Bit field	Bit field name	Bit width	reset value	Visit	description
31:24	IntrInfo [31:24]	32	0xF8	R	Keep
23: 2	IntrInfo [23: 2]	twenty two	0x0	R / W	IntrInfo [23: 2], when the PIC interrupt is issued, the value of IntrInfo Used to represent interrupt vector
1: 0	Reserved	2	0x0	R	Keep

Offset: 0xcc

Reset value: 0x00000000

Name: IntrInfo [63:32]

Bit field	Bit field name	Bit width	reset value	Visit	description
31: 0	IntrInfo [63:32]	32	0x0	R	Keep

9.5.11 POST 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 high bit of MASK should be all 1s and the low bit should be all 0s. 0 in MASK

The actual number of bits indicates the size of the address window.

The address in this window is the address received on the AXI bus. All write accesses that fall in this window will immediately be on the AXI B channel

Return and send to the HT bus in the format of POST WRITE command. Instead of writing requests in this window, NONPOST

WRITE is sent to the HT bus, and waits 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)

Bit field	Bit field name	Bit width	reset value	Visit	description
31	ht_post0_en	1	0x0	R / W	HT bus POST address window 0, enable signal
30:23	Reserved	15	0x0		Keep
15: 0	ht_post0_trans [39:24]	16	0x0	R / W	HT bus POST address window 0, the translated address [39:24]

Offset: 0xd4

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31:16	ht_post0_base [39:24]	16	0x0	R / W	HT bus POST address window 0, address base address [39:24]
15: 0	ht_post0_mask [39:24]	16	0x0	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)

Bit field	Bit field name	Bit width	reset value	Visit	description

Bit field	Bit field name	Bit width	reset value	Visit	description
31	ht_post1_en	1	0x0	R / W	HT bus POST address window 1, enable signal
30:23	Reserved	15	0x0		Keep
15: 0	ht_post1_trans [39:24]	16	0x0	R / W	HT bus POST address window 1, the translated address [39:24]

Offset: 0xdc

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit	description
31:16	ht_post1_base [39:24]	16	0x0	R / W	HT bus POST address window 1, address base address [39:24]
15: 0	ht_post1_mask [39:24]	16	0x0	R / W	HT bus POST address window 1, address masked [39:24]

9.5.12 Prefetch 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 high bit of MASK should be all 1s and the low bit should be all 0s. 0 in MASK

The actual number of bits indicates the size of the address window.

The address in this window is the address received on the AXI bus. The instruction fetch instruction that falls in this window, CACHE access will be issued To the HT bus, other fetch instructions or CACHE access will not be sent to the HT bus, but will return immediately.

Read command will return the corresponding number of invalid read data.

Offset: 0xe0

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_prefetch0_en	1	0x0	R / W HT bus can prefetch address window 0, enable signal
30:23	Reserved	15	0x0	Keep
15: 0	ht_prefetch0_trans [39:24]	16	0x0	R / W HT bus can prefetch the address window 0, the translated address [39:24]

Offset: 0xe4

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_prefetch0_ base [39:24]	16	0x0	R / W HT bus can pre-fetch address window 0, address base address [39:24] Bit address
15: 0	ht_prefetch0_ mask [39:24]	16	0x0	R / W HT bus can prefetch address window 0, address masked [39:24]

Offset: 0xe8

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_prefetch1_en	1	0x0	R / W HT bus can prefetch address window 1, enable signal
30:23	Reserved	15	0x0	Keep
15: 0	ht_prefetch1_ trans [39:24]	16	0x0	R / W HT bus can pre-fetch the address window 1, the translated address [39:24]

Offset: 0xec

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_prefetch1_ base [39:24]	16	0x0	R / W HT bus can prefetch address window 1, address base address [39:24]
15: 0	ht_prefetch1_ mask [39:24]	16	0x0	R / W HT bus can prefetch address window 1, address masked [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 | ADDR & ~ MASK: ADDR

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

The actual number of bits indicates the size of the address window.

The address in this window is the address received on the HT bus. Read and write commands that fall into this window address will not be sent to the second level

CACHE does not invalidate the first-level CACHE, but is sent directly to memory or other address spaces.

This means that the read and write commands in this address window will not maintain the CACHE consistency of IO. The main pin of this window

For some operations that will not hit in CACHE and can improve the efficiency of questions, such as access to video memory.

Offset: 0xf0

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_uncache0_en	1	0x0	R / W HT bus uncache address window 0, enable signal
30	ht_uncache0_ trans_en	1	0x0	R / W HT bus uncache address window 1, mapping enable signal
29: 0	ht_uncache0_ trans [53:24]	16	0x0	R / W HT bus uncache address window 0, the translated address

[53:24], for LS3A1000D and below, [29:23]
Fixed at 0

Offset: 0xf4

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_uncache0_ base [39:24]	16	0x0	R / W HT bus uncache address window 0, address base address [39:24]
15: 0	ht_uncache0_ mask [39:24]	16	0x0	R / W HT bus uncache address window 0, address masked [39:24]

Offset: 0xf8

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31	ht_uncache1_en	1	0x0	R / W HT bus uncache address window 1, enable signal
30	ht_uncache1_ trans_en	1	0x0	R / W HT bus uncache address window 1, mapping enable signal
29: 0	ht_uncache1_ trans [53:24]	16	0x0	R / W HT bus uncache address window 1, the translated address [53:24], for LS3A1000D and below, [29:23] Fixed at 0

Offset: 0xfc

Reset value: 0x00000000

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

Bit field	Bit field name	Bit width	reset value	Visit description
31:16	ht_uncache1_ base [39:24]	16	0x0	R / W HT bus uncache address window 1, address base address [39:24]
15: 0	ht_uncache1_ mask [39:24]	16	0x0	R / W HT bus uncache address window 1, address masked [39:24]

9.5.14 **HyperTransport** bus configuration space access method

The protocol of the HyperTransport interface software layer is basically the same as the PCI protocol. Configuration access is directly related to the underlying protocol Off, the method of access may be slightly different. As shown in Table 9-5, the configuration access space is located at the address

0xFD_FE00_0000 to 0xFD_FFFF_FFFFh. For configuration access in the PCI protocol, in Loongson No. 3, such as
The next realization.

Type 0:

Type 1:

Figure 9-1 HT protocol configuration access in Loongson 3

9.6 HyperTransport multiprocessor support

Loongson No. 3 processor uses HyperTransport interface for multi-processor interconnection, and can automatically maintain 4 hardware
Consistency request between chips. The following provides two multiprocessor interconnection methods:

Four piece Loongson No. 3 interconnection structure

The four CPUs are connected in pairs to form a ring structure. Each CPU uses two 8-bit controllers of HT0 to connect with two adjacent chips,
Among them, HTx_LO is the master device, and HTx_HI is the slave device, and the interconnection structure as shown below is obtained:

Page 109

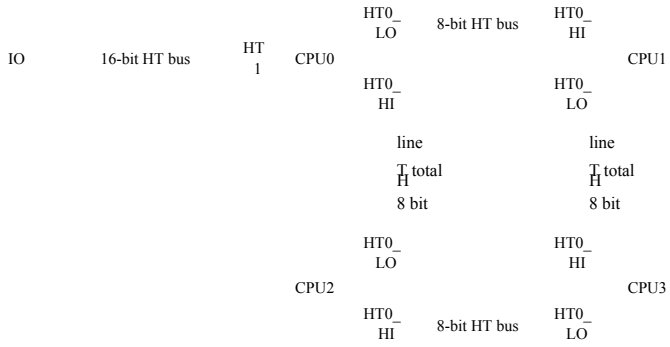


Figure 9-2 Four-chip Loongson No. 3 interconnection structure

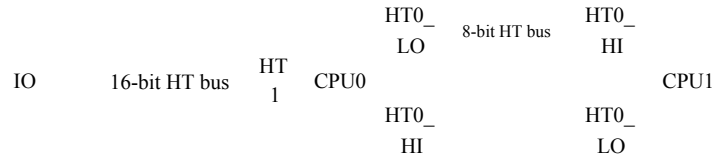
Loongson 3 interconnection routing

Loongson No. 3 interconnection routing adopts simple XY routing method. That is, when routing, X then Y, taking four chips as an example, ID
The numbers are 00, 01, 10, and 11, respectively. If a request is made from 11 to 00, it is a route from 11 to 00, first go to X
Direction, from 11 to 10, then Y direction, from 10 to 00. When the response to the request returns 11 from 00, the route
First go in the X direction, from 00 to 01, then go in the Y direction, from 01 to 11. As you can see, these are two different routes
line. Due to the characteristics of this algorithm, we will adopt different methods when constructing the interconnection of two chips.

Two piece Loongson No. 3 interconnection structure

Due to the nature of the fixed routing algorithm, we have two different methods when constructing the interconnection of two chips. The first is to adopt 8
Bit HT bus interconnection. In this interconnection method, only 8-bit HT interconnection can be used between the two processors. Two chip numbers
Don't be 00 and 01. From the routing algorithm, we can know that when two chips access each other, they are interconnected with
8-bit HT bus. As follows:

Figure 9-3 Two-chip Loongson No. 3 8-bit interconnection structure



However, our widest HT bus can use 16-bit mode, so the connection method to maximize bandwidth should be 16

97

Page 110

Bit interconnect structure. In Godson III, as long as the HT0 controller is set to 16-bit mode, all are sent to the HT0 controller
Will be sent to HT0_LO instead of HT0_HI or HT0_LO according to the routing table.
In this way, we can use the 16-bit bus when interconnecting. Therefore, we only need to convert the 16-bit modulo of CPU0 and CPU1
The 16-bit HT bus interconnection can be used if the configuration is correctly configured and the high and low busses are correctly connected. And this interconnection structure al
The 8-bit HT bus protocol can be used for mutual access. The resulting interconnection structure is as follows:

Figure 9-4 Two-chip Loongson No. 3 16-bit interconnection structure

10 Low-speed IO controller configuration

Loongson No. 3 I / O controller includes PCI / PCI-X controller, LPC controller, UART controller, SPI controller, GPIO and configuration registers. These I / O controllers share an AXI port, after the CPU's request is decoded by the address Send to the appropriate device.

10.1 PCI / PCI-X controller

The PCI / PCI-X controller of Loongson 3 can be used as the main bridge to control the entire system, or it can be used as an ordinary PCI / PCI-X devices work on the PCI / PCI-X bus. Its implementation conforms to PCI-X 1.0b and PCI 2.3 specifications. Dragon The PCI / PCI-X controller of Core 3 also has a built-in PCI / PCI-X arbiter.

The configuration header of 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

Byte 3	Byte 2	Byte 1	Byte 0	address
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08

BIST	Header Type	Latency Timer	CacheLine Size	0C
		Base Address Register 0		10
		Base Address Register 1		14
		Base Address Register 2		18
		Base Address Register 3		1C
		Base Address Register 4		20
		Base Address Register 5		twenty four
				28
Subsystem ID		Subsystem Vendor ID		2C
				30
			Capabilities Pointer	34
				38
Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line	3C
		Implementation Specific Register (ISR40)		40
		Implementation Specific Register (ISR44)		44
		Implementation Specific Register (ISR48)		48
		Implementation Specific Register (ISR4C)		4C
		Implementation Specific Register (ISR50)		50
		Implementation Specific Register (ISR54)		54
		Implementation Specific Register (ISR58)		58
				...
		PCIX Command Register		E0
		PCIX Status Register		E4

99

Page 112

The PCIX controller of Loongson 3A1000 supports three 64-bit windows, composed of {BAR1, BAR0}, {BAR3, BAR2}, {BAR5, BAR4} The base address of three pairs of register configuration windows 0, 1, 2. The size, enable, and other details of the The three corresponding registers PCI_Hit0_Sel, PCI_Hit1_Sel, PCI_Hit2_Sel control, please refer to Table 2 for specific bit fields.

Table 10-2 PCI Control Register

Bit field	Field name	access	Reset value	Explanation
REG_40				
31	tar_read_io	Read and write (Write 1 clear)	0	Target end receives access to IO or non-prefetchable area
30	tar_read_discard	Read and write (Write 1 clear)	0	The delay request on the target side is discarded
29	tar_resp_delay	Read and write	0	When target access is given delay / split 0: After timeout 1: right away
28	tar_delay_retry	Read and write	0	target access retry strategy 0: According to internal logic (see bit 29) 1: Retry now
27	tar_read_abort_en	Read and write	0	If the target times out for internal read requests, whether to let target-abort respond
26:25	Reserved	-	0	
24	tar_write_abort_en	Read and write	0	If the target's internal write request times out, whether to respond with target-abort
23	tar_master_abort	Read and write	0	Whether to allow master-abort
22:20	tar_subseq_timeout	Read and write	000	target subsequent delay timeout 000: 8 cycles Other: Not supported
				target initial delay timeout In PCI mode 0: 16 cycles

			1-7: Disable counter 8-15: 8-15 cycle In PCIX mode, the timeout count is fixed at 8 cycles.
19:16 tar_init_timeout	Read and write	0000	delay visits 0: 8 delay access 8: 1 delay access 9: 2 delay visit 10: 3 delay visit 11: 4 delay visit 12: 5 delay visit 13: 6 delay visit 14: 7 delay visit 15: 8 delay visit
15: 4 tar_pref_boundary	Read and write	000h	Prefetchable boundary configuration (in units of 16 bytes) FFF: 64KB to 16byte FFE: 64KB to 32byte FF8: 64KB to 128byte Configuration using tar_pref_boundary
3 tar_pref_bound_en	Read and write	0	0: prefetch to device boundary 1: Use tar_pref_boundary
2 Reserved	-	0	
1 tar_splitw_ctrl	Read and write	0	target split write control 0: Block access other than Posted Memory Write 1: Block all access until the split is completed
0 mas_lat_timeout	Read and write	0	Disable mater access timeout 0: Allow master access timeout 1: not allowed

REG_44

31: 0 Reserved	-	-	

REG_48

31: 0 tar_pending_seq	Read and write	0	target unprocessed request number bit vector The corresponding bit can be cleared by writing 1

REG_4C

31:30 Reserved	-	-	
29 mas_write_defer	Read and write	0	Allow subsequent reads to skip past unfinished writes (Only valid for PCI)
28 mas_read_defer	Read and write	0	Allow subsequent reads and writes to bypass previous unfinished reads (Only valid for PCI)
27 mas_io_defer_cnt	Read and write	0	Maximum number of IO requests out 0: controlled by 1: 1
26:24 mas_read_defer_cnt	read and write	010	The maximum number of master supports reading outside (only valid for PCI) 0: 8 1-7: 1-7
23:16 err_seq_id	Read only	00h	Note: A dual address cycle access accounts for two target / master error number
15 err_type	Read only	0	Command type of target / master error 0: The wrong module
14 err_module	Read only	0	0: target 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 parity error (write 1 clear)

100

Page 113

101

Page 114

10: 0 Reserved	-	-	
REG_50			
31: 0 mas_pending_seq	Read and write	0	Vector of unprocessed request number of master The corresponding bit can be cleared by writing 1
REG_54			
31: 0 mas_split_err	Read and write	0	split returns the wrong request number vector
REG_58			
31:30 Reserved	-	-	
29:28 tar_split_priority	Read and write	0	target split returns priority 0 highest, 3 lowest
27:26 mas_req_priority	Read and write	0	master external priority 0 highest, 3 lowest
25 Priority_en	Read and write	0	Arbitration algorithm (arbitration between master's access and target's split return) 0: fixed priority 1: rotation
24:18 Reserved	-	-	
17 mas_retry_aborted	Read and write	0	master retry cancellation (write 1 to clear)
16 mas_trdy_timeout	Read and write	0	master TRDY timeout count
15: 8 mas_retry_value	Read and write	0xff	master retries 0: unlimited retry 1-255: 1-255 times
7: 0 mas_trdy_count	Read and write	0xff	master TRDY timeout counter 0: disabled 1-255: 1-255 beat

Before initiating reading and writing in the configuration space, the application should first configure the PCIMap_Cfg register and tell the The type of configuration operation and the value on the upper 16-bit address line. Then read the 2K space starting at 0x1fe80000 Write to access the configuration header of the corresponding device. The device number is obtained by coding according to PCIMap_Cfg [15: 0] from low to high priority.

The configuration operation address generation is shown in Figure 10-1.

102

Page 115

Figure 10-1 Configure the read and write bus address generation

The PCI / PCIX arbiter implements two-level round robin arbitration, bus docking, and isolation of damaged master devices. Its configuration and status See PXArb_Config and PXArb_Status registers. The assignment of PCI / PCIX bus request and response lines is shown in Table 13-3.

Table 10-3 PCI / PCIX bus request and response line assignment

Request and answer line	description
0	Internal integrated PCI / PCIX controller
7: 1	External request 6 ~ 0

The rotation-based arbitration algorithm provides two levels, and the second level as a whole is scheduled as a member of the first level. Dangduo When a device applies for the bus at the same time, the first level device is rotated once, and the highest priority device in the second level can get line.

The arbiter is designed to be switched at any time as long as conditions permit. For some PCI devices that do not conform to the protocol, Doing so may make it abnormal. Using mandatory priority allows these devices to occupy the bus through continuous requests.

Bus docking refers to whether or not to select one to give an enable signal when no device requests to use the bus. For already As far as allowed devices are concerned, directly initiating bus operations can improve efficiency. Loongson 2F's PCI arbiter provides two kinds of stop By mode: the last master device and the default master device. If you cannot dock in special occasions, you can set the arbiter To dock to the default No. 0 master device (internal controller), and rely on delay 0.

10.2 LPC controller

The LPC controller has the following characteristics:

- Conform to LPC1.1 specification
- Support LPC access timeout counter
- Supports Memory Read and Memory write access types
- Support Firmware Memory Read, Firmware Memory Write access type (single byte)
- Supports I / O read and I / O write access types
- Support memory access type address conversion
- Support Serizlized IRQ specification, provide 17 interrupt sources

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

Table 10-4 LPC Controller Address Space Distribution

Address name	Address range	size
LPC Boot	0X1FC0_0000-0X1FD0_0000	1MByte

LPC Memory	0X1C00_0000-0X1E00_0000	32MByte
LPC I / O	0X1FF0_0000-0X1FF1_0000	64KByte
LPC Register	0X1FE0_0200-0X1FE0_0300	256Byte

The LPC Boot address space is the address space that the processor first accesses when the system starts. This address space supports LPC Memory or Firmware Memory access type. What type of access is issued by the system at startup LPC_ROM_INTEL pin control. LPC Firmware Memory is issued when the LPC_ROM_INTEL pin is pulled up Access, LPC Memory access type is issued when the LPC_ROM_INTEL pin is pulled down.

The LPC Memory address space is the address space accessed by the system with Memory / Firmware Memory. LPC control The type of memory access issued by the controller is determined by the configuration register LPC_MEM_IS_FWH of the LPC controller. The address sent by the processor to this address space can perform address translation. The converted address is sent by the configuration of the LPC controller Register LPC_MEM_TRANS.

The processor's access to the LPC I / O address space is sent to the LPC bus according to the LPC I / O access type. Address is address The space is 16 bits lower.

There are three 32-bit registers in the LPC controller configuration register. The meaning of the configuration register is shown in Table 13-5:

Table 10-5 Meaning of LPC Configuration Register

Bit field	Field name	Access	reset value	description
		REG0		
REG0 [31:31]	SIRQ_EN	Read-write	0	SIRQ enable control
REG0 [23:16]	LPC_MEM_TRANS	Read-write	0	LPC Memory Space Address Translation Control
REG0 [15: 0]	LPC_SYNC_TIMEOUT	Read-write	0	LPC access timeout counter
		REG1		
REG1 [31:31]	LPC_MEM_IS_FWH	Read-write	0	LPC Memory Space Firmware Memory access type settings
REG1 [17: 0]	LPC_INT_EN	Read-write	0	LPC SIRQ interrupt enable
		REG2		
REG2 [17: 0]	LPC_INT_SRC	Read-write	0	LPC SIRQ interrupt source indication
		REG3		
REG3 [17: 0]	LPC_INT_CLEAR	write	0	LPC SIRQ interrupt clear

10.3 UART controller

The UART controller has the following features

- Full duplex asynchronous data receiving / sending
- Programmable data format
- 16-bit programmable clock counter
- Support receiving timeout detection
- Multi-interrupt system with arbitration
- Only work in FIFO mode
- Compatible with NS16550A in register and function

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

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

The base address of the physical address of the 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

Bit field	Bit field name	Bit width	access	description
7: 0	Tx FIFO	8	W	Data transfer register

10.3.2 Interrupt enable register (**IER**)

Chinese name: Interrupt enable register

Register bit width: [7: 0]

Offset: 0x01

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description
7: 4	Reserved	4	RW	Keep
3	IME	1	RW	Modem status interrupt enable '0' – close '1' – open
2	ILE	1	RW	Receiver line status interrupt enable '0' – close '1' – open
1	ITxE	1	RW	The transfer save register is empty and interrupt enable '0'-turn off '1'-hit open
0	IRxE	1	RW	Receive valid data interrupt enable '0' – close '1' – open

10.3.3 Interrupt Identification Register (**IIR**)

Chinese name: Interrupt source register

105

Register bit width: [7: 0]
Offset: 0x02
Reset value: 0xc1

Bit field	Bit field name	Bit width	access	description
7: 4	Reserved	4	R	Keep
3: 1	II	3	R	Interrupt source display bit, see the table below for details
0	INTp	1	R	Interrupt indication bit

Interrupt control function table

Bit 3	Bit 2	Bit 1	Priority	Type of interrupt	Interrupt source	Interrupt reset control
0	1	1	1st	Receiving line state	Parity, overflow, or frame error, or Interrupt	Read LSR
0	1	0	2nd	Received valid data	The number of characters in the FIFO reaches trigger level	FIFO number of characters in FIFO Value for trigger
1	1	0	2nd	Receive timeout	There is at least one character in the FIFO, But not in 4 character time Any operation, including read and write operations Make	Read receive FIFO
0	0	1	3rd	Transmit, save, send The memory is empty	Transfer save register is empty	Write data to THR or Multi IIR
0	0	0	4th	Modem status 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

Bit field	Bit field name	Bit width	access	description
7: 6	TL	2	W	Trigger value '00' of the interrupt request from the receiving FIFO – 1 byte '01' – 4 bytes '10' – 8 bytes '11' – 14 bytes

106

Page 119

5: 3	Reserved	3	W	Keep
2	Txset	1	W	'1' Clear the content of transmit FIFO, reset its logic
1	Rxset	1	W	'1' Clear the content of the receive FIFO, reset its logic
0	Reserved	1	W	Keep

10.3.5 Line Control Register (**LCR**)

Chinese name: Line Control Register

Register bit width: [7: 0]

Offset: 0x03

Reset value: 0x03

Bit field	Bit field name	Bit width	access	description
7	dlab	1	RW	Divider latch access bit '1'-access to the operation divider latch '0'-access to normal operation register
6	bcb	1	RW	Interrupt control bit '1'-At this time the output of the serial port is set to 0 (interrupted state). '0'-normal operation
5	spb	1	RW	Specify parity '0' – no parity bit specified '1' – transmission and check parity if LCR [4] bit is 1 The bit is 0. If the LCR [4] bit is 0, transmit and check the parity The checkpoint is 1.
4	eps	1	RW	Parity bit selection '0' – There are an odd number of 1s in each character (including data and odd Even parity bit) '1' – there are an even number of 1s in each character
3	pe	1	RW	Parity bit enable '0' – no parity bit '1'-generate parity bit on output, judge odd on input Even parity
2	sb	1	RW	Define the number of generated stop bits

107

Page 120

'0' – 1 stop bit

'1' – 1.5 stop bits when 5 characters long, others

				The length is 2 stop bits
1: 0	bec	2	RW	Set the number of digits for each character '00' – 5 digits '01' – 6 digits '10' – 7 digits '11' – 8 digits

10.3.6 **MODEM** control register (**MCR**)

Chinese name: Modem control register

Register bit width: [7: 0]

Offset: 0x04

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description
7: 5	Reserved	3	W	Keep
4	Loop	1	W	Loopback mode control bit '0'-normal operation '1' – Loopback mode. In loopback mode, TXD outputs a Straight to 1, the output shift register is directly connected to the input shift register 器 中. The other connections are as follows. DTR → DSR RTS → CTS Out1 → RI Out2 → DCD
3	OUT2	1	W	Connect to DCD input in loopback mode
2	OUT1	1	W	Connect to RI input in loopback mode
1	RTSC	1	W	RTS signal control bit
0	DTRC	1	W	DTR signal control bit

10.3.7 Line Status Register (**LSR**)

Chinese name: Line Status Register

Register bit width: [7: 0]

Offset: 0x05

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description
7	ERROR	1	R	Error indication bit '1'-at least parity error, framing error or interruption The broken one. '0' – no errors
6	TE	1	R	Transmission is empty '1' – Both the transmission FIFO and the transmission shift register are empty. give Clear when the transmit FIFO writes data '0' – with data
5	TFE	1	R	Transmit FIFO bit empty representation bit

				'1' – The current transmit FIFO is empty, write data to the transmit FIFO
				Time zero
				'0' – with data
4	BI	1	R	Interrupt interruption bit
				'1'-Start bit + data + parity bit + stop bit received
				Is 0, that is interrupted
				'0'-no interruption
3	FE	1	R	Frame error indication bit
				'1' – received data has no stop bit
				'0' – no errors
2	PE	1	R	Parity bit error indicates bit
				'1'-The current received data has a parity error
				'0' – no parity error
1	OE	1	R	Data overflow indication bit
				'1'-There is data overflow
				'0' – no overflow
0	DR	1	R	Receive data valid representation bit
				'0' – No data in FIFO
				'1' – There is data in the FIFO

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

10.3.8 **MODEM** status register (**MSR**)

Chinese name: Modem Status Register

Register bit width: [7: 0]

Offset: 0x06

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description
7	CDCD	1	R	Inverse of DCD input value, or connect to Out2 in loopback mode
6	CRI	1	R	Inverse of RI input value, or connect to OUT1 in loopback mode
5	CDSR	1	R	Inverse of DSR input value, or connect to DTR in loopback mode
4	CCTS	1	R	Inverse of CTS input value, or connect to RTS in loopback mode
3	DDCD	1	R	DDCD indicator
2	TERI	1	R	RI edge detection. RI state changes from low to high
1	DDSR	1	R	DDSR indicator
0	DCTS	1	R	DCTS indicator

10.3.9 Frequency divider latch

Chinese name: Frequency Division Latch 1

Register bit width: [7: 0]

Offset: 0x00

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description

7: 0	LSB	8	RW	Store the lower 8 bits of the divider latch

Chinese name: Frequency Division Latch 2

Register bit width: [7: 0]

Offset: 0x01

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description
7: 0	MSB	8	RW	Stores the upper 8 bits of the divider latch

10.4 SPI controller

The SPI controller has the following features:

- Full duplex synchronous serial data transmission
- Supports up to 4 variable-length byte transmission
- Main mode support
- Mode failure generates an error flag and issues an interrupt request
- Double buffer receiver
- Serial clock with programmable polarity and phase
- Can control SPI in wait mode

The base address of the physical address of the SPI controller module register is 0x1FE001F0.

10.4.1 Control Register (**SPCR**)

Chinese name: Control Register

Register bit width: [7: 0]

Offset: 0x00

Reset value: 0x10

Bit field	Bit field name	Bit width	access	description
7	Spie	1	RW	Interrupt output enable signal is high and effective
6	spe	1	RW	System work enable signal is highly effective
5	Reserved	1	RW	Keep
4	mstr	1	RW	master mode selection bit, this bit keeps 1
3	cpol	1	RW	Clock polarity bit
2	cpha	1	RW	Clock phase bit 1 is the opposite phase, and 0 is the same
1: 0	spr	2	RW	sclk_o crossover setting, need to be used with sper spre

10.4.2 Status Register (**SPSR**)

Chinese name: Status Register

Register bit width: [7: 0]

Offset: 0x01

Reset value: 0x05

Bit field	Bit field name	Bit width	access	description
7	spif	1	RW	Interrupt flag bit 1 indicates that there is an interrupt request, write 1 to clear
6	wcol	1	RW	Write register overflow flag bit is 1 indicates that it has overflowed, write 1 to

Clear

111

5: 4	Reserved	2	RW	Keep
3	wffull	1	RW	Write register full flag 1 means full
2	wfempty	1	RW	Write register empty flag 1 means empty
1	rffull	1	RW	Read register full flag 1 means full
0	rfempty	1	RW	Read register empty flag 1 means empty

10.4.3 Data Register (**TxFIFO**)

Chinese name: Data Transfer Register

Register bit width: [7: 0]

Offset: 0x02

Reset value: 0x00

Bit field	Bit field name	Bit width	access	description
7: 0	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

Bit field	Bit field name	Bit width	access	description
7: 6	icnt	2	RW	Send an interrupt request signal after how many bytes are transferred 00 – 1 byte 01-2 bytes 10-3 bytes 11-3 bytes
5: 2	Reserved	4	RW	Keep
1: 0	spre	2	RW	Set the frequency division ratio with Spr

Frequency division factor:

spre	00	00	00	00	01	01	01	01	10	10	10	10
spr	00	01	10	11	00	01	10	11	00	01	10	11

Frequency division factor 1 2 4 8 32 64 128 256 512 1024 2048 4096

112

10.5 IO controller configuration

The configuration register is mainly used to configure the address window, arbiter and GPIO controller of the PCI / PIC-X controller. Table 10-6 lists these registers, and Table 10-7 gives a detailed description of the registers. The base address of this part of the register is 0x1FE00100.

Table 10-6 IO Control Register

address	register	Explanation
00	PonCfg	Power-on configuration
04	GenCfg	General configuration
08	Keep	
0C	Keep	
10	PCIMap	PCI mapping
14	PCIX_Bridge_Cfg	PCI / X bridge related configuration
18	PCIMap_Cfg	PCI configuration read and write device address
1C	GPIO_Data	GPIO data
20	GPIO_EN	GPIO direction
twenty four	Keep	
28	Keep	
2C	Keep	
30	Keep	
34	Keep	
38	Keep	
3C	Keep	
40	Mem_Win_Base_L	Prefetch the lower 32 bits of the base address of the window
44	Mem_Win_Base_H	Pre-fetch window base 32 higher bits
48	Mem_Win_Mask_L	Prefetchable window mask lower 32 bits
4C	Mem_Win_Mask_H	Pre-fetch window mask high 32 bits
50	PCI_Hit0_Sel_L	PCI window 0 controls the lower 32 bits
54	PCI_Hit0_Sel_H	PCI window 0 controls the upper 32 bits

58	PCI_Hit1_Sel_L	PCI Window 1 controls the lower 32 bits
5C	PCI_Hit1_Sel_H	PCI Window 1 controls the upper 32 bits
60	PCI_Hit2_Sel_L	PCI Window 2 controls the lower 32 bits
64	PCI_Hit2_Sel_H	PCI Window 2 controls the upper 32 bits
68	PXArb_Config	PCIX arbiter configuration
6C	PXArb_Status	PCIX arbiter status
70		
74		
78		
7C		

80	Chip Config	Chip configuration register
84		
88		
8C		
90	Chip Sample	Chip sampling register

Table 10-7 Register detailed description

Bit field	Field name	access	Reset value	Explanation
CR00: PonCfg				
15: 0	pcix_bus_dev	Read-only	lio_ad [7: 0]	In PCIX Agent mode, the total CPU usage Line, equipment number
15: 8	Keep	Read-only	lio_ad [15: 8]	
23:16	pon_pci_configi	Read-only	pci_configi	PCI_Configi pin value
31:24	Reserved	Read only		
CR04: reserved				
31: 0	Keep	Read only	0	
CR08: reserved				
31: 0	Keep	Read only	0	
CR10: PCIMap				
5: 0	trans_lo0	Read-write	0	PCI_Mem_Lo0 window map address high 6 bits
11: 6	trans_lo1	Read-write	0	PCI_Mem_Lo1 window map address high 6 bits
17:12	trans_lo2	Read-write	0	PCI_Mem_Lo2 window map address high 6 bits
31:18	Reserved	Read only	0	
CR14: PCIX_Bridge_Cfg				

5: 0	pcix_rgate	Read and write	6'h18	Threshold for sending data to DDR2 in PCIX mode
6	pcix_ro_en	Read-write	0	Does the PCIX bridge allow write over read
31:18	Reserved	Read only	0	
CR18: PCIMap_Cfg				
15: 0	dev_addr	Read-write	0	The upper 16 bits of the AD line in PCI configuration
16	conf_type	Read-write	0	Configure the type of read and write
31:17	Reserved	Read only	0	
CR1C: GPIO_Data				
15: 0	gpio_out	Read-write	0	GPIO output data
31:16	gpio_in	Read-write	0	GPIO input data
CR20: GPIO_EN				
15: 0	gpio_en	Read and write	16'hFFFF	High is input, low output
31:16	Reserved	Read only	0	
CR3C: reserved				
31: 0	Keep	Read only	0	Keep
CR24, 2C, 30, 34, 38: reserved				
See table 11-3				
CR50,54 / 58,5C / 60,64: PCI_Hit * _Sel_ *				
0	Keep	Read only	0	
2: 1	pci_img_size	Read and write	2'b11	00: 32 bits; 10: 64 bits; others: invalid
3	pref_en	Read-write	0	Prefetch enable
11: 4	Keep	Read only	0	
62:12	bar_mask	Read-write	0	Window size mask (high order 1, low order 0)
63	burst_cap	Read and write	1	Whether to allow burst transfer

CR68: PXArb_Config

0	device_en	Read and write 1	Permitted by external equipment
1	disable_broken	Read-write 0	Disable damaged master device
2	default_mas_en	Read and write 1	The bus is docked to the default master 0: dock to the last master device 1: dock to the default master device
5: 3	default_master	Read-write 0	Bus docking default master device number
7: 6	park_delay	Read and write 2'b11	Starting from no device requesting the bus to triggering the docking default Delay in device behavior 00: 0 cycles 01: 8 cycles 10: 32 cycles 11: 128 cycles
15: 8	level	Read and write 8'h01	Equipment in the first level
23:16	rude_dev	Read-write 0	Mandatory priority device

115

Page 128

			The PCI device corresponding to the 1 bit can be obtained after the bus To occupy the bus with continuous requests
31:13	Reserved	Read only 0	

CR6C: PXArb_Status

7: 0	broken_master	Read only 0	Damaged master device (cleared when changing the disable policy)
10: 8	Last_master	Read only 0	Last master device using the bus
31:11	Keep	Read only 0	

CR80: Chip config

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 frequency multiplication
8	Disable_ddr2_confspace	Read and write 1'b0	Whether to disable the DDR configuration space
9	DDR_buffer_cpu	Read and write 1'b0	Whether to open DDR read access buffer
12	Core0_en	Read and write 1'b1	Whether to enable processor core 0
13	Core1_en	Read and write 1'b1	Whether to enable processor core 1
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
twenty two	HT0_en	Read and write 1'b1	Whether to enable the HT controller 0
twenty three	HT1_en	Read and write 1'b1	Whether to enable the HT controller 1 1: open 0: disabled
28:24	DDR_Clksel	Read and write 5'b11111	Software configuration DDR clock multiplier relationship (when (Valid when DDR_Clksel_en is 1)
31:29	HT_freq_scale_ctrl0	Read and write 3'b111	HT controller divide by 0
34:32	HT_freq_scale_ctrl0	Read and write 3'b111	HT controller divided by 1
35	Mc0_prefetch_disable	Read and write 1'b0	Whether to disable the MC0 prefetch function (for different program lines Because it will have different performance impact)
36	Mc1_prefetch_disable	Read and write 1'b0	Whether to disable the MC1 prefetch function (for different program lines Because it will have different performance impact)
other		Read only	Keep

CR90: Chip Sample

15: 0	Pad2v5_ctrl	Read and write 16'h780	2v5pad control
31:16	Pad3v3_ctrl	Read and write 16'h780	3v3pad control
47:32	Sys_clksel	Read only	Onboard frequency setting
51:48	Bad_ip_core	Read only	4 processor cores are bad

53:52	Bad_ip_ddr	Read only	Whether 2 DDR controllers are bad
57:56	Bad_ip_ht	Read only	Whether 2 HT controllers are bad
102: 96	Thsens0_out	Read only	Temperature sensor 0 temperature, used to monitor the secondary cache attached Near temperature, accuracy is +/- 6 degrees Celsius
116			

103	Thsens0_overflow	Read only	Temperature sensor 0 temperature overflow (over 128 degrees)
110: 104	Thsens1_out	Read only	Temperature sensor 1 temperature, used to monitor the processor core attached Near temperature, accuracy is +/- 6 degrees Celsius
111	Thsens1_overflow	Read only	Temperature sensor 1 temperature overflow (over 128 degrees)
other		Read only	Keep

117

the second part

System Software Programming Guide

11 Interrupt configuration and use

11.1 Interrupted process

The process of Loongson 3A1000 handling interrupts, from the external interrupt request to the kernel's handling of interrupts, the process is the same of. The following figure shows the flow chart of the interrupt processing of 3A-690e board.

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

11.2 Interrupt routing and interrupt enable

Loongson 3A1000 chip supports up to 32 interrupt sources, managed in a unified manner, as shown in the following figure, any one An IO interrupt source can be configured to enable, trigger, and route the interrupt pin of the target processor core.

HT-1 INT7	31	can Match Set in Break road by	IP0 IP1 IP2 IP3	CORE 0
...	...			
HT-1 INT0	twenty four			
HT-0 INT7	twenty three			
...	...			
HT-0 INT0	16		IP0 IP1 IP2 IP3	CORE 1
PCI perr & serr	15			
Reserve	14			
Barrier INT	13			
DDR2-1 INT	12		IP0 IP1 IP2 IP3	CORE 2
DDR2-0 INT	11			
LPC INT	10			
MT-1 INT	9			
MT-0 INT	8			
PCI INTn3	7			
PCI INTn2	6			
PCI INTn1	5			
PCI INTn0	4			
INTn3	3		IP0 IP1 IP2 IP3	CORE 3
INTn2	2			
INTn1	1			
INTn0	0			

Figure 11-2 Schematic diagram of Loongson 3A1000 processor interrupt routing

11.2.1 Interrupt routing

Four processor cores are integrated in Loongson 3A1000. The above 32-bit interrupt sources can be selected through software configuration.

The target processor core is expected to be interrupted. Further, the interrupt source can be optionally routed to the processor core interrupt INT0 to INT3

Any one, namely IP2 to IP5 corresponding to CP0_Status. In other words, the CORE0 ~ CORE3 shown above

IP0 ~ IP3 correspond to CP2_IP5 of CP0_Status. Each of the 32 I / O interrupt sources corresponds to an 8-bit

Routing controller, its format and address are shown in Table 1 and Table 2 below. The routing register is routed in a vector way

Select, such as 0x48 to route to INT2 of processor 3.

Table 11-1 Description of Interrupt Routing Register

Bit field	Explanation
3: 0	Routed processor core vector number

7: 4	Routed processor core interrupt pin vector number

Table 11-2 Interrupt Routing Register Address

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

For ease of understanding, the following will give the configuration of the 3A-690e and 3A-KD60 cards on the interrupt routing:

a) 3A-690e

The hardware connection is "CPU serial port + HT1 connected to North and South Bridge". The routing is set to:

/ * Route the LPC interrupt to Core0 INT0, corresponding to IP2 of Cp0_Status * /

* (volatile unsigned char *) 0x900000003ff0140a = 0x11;

/ * Route the HT1 interrupt to Core0 INT1, corresponding to IP3 of Cp0_Status * /

* (volatile unsigned char *) 0x900000003ff01418 = 0x21;

```
* (volatile unsigned char *) 0x900000003ff01419 = 0x21;
* (volatile unsigned char *) 0x900000003ff0141a = 0x21;
* (volatile unsigned char *) 0x900000003ff0141b = 0x21;
* (volatile unsigned char *) 0x900000003ff0141c = 0x21;
* (volatile unsigned char *) 0x900000003ff0141d = 0x21;
```

121

Page 134


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

b) 3A-KD60

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

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

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

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

11.2.2 Interrupt enable

Interrupt related configuration registers are used to control the corresponding interrupt lines in the form of bits.

See Table 11-3 for configuration , The interrupt enable (Enable) configuration has three registers: Intenset, Intenclr and Inten. Intenset

Set the interrupt enable, and the interrupt corresponding to the write 1 bit in the Intenset register is enabled. Intenclr clear interrupt enable, Intenclr

The interrupt corresponding to the register write 1 is cleared. The Inten register reads the current status of each interrupt enable. Pulsed

The interrupt signal (such as PCI_SERR) is selected by the Intedge configuration register, write 1 means pulse trigger, write 0 means

Level trigger. The interrupt handler can clear the pulse record through the corresponding bit of Intenclr.

Table 11-3 Interrupt control bit connection and attribute configuration

name	Address offset	description
Intisr	0x1420	32-bit interrupt status register
Inten	0x1424	32-bit interrupt enable status register
Intenset	0x1428	32-bit setting enable register
Intenclr	0x142c	32-bit clear enable register
Intedge	0x1438	32-bit trigger mode register
CORE0_INTISR	0x1440	32-bit interrupt status routed to CORE0
CORE1_INTISR	0x1448	32-bit interrupt status routed to CORE1
CORE2_INTISR	0x1450	32-bit interrupt status routed to CORE2
CORE3_INTISR	0x1458	32-bit interrupt status routed to CORE3

Not only need to enable the IO controller, specifically to the connected IO, if it has its own interrupt controller, it also needs to

122

Page 135

Independent enable, such as LPC interrupt controller, HT interrupt controller, specific register configuration can check the register manual. below

Lists some interrupt controllers that may need to be enabled:

```
/ * 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 *) (0xffffffffbfe00200 + 0x00) = 0x80000000;
/ * the 18-bit interrpt enable bit * /
* (volatile unsigned int *) (0xffffffffbfe00200 + 0x04) = 0x0;

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

11.3 Interrupt distribution

When an interrupt occurs, the hardware sets the Excode field of the cause register and related IP bits. Then enter the software department

Process, the software determines which type of exception by querying the Excode field and chooses which exception handling example to use

Cheng. If it is the hardware interrupt we want to discuss, it will enter the platform-related interrupt dispatch function. Interrupt distribution function and then root

According to the IP bit and IM bit (interrupt mask) of the cause register, the first level interrupt is distributed, and then according to the interrupt number.

Bit field to perform secondary distribution, and then execute the specific do_IRQ interrupt operation processing.

During the startup phase of the kernel, each exception vector is mapped to each exception handling routine. There are 32 anomalies,

Here we only discuss the exception 0, which is a hardware interrupt. In the trap_init () function, the abnormal general entry address is set

Set to 0x80000180, this address holds a function pointer as expect_vec3_generic, see the kernel

arch / mips / kernel / genex.S. The expect_vec3_generic () function will enter the phase according to the obtained Excode code

The corresponding routine function. The exception code 0 in trap_init () is that the hardware interrupt is related to the handle_int routine, handle_int

See kernel arch / mips / kernel / genex.S. hanle_int finally jumps to the platform-related plat_irq_diapatch ()

Break the distribution function to perform the first-level distribution of interrupts.

Taking 3A-690e as an example, IP0 and IP1 of the Cause register correspond to soft interrupts, and IP6 corresponds to the core

Off, IP7 corresponds to the clock interrupt, IP2 corresponds to the CPU serial port and LPC interrupt, IP3 corresponds to the route to

HT1 interrupt. HT1 connects to North Bridge 690E, North Bridge then connects to South Bridge SB600, the interruption of North and South Bridges is all from 8259 on South Bridg

Control, the interruption of various peripherals such as USB, sata, etc. is routed to the 8259 controller. See arch / mips / kernel / fixup_ev3a.c

123

The godson3a_smbus_fixup function in the function, the definition of the register, see the AMD Southbridge manual "AMD SB600

Register Reference Manual.pdf ". The pcibios_map_irq function scans and centralizes PCI and PCIE slots

Number allocation, other functions in the file fixup_ev3a.c are mainly to allocate the interrupt number of some fixed PCI devices,

For detailed interrupt assignment and routing to 8259, please refer to the code and comment of file fixup_ev3a.c

Use to view AMD's Southbridge manual.

After the interrupt is distributed, run the do_IRQ () function and jump to the corresponding specific driver for execution.

124

12 Serial port configuration and use

12.1 Optional serial port

As a communication interface, serial port is mainly used for system debugging. The working principle is to configure the serial port baud Registers related to rate, data bits, stop bits, and parity bits enable the serial port to send and receive bytes bit by bit.

There are currently two types of UART available on the 3A1000: one type is the UART of the CPU, there are UART0 and UART1, The base address of the serial register of UART0 is 0xbfe001e0, the base address of the serial register of UART1 is 0xbfe001e8 The baud rate is 115200; there is another type of LPC UART, whose base address is 0xbff003f8, and the baud rate is 57600. In the setting, the data bits are set to 8 bits, the stop bit is 1 bit, no parity, no flow control.

12.2 Serial configuration of PMON

In pmon, the macro USE_LPC_UART is used to distinguish the above two types of serial ports. The setting of pmon mainly involves And the files are start.S and tgt_machdep.c.

The following takes the UART0 of the CPU as an example. First, there is a function to initialize the serial port in pmon start.S (register For the use of the controller, see the UART controller section):

```
LEAF (initserial)
    li a0, GS3_UART_BASE
    li      t1,128
    sb      t1,3 (a0)       // Access the divider register
    li      t1,0x12 # divider, highest possible baud rate
    sb      t1,0 (a0)       // Frequency divider register 1, stores the lower 8 bits of the frequency divider register, the calculation formula is
                            // Working clock / (baud rate * 16), in this case 33M / (115200 * 16)
    li      t1,0x0          # divider, highest possible baud rate
    sb      t1,1 (a0)       // Frequency divider register 2, stores the upper 8 bits of the frequency divider register
    li      t1,3
    sb      t1,3 (a0)       // The data bits are 8 bits

    li      t1,0
    sb      t1,1 (a0)       // Do not use interrupt
```

125


```
    li      t1,71
    sb      t1,2 (a0)       // Set FIFO control register
    jr      ra
    nop
END (initserial)
```

GS3_UART_BASE is also defined in the start.S file, which is 0xbfe001e0.

In tgt_machdep.c, the structure ConfigEntry also gives the UART settings, the function ns16550 also

It is the processing function of UART sending and receiving.

12.3 Serial configuration of Linux kernel

There are three main files for configuring the serial port in the Linux kernel: include / asm / serial.h,

arch / mips / kernel / 8250-platform.c, arch / mips / lemote / ev3a / dbg_io.c. These three configuration files involve

The setting of the serial port base address depends on the specific serial port used. In the kernel, in arch / mips / Kconfig

There is also a macro definition CONFIG_CPU_UART, used to select whether to use LPC serial port or CPU serial port. vertical

Although I chose the CPU UART, but whether it is UART0 or UART1, I still need to check the above three files.

Whether the definition of the serial port base address is correct.

In arch / mips / lemote / ev3a / dbg_io.c is mainly used in the kernel startup process, there is no initial

A simple serial port printing method that is used for the convenience of kernel debugging at the stage of conversion

It is often used in the kernel debugging stage, it will call putDebugChar () to print the characters one by one

Print to the console. include / asm / serial.h provides a macro definition for the serial driver / serial / 8250.c,

If you use the UART0 of the CPU, the definition is 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 *) (0xffffffffbfe001e0), \
.io_type = SERIAL_IO_MEM}
```

The driver used is the standard 8250 / 16x50 serial port driver. The interrupt number of the serial port is No. 58, according to

The serial port of the CPU or the serial port of the LPC is used. The distribution of interrupts is also slightly different. The section on serial interrupts in irq.c

Points are as follows:

126

Page 139


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

If it is the serial port of the CPU, it directly handles interrupt 58, but if it is an LPC serial port, you need to read the LPC control

To determine whether the interrupt is a keyboard interrupt, a mouse interrupt or a serial interrupt, the serial interrupt number is still

No. 58.

127

13 EJTAG debugging

13.1 Introduction to EJTAG

EJTAG (Enhanced JTAG) is defined by MIPS according to the basic structure and function extension of IEEE1149.1 protocol
The specification is a hardware / software subsystem used to support on-chip debugging. The EJTAG specification defines a new type of debugging
Modes, including dedicated instructions, operating modes, and address space, are well integrated with the original MIPS architecture
together. The basic idea of the debug mode is to adopt an exception handling mechanism, so that the software being debugged cannot detect the existence of the debugger.
In addition, the processor expands the address space in debug mode, and can access the debug control register and map to the
Debug interface for storage area. The following figure shows the composition of a common EJTAG debugging system, including:

♦ Debug Host: Run the debug application and control the EJTAG cable

♦ Target Board: the board containing the debugged chip, providing EJTAG interface

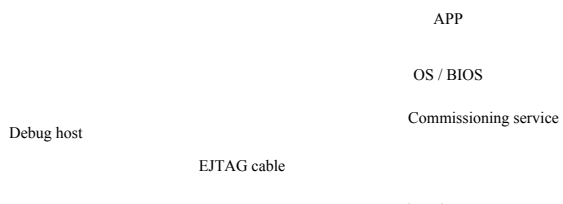


Figure 13-1 EJTAG debugging system

The debug host can use the EJTAG line to put the processor on the target board into debug mode and turn to execute the debug service
Cheng. There are two entrances to the debug service, which are located at 0xbfc00480 in the BIOS and at the EJTAG debug memory interface.
0xff200200 of the port is selected by the debugging host.

The EJTAG specification draws two blocks in the debug mode address space of the processor and maps them to the debug control register (drseg)
And debug memory interface (dmseg). When the processor executes the debugging service to access the debugging interface, the memory access request will be blocked
Plugged in the debug interface. At this time, the debugging host can detect the memory access request of the debugging memory interface and respond to it.
Simply put, the debug service program can access a section of memory space located on the debug host.

Using the debugging memory space controlled by the debugging host, the debugging service can complete almost any conceivable function, because
The code for the debugging service program itself can be provided by the debugging host (currently not available in the samples of Godson 3A1000 and 2G
Get instructions from dmseg, but can execute memory access instructions).

Godson 3A1000 / 2G also supports the PC sampling function of the EJTAG specification.

13.2 Use of EJTAG tools

13.2.1 Environmental preparation

♦ EJTAG cable, parallel port to 14-pin EJTAG port

♦ Linux debugging host with parallel port, with root authority

♦ Add debugging service program in PMON

13.2.2 **PC** sampling

Running jtag_1 / 4/16 on the debug host will complete a PC sampling of the processor core. Where the program suffix refers to

The number of processor cores in the EJTAG interface, one 2G is 1, one 3A1000 is 4, and four 3A1000 strings are together

16. PC sampling does not require debugging service programs.

The value of PC sampling in 3A1000 / 2G is inaccurate and there is jitter. The user needs to ignore the lower 2 bits of the sampling result. deal with

The real PC pointer in the core may be 0 ~ 4 instructions behind the program flow of the corresponding instruction of the sampled PC, that is

Move, a real PC may fall near the transfer target.

If it is not a regular cycle or software-controlled stop clock, PC sampling should not stand still. If it appears,

This means that the processor is not operating properly.

13.2.3 Read and write memory

Run pracc_1 / 4/16, specify the processor core number, access type, access address,

The value to write etc. Its working principle is to initialize a parameter area in the debugging memory space according to the task to be completed, and

The debugging service program cooperates to control the execution of the debugging service. The debugging service will first save several registers to the debugging host,

Then run based on these vacated registers, read and execute the relevant parameters, and finally restore the previously saved registers.

It should be noted that this function requires that the processor core executing the debugging service can also execute instructions. If it works abnormally, then

There may be a hardware failure.

13.2.4 Implementation instructions

Read bfc00480 | The meaning of the value

```
cpu @ ubuntu: ~ / ejtag $ ./pracc_4 0 0xffffffffbfc00480 dr
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 = 0000000000000027, value = ffffffff8000b899 | t4
pracc write, size = 000000000000002f, value = ffffffffffffffff | t5
pracc write, size = 0000000000000037, value = ffffffff80110000 | t6
pracc write, size = 000000000000003f, value = ffffffff800f7428 | t7
pracc write, size = 0000000000000047, value = 00000000340000e0 | status
pracc write, size = 000000000000004f, value = 0000000080034483 | config
pracc write, size = 0000000000000057, value = 0000000040008000 | cause
pracc write, size = 000000000000005f, value = 0000000000000033 | a0
pracc write, size = 0000000000000067, value = ffffffff80120000 | a1
pracc write, size = 000000000000006f, value = 0000000000000000 | a2
pracc read, size = 3, address = 0000000000000207, value = fff3ffffbfc00480
pracc write, size = 0000000000000107, value = fff3ffffbfc00480
pracc write, size = 0000000000000107, value = 0000000000000003
pracc write, size = 0000000000000107, value = 00000000000001fc
pracc write, size = 0000000000000107, value = 0000000000000001
pracc write, size = 0000000000000107, value = ffffffffbfc00480
pracc write, size = 000000000000020f, value = 3c08ff2040a8f800
return 3c08ff2040a8f800 |
Read return value
pracc read, size = 3, address = 0000000000000217, value = 0000000000000000
pracc read, size = 000000000000021f, value = 0000000000000000
pracc read, size = 000000000000000f, value = 0000000000000000
pracc read, size = 0000000000000017, value = 0000000000000000
pracc read, size = 000000000000001f, value = ffffffff80e1060
pracc read, size = 0000000000000027, value = ffffffff8000b899
pracc read, size = 000000000000002f, value = ffffffffffffffff
pracc read, size = 0000000000000037, value = ffffffff80110000
pracc read, size = 000000000000003f, value = ffffffff800f7428
```

Write bfc00480 (actually not written in)

```
cpu @ ubuntu: / home / cpu / ejtag $ ./pracc_4 0 0xffffffffbfc00480 dw 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
```

130

Page 143


```
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 annotation
# start.S
/ * Debug exception * /
        .align 7                                        / * bfc00480 * /
////////////////////////////////////////////
#define COP_0_DESAVE $ 31
    .set mips64
    // save context
    dmtc0 t0, COP_0_DESAVE                  // save a register for dmseg pointer
    lui t0, 0xff20                          //
    sd t1, 0x08 (t0)                        // push the stack
    sd t2, 0x10 (t0)
    sd t3, 0x18 (t0)
    sd t4, 0x20 (t0)
    sd t5, 0x28 (t0)
    sd t6, 0x30 (t0)
    sd t7, 0x38 (t0)
    dmfc0 t1, COP_0_STATUS_REG // output several cp0 registers
    sd t1, 0x40 (t0)
    dmfc0 t1, COP_0_CONFIG
    sd t1, 0x48 (t0)
    dmfc0 t1, COP_0_CAUSE_REG
    sd t1, 0x50 (t0)
    sd a0, 0x58 (t0)                        // Other interested registers
    sd a1, 0x60 (t0)
    sd a2, 0x68 (t0)

#define _t1 9
#define _t2 10
#define _t3 11
#define _t4 12
#define dextu (dest, src, msbd, dlsb) \
    .word
((0x1f << 26) | ((src & 0x1f) << 21) | ((dest & 0x1f) << 16) | (((msbd) & 0x1f) << 11) | (((dlsb) & 0x1f) << 6)
| (0x2))
```

```
#define dinsu (dest, src, dmsb, dlsb) \
        .word
((0x1f << 26) | ((src & 0x1f) << 21) | ((dest & 0x1f) << 16) | (((dmsb) & 0x1f) << 11) | (((dlsb) & 0x1f) << 6) |
(0x6))


        // exec_main
        ld t1, 0x200 (t0)                       // Read the parameter addr / size / count
        beqz t1, read_end
        sd t1, 0x100 (t0)                       // debug ...
        dextu (_t2, _t1, 2-1, 48-32)
```

131

Page 144


```
        sd t2, 0x100 (t0)                       // debug ...
        dextu (_t3, _t1, 9-1, 50-32)
        sd t3, 0x100 (t0)                       // debug ...
        dextu (_t4, _t1, 1-1, 47-32)
        sd t4, 0x100 (t0)                       // debug ...
        dsubu t4, $ 0, t4                       // sign bit extend
        dinsu (_t1, _t4, 58-32, 48-32) // address back
        sd t1, 0x100 (t0)                       // debug ...
        // case t2 0,1,2,3-> lb, lh, lw, ld
        beqzl t2, 1f
        lb t5, 0x0 (t1)
        addiu t2, t2, -1

        beqzl t2, 1f
        lh t5, 0x0 (t1)
        addiu t2, t2, -1

        beqzl t2, 1f
        lw t5, 0x0 (t1)
        addiu t2, t2, -1

        ld t5, 0x0 (t1)
1:
        sd t5, 0x208 (t0)                       // read the return value
read_end:
        // write
        ld t1, 0x210 (t0)                       // write parameter addr / size / count
        beqz t1, write_end
        ld t5, 0x218 (t0)                       // write parameters
        sd t1, 0x100 (t0)                       // debug ...
        dextu (_t2, _t1, 2-1, 48-32)
        sd t2, 0x100 (t0)                       // debug ...
        dextu (_t3, _t1, 9-1, 50-32)
        sd t3, 0x100 (t0)                       // debug ...
        dextu (_t4, _t1, 1-1, 47-32)
        sd t4, 0x100 (t0)                       // debug ...
        dsubu t4, $ 0, t4                       // sign bit extend
        dinsu (_t1, _t4, 58-32, 48-32) // address back
        sd t1, 0x100 (t0)                       // debug ...
        // case t2 0,1,2,3-> sb, sh, sw, sd
        beqzl t2, 1f
        sb t5, 0x0 (t1)
        addiu t2, t2, -1

        beqzl t2, 1f
        sh t5, 0x0 (t1)
        addiu t2, t2, -1

        beqzl t2, 1f
        sw t5, 0x0 (t1)
        addiu t2, t2, -1

        sd t5, 0x0 (t1)
1:
        sd t5, 0x218 (t0)                       // write response
write_end:
```

132

Page 145


```
// restore context
ld t1, 0x08 (t0)                    // unstack
ld t2, 0x10 (t0)
ld t3, 0x18 (t0)
ld t4, 0x20 (t0)
ld t5, 0x28 (t0)
ld t6, 0x30 (t0)
ld t7, 0x38 (t0)
dmfc0 t0, COP_0_DESAVE
deret                               // Debug exception return
```

13.2.5 Online **GDB** debugging

Due to the bugs in the current sample, the online debugging function needs to be modified in the common MIPS debugging platform

And increase the corresponding debugging service program. This feature has not been implemented.

Page 146

14 Address window configuration conversion

Loongson 3A1000 adopts two-stage cross switch structure, two-stage cross switch window can be configured separately for controlling

The address is sent to a specific receiver for processing. In addition, the HyperTransport controller is also internal to the chip and external to the chip.

Access to the address window is controlled.

14.1 Method for configuring the address window of the first and second level crossbar

Each main port on the crossbar has 8 address windows for configuration. Each address window consists of BASE, MASK Composed of three 64-bit registers with MMAP, BASE is aligned with K bytes, that is, the space allocated for each address window is the least 1KB; MASK is the window mask; MMAP is the window mapped address, and [2: 0] means the corresponding target slave port , MMAP [4] means to allow instruction fetch, MMAP [5] means to allow block reading, MMAP [7] means to enable window.

The judgment of window hit is as follows:

Master port address & MASK == BASE

The mapped slave port address conversion formula is as follows:

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

It should be noted that for the first-level crossbar switch, MMAP [4] and MMAP [5] must be 1. And for the two-level crossover Off, the slave port that does not allow cache access or fetch access can set MMAP [4] or MMAP [5] to 0.

In addition, if the first-level crossbar is used to map the second-level cache address, the mapped address is also called "from The "port address" must be consistent with the address before mapping, that is, the "primary port address." The address and the configuration on the secondary crossbar switch are not subject to this restriction.

14.2 Address window of primary crossbar switch

The first-level crossbar has the default routing setting. This setting is not displayed in the address window configuration register, only in The address hit by any address window will be interpreted by this default route.

For the main port of the first-level crossbar, that is, the main device side that sends out requests, includes the following:

Main port 0: processor core 0

Main port 1: processor core 1

Main port 2: processor core 2

Main port 3: processor core 3

Main port 6: HyperTransport 0

Main port 7: HyperTransport 1

For the slave port of the first-level crossbar, that is, the slave device that sends requests to the outside, it includes the following:

Slave port 0: Secondary cache 0

Slave port 1: secondary cache 1

Slave port 2: secondary cache 2

Slave port 3: secondary cache 3

Slave port 6: HyperTransport 0

Slave port 7: HyperTransport 1

The configuration of the address window of each master port is independent of each other, and each has 8 configurable windows. Priority of configuration window Down, starting from configuration window 0, the first hit window routes this address. The priority order is as follows:

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	
lowest	System default window	See table below

Among them, the "system default window" is only when all 8 "configuration windows" have not hit a certain address
Will take effect. In other words, before the "configuration window" is configured, all read and write requests will follow the "system
The default window is used for routing. The description of the "system default window" is as follows:

starting address	End address	aims	Explanation
0xn000_0000_0000	0xnFFF_FFFF_FFFF	HyperTransport 0	When n! = NODE_ID
0x0000_0000_0000	0x0BFF_FFFF_FFFF	Secondary Cache	According to the configuration of scid_sel Which two are mapped to different four secondary Cache. See section 2.4 of the user manual for details. E.g, When scid_sel = 0, 0x000: route to secondary cache 0 0x020: Route to secondary cache 1 0x040: Route to secondary cache 2 0x060: Route to secondary cache 3 When scid_sel = 2, 0x000: route to secondary cache 0 0x400: route to secondary cache 1 0x800: Route to secondary cache 2 0xc00: Route to secondary cache 3
0x0C00_0000_0000	0x0DFF_FFFF_FFFF	HyperTransport 0	
0x0E00_0000_0000	0x0FFF_FFFF_FFFF	HyperTransport 1	

135

Page 148

14.3 Timing of configuring the address window of the first-level crossbar

The first-level crossbar address window configuration is very important for the window configuration that needs to be routed to the second-level cache request
Strictly, the data consistency between the second-level cache and the first-level cache needs to be ensured before and after configuration, that is, it is never allowed
After configuration, Xu appears as follows:

Before the configuration, there is a backup in the second-level cache, and the corresponding backup in the first-level cache, but after the configuration
The request to close this backup address will be routed to other slave ports.

The above situation will eventually lead to the disorder of the first and second cache data. Therefore, the best way to configure each window
The timing is before the system has not performed the Cache operation. In other cases, if you need to configure the first-level crossbar
It must be ensured that the above situation does not occur.

It should be noted that changing the value of scid_sel after the Cache operation itself will also bring about this inconsistency,
Because the address space and the mapped secondary cache number have changed.

14.4 Address window of secondary crossbar switch

The secondary crossbar also has a default route, and all addresses that are not hit by any address window will be routed to the slave Port 3, that is, the system configuration register space.

For the main port of the two-level crossbar switch, that is, the main device side that sends out requests, the following include

- ♦ Main port 0: Secondary cache 0-3
- ♦ No. 1 main port: PCI main port

136

For the slave port of the second-level crossbar, that is, the slave device that sends out requests, the following include:

- ♦ Slave port 0: memory controller 0
- ♦ Slave port 1: memory controller 1
- ♦ Slave port 2: low-speed device interface, including PCI slave port, LPC interface, UART interface, SPI interface Port, PCI register space
- ♦ Slave port 3: System configuration register

Compared with the first-level crossbar switch, the restrictions of the address window configuration of the second-level crossbar switch will be weaker. The software can ensure that the access content after reconfiguring the address window will not be error.

For example, before mapping 0x0000_0000_0000 – 0x0000_0FFF_FFFF of the system address to memory control 0x0000_0000_0000 – 0x0000_0FFF_FFFF on device 0 Work, stored some valid data. After configuring the address window, set the system address to 0x0000_2000_0000 – 0x0000_2FFF_FFFF is mapped to 0x0000_0000_0000 of memory controller 0 – 0x0000_0FFF_FFFF. At this time, the access to 0x0000_2000_0000 will be used originally 0x0000_0000_0000 The value stored in the address.

In this process, it should be noted that the content in the secondary cache does not change according to the configuration of the address window. Change, that is to say, if the original write access to 0x0000_0000_0000 uses Cache, then it is likely The access to 0x0000_20000_0000 after the address window update will get an old value.

14.5 Special treatment for address window configuration

Since Loongson 3A1000 processor cores will have some guessed access to the outside, these guessed accesses may fall to Address space. However, not all devices are allowed to be accessed by guessing, especially for PCI devices, a guess The measured read access is likely to cause the destruction of a "read clear" register data, and may also cause some The access cannot return normally, and the processor freezes.

We have configured the first and second level crossbar switches to prevent these situations from happening, and set some non-guessable addresses Space is forbidden. For example, we can prevent the PCI space by setting the second-level crossbar as follows Speculative access to 0x1000_0000, but at the same time allow speculative access to 0x1FC0_0000.

BASE MASK MMAP

137

Page 150

Window 0 0x0000_0000_1000_0000	0xFFFF_FFFF_F000_0000 0x0000_0000_1000_0082	
Window 1 0x0000_0000_1FC0_0000	0xFFFF_FFFF_FFF0_0000 0x0000_0000_1FC0_00F2	

For the first-level crossbar switch, the mapping of the second-level cache address is also affected by scid_sel. These two

Those must not conflict.

If you need to map an address space to the secondary cache, you need to consider the impact of the secondary cache. which is

Map this address space to each secondary cache. Because each address window can only correspond to one slave port, then

The mapping to the secondary cache needs to be completed through the mapping of four address windows.

In addition, because the minimum unit of the address window is 1KB, if you configure the secondary cache space at this time,

It is necessary to set the value of scid_sel to 2 or more, that is, to use an address of 10 bits or more for hashing. As an example in the table below, for one

The level crossbar is configured to map all address accesses issued by the processor core to the level 2 cache.

Scid_sel = 2

	BASE	MASK	MMAP
Window 4	0x0000_0000_0000_0000	0x0000_0000_0000_0C00	0x0000_0000_0000_00F0
Window 5	0x0000_0000_0000_0400	0x0000_0000_0000_0C00	0x0000_0000_0000_04F1
Window 6	0x0000_0000_0000_0800	0x0000_0000_0000_0C00	0x0000_0000_0000_08F2
Window 7	0x0000_0000_0000_0C00	0x0000_0000_0000_0C00	0x0000_0000_0000_0CF3

From this window, we can see that all addresses that did not hit in windows 0-3 will be hit in these 4 windows, and according to

scid_sel hashes the entire address space into four correct secondary caches.

14.6 HyperTransport Address Window

The HyperTransport controller can not only send read and write access to the outside, but also enable external devices to access the processor.

DMA access to the internal memory. The access address spaces of these two visits are independent of each other, which are introduced separately below.

14.6.1 External access window of processor core

Loongson 3A1000 chip has 4 HyperTransport controllers, namely HT0_LO, HT0_HI,

HT1_LO, HT1_HI, where HT0_LO and HT0_HI share a physical interface, HT1_LO and HT1_HI

Sharing a physical interface, when the chip pin HTx_8x2 is set low, only HTx_LO is visible to the user, and HTx_HI

'S address space is invalid. According to the default routing of the first-level crossbar, the address space of each controller is as follows:

Base address	End address	size	definition	Explanation
0x0C00_0000_0000	0x0CFF_FFFF_FFFF	1 Tbytes	HT0_LO window	
0x0D00_0000_0000	0x0DFF_FFFF_FFFF	1 Tbytes	HT0_HI window	Effective when HT0_8x2 = 1

Page 151

0x0E00_0000_0000	0x0EFF_FFFF_FFFF	1 Tbytes	HT1_LO window	
0x0F00_0000_0000	0x0FFF_FFFF_FFFF	1 Tbytes	HT1_HI window	Effective when HT1_8x2 = 1

Each HyperTransport controller has a 40-bit address space. According to the HyperTransport protocol, this

The 40-bit address is divided as follows:

Base address	End address	size	definition
0x00_0000_0000	0xFC_FFFF_FFFF	1012 Gbytes	MEM space
0xFD_0000_0000	0xFD_F7FF_FFFF	3968 Mbytes	Keep
0xFD_F800_0000	0xFD_F8FF_FFFF	16 Mbytes	Interrupt
0xFD_F900_0000	0xFD_F90F_FFFF	1 Mbyte	PIC interrupt response
0xFD_F910_0000	0xFD_F91F_FFFF	1 Mbyte	system message
0xFD_F920_0000	0xFD_FAFF_FFFF	30 Mbytes	Keep
0xFD_FB00_0000	0xFD_FBFF_FFFF	16 Mbytes	HT controller configuration space
0xFD_FC00_0000	0xFD_FDFF_FFFF	32 Mbytes	I / O space
0xFD_FE00_0000	0xFD_FFFF_FFFF	32 Mbytes	HT bus configuration space
0xFE_0000_0000	0xFF_FFFF_FFFF	8 Gbytes	Keep

Among them, MEM space, I / O space, and HT bus configuration space correspond to the three accesses on the traditional PCI space, respectively. The forms are 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 "bus number", "device number", "function number", "register offset" of the external device to The following rules allow direct read and write access to the address.

Type 0:

Type 1:

14.6.2 Address window for external device to **DMA** access to processor chip memory

In order to protect the memory data in the processor chip, the HyperTransport controller provides DMA access for external devices. A set of windows is provided, namely the “Receive Address Window” in Section 9.5.4 of the user manual, only the DMA locations that fall in this set of windows

The address will be truly operated on the memory space in the chip, otherwise it will make an error response to the peripheral that initiated the access.

This group of windows is determined by the following parameters, and the window hitting rules are 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 sequentially.

14.6.3 Low-speed device address window

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

Address name	Address range	size

LPC Memory	0x1C00_0000 – 0x1DFF_FFFF	32 MByte
LPC Boot	0x1FC0_0000 – 0x1FCF_FFFF	1 MByte
PCI IO space	0x1FD0_0000 – 0x1FDF_FFFF	1 MByte
PCI controller configuration space	0x1FE0_0000 – 0x1FE0_00FF	256 Byte
IO register space	0x1FE0_0100 – 0x1FE0_01DF	256 Byte
UART 0	0x1FE0_01E0 – 0x1FE0_01E7	8 Byte
UART 1	0x1FE0_01E8 – 0x1FE0_01EF	8 Byte
SPI	0x1FE0_01F0 – 0x1FE0_01FF	16 Byte
LPC Register	0x1FE0_0200 – 0x1FE0_02FF	256 Byte
PCI configuration space	0x1FE8_0000 – 0x1FE8_FFFF	64 KByte
LPC I / O	0x1FF0_0000 – 0x1FF0_FFFF	64 Kbyte
PCI MEM	other	

14.7 Example Analysis of Address Space Configuration

The following describes the configuration of the two-level crossbar in PMON. In the following example, the HT device Use HT1 interface connection, HT0 interface is not used.

14.7.1 Example of a level 1 crossbar **1**

One configuration of the first-level crossbar is as follows:

	BASE	MASK	MMAP
Window 0	0x0000_0000_1800_0000	0xFFFF_FFFF_FC00_0000	0x0000_0EFD_FC00_00F7
Window 1	0x0000_0000_1000_0000	0xFFFF_FFFF_F800_0000	0x0000_0E00_1000_00F7
Window 2	0x0000_0000_1E00_0000	0xFFFF_FFFF_FF00_0000	0x0000_0E00_0000_00F7
Window 3			
Window 4	0x0000_0C00_0000_0000	0xFFFF_FC00_0000_0000	0x0000_0C00_0000_00F7
Window 5			
Window 6	0x0000_1000_0000_0000	0x0000_1000_0000_0000	0x0000_1000_0000_00F7
Window 7	0x0000_2000_0000_0000	0x0000_2000_0000_0000	0x0000_2000_0000_00F7

The following analyzes the role of each configuration window.

Window 0, convert the address of 0x1800_0000 to 0x0000_0EFD_FC00_0000, and route to HT1 Controller. In this way, the HT IO space and HT configuration space that originally needed to be accessed using 64-bit addresses are directly The 32-bit address space is used for mapping, and the space after the mapping can be accessed using a 32-bit address.

	Address before conversion	Address after conversion	Explanation
Address 0	0x0000_0000_18xx_xxxx	0x0000_0EFD_FCxx_xxxx	HT IO space
Address 1	0x0000_0000_19xx_xxxx	0x0000_0EFD_FDxx_xxxx	
Address 2	0x0000_0000_1Axx_xxxx	0x0000_0EFD_FExx_xxxx	HT configuration space: Type 0
Address 3	0x0000_0000_1Bxx_xxxx	0x0000_0EFD_FFxx_xxxx	HT configuration space: Type 1

Window 1, convert the address of 0x1000_0000 to 0x0000_0E00_1000_0000, and route it to the HT1 controller Controller. In this way, a part of the HT MEM space that originally needs to be accessed using a 64-bit address is directly used

The 32-bit address space is used for mapping, and the space after the mapping can be accessed using a 32-bit address. Although there is no mapping All HT MEM space, but up to 128MB of HT MEM space can already be used here.

Address before conversion	Address after conversion	Explanation
Address 0 0x0000_0000_1xxx_xxxx	0x0000_0E00_1xxx_xxxx	HT MEM space

Window 2, convert the address of 0x1E00_0000 to 0x0000_0E00_0000_0000, and route it to the HT1 controller

Controller. In this way, the lowest 16MB address of the HT MEM space that originally needs to be accessed using a 64-bit address

The 32-bit address space is directly used for mapping, and the space after the mapping can be accessed using a 32-bit address. Of

141

Page 154

So this part of the address is mapped because some traditional devices need to use this reserved space for fixed decoding, such as

Graphics devices, etc.

Address before conversion	Address after conversion	Explanation
Address 0 0x0000_0000_1Exx_xxxx	0x0000_0E00_1Exx_xxxx	16MB lower HT MEM space

The setting of the previous windows is to use the 32-bit address directly in PMON without TLB conversion

You can access the HT space and HT equipment to facilitate the compatible design of the software version. In the linux system, because

In order to directly use 64-bit addresses for access, these address translations are not required. But it should be noted that the basic

Due to the way Linux handles the HT MEM space, it still needs to convert the HT MEM space to simplify the 32

Access to peripherals addressed by bit addresses.

The remaining windows are used to route all the addresses of non-responsive devices to the HT1 controller, and the HT1 controller will respond

should. These addresses will not appear actively during normal program execution, but due to the speculative execution of the processor core, any

Address access may occur. If the correct response is not obtained, the processor may crash. Godson 3A1000

The HT controller can correctly identify and handle such access. Therefore, all these potential guess visits need to be routed to

HT controller.

Except these abnormal addresses, other addresses will be routed to the secondary cache according to the default routing method

Level crossbar forwarding.

14.7.2 Level 1 Crossbar Example **2**

Another configuration of the first-level crossbar is as follows:

Scid_sel = 2

BASE	MASK	MMAP
Window 0 0x0000_0000_1800_0000	0xFFFF_FFFF_FC00_0000	0x0000_0EFD_FC00_00F7
Window 1 0x0000_0000_1000_0000	0xFFFF_FFFF_F800_0000	0x0000_0E00_1000_00F7
Window 2 0x0000_0000_1E00_0000	0xFFFF_FFFF_FF00_0000	0x0000_0E00_0000_00F7
Window 3 0x0000_0E00_0000_0000	0xFFFF_FE00_0000_0000	0x0000_0E00_0000_00F7
Window 4 0x0000_0000_0000_0000	0x0000_0000_0000_0C00	0x0000_0000_0000_00F0
Window 5 0x0000_0000_0000_0400	0x0000_0000_0000_0C00	0x0000_0000_0000_04F1
Window 6 0x0000_0000_0000_0800	0x0000_0000_0000_0C00	0x0000_0000_0000_08F2
Window 7 0x0000_0000_0000_0C00	0x0000_0000_0000_0C00	0x0000_0000_0000_0CF3

In this configuration, the first three windows are the same as the previous configuration, and will not be repeated here.

142

Page 155

Window 3 routes all the addresses of 0x0000_0E00_0000_0000 to the HT1 controller, which is the default

A routing method, this configuration is done because in windows 4-7, all addresses are routed to four different

In the secondary cache, so the default route will no longer take effect.

The configuration of windows 4-7 has been explained in Section 1.5, the most important of which is routing to each

The level of Cache should be consistent with the configuration of scid_sel. The purpose of this configuration is the same as the first configuration, all for

In order to prevent some addresses that do not respond to the device to cause the processor to freeze.

14.7.3 Example of a secondary crossbar **1**

One configuration of the two-level crossbar is as follows. Only one memory controller is used in this configuration. Use memory

256MB space.

	BASE	MASK	MMAP
Window 0	0x0000_0000_1000_0000	0xFFFF_FFFF_F000_0000	0x0000_0000_1000_0082
Window 1	0x0000_0000_1FC0_0000	0xFFFF_FFFF_FFF0_0000	0x0000_0000_1FC0_00F2
Window 2	0x0000_0000_0000_0000	0xFFFF_FFFF_F000_0000	0x0000_0000_0000_00F0
Window 3			
Window 4			
Window 5			
Window 6			
Window 7			

Window 0 opens the uncache and non-fetch access of the low-speed device space, so that it can be guaranteed to fall in this window

The visit of the mouth is the visit that the procedure needs to make.

Window 1 opens all types of access to the BOOT space in the low-speed device space, including cache access and instruction fetch

Normal visits including visits, guess visits can make normal visits to this space.

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

All other accesses will be routed to the system configuration register space according to the default route.

Combined with the first-level crossbar configuration in 1.8.1, the resulting full chip address space is as follows:

	starting address	End address	Explanation
Address 0	0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF	Memory controller 0
Address 1	0x0000_0000_1000_0000	0x0000_0000_17FF_FFFF	HT1 MEM space
Address 2	0x0000_0000_1800_0000	0x0000_0000_19FF_FFFF	HT1 IO space
Address 3	0x0000_0000_1A00_0000	0x0000_0000_1BFF_FFFF	HT1 configuration space
Address 4	0x0000_0000_1C00_0000	0x0000_0000_1DFF_FFFF LPC Memory	
Address 5	0x0000_0000_1FC0_0000	0x0000_0000_1FCF_FFFF LPC Boot	

143

Page 156

Address 6	0x0000_0000_1FD0_0000	0x0000_0000_1FDF_FFFF PCI IO space	
Address 7	0x0000_0000_1FE0_0000	0x0000_0000_1FE0_00FF	PCI controller configuration space
Address 8	0x0000_0000_1FE0_0100	0x0000_0000_1FE0_01DF	IO register space
Address 9	0x0000_0000_1FE0_01E0	0x0000_0000_1FE0_01E7	UART 0

Address 10	0x0000_0000_1FE0_01E8	0x0000_0000_1FE0_01EF	UART 1
Address 11	0x0000_0000_1FE0_01F0	0x0000_0000_1FE0_01FF	SPI
Address 12	0x0000_0000_1FE0_0200	0x0000_0000_1FE0_02FF	LPC Register
Address 13	0x0000_0000_1FE8_0000	0x0000_0000_1FE8_FFFF	PCI configuration space
Address 14	0x0000_0000_1FF0_0000	0x0000_0000_1FF0_FFFF	LPC I / O
Address 15	0x0000_0C00_0000_0000	0x0000_0FFF_FFFF_FFFF	HT1 controller, various spaces
Address 16	0x0000_1000_0000_0000	0x0000_3FFF_FFFF_FFFF	HT1 controller, guess space
Address 17	Other address		System configuration space

14.7.4 Two-level crossbar example **2**

Another configuration of the two-level crossbar is as follows. Two memory controllers are used in this configuration. Each memory controller makes

Use 1GB of memory

	BASE	MASK	MMAP
Window 0	0x0000_0000_1000_0000	0xFFFF_FFFF_F000_0000	0x0000_0000_1000_0082
Window 1	0x0000_0000_1FC0_0000	0xFFFF_FFFF_FFF0_0000	0x0000_0000_1FC0_00F2
Window 2	0x0000_0000_0000_0000	0xFFFF_FFFF_F000_0000	0x0000_0000_0000_00F0
Window 3			
Window 4	0x0000_0000_8000_0000	0xFFFF_FFFF_C000_0000	0x0000_0000_0000_00F0
Window 5			
Window 6	0x0000_0000_C000_0000	0xFFFF_FFFF_C000_0000	0x0000_0000_0000_00F1
Window 7			

Window 0 opens the uncache and non-fetch access of the low-speed device space, so that it can be guaranteed to fall in this window

The visit of the mouth is the visit that the procedure needs to make.

Window 1 opens all types of access to the BOOT space in the low-speed device space, including cache access and instruction fetch

Normal visits including visits, guess visits can make normal visits to this space.

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

Window 4 opens all 1GB of space on memory controller 0, the system uses 0x8000_0000 –

0xBFFF_FFFF address is accessed, it should be noted that the system 0x8000_0000 – 0x8FFF_FFFF is empty

Between 0x0000_0000 and 0x0FFF_FFFF. The system software must ensure that the data is correct.

It must be ensured that only one of these addresses is used to access this part. Taking Linux as an example, you must use 0x0000_0000 in the system.

– The address of 0x0FFF_FFFF, then, for this space, the access of 0x8000_0000-8FFF_FFFF is

144

Page 157

Godson **3A1000** Processor User Manual Part 2

Prohibited.

Window 6 opens all 1GB of space on memory controller 1, the system uses 0xC0000_0000 –

0xFFFF_FFFF accesses this memory.

Combined with the first-level crossbar configuration in 1.8.2, the resulting full chip address space distribution is as follows:

	starting address	End address	Explanation
Address 0	0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF	Memory controller 0
Address 1	0x0000_0000_1000_0000	0x0000_0000_17FF_FFFF	HT1 MEM space
Address 2	0x0000_0000_1800_0000	0x0000_0000_19FF_FFFF	HT1 IO space
Address 3	0x0000_0000_1A00_0000	0x0000_0000_1BFF_FFFF	HT1 configuration space
Address 4	0x0000_0000_1C00_0000	0x0000_0000_1DFF_FFFF	LPC Memory
Address 5	0x0000_0000_1FC0_0000	0x0000_0000_1FCF_FFFF	LPC Boot
Address 6	0x0000_0000_1FD0_0000	0x0000_0000_1FDF_FFFF	PCI IO space
Address 7	0x0000_0000_1FE0_0000	0x0000_0000_1FE0_00FF	PCI controller configuration space
Address 8	0x0000_0000_1FE0_0100	0x0000_0000_1FE0_01DF	IO register space

Address 9	0x0000_0000_1FE0_01E0	0x0000_0000_1FE0_01E7	UART 0
Address 10	0x0000_0000_1FE0_01E8	0x0000_0000_1FE0_01EF	UART 1
Address 11	0x0000_0000_1FE0_01F0	0x0000_0000_1FE0_01FF	SPI
Address 12	0x0000_0000_1FE0_0200	0x0000_0000_1FE0_02FF	LPC Register
Address 13	0x0000_0000_1FE8_0000	0x0000_0000_1FE8_FFFF	PCI configuration space
Address 14	0x0000_0000_1FF0_0000	0x0000_0000_1FF0_FFFF	LPC I / O
Address 15	0x0000_0000_8000_0000	0x0000_0000_BFFF_FFFF	Memory controller 0
Address 16	0x0000_0000_C000_0000	0x0000_0000_FFFF_FFFF	Memory controller 1
Address 17	0x0000_0E00_0000_0000	0x0000_0FFF_FFFF_FFFF	HT1 controller, various spaces
Address 18	Other address		System configuration space

145

Page 158

15 System memory space distribution design

15.1 System memory space

Loongson 3A1000 processor has two memory controllers, if the address space can be interleaved in two memories

On the controller, it will bring benefits to the average access delay and average access bandwidth of the system, but it is subject to the configuration of the crossbar

To limit the number of windows, certain rules must be adopted to design a certain method for address space distribution.

Based on the above considerations, at the same time, in order to maintain the different needs of different memory space sizes of the Linux system, and ensure that the memory

The distribution is simple and beautiful, you can use the following rules to design the memory address space. Of course, according to the system designer

You can also customize the memory space distribution rules.

(1) No matter how big the memory size is, you must ensure that the lower 256MB of 0x0000_0000 – 0x0FFF_FFFF

space;

(2) In order to reserve the necessary direct access address space for IO devices, 0x1000_0000 – 0x1FFF_FFFF

Reserved for non-spatial address space;

(3) Therefore, the remaining part of the memory space of 1GB or more is defined according to the following formula:

Base = Size + 0x1000_0000

Limit = Size + Size-1

Among them, Base and Limit are the base address and high address of this space, Size is all memory

size.

For example, if the memory size is 1GB, the address space of the memory in the system is as follows:

	starting address	End address	Explanation
Address 0	0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF	0 – 256MB
Address 1	0x0000_0000_5000_0000	0x0000_0000_7FFF_FFFF	256MB – 1GB

If the memory size is 2GB, the address space of the memory in the system is as follows:

	starting address	End address	Explanation
Address 0	0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF	0 – 256MB
Address 1	0x0000_0000_9000_0000	0x0000_0000_FFFF_FFFF	256MB – 2GB

If the memory size is 4GB, the address space of the memory in the system is as follows:

	starting address	End address	Explanation

Page 159

Address 0	0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF	0 – 256MB
Address 1	0x0000_0001_1000_0000	0x0000_0001_FFFF_FFFF	256MB – 4GB

And so on.

In addition, in order to enable the two memory controllers to be interleaved, we configure the two-level crossbar as follows

On the memory address space.

Explanation		window		
Used to enable access to the BIOS space		0	BASE	0x00000000_1FC00000
			MASK	0xFFFFFFFF_FFF00000
			MMAP	0x00000000_1FC000F2
Used to enable access to PCI space (only non-access is allowed		1	BASE	0x00000000_10000000
Refers to UNCACHE access via)			MASK	0xFFFFFFFF_F0000000
			MMAP	0x00000000_10000082
Used to enable alignment	MC0 single channel	2	BASE	0x00000000_00000000
Low 256M space	256MB and above		MASK	0xFFFFFFFF_F0000000
Access			MMAP	0x00000000_000000F0
		3		
	Dual channel	2	BASE	0x00000000_00000000
	256MB x 2 and above		MASK	0xFFFFFFFF_F0000400
	(Interleaved with address [10])		MMAP	0x00000000_000000F0
		3	BASE	0x00000000_00000400
			MASK	0xFFFFFFFF_F0000400
			MMAP	0x00000000_000000F1
	MC0 single channel 256M	4		
		5		
		6		
		7		
Used to enable internal	MC0 single channel 512M	4	BASE	0x00000000_20000000
Store high address space			MASK	0xFFFFFFFF_F0000000
Access			MMAP	0x00000000_100000F0
		5		
		6		
		7		
	MC0 single channel 1G	4	BASE	0x00000000_40000000
			MASK	0xFFFFFFFF_C0000000
			MMAP	0x00000000_000000F0
		5		
		6		
		7		

147

Page 160

MC0 single channel 2G	4	BASE	0x00000000_80000000
		MASK	0xFFFFFFFF_80000000
		MMAP	0x00000000_000000F0
	5		
	6		
	7		
Dual channel 256M x 2 (Interleaved with address [10])	4	BASE	0x00000000_20000000
		MASK	0xFFFFFFFF_F0000400
		MMAP	0x00000000_000004F0
	5	BASE	0x00000000_20000400
		MASK	0xFFFFFFFF_F0000400
		MMAP	0x00000000_000004F1
	6		
	7		
Dual channel 512M x 2 (Interleaved with address [10])	4	BASE	0x00000000_40000000
		MASK	0xFFFFFFFF_E0000400
		MMAP	0x00000000_000000F0
	5	BASE	0x00000000_40000400
		MASK	0xFFFFFFFF_E0000400
		MMAP	0x00000000_000000F1
	6	BASE	0x00000000_60000000
		MASK	0xFFFFFFFF_E0000400
		MMAP	0x00000000_000004F0
	7	BASE	0x00000000_60000400
		MASK	0xFFFFFFFF_E0000400
		MMAP	0x00000000_000004F1
Dual channel 1G x 2 (Interleaved with address [10])	4	BASE	0x00000000_80000000
		MASK	0xFFFFFFFF_C0000400
		MMAP	0x00000000_000000F0
	5	BASE	0x00000000_80000400
		MASK	0xFFFFFFFF_C0000400
		MMAP	0x00000000_000000F1
	6	BASE	0x00000000_C0000000
		MASK	0xFFFFFFFF_C0000400
		MMAP	0x00000000_000004F0
	7	BASE	0x00000000_C0000400
		MASK	0xFFFFFFFF_C0000400
		MMAP	0x00000000_000004F1
Dual channel 2G x 2 (Interleaved with address [10])	4	BASE	0x00000001_00000000
		MASK	0xFFFFFFFF_80000400

148

Page 161

		MMAP	0x00000000_000000F0
	5	BASE	0x00000001_00000400

	MASK	0xFFFFFFFF_80000400
	MMAP	0x00000000_000000F1
6	BASE	0x00000001_80000000
	MASK	0xFFFFFFFF_80000400
	MMAP	0x00000000_000004F0
7	BASE	0x00000001_80000400
	MASK	0xFFFFFFFF_80000400
	MMAP	0x00000000_000004F1

15.2 Mapping relationship between system memory space and peripheral DMA space

After this configuration, we will introduce the use of memory addresses in DMA. Traditional PCI DMA empty

The address above 0x8000_0000 exists, when the device performs DMA operation, the access of 0x8000_0000 is mapped

Shot into the 0x0000_0000 space, and then one by one with the system memory mapping.

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

(1) System memory space 0x0000_0000 – 0x0FFF_FFFF When used as DMA space, peripherals

Use 0x8000_0000 – 0x8FFF_FFFF to access;

(2) When the memory space of other systems is used as DMA space, it can be used directly without address conversion.

Therefore, taking the 2GB memory space as an example, the following address translation table can be obtained:

	Explanation		starting address	End address
Address 0	0 – 256MB	System space	0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF
		DMA space	0x0000_0000_8000_0000	0x0000_0000_8FFF_FFFF
Address 1	256MB – 2GB	System space	0x0000_0000_9000_0000	0x0000_0000_FFFF_FFFF
		DMA space	0x0000_0000_9000_0000	0x0000_0000_FFFF_FFFF

When using the HyperTransport interface, the above address conversion method can be accessed through the HyperTransport

Receive address window "configuration to achieve. See section 1.6.2. Use two sets of address windows to complete this conversion.

	Explanation	Window enable register	Window base register
Window 0	0 – 256MB	0xC000_0000	0x0080_FFF0
Window 1	256MB – 2GB	0xC000_0080	0x0080_FF80

Window 0 translates the address of 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. According to the priority rule hit by the window, we can know that the address of 0x8000_0000 – 0x8FFF_FFFF is actually

149

The above will only be mapped by window 0, so the address handled by window 1 is actually 0x9000_0000 – 0xFFFF_FFFF.

15.3 Other mapping methods of system memory space

The system memory space mapping method introduced in the previous two sections is only an effective reference method, and system designers can also

Redefine the memory mapping rules according to your needs.

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

150

Page 163

Godson **3A1000** Processor User Manual Part 2

16 X system memory allocation

The memory allocation problem of the X system describes the video memory allocation problem. In 3A-690e, the graphics card is set
It is inside the North Bridge 690E and is a device of PCIE. The display core of the graphics card is ATI X1250, integrated inside
128M video memory also supports shared video memory. The maximum shared video memory can be up to 128M. The display process of the graphics card is this
Similar: The CPU transfers the instructions and data related to drawing to the graphics card through the PCIE bus. GPU according to CPU requirements
To complete the image processing process and save the final image data in the video memory.

Figure 16-1 Graphics card processing image display process

For the case of using independent video memory, since the video memory is inside the graphics card, the process becomes
The card can directly write the content to the video memory. For shared video memory, the process is a bit more complicated, the GPU will write
After the video memory address is told to the CPU, there will be two situations: 1. The video memory address is a PCI space address. CPU will put
The content is directly written to the PCIE bus, and PCIE performs another address conversion to the actual memory address, which is
Is the physical address; second, the memory address is the memory address. At this time, the CPU will directly write the content to the video memory in the memory
position. Obviously, for the method of shared memory, the second scheme will be more efficient.

The following specifically describes how to implement the second way of sharing video memory in PMON Chinese. To expand our PCI
Space, we use TLB mapping. In bonito.h of PMON, we define it like this:

```
#define BONITO_PCILO_BASE                    0x10000000
#define BONITO_PCILO_BASE_VA                 0xd0000000
#define BONITO_PCIIO_BASE                    0x18000000
#define BONITO_PCIIO_BASE_VA                 0xb8000000
```

It means that the 256M PCI space (0x10000000 ~ 0x20000000) is divided into two parts:

0x10000000 ~ 0x17ffffff is mem space, 0x18000000 ~ 0x20000000 is IO space. And mem space

The virtual address 0xd0000000 to the physical address 0x10000000 is converted by manually filling the TLB. in

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

0x78000000, this is also achieved through TLB mapping. The code for TLB mapping is as follows:

151


```
        li t0, 15
        li t3, 0xf0000000 # entry_hi, the starting address of the virtual address to be mapped
        li a0, 0x3f000000
        bleu msize, a0, 1f // judge whether the memory is 1G or 2G, if it is 1G, jump to 1 to execute
        nop
        li t4, 0x0000f000 // 2G situation, turn 0xf0000000 to 0xf0000000
        b 2f                                    // jump to 2 to execute
        nop
1:
        li t4, 0x00007000 // 1G, turn 0xf0000000 to 0x70000000
2:
        .set mips64
        dsll t4, t4, 10
        .set mips3
        ori t4, t4, 0x1f
        li t5, (0x1000000 >> 6)                 # 16M stride, a page is 16M
        li t6, 0x2000000                        # VPN2 32M stride
        .set mips64
1:
        dmtc0 t3, COP_0_TLB_HI                  // Fill in the TLB entry
        daddu t3, t3, t6
        dmtc0 t4, COP_0_TLB_LO0
        daddu t4, t4, t5
        dmtc0 t4, COP_0_TLB_LO1
        daddu t4, t4, t5
        .set mips3
        addiu t1, t0, 16
        mtc0 t1, COP_0_TLB_INDEX                # 16MB page
```

```
nop
nop
```

152

Page 165


```
nop
nop
nop
tlbwi
bnez t0, 1b                    // Total mapping size 16 * 16 = 256M
addiu t0, t0, -1
```

In this way, when the graphics card accesses the video memory, we can all use the address of 0xf8000000 as the starting address of the video memory. In this way, it actually uses the upper part of the memory. This is the structure of ati_nb_cfg in rs690_struct.c

Set in the body, set system_memory_tom_lo to 0x1000M, which is 0x100000000, if the video memory is

128M, then the starting address is the address of 0x1000M-128M = 0xf8000000, and this address is what is said above

The virtual address of the video memory can be obtained according to the TLB mapping. At this point, the graphics card directly accesses the display

This is how deposit is achieved.

153