



Instruction manual of Loongson 3A4000 processor register

# Multi-core processor architecture, register description and System Software Programming Guide V1.5

 ***Loongson Zhongke Technology Co. LTD***



**Copyright statement**

The copyright of this document belongs to Loongson Technology Co., LTD., and all rights are reserved. No company or individual may publish, reprint or otherwise distribute any part of this document to third parties without written permission. Otherwise, it will be held legally liable.

**disclaimer**

This document only provides phased information. The contents can be updated according to the actual situation of the product at any time without prior notice. The Company shall not be liable for any direct or indirect losses caused by improper use of documents.

**Loongson Zhongke Technology Co. LTD**

Loongson Technology Corporation Limited

Address: Building 2, Loongson Industrial Park, Zhongguancun Environmental Protection Technology Demonstration Park, Haidian District, Beijing

Building No.2, Loongson Industrial Park,

Zhongguancun Environmental Protection Park, Haidian District, Beijing

Tel: 010-62546668 Fax: 010-62600826

### Reading guide

The Instruction manual for The Processor Registers of Loongson 3A4000 introduces the architecture and registers of the multi-core processor of Loongson 3A4000, and gives a detailed description of the chip system architecture, functions and configurations of main modules, register lists and bit fields.

# Revision history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document update record** | | | The document name: | | Instruction manual of Loongson 3A4000 processor register |
| The version number | | V1.5 |
| founder | | Chip R&d Department |
| Creation date | | 2019-12-20 |
| **Update history** | | | | | |
| **The serial number** | **Updated date** | **The version number** | | **Update the content** | |
| 1 | 2018-05-08 | V0.1 | | The initial release | |
| 2 | 2018-05-28 | V0.2 | | Update various chip configuration registers | |
| 3 | 2018-06-02 | V0.3 | | Add the section of frequency division control and update GPIO, UART, I2C and SPI | |
| 4 | 2018-06-04 | V0.4 | | Modify the routing | |
| 5 | 2018-06-05 | V0.5 | | Modify memory controller section to add software clock system | |
| 6 | 2018-06-13 | V0.6 | | Add clock description; Add GPIO interrupt description; Increase temperature status detection; Add HT interruption description; Add 3A3000 compatibility description; Modify EXTIOI to support 8-node interconnection; Modify the processor core description. | |
| 7 | 2018-09-11 | V0.7 | | Update the configuration register list, adding some register field descriptions. | |
| 8 | 2018-09-13 | V0.8 | | Update the stable Clock structure description. | |
| 9 | 2018-10-26 | V0.9 | | Update the DDR section | |
| 10 | 2019-02-19 | V1.0 | | Internal debug version | |
| 11 | 2019-05-29 | V1.1 | | Update configuration register, temperature sensor, stable Counter, extension interrupt, Scache interrupt, processor CPUCFG section description | |
| 12 | 2019-07-01 | V1.2 | | Some statement errors have been fixed | |
| 13 | 2019-09-11 | V1.3 | | Chapter 4 adds part of characteristic control description  11.1.1 Fixed int\_EDGE address offset | |
| 14 | 2019-10-11 | V1.4 | | Modify the temperature sensor register description | |
| 15 | 2019-12-17 | V1.5 | | Fixed some formatting errors | |

##### Feedback of manual information: service@loongson.cn

Problem feedback web site, http://bugs.loongnix.org/, also can be submitted to our chip production



###### Instruction manual of Longson 3A4000 processor register

Problems in the process of product use, and obtain technical support.

**directory**

目录

[Multi-core processor architecture, register description and System Software Programming Guide V1.5 1](#_Toc44342065)

[Reading guide 3](#_Toc44342066)

[Revision history 4](#_Toc44342067)

[1 An overview of the 20](#_Toc44342068)

[1.1 Introduction to loong chip series processor 20](#_Toc44342069)

[1.2 Introduction to Loongson 3A4000 21](#_Toc44342070)

[2 System configuration and control 24](#_Toc44342071)

[2.1 Chip operation mode 24](#_Toc44342072)

[2.2 Control pin description 25](#_Toc44342073)

[3 Physical address space distribution 27](#_Toc44342074)

[3.1 Physical address space distribution between nodes 27](#_Toc44342075)

[3.2 Physical address space distribution within nodes 28](#_Toc44342076)

[3.3 Address routing distribution and configuration 30](#_Toc44342077)

[4 Chip configuration register 38](#_Toc44342078)

[Version 4.1 Register (0x0000) 38](#_Toc44342079)

[4.2 Chip Feature Register (0x0008) 38](#_Toc44342080)

[4.3 Manufacturer name (0x0010) 39](#_Toc44342081)

[4.4 Chip Name (0x0020) 40](#_Toc44342082)

[4.5 Function Setting register (0x0180) 40](#_Toc44342083)

[4.6 Pin Drive Setup register (0x0188) 42](#_Toc44342084)

[4.7 Functional Sampling register (0x0190) 42](#_Toc44342085)

[4.8 Temperature sampling register (0x0198) 42](#_Toc44342086)

[4.9 Bias configuration register (0x01A0) 43](#_Toc44342087)

[4.10 Frequency configuration register (0x01B0) 44](#_Toc44342088)

[4.11 Processor core frequency division setting register (0x01D0) 46](#_Toc44342089)

[4.12 Processor core reset Control Register (0x01D8) 48](#_Toc44342090)

[4.13 Route Setup register (0x0400) 48](#_Toc44342091)

[4.14 Other Functions Set register (0x0420) 49](#_Toc44342092)

[4.15 Celsius temperature register (0x0428) 50](#_Toc44342093)

[4.16 SRAM Adjustment register (0x0430) 50](#_Toc44342094)

[4.17 FUSE0 Observation register (0x0460) 51](#_Toc44342095)

[4.18 FUSE1 Observation register (0x0470) 52](#_Toc44342096)

[5 Chip clock frequency division and enable control 53](#_Toc44342097)

[5.1 Chip module clock introduction 53](#_Toc44342098)

[5.2 Processor core frequency division and enabling control 54](#_Toc44342099)

[5.3 Node clock frequency division and enabling control 57](#_Toc44342100)

[5.4 HT controller frequency division and enabling control 60](#_Toc44342101)

[5.5 Stable Counter split frequency and enable control 61](#_Toc44342102)

[6 Software clock system 63](#_Toc44342103)

[6.1 Stable Counter 63](#_Toc44342104)

[6.1.1 Configuration address of Stable Timer 63](#_Toc44342105)

[6.2 The Node Counter 67](#_Toc44342106)

[6.3 Summary of clock system 68](#_Toc44342107)

[7 GPIO control 69](#_Toc44342108)

[7.1 Output enable register (0x0500) 69](#_Toc44342109)

[7.2 Input and Output register (0x0508) 69](#_Toc44342110)

[7.3 Interrupt Control register (0x0510) 69](#_Toc44342111)

[7.4 GPIO pin function multiplexing table 71](#_Toc44342112)

[7.5 GPIO interrupt control 72](#_Toc44342113)

[8 GS464V processor core 74](#_Toc44342114)

[8.1 3A4000 implements instruction set features 75](#_Toc44342115)

[8.2 3A4000 configuration status register access 79](#_Toc44342116)

[9 Shared Cache (SCache) 81](#_Toc44342117)

[10 Interrupt communication between processor cores 85](#_Toc44342118)

[10.1 Access mode by address 85](#_Toc44342119)

[10.2 Configure register instruction access 87](#_Toc44342120)

[11 I/O interrupt 90](#_Toc44342121)

[11.1 Traditional I/O interrupt 90](#_Toc44342122)

[11.2 Extend I/O interrupt 95](#_Toc44342123)

[12 Temperature sensor 103](#_Toc44342124)

[12.1 Real-time temperature sampling 103](#_Toc44342125)

[12.2 High and low temperature interrupt triggers 104](#_Toc44342126)

[12.3 High temperature automatic frequency reduction setting 106](#_Toc44342127)

[12.4 Temperature status detection and control 107](#_Toc44342128)

[12.5 Temperature sensor control 109](#_Toc44342129)

[13 Ddr3/4 SDRAM controller configuration 111](#_Toc44342130)

[13.1 Ddr3/4 SDRAM controller features Overview 111](#_Toc44342131)

[13.2 Ddr3/4 SDRAM read operation protocol 112](#_Toc44342132)

[13.3 Ddr3/4 SDRAM write operation protocol 113](#_Toc44342133)

[13.4 Ddr3/4 SDRAM parameter configuration format 114](#_Toc44342134)

[13.5 Software Programming Guide 128](#_Toc44342135)

[14 HyperTransport controller 140](#_Toc44342136)

[14.1 HyperTransport hardware setup and initialization 140](#_Toc44342137)

[14.2 HyperTransport protocol support 144](#_Toc44342138)

[14.3 HyperTransport interrupt support 146](#_Toc44342139)

[14.4 HyperTransport address window 148](#_Toc44342140)

[14.5 Configuration register 151](#_Toc44342141)

[14.6 The HyperTransport bus concodes access methods for Spaces 211](#_Toc44342142)

[14.7 HyperTransport multiprocessor support 213](#_Toc44342143)

[15 Low speed IO controller configuration 218](#_Toc44342144)

[15.1 UART controller 218](#_Toc44342145)

[15.2 SPI controller 157](#_Toc44342146)

[15.3 I2C controller 167](#_Toc44342147)

[16 3A3000 kernel compatibility 169](#_Toc44342148)

[16.1 Compatible with 3A3000 kernel 169](#_Toc44342149)

[16.2 New feature support 172](#_Toc44342150)

[16.3 Configure register instruction debugging support 175](#_Toc44342151)

Figure orders to record

[Figure 1-1 System Structure 1 of Loongson 3](#_bookmark2)

[FIG. 1-2 Structure of No3 of The Loong Chip 2](#_bookmark3)

[Figure 1-3 Structure of Loongson 3A4000 Chip 3](#_bookmark5)

[Figure 6-1 Stable reset control 35 for multi-chip interconnection](#_bookmark90)

[Figure 8-1 GS464V Structure Figure 42](#_bookmark108)

[FIG. 11-1 Schematic diagram of interrupt routing for longson 3A4000 processor](#_bookmark127)

[Figure 13- 1 DDR3 SDRAM read operation protocol 70](#_bookmark170)

[Figure 13- 2 DDR3 SDRAM write protocol 71](#_bookmark172)

[Figure 14-1 Configuration access of HT protocol in Longson 3A4000](#_bookmark330)

[FIG. 14-2 Four-chip Longshon No.3 interconnection structure 144](#_bookmark332)

[FIG. 14-3 Eight-chip Longshon No.3 Interconnection structure 144](#_bookmark333)

[Figure 14-4 Two-chip Longson no.3 8-bit interconnection Structure 145](#_bookmark334)

[Figure 14-5 Two-chip Longshon No.3 16-bit interconnection structure 146](#_bookmark335)

Table item record

[Table 2-1 Control pin description 5](#_bookmark9)

[Table 3-1 Global address distribution of the system at node level 7](#_bookmark12)

[Table 3-2 Address distribution in nodes 8](#_bookmark14)

[Table 3-3 SCID\_SEL address bit setting 8](#_bookmark15)

[Table 3-4 physical address distribution of 44 bits in node 8](#_bookmark16)

[Table 3-5 MMAP field corresponding to the space access property 9](#_bookmark18)

[Table 3-6 Address window register Table 9](#_bookmark19)

[Table 3-7 Correspondence between the slave device number and the module 15](#_bookmark20)

[Table 3-8 MMAP field corresponding to the space access property 15](#_bookmark21)

[Table 4-1 register 17](#_bookmark24)

[Table 4-2 Chip feature register 17](#_bookmark26)

[Table 4-3 Manufacturer name register 18](#_bookmark28)

[Table 4-4 Chip name register 18](#_bookmark30)

[Table 4-5 Function Settings register 18](#_bookmark32)

[Table 4-6 Pin drive setup register 19](#_bookmark34)

[Table 4-7 Functional Sampling register 19](#_bookmark36)

[Table 4-8 Temperature sampling register 19](#_bookmark38)

[Table 4-9 Bias setting register 20](#_bookmark40)

[Table 4-10 Clock software frequency doubling set register 21](#_bookmark42)

[Table 4-11 Memory clock software frequency doubling set register 21](#_bookmark43)

[Table 4-12 Processor core software frequency division setting register 22](#_bookmark45)

[Table 4-13 Processor core software frequency division set Register 23](#_bookmark47)

[Table 4-14 Chip routing setup register 23](#_bookmark49)

[Table 4-15 Other Function Settings register 24](#_bookmark51)

[Table 4-16 Temperature observation register 25](#_bookmark53)

[Table 4-17 Processor core SRAM adjustment register 25](#_bookmark55)

[Table 4-18 FUSE observation register 25](#_bookmark57)

[Table 4-19 FUSE observation register 26](#_bookmark59)

[Table 5-1 Register 28 for frequency division setting of processor core software](#_bookmark64)

[Table 5-2 Register 28 for other Function Settings](#_bookmark65)

[Table 5-3 Other Function Settings register 29](#_bookmark67)

[Table 5-4 Processor core private divider register 29](#_bookmark68)

[Table 5-5 Function Settings register 30](#_bookmark71)

[Table 5-6 Register 30 for other Functions](#_bookmark72)

[Table 5-7 High temperature and frequency drop control register description 30](#_bookmark74)

[Table 5-8 Function Settings register 31](#_bookmark76)

[Table 5-9 Other Function Settings Register 32](#_bookmark77)

[Table 5-10 Other Function Settings Register 32](#_bookmark79)

[Table 5-11 GPIO output enable register 32](#_bookmark80)

[Table 6-1 Address access method 33](#_bookmark84)

[Table 6-2 Configuration register instruction access mode](#_bookmark85)

[Table 6-3 The meanings of registers 34](#_bookmark86)

[Table 6-4 Other Function Settings Register 34](#_bookmark88)

[Table 6-5 Node Counter register 36](#_bookmark93)

[Table 7-1 Output enable register 37](#_bookmark98)

[Table 7-2 I/O register 37](#_bookmark100)

[Table 7-3 Interrupt control register 37](#_bookmark102)

[Table 7-4 GPIO function reuse Table 38](#_bookmark104)

[Table 7-5 Interrupt control register 39](#_bookmark106)

[Table 8-1 Configuration information of instruction set functions implemented by 3A4000 is listed as 43](#_bookmark110)

[Table 8-2 List of internal configuration status registers](#_bookmark112)

[Table 9-1 Shared Cache lock window register configuration 48](#_bookmark114)

[Table 10-1. Registers and their functions related to interrupt between processor cores](#_bookmark117)

[Table 10- 20 Intercore interrupt and communication registers for processor core No. 0](#_bookmark118)

[Table 10-3 Intercore interrupt and communication registers for Processor core No. 1](#_bookmark119)

[Table 10-4 Intercore interrupt and communication registers for No. 2 processor core 51](#_bookmark120)

[Table 10-5 Intercore interrupt and communication registers for Processor core No. 3](#_bookmark121)

[Table 10-6 List of interrupt and communication registers between current processor cores](#_bookmark123)

[Table 10-7 Processor inter-core communication register 52](#_bookmark124)

[Table 11-1 Interrupt control register 55](#_bookmark128)

[Table 11-2 IO control register address 56](#_bookmark130)

[Table 11-3 Description of interrupt routing register](#_bookmark131)

[Table 11-4 Interrupt routing register address 56](#_bookmark132)

[Table 11-5 Processor core private interrupt status register 57](#_bookmark134)

[Table 11-6 Register 57 for other Function Settings](#_bookmark136)

[Table 11-7 Extended IO interrupt enable register 58](#_bookmark138)

[Table 11-8 Extension IO interrupt Automatic rotation enable register 58](#_bookmark139)

[Table 11-9 Extended IO interrupt status register 58](#_bookmark140)

[Table 11-10 Extended IO interrupt status register for each processor core 58](#_bookmark141)

[Table 11-11 Description of interrupt pin routing register](#_bookmark142)

[Table 11-12 Interrupt routing register address 59](#_bookmark143)

[Table 11-13 Description of interrupt target processor core routing register 60](#_bookmark144)

[Table 11-14 Interrupt target processor core routing register address 60](#_bookmark145)

[Table 11-15 Interrupt target node mapping mode configuration 60](#_bookmark146)

[Table 11-16 Extended IO interrupt status register for current processor core 61](#_bookmark148)

[Table 11-17 Extended IO interrupt trigger register 61](#_bookmark150)

[Table 12-1 Temperature sampling register description 63](#_bookmark154)

[Table 12-2 Extended IO interrupt trigger register 63](#_bookmark155)

[Table 12-3 High and low temperature interrupt register description 64](#_bookmark157)

[Table 12-4 High temperature and frequency drop control register description 65](#_bookmark159)

[Table 12-5 Temperature status detection and control register description 66](#_bookmark161)

[Table 12-6 Temperature sensor configuration register description 67](#_bookmark163)

[Table 12-7 Temperature sensor data register instruction 67](#_bookmark164)

[Table 12-8 Monitoring points of temperature sensor](#_bookmark165)

[Table 13-1 DDR3/4 Address control signal multiplexing 69](#_bookmark168)

[Table 13-2 Memory controller software visible parameters list 71](#_bookmark175)

[Table 13-3 Memory controller error state observation register 89](#_bookmark186)

[Table 13-4 Memory controller No. 1 error state observation register 90](#_bookmark187)

[Table 14-1 HyperTransport Bus related pin signals 94](#_bookmark190)

[Table 14-2 Commands that the HyperTransport receiver can receive](#_bookmark192)

[Table 14-3 Command to be sent out in both modes 96](#_bookmark193)

[Table 14-4 Other Function Settings register 98](#_bookmark198)

[Table 14-5 Distribution of address Windows for the four default HyperTransport interfaces](#_bookmark201)

[Table 14-6 Distribution of address Windows within the HyperTransport interface of Loongson 3 processor](#_bookmark202)

[Table 14-7 Address window 99 provided in longson 3A4000 processor HyperTransport interface](#_bookmark204)

[Table 14-8 Bus Reset Control register definition 104](#_bookmark207)

[Table 14-9 Command, Capabilities Pointer, Capability ID register definition 104](#_bookmark209)

[Table 14-10 Link Config, Link Control register definition 105](#_bookmark210)

[Table 14-11 Revision ID, Link Freq, Link Error, Link Freq Cap register definition 106](#_bookmark211)

[Table 14-12 Definition of Feature Capability Register](#_bookmark212)

[Table 14-13 Error Retry control register 107](#_bookmark214)

[Table 14-14 Retry Count register 108](#_bookmark216)

[Table 14-15 Revision ID register 108](#_bookmark218)

[Table 14-16 Interrupt register defines 108](#_bookmark220)

[Table 14-17 Dataport register definitions 109](#_bookmark221)

[Table 14-18 IntrInfo register definition (1) 109](#_bookmark222)

[Table 14-19 IntrInfo register definition (2) 109](#_bookmark223)

[Table 14-20 HT bus interrupt vector register definition (1) 111](#_bookmark225)

[Table 14-21 HT bus interrupt vector register definition (2) 111](#_bookmark226)

[Table 14-22 HT bus interrupt vector register definition (3) 111](#_bookmark227)

[Table 14-23 HT bus interrupt vector register definition (4) 111](#_bookmark228)

[Table 14-24 HT bus interrupt vector register definition (6) 112](#_bookmark229)

[Table 14-25 HT bus interrupt vector register definition (7) 112](#_bookmark230)

[Table 14-26 HT bus interrupt vector register definition (8) 112](#_bookmark231)

[Table 14-27 HT bus interrupt enable register definition (1) 113](#_bookmark233)

[Table 14-28 HT bus interrupt enable register definition (2) 114](#_bookmark234)

[Table 14-29 HT bus interrupt enable register definition (3) 114](#_bookmark235)

[Table 14-30 HT bus interrupt enable register definition (4) 114](#_bookmark236)

[Table 14-31 HT bus interrupt enabled register definition (5) 114](#_bookmark237)

[Table 14-32 HT bus interrupt enable register definition (6) 114](#_bookmark238)

[Table 14-33 HT bus interrupt enabled register definition (7) 115](#_bookmark239)

[Table 14-34 HT bus interrupt enable register definition (8) 115](#_bookmark240)

[Table 14-35 Link Train register 115](#_bookmark242)

[Table 14-36 HT bus receive address window 0 enable (external access) register definition 116](#_bookmark244)

[Table 14-37 HT bus receive address window 0 Base address (external access) register definition 117](#_bookmark245)

[Table 14-38 HT bus receiver address window 1 enables (externally accessible) register definition 117](#_bookmark246)

[Table 14-39 HT bus receiver address window 1 Base address (external access) register definition 117](#_bookmark247)

[Table 14-40 HT bus receive address window 2 enable (external access) register definition 118](#_bookmark248)

[Table 14-41 HT bus receiver address window 2 Base address (external access) register definition 118](#_bookmark249)

[Table 14-42 HT bus receive address window 3 enable (external access) register definition 118](#_bookmark250)

[Table 14-43 HT bus receiver address window 3 Base address (external access) register definition 119](#_bookmark251)

[Table 14-44 HT bus receiver address window 4 enables (externally accessible) register definition 119](#_bookmark252)

[Table 14-45 HT bus receiver address window 4 Base address (external access) register definition 119](#_bookmark253)

[Table 14-46 Configuration space extension address translation register definition 120](#_bookmark255)

[Table 14-47 Extended address translation register definition 120](#_bookmark256)

[Table 14-48 HT bus POST address window 0 enabled (internal access) 121](#_bookmark258)

[Table 14-49 HT bus POST address window 0 base address (internal access) 121](#_bookmark259)

[Table 14-50 HT bus POST address window 1 enabling (internal access) 121](#_bookmark260)

[Table 14-51 HT bus POST address window 1 Base address (internal access) 122](#_bookmark261)

[Table 14-52 HT bus can prefetch address window 0 enabling (internal access) 122](#_bookmark263)

[Table 14-53 HT bus prefetchable address window 0 base address (internal access) 122](#_bookmark264)

[Table 14-54 THE HT bus can prefetch address window 1 enabling (internal access) 123](#_bookmark265)

[Table 14-55 HT bus can prefetch address window 1 base address (internal access) 123](#_bookmark266)

[Table 14-56 HT bus Uncache Address window 0 enables (internal access) 123](#_bookmark268)

[Table 14-57 HT Bus Uncache Address window 0 base address (internal access) 124](#_bookmark269)

[Table 14-58 HT bus Uncache Address window 1 enabling (internal access) 124](#_bookmark270)

[Table 14-59 HT bus Uncache Address window 1 Base address (internal access) 125](#_bookmark271)

[Table 14-60 HT bus Uncache Address window 2 enables (internal access) 125](#_bookmark272)

[Table 14-61 HT Bus Uncache Address window 2 Base address (internal access) 125](#_bookmark273)

[Table 14-62 HT bus Uncache Address window 3 enables (internal access) 125](#_bookmark274)

[Table 14-63 HT Bus Uncache Address window 3 Base address (internal access) 126](#_bookmark275)

[Table 14-64 HT bus P2P address window 0 enable (external access) register definition 126](#_bookmark277)

[Table 14-65 HT bus P2P address window 0 base address (external access) register definition 127](#_bookmark278)

[Table 14-66 HT bus P2P address window 1 enables (externally accessible) register definition 127](#_bookmark279)

[Table 14-67 HT bus P2P address window 1 base address (external access) register definition 127](#_bookmark280)

[Table 14-68 Controller parameter configuration register 0 defines 127](#_bookmark282)

[Table 14-69 Controller parameter configuration register 1 defines 128](#_bookmark283)

[Table 14-70 receives diagnostic register 130](#_bookmark285)

[Table 14-71 PHY status register 130](#_bookmark287)

[Table 14-72 sends the cache size register 131](#_bookmark289)

[Table 14-73 Data send cache size register 131](#_bookmark291)

[Table 14-74 Send cache debug register 132](#_bookmark293)

[Table 14-75 Receive buffer initial register 133](#_bookmark295)

[Table 14-76 Short timeout register of Training 0](#_bookmark297)

[Table 14-77 Training 0 timeout long count register 134](#_bookmark299)

[Table 14-78 Register 134 of Training 1](#_bookmark301)

[Table 14-79 Training 2 counting register 135](#_bookmark303)

[Table 14-80 Training 3 counting register 135](#_bookmark305)

[Table 14-81 Software frequency configuration register 136](#_bookmark307)

[Table 14-82 Impedance matching control register 137](#_bookmark309)

[Table 14-83 PHY configuration register 137](#_bookmark311)

[Table 14-84 Link Initialization debug register 139](#_bookmark313)

[Table 14-85 LDT debug register 1139](#_bookmark315)

[Table 14-86 LDT debug register 2139](#_bookmark316)

[Table 14-87 LDT debug register 3139](#_bookmark317)

[Table 14-88 LDT debug register 4140](#_bookmark318)

[Table 14-89 LDT debug register 5140](#_bookmark319)

[Table 14-90 LDT debug register 5140](#_bookmark320)

[Table 14-91 HT TX POST ID WIN0141](#_bookmark322)

[Table 14-92 HT TX POST ID WIN1141](#_bookmark323)

[Table 14-93 HT TX POST ID WIN2141](#_bookmark324)

[Table 14-94 HT TX POST ID WIN3141](#_bookmark325)

[Table 14-95 HT RX INT TRANS LO142](#_bookmark327)

[Table 14-96 HT RX INT TRANS Hi142](#_bookmark328)

[Table 15-1 SPI controller address space distribution 157](#_bookmark351)

[Table 16-1 Chip feature register 168](#_bookmark379)

[Table 16-2 Manufacturer name register 169](#_bookmark381)

[Table 16-3 Chip name register 169](#_bookmark382)

[Table 16-4 HT RX INT TRANS LO171](#_bookmark386)

[Table 16-5 HT RX INT TRANS Hi171](#_bookmark387)

[Table 16-6 Other Function Settings register 172](#_bookmark388)

[Table 16-7 Processor inter-core communication register 172](#_bookmark390)

## An overview of the

## Introduction to loong chip series processor

The godson processor mainly consists of three series. Loongson No. 1 processor and its IP series are mainly for embedded applications, Loongson No. 2 superstandard processor and its IP series are mainly for desktop applications, and Loongson No. 3 multi-core processor series are mainly for server and high-performance computer applications. According to the needs of the application, part of the loongson 2 can also be oriented towards part of the high-end embedded

Yes, some low-end Longson 3 can also be used for some desktop applications. The three series developed in parallel.

Based on the scalable multi-core interconnection architecture, the Loongson 3 multi-core series processor integrates multiple high-performance processor cores and a large number of 2-level Caches on a single chip, and realizes multi-chip interconnection through high-speed I/O interface to form a larger scale system.

The telescopic interconnection structure adopted by Loongson no. 3 is shown in Figure 1-1 below. The godson 3 and more pieces system interconnection ports with similar to node to implement interconnection structure unit, in which each node is composed of 8 \* 8 cross switch, each cross switch connected four cores and four Shared Cache, and the east (E) south west (W) north (N) (N) interconnection of other nodes in four directions.



**L2 L2 L2 L2**

(a)

**P0, P1 P2, P3**

**8 by 8 switch**

Node 3 and 2-D interconnection structure, (a) node structure, (b) mesh network of 2\*2 connected 16 processors, (c)

The 4\*4 mesh network connects 64 processors.

(c)

(b)

**E S W**

**N**

**E S W**

**N**

Figure 1-1 System structure of No. 3 Loong Chip

The node structure of No.3 is shown in Figure 1-2 below. Each node has two levels of AXI switch to connect processor and share

Cache, memory controller, and IO controller. The first level of AXI cross-switches (called the X1 Switch, or X1 for short) connects the processor to the Shared Cache. The second level cross Switch (called the X2 Switch, or X2 for short) connects the Shared Cache to the memory controller.



**The MC**

**Xcon**

SS SM

MA0MA1MA2MA3

**X2 Switch**

WSES

**S0S1S2**

s5s6s7

**The X1 Switch**

s0s1s2s3s4

m7

m3m4m5m6

m1m2

m0

WM

EM

**P0P1P2P3**

NSNM

**S3**

Figure 1-2 Structure of No. 3 node of the Loong Chip

In each node, the X1 cross switch of up to 8\*8 connects four GS464 processor cores through four Master ports

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

X2 cross-switch connects four Shared caches through four Master ports, at least one Slave port connects to a memory controller, at least one Slave port connects to a cross-switch configuration module (Xconf), which is used to configure the address window of X1 and X2 of this node. You can also connect more memory controllers and IO ports as needed.

## Introduction to Loongson 3A4000

Longshon 3A4000 is a quad-core longshon processor manufactured by a 28nm process with a stable operating frequency of 1.5-2.0GHz. Its main technical features are as follows:

* Four 64-bit quad-emission superstandard GS464V high-performance processor cores are integrated on the chip.
* Integrated 8MB split Shared three-level Cache(composed of 4 individual modules, each with a capacity of 2MB);
* Maintain Cache consistency of multi-core and I/O DMA access through directory protocol;
* Integrated two 64-bit DDR3/4 controllers with ECC and 800MHz on the chip;
* Two 16-bit HyperTransport controllers (HT) are integrated on the chip;
* Each 16-bit HT port is divided into two 8-way HT ports for use.
* Two I2C, one UART, one SPI and 16-channel GPIO interfaces are integrated on the chip.

On the basis of 3A2000/3A3000, the top structure design of Loongson 3A4000 is greatly optimized, and the main improvements are as follows:

* The structure of on-chip interconnection is adjusted and address routing is simplified. RING structure is adopted for the interconnection between IO modules.
* The bandwidth utilization and cross-chip delay of HT controller are optimized.
* Optimized the structure of memory controller, increased the support of MEMORY controller DDR4, and supported memory slot connection acceleration card;
* The register space and access mode are standardized, and the REGISTER access mechanism of CSR is introduced.
* The interrupt controller structure is optimized and the support vector interrupt hardware distribution mechanism is optimized.
* Added 8 - way interconnection support.

The overall architecture of Loongson 3A4000 chip is realized based on multi-level interconnection. The structure is shown in Figure 1-3 below.

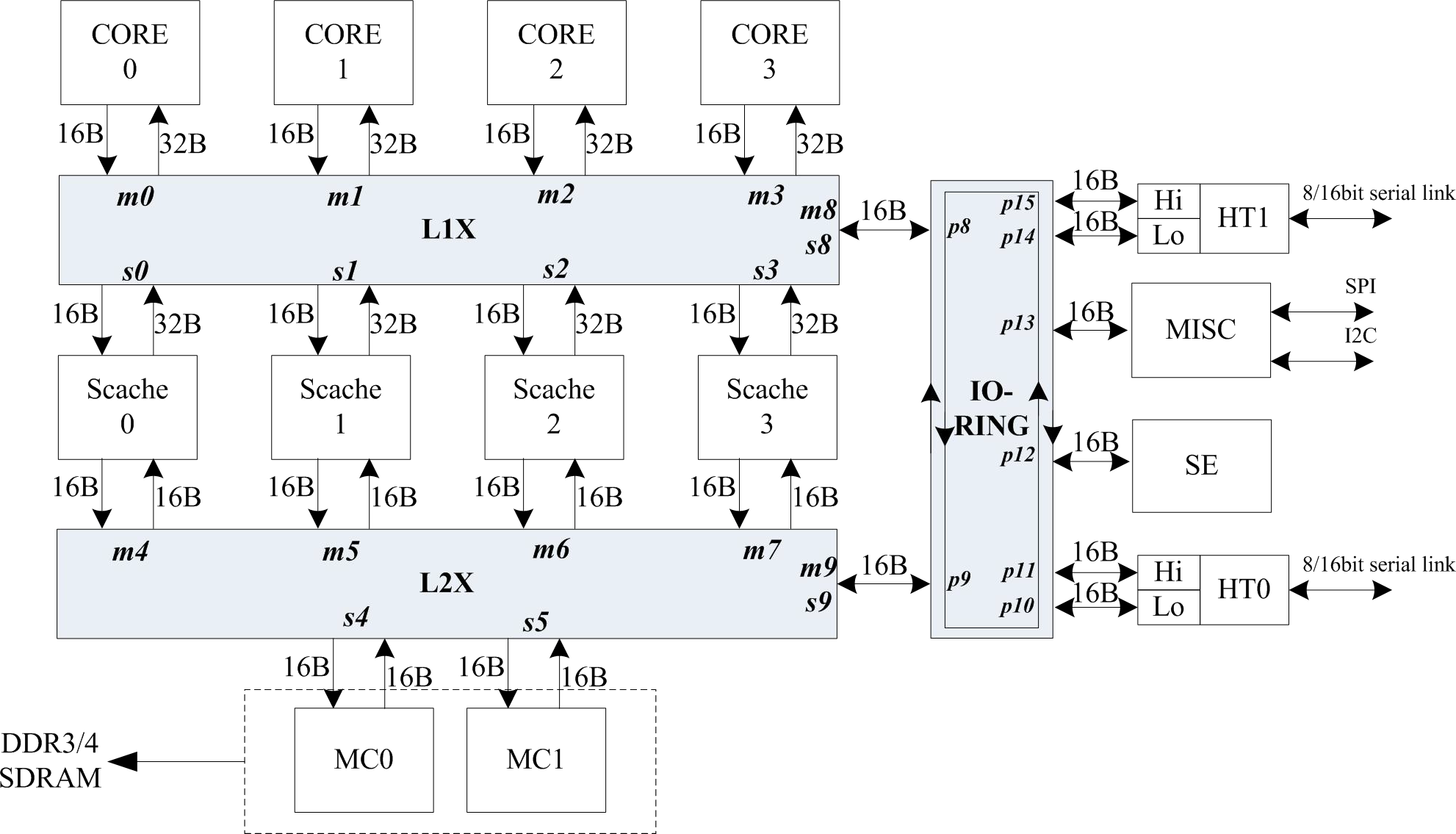


Figure 1-3 Structure of Loongson 3A4000 chip

The first interconnection USES a 5x5 cross switch for connecting four GS464v cores (as the primary device), four Shared Cache modules (as the slave device), and an IO port to the IO-Ring. IO port USES one Master and one Slave.

The second interconnection USES a 5x3 cross-switch to connect four Shared Cache modules (as the primary device), two DDR3/4 memory controllers, and an IO port to the IO-Ring.

Io-ring contains 8 ports, connection includes 4 HT controller, MISC module, SE module and two-stage cross switch. Two HT controllers (LO/HI) share the 16-bit HT bus, which can be used as two 8-bit HT buses, or lo can monopolized the 16-bit HT bus. HT controller integrates a DMA controller, which is responsible for DMA control of IO and maintenance of inter-chip consistency.

The interconnection structure USES a read-write separated data channel with a width of 128 bits and operates at the same frequency as the processor core to provide high-speed on-chip data transmission. In addition, a one-stage cross-switch connects the four processor cores to the scache's read data channel for 256 bits to improve the read bandwidth of the on-chip processor cores accessing scache.

## System configuration and control

## Chip operation mode

According to the structure of the system, longson 3A4000 mainly includes two working modes:

* + - Single chip mode. The system contains only one piece of longson 3A4000, which is a symmetric multiprocessor system (SMP).
    - Multi-chip interconnection mode. The system consists of 2, 4 or 8 loongson 3A4000 connected through HT port to form a non-uniform access multiprocessor system (CC-NUMA).

## Control pin description

Main control pins include DO\_TEST, ICCC\_EN, NODE\_ID[2:0], CLKSEL[9:0], CHIP\_CONFIG[5:0].

Table 2-1 Description of control pins

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| signal | Pull up and down | role | | | |
| DO\_TEST | On the pull | 1 'b1 represents the functional mode  1 'b0 represents the test mode | | | |
| ICCC\_EN | The drop-down | 1 'B1 represents the multi-chip consistent interconnection mode  1 'b0 represents the single-chip mode | | | |
| NODE\_ID [2-0] |  | Represents the processor number in multi-chip consistent interconnection mode | | | |
| CLKSEL [9:0] |  | HT clock control | | | |
|  | signal | role |  |
| CLKSEL [9] | 1 'B1: means that HT PLL clock is controlled by CLKSEL[9:4]  1 'b0: The initial frequency multiplier is 1, which can be reconfigured by the software |
| CLKSEL [8] | 1 'b1: Means HT PLL USES SYSCLK clock input  1 'b0: means HT PLL adopts differential clock input |
| CLKSEL [but] | 2 'b00 means the PHY clock frequency is 1.6ghz  2 'b01 means the PHY clock frequency is 3.2ghz (reference clock is  1.6ghz at 25MHz)  2 'b10 means THE PHY clock frequency is 1.2ghz  2 'b11 means THE PHY clock frequency is 2.4ghz |
| CLKSEL [5] | 1 'b1: Refers to HT PLL clock is in bypass mode, direct  External input 100MHz/25MHz reference clock is used |

MEM clock control (clock frequency should be interface clock 1/2)

Main clock control (maximum frequency of network and processor core)

Chip configuration control

On-chip clock debugging enabler (DCDL)

CHIP\_CONFIG [5]

HT logic function interchange, HT0/HT1 interchange

CHIP\_CONFIG [4]

Ht0/1-hi enters consistency mode by default and is used to support 8-way mutual

even

CHIP\_CONFIG [3]

reserve

CHIP\_CONFIG [2]

Default HT Gen1 mode

CHIP\_CONFIG [1]

SE functional enablement

CHIP\_CONFIG [0]

CHIP\_CONFIG [beat]

SYSCLK (100 MHZ / 25 MHZ)

2 'bl1

Software configuration (PLL clock multiplier 1.6-3.2ghz)

2 'b10

2 GHZ.

2 'b01

1 GHZ.

2 'b00

The output frequency

CLKSEL [1:0]

The 1-reference clock is 25MHz and the 0-reference clock is 100MHz

CLKSEL [4]

|  |  |
| --- | --- |
| CLKSEL (3:2) | The output frequency |
| 2 'b00 | 466 MHZ |
| 2 'b01 | 600 MHZ |
| 2 'b10 | Software configuration (PLL clock multiplier 1.6-3.2ghz) |
| 2 'bl1 | SYSCLK (100 MHZ / 25 MHZ) |

## Physical address space distribution

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

System development extension compatibility. The physical address width of the entire system is 48 bits. According to the address of the high 4 bits, the entire address is empty

In other words, each node is allocated a 44-bit address space.

## Physical address space distribution between nodes

Longson 3A4000 processor can directly use 2/4/8 3A4000 chips to build CC-NUMA system. The processor number of each chip is determined by pin NODEID. The address space distribution of each chip is as follows:

Table 3-1 Global address distribution of the system at node level

|  |  |  |  |
| --- | --- | --- | --- |
| Chip NODEID | Address [47:44] | The starting address | End address |
| 0 | 0 | 0 x0000\_0000\_0000 | 0 x0fff\_ffff\_ffff |
| 1 | 1 | 0 x1000\_0000\_0000 | 0 x1fff\_ffff\_ffff |
| 2 | 2 | 0 x2000\_0000\_0000 | 0 x2fff\_ffff\_ffff |
| 3 | 3 | 0 x3000\_0000\_0000 | 0 x3fff\_ffff\_ffff |
| 4 | 4 | 0 x4000\_0000\_0000 | 0 x4fff\_ffff\_ffff |
| 5 | 5 | 0 x5000\_0000\_0000 | 0 x5fff\_ffff\_ffff |
| 6 | 6 | 0 x6000\_0000\_0000 | 0 x6fff\_ffff\_ffff |
| 7 | 7 | 0 x7000\_0000\_0000 | 0 x7fff\_ffff\_ffff |

When the number of system nodes is less than 8, the nodemask field of the route setting register (0x1FE00400) should be set to ensure that the response can be obtained even without the address of the physical node when guess access occurs. (2 channels: 0x1; Route 4:0x3)

## Physical address space distribution within nodes

The single node 4-core configuration is adopted for the longson 3A4000, so the DDR memory controller integrated with the longson 3A4000 chip and the corresponding address of the HT bus are all contained in the 44-bit address space from 0x0 (including) to 0x1000\_0000\_0000 (excluding). Within each node, the 44-bit address space is further divided among all devices connected within the node, and requests are routed to four Shared Cache modules only if the access type is cached. Depending on the configuration of the chip and system structure, if there is no slave device connected on a port, the corresponding address space is reserved address space and is not allowed to access.

The address space corresponding to each slave terminal of internal interconnection of Loongson 3A4000 chip is as follows:

Table 3-2 Address distribution in nodes

|  |  |  |  |
| --- | --- | --- | --- |
| equipment | Address [43:40] | The starting address of a node | Node end address |
| MC0 | 4 | 0 x400\_0000\_0000 | 0 x4ff\_ffff\_ffff |
| MC1 | 5 | 0 x500\_0000\_0000 | 0 x5ff\_ffff\_ffff |
| SE | c | 0 xc00\_0000\_0000 | 0 xcff\_ffff\_ffff |
| HT0 Lo controller | a. | 0 xa00\_0000\_0000 | 0 xaff\_ffff\_ffff |
| HT0 Hi controller | b | 0 xb00\_0000\_0000 | 0 xbff\_ffff\_ffff |
| HT1 Lo controller | e | 0 xe00\_0000\_0000 | 0 xeff\_ffff\_ffff |
| HT1 Hi controller | f | 0 xf00\_0000\_0000 | 0 xfff\_ffff\_ffff |

Different from the mapping relationship of directional ports, longson 3A4000 can determine the cross-addressing mode of Shared Cache based on the actual application access behavior. The address space corresponding to the four Shared Cache modules in the node is determined according to some two-bit selection bits of address bits, and can be dynamically configured by software. A configuration register called SCID\_SEL is set up to determine the address selection bit, as shown in the table below. By default, the distribution takes the form of [7:6] address hash, in which the [7:6] two digits of the address determine the corresponding Shared Cache number. The register address is 0x3FF00400 or 0x1fe00400.

Table 3-3 SCID\_SEL address bit Settings

|  |  |  |  |
| --- | --- | --- | --- |
| SCID\_SEL | Address bit selection | SCID\_SEL | Address bit selection |
| 4 'h0 | 7:6 | 4 'h8 | " |
| 4 'h1 | 9:8 | 4 'h9 | Thus for |
| 4 'h2 | " | 4 'ha | But after |
| 4 'h3 | She answered | 4 'hb | then |
| 4 'h4 | The lowest | 4 'hc | charm |
| 4 'h5 | " | 4 'hd | 33:32 |
| 4 'h6 | 7 | 4 'he | " |
| 4 'h7 | mark | 4 'hf | meanwhile |

The default distribution of the internal 44-bit physical address of each node in Longson 3A4000 processor is shown in the following table:

Table 4-4 Physical address distribution in nodes 3-4

|  |  |  |
| --- | --- | --- |
| Address range | To access attributes | destination |
| Addr [43:40] = = 4 "ha | Local node,uncache | HT0\_LO |
| Addr [43:40] = = '4 hb | Local node,uncache | HT0\_HI |
| Addr [43:40] = = 4 'hc | Local node,uncache | SE |
| 'he addr [43:40] = = 4 | Local node,uncache | HT1\_LO |
| Addr [43:40] = = '4 hf | Local node,uncache | HT1\_HI |
| 0x10000000- 0x1FFFFFFf, 0x3FF00000-0x3FF0FFff (closed) | Local node,uncache | MISC |
| Mc Interleave is 0 and is not the above address | Local node,uncache | MC0 |
| Mc Interleave is 1 and is not the above address | Local node,uncache | MC1 |
| Scache interleave is 0(address bit selection determined by scID\_sel) | Local node,cache | Scache0 |
| Scache interleave is 1(address bit selection determined by SCID\_sel) | Local node,cache | Scache1 |
| Scache interleave is 2(address bit selection determined by SCID\_sel) | Local node,cache | Scache2 |
| Scache interleave is 3(address bit selection determined by SCID\_sel) | Local node,cache | Scache3 |

## Address routing distribution and configuration

The routing of Loongson 3A4000 is mainly realized through the two-stage cross switch and IO-ring of the system. The software can carry out routing configuration for the requests received by each Master port. Each Master port has 8 address Windows, which can complete the target routing selection of 8 address Windows. Each address window is composed of three 64-bit registers, BASE, MASK and MMAP. The BASE is aligned with K bytes. MASK adopts a format similar to network MASK in which the high digit is 1. The low four-digit MMAP represents the number corresponding to the target Slave port, MMAP[4] represents the enabled point, MMAP[5] represents the enabled block, MMAP[6] represents the enabled window.

Table 3-5 MMAP field corresponding to the space access properties

|  |  |  |  |
| --- | --- | --- | --- |
| [7] | [6] | [5] | [4] |
| The window can make | Allows interleaving access to SCACHE/ memory | Allow the block read | Allowed to take to |

Window hit formula :(IN\_ADDR & MASK) == BASE

As the default route is fixed, the configuration window is closed when starting power on, and the system software is required to enable configuration of the loongson no. 3.

SCACHE/ memory interleave access configuration enabled, Slave number is only valid when 0 or 4. Zero represents routing to SCACHE and it is up to SCID\_SEL to decide how to interleave access across the four SCACHE. A 4 represents routing to memory, with interleave\_bit deciding how to interleave access between the two MCS.

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

Table 3-6 Address window register table

|  |  |  |  |
| --- | --- | --- | --- |
| address | register | address | register |
| 0 x3ff0\_2000 | CORE0\_WIN0\_BASE | 0 x3ff0\_2100 | CORE1\_WIN0\_BASE |
| 0 x3ff0\_2008 | CORE0\_WIN1\_BASE | 0 x3ff0\_2108 | CORE1\_WIN1\_BASE |
| 0 x3ff0\_2010 | CORE0\_WIN2\_BASE | 0 x3ff0\_2110 | CORE1\_WIN2\_BASE |
| 0 x3ff0\_2018 | CORE0\_WIN3\_BASE | 0 x3ff0\_2118 | CORE1\_WIN3\_BASE |
| 0 x3ff0\_2020 | CORE0\_WIN4\_BASE | 0 x3ff0\_2120 | CORE1\_WIN4\_BASE |
| 0 x3ff0\_2028 | CORE0\_WIN5\_BASE | 0 x3ff0\_2128 | CORE1\_WIN5\_BASE |
| 0 x3ff0\_2030 | CORE0\_WIN6\_BASE | 0 x3ff0\_2130 | CORE1\_WIN6\_BASE |
| 0 x3ff0\_2038 | CORE0\_WIN7\_BASE | 0 x3ff0\_2138 | CORE1\_WIN7\_BASE |
| 0 x3ff0\_2040 | CORE0\_WIN0\_MASK | 0 x3ff0\_2140 | CORE1\_WIN0\_MASK |
| 0 x3ff0\_2048 | CORE0\_WIN1\_MASK | 0 x3ff0\_2148 | CORE1\_WIN1\_MASK |
| 0 x3ff0\_2050 | CORE0\_WIN2\_MASK | 0 x3ff0\_2150 | CORE1\_WIN2\_MASK |
| 0 x3ff0\_2058 | CORE0\_WIN3\_MASK | 0 x3ff0\_2158 | CORE1\_WIN3\_MASK |
| 0 x3ff0\_2060 | CORE0\_WIN4\_MASK | 0 x3ff0\_2160 | CORE1\_WIN4\_MASK |
| 0 x3ff0\_2068 | CORE0\_WIN5\_MASK | 0 x3ff0\_2168 | CORE1\_WIN5\_MASK |
| 0 x3ff0\_2070 | CORE0\_WIN6\_MASK | 0 x3ff0\_2170 | CORE1\_WIN6\_MASK |
| 0 x3ff0\_2078 | CORE0\_WIN7\_MASK | 0 x3ff0\_2178 | CORE1\_WIN7\_MASK |
| 0 x3ff0\_2080 | CORE0\_WIN0\_MMAP | 0 x3ff0\_2180 | CORE1\_WIN0\_MMAP |
| 0 x3ff0\_2088 | CORE0\_WIN1\_MMAP | 0 x3ff0\_2188 | CORE1\_WIN1\_MMAP |
| 0 x3ff0\_2090 | CORE0\_WIN2\_MMAP | 0 x3ff0\_2190 | CORE1\_WIN2\_MMAP |
| 0 x3ff0\_2098 | CORE0\_WIN3\_MMAP | 0 x3ff0\_2198 | CORE1\_WIN3\_MMAP |
| 0 x3ff0\_20a0 | CORE0\_WIN4\_MMAP | 0 x3ff0\_21a0 | CORE1\_WIN4\_MMAP |
| 0 x3ff0\_20a8 | CORE0\_WIN5\_MMAP | 0 x3ff0\_21a8 | CORE1\_WIN5\_MMAP |
| 0 x3ff0\_20b0 | CORE0\_WIN6\_MMAP | 0 x3ff0\_21b0 | CORE1\_WIN6\_MMAP |
| 0 x3ff0\_20b8 | CORE0\_WIN7\_MMAP | 0 x3ff0\_21b8 | CORE1\_WIN7\_MMAP |
| 0 x3ff0\_2200 | CORE2\_WIN0\_BASE | 0 x3ff0\_2300 | CORE3\_WIN0\_BASE |
| 0 x3ff0\_2208 | CORE2\_WIN1\_BASE | 0 x3ff0\_2308 | CORE3\_WIN1\_BASE |
| 0 x3ff0\_2210 | CORE2\_WIN2\_BASE | 0 x3ff0\_2310 | CORE3\_WIN2\_BASE |
| 0 x3ff0\_2218 | CORE2\_WIN3\_BASE | 0 x3ff0\_2318 | CORE3\_WIN3\_BASE |
| 0 x3ff0\_2220 | CORE2\_WIN4\_BASE | 0 x3ff0\_2320 | CORE3\_WIN4\_BASE |
| 0 x3ff0\_2228 | CORE2\_WIN5\_BASE | 0 x3ff0\_2328 | CORE3\_WIN5\_BASE |
| 0 x3ff0\_2230 | CORE2\_WIN6\_BASE | 0 x3ff0\_2330 | CORE3\_WIN6\_BASE |
| 0 x3ff0\_2238 | CORE2\_WIN7\_BASE | 0 x3ff0\_2338 | CORE3\_WIN7\_BASE |
| 0 x3ff0\_2240 | CORE2\_WIN0\_MASK | 0 x3ff0\_2340 | CORE3\_WIN0\_MASK |
| 0 x3ff0\_2248 | CORE2\_WIN1\_MASK | 0 x3ff0\_2348 | CORE3\_WIN1\_MASK |
| 0 x3ff0\_2250 | CORE2\_WIN2\_MASK | 0 x3ff0\_2350 | CORE3\_WIN2\_MASK |
| 0 x3ff0\_2258 | CORE2\_WIN3\_MASK | 0 x3ff0\_2358 | CORE3\_WIN3\_MASK |
| 0 x3ff0\_2260 | CORE2\_WIN4\_MASK | 0 x3ff0\_2360 | CORE3\_WIN4\_MASK |
| 0 x3ff0\_2268 | CORE2\_WIN5\_MASK | 0 x3ff0\_2368 | CORE3\_WIN5\_MASK |
| 0 x3ff0\_2270 | CORE2\_WIN6\_MASK | 0 x3ff0\_2370 | CORE3\_WIN6\_MASK |
| 0 x3ff0\_2278 | CORE2\_WIN7\_MASK | 0 x3ff0\_2378 | CORE3\_WIN7\_MASK |
| 0 x3ff0\_2280 | CORE2\_WIN0\_MMAP | 0 x3ff0\_2380 | CORE3\_WIN0\_MMAP |
| 0 x3ff0\_2288 | CORE2\_WIN1\_MMAP | 0 x3ff0\_2388 | CORE3\_WIN1\_MMAP |
| 0 x3ff0\_2290 | CORE2\_WIN2\_MMAP | 0 x3ff0\_2390 | CORE3\_WIN2\_MMAP |
| 0 x3ff0\_2298 | CORE2\_WIN3\_MMAP | 0 x3ff0\_2398 | CORE3\_WIN3\_MMAP |
| 0 x3ff0\_22a0 | CORE2\_WIN4\_MMAP | 0 x3ff0\_23a0 | CORE3\_WIN4\_MMAP |
| 0 x3ff0\_22a8 | CORE2\_WIN5\_MMAP | 0 x3ff0\_23a8 | CORE3\_WIN5\_MMAP |
| 0 x3ff0\_22b0 | CORE2\_WIN6\_MMAP | 0 x3ff0\_23b0 | CORE3\_WIN6\_MMAP |
| 0 x3ff0\_22b8 | CORE2\_WIN7\_MMAP | 0 x3ff0\_23b8 | CORE3\_WIN7\_MMAP |
| 0 x3ff0\_2400 | SCACHE0\_WIN0\_BASE | 0 x3ff0\_2500 | SCACHE1\_WIN0\_BASE |
| 0 x3ff0\_2408 | SCACHE0\_WIN1\_BASE | 0 x3ff0\_2508 | SCACHE1\_WIN1\_BASE |
| 0 x3ff0\_2410 | SCACHE0\_WIN2\_BASE | 0 x3ff0\_2510 | SCACHE1\_WIN2\_BASE |
| 0 x3ff0\_2418 | SCACHE0\_WIN3\_BASE | 0 x3ff0\_2518 | SCACHE1\_WIN3\_BASE |
| 0 x3ff0\_2420 | SCACHE0\_WIN4\_BASE | 0 x3ff0\_2520 | SCACHE1\_WIN4\_BASE |
| 0 x3ff0\_2428 | SCACHE0\_WIN5\_BASE | 0 x3ff0\_2528 | SCACHE1\_WIN5\_BASE |
| 0 x3ff0\_2430 | SCACHE0\_WIN6\_BASE | 0 x3ff0\_2530 | SCACHE1\_WIN6\_BASE |
| 0 x3ff0\_2438 | SCACHE0\_WIN7\_BASE | 0 x3ff0\_2538 | SCACHE1\_WIN7\_BASE |
| 0 x3ff0\_2440 | SCACHE0\_WIN0\_MASK | 0 x3ff0\_2540 | SCACHE1\_WIN0\_MASK |
| 0 x3ff0\_2448 | SCACHE0\_WIN1\_MASK | 0 x3ff0\_2548 | SCACHE1\_WIN1\_MASK |
| 0 x3ff0\_2450 | SCACHE0\_WIN2\_MASK | 0 x3ff0\_2550 | SCACHE1\_WIN2\_MASK |
| 0 x3ff0\_2458 | SCACHE0\_WIN3\_MASK | 0 x3ff0\_2558 | SCACHE1\_WIN3\_MASK |
| 0 x3ff0\_2460 | SCACHE0\_WIN4\_MASK | 0 x3ff0\_2560 | SCACHE1\_WIN4\_MASK |
| 0 x3ff0\_2468 | SCACHE0\_WIN5\_MASK | 0 x3ff0\_2568 | SCACHE1\_WIN5\_MASK |
| 0 x3ff0\_2470 | SCACHE0\_WIN6\_MASK | 0 x3ff0\_2570 | SCACHE1\_WIN6\_MASK |
| 0 x3ff0\_2478 | SCACHE0\_WIN7\_MASK | 0 x3ff0\_2578 | SCACHE1\_WIN7\_MASK |
| 0 x3ff0\_2480 | SCACHE0\_WIN0\_MMAP | 0 x3ff0\_2580 | SCACHE1\_WIN0\_MMAP |
| 0 x3ff0\_2488 | SCACHE0\_WIN1\_MMAP | 0 x3ff0\_2588 | SCACHE1\_WIN1\_MMAP |
| 0 x3ff0\_2490 | SCACHE0\_WIN2\_MMAP | 0 x3ff0\_2590 | SCACHE1\_WIN2\_MMAP |
| 0 x3ff0\_2498 | SCACHE0\_WIN3\_MMAP | 0 x3ff0\_2598 | SCACHE1\_WIN3\_MMAP |
| 0 x3ff0\_24a0 | SCACHE0\_WIN4\_MMAP | 0 x3ff0\_25a0 | SCACHE1\_WIN4\_MMAP |
| 0 x3ff0\_24a8 | SCACHE0\_WIN5\_MMAP | 0 x3ff0\_25a8 | SCACHE1\_WIN5\_MMAP |
| 0 x3ff0\_24b0 | SCACHE0\_WIN6\_MMAP | 0 x3ff0\_25b0 | SCACHE1\_WIN6\_MMAP |
| 0 x3ff0\_24b8 | SCACHE0\_WIN7\_MMAP | 0 x3ff0\_25b8 | SCACHE1\_WIN7\_MMAP |
| 0 x3ff0\_2600 | SCACHE2\_WIN0\_BASE | 0 x3ff0\_2700 | SCACHE3\_WIN0\_BASE |
| 0 x3ff0\_2608 | SCACHE2\_WIN1\_BASE | 0 x3ff0\_2708 | SCACHE3\_WIN1\_BASE |
| 0 x3ff0\_2610 | SCACHE2\_WIN2\_BASE | 0 x3ff0\_2710 | SCACHE3\_WIN2\_BASE |
| 0 x3ff0\_2618 | SCACHE2\_WIN3\_BASE | 0 x3ff0\_2718 | SCACHE3\_WIN3\_BASE |
| 0 x3ff0\_2620 | SCACHE2\_WIN4\_BASE | 0 x3ff0\_2720 | SCACHE3\_WIN4\_BASE |
| 0 x3ff0\_2628 | SCACHE2\_WIN5\_BASE | 0 x3ff0\_2728 | SCACHE3\_WIN5\_BASE |
| 0 x3ff0\_2630 | SCACHE2\_WIN6\_BASE | 0 x3ff0\_2730 | SCACHE3\_WIN6\_BASE |
| 0 x3ff0\_2638 | SCACHE2\_WIN7\_BASE | 0 x3ff0\_2738 | SCACHE3\_WIN7\_BASE |
| 0 x3ff0\_2640 | SCACHE2\_WIN0\_MASK | 0 x3ff0\_2740 | SCACHE3\_WIN0\_MASK |
| 0 x3ff0\_2648 | SCACHE2\_WIN1\_MASK | 0 x3ff0\_2748 | SCACHE3\_WIN1\_MASK |
| 0 x3ff0\_2650 | SCACHE2\_WIN2\_MASK | 0 x3ff0\_2750 | SCACHE3\_WIN2\_MASK |
| 0 x3ff0\_2658 | SCACHE2\_WIN3\_MASK | 0 x3ff0\_2758 | SCACHE3\_WIN3\_MASK |
| 0 x3ff0\_2660 | SCACHE2\_WIN4\_MASK | 0 x3ff0\_2760 | SCACHE3\_WIN4\_MASK |
| 0 x3ff0\_2668 | SCACHE2\_WIN5\_MASK | 0 x3ff0\_2768 | SCACHE3\_WIN5\_MASK |
| 0 x3ff0\_2670 | SCACHE2\_WIN6\_MASK | 0 x3ff0\_2770 | SCACHE3\_WIN6\_MASK |
| 0 x3ff0\_2678 | SCACHE2\_WIN7\_MASK | 0 x3ff0\_2778 | SCACHE3\_WIN7\_MASK |
| 0 x3ff0\_2680 | SCACHE2\_WIN0\_MMAP | 0 x3ff0\_2780 | SCACHE3\_WIN0\_MMAP |
| 0 x3ff0\_2688 | SCACHE2\_WIN1\_MMAP | 0 x3ff0\_2788 | SCACHE3\_WIN1\_MMAP |
| 0 x3ff0\_2690 | SCACHE2\_WIN2\_MMAP | 0 x3ff0\_2790 | SCACHE3\_WIN2\_MMAP |
| 0 x3ff0\_2698 | SCACHE2\_WIN3\_MMAP | 0 x3ff0\_2798 | SCACHE3\_WIN3\_MMAP |
| 0 x3ff0\_26a0 | SCACHE2\_WIN4\_MMAP | 0 x3ff0\_27a0 | SCACHE3\_WIN4\_MMAP |
| 0 x3ff0\_26a8 | SCACHE2\_WIN5\_MMAP | 0 x3ff0\_27a8 | SCACHE3\_WIN5\_MMAP |
| 0 x3ff0\_26b0 | SCACHE2\_WIN6\_MMAP | 0 x3ff0\_27b0 | SCACHE3\_WIN6\_MMAP |
| 0 x3ff0\_26b8 | SCACHE2\_WIN7\_MMAP | 0 x3ff0\_27b8 | SCACHE3\_WIN7\_MMAP |
| - | - | 0 x3ff0\_2900 | IO\_L2X\_WIN0\_BASE |
| - | - | 0 x3ff0\_2908 | IO\_L2X\_WIN1\_BASE |
| - | - | 0 x3ff0\_2910 | IO\_L2X\_WIN2\_BASE |
| - | - | 0 x3ff0\_2918 | IO\_L2X\_WIN3\_BASE |
| - | - | 0 x3ff0\_2920 | IO\_L2X\_WIN4\_BASE |
| - | - | 0 x3ff0\_2928 | IO\_L2X\_WIN5\_BASE |
| - | - | 0 x3ff0\_2930 | IO\_L2X\_WIN6\_BASE |
| - | - | 0 x3ff0\_2938 | IO\_L2X\_WIN7\_BASE |
| - | - | 0 x3ff0\_2940 | IO\_L2X\_WIN0\_MASK |
| - | - | 0 x3ff0\_2948 | IO\_L2X\_WIN1\_MASK |
| - | - | 0 x3ff0\_2950 | IO\_L2X\_WIN2\_MASK |
| - | - | 0 x3ff0\_2958 | IO\_L2X\_WIN3\_MASK |
| - | - | 0 x3ff0\_2960 | IO\_L2X\_WIN4\_MASK |
| - | - | 0 x3ff0\_2968 | IO\_L2X\_WIN5\_MASK |
| - | - | 0 x3ff0\_2970 | IO\_L2X\_WIN6\_MASK |
| - | - | 0 x3ff0\_2978 | IO\_L2X\_WIN7\_MASK |
| - | - | 0 x3ff0\_2980 | IO\_L2X\_WIN0\_MMAP |
| - | - | 0 x3ff0\_2988 | IO\_L2X\_WIN1\_MMAP |
| - | - | 0 x3ff0\_2990 | IO\_L2X\_WIN2\_MMAP |
| - | - | 0 x3ff0\_2998 | IO\_L2X\_WIN3\_MMAP |
| - | - | 0 x3ff0\_29a0 | IO\_L2X\_WIN4\_MMAP |
| - | - | 0 x3ff0\_29a8 | IO\_L2X\_WIN5\_MMAP |
| - | - | 0 x3ff0\_29b0 | IO\_L2X\_WIN6\_MMAP |
| - | - | 0 x3ff0\_29b8 | IO\_L2X\_WIN7\_MMAP |
| 0 x3ff0\_2a00 | HT0\_LO\_WIN0\_BASE | 0 x3ff0\_2b00 | HT0\_HI\_WIN0\_BASE |
| 0 x3ff0\_2a08 | HT0\_LO\_WIN1\_BASE | 0 x3ff0\_2b08 | HT0\_HI\_WIN1\_BASE |
| 0 x3ff0\_2a10 | HT0\_LO\_WIN2\_BASE | 0 x3ff0\_2b10 | HT0\_HI\_WIN2\_BASE |
| 0 x3ff0\_2a18 | HT0\_LO\_WIN3\_BASE | 0 x3ff0\_2b18 | HT0\_HI\_WIN3\_BASE |
| 0 x3ff0\_2a20 | HT0\_LO\_WIN4\_BASE | 0 x3ff0\_2b20 | HT0\_HI\_WIN4\_BASE |
| 0 x3ff0\_2a28 | HT0\_LO\_WIN5\_BASE | 0 x3ff0\_2b28 | HT0\_HI\_WIN5\_BASE |
| 0 x3ff0\_2a30 | HT0\_LO\_WIN6\_BASE | 0 x3ff0\_2b30 | HT0\_HI\_WIN6\_BASE |
| 0 x3ff0\_2a38 | HT0\_LO\_WIN7\_BASE | 0 x3ff0\_2b38 | HT0\_HI\_WIN7\_BASE |
| 0 x3ff0\_2a40 | HT0\_LO\_WIN0\_MASK | 0 x3ff0\_2b40 | HT0\_HI\_WIN0\_MASK |
| 0 x3ff0\_2a48 | HT0\_LO\_WIN1\_MASK | 0 x3ff0\_2b48 | HT0\_HI\_WIN1\_MASK |
| 0 x3ff0\_2a50 | HT0\_LO\_WIN2\_MASK | 0 x3ff0\_2b50 | HT0\_HI\_WIN2\_MASK |
| 0 x3ff0\_2a58 | HT0\_LO\_WIN3\_MASK | 0 x3ff0\_2b58 | HT0\_HI\_WIN3\_MASK |
| 0 x3ff0\_2a60 | HT0\_LO\_WIN4\_MASK | 0 x3ff0\_2b60 | HT0\_HI\_WIN4\_MASK |
| 0 x3ff0\_2a68 | HT0\_LO\_WIN5\_MASK | 0 x3ff0\_2b68 | HT0\_HI\_WIN5\_MASK |
| 0 x3ff0\_2a70 | HT0\_LO\_WIN6\_MASK | 0 x3ff0\_2b70 | HT0\_HI\_WIN6\_MASK |
| 0 x3ff0\_2a78 | HT0\_LO\_WIN7\_MASK | 0 x3ff0\_2b78 | HT0\_HI\_WIN7\_MASK |
| 0 x3ff0\_2a80 | HT0\_LO\_WIN0\_MMAP | 0 x3ff0\_2b80 | HT0\_HI\_WIN0\_MMAP |
| 0 x3ff0\_2a88 | HT0\_LO\_WIN1\_MMAP | 0 x3ff0\_2b88 | HT0\_HI\_WIN1\_MMAP |
| 0 x3ff0\_2a90 | HT0\_LO\_WIN2\_MMAP | 0 x3ff0\_2b90 | HT0\_HI\_WIN2\_MMAP |
| 0 x3ff0\_2a98 | HT0\_LO\_WIN3\_MMAP | 0 x3ff0\_2b98 | HT0\_HI\_WIN3\_MMAP |
| 0 x3ff0\_2aa0 | HT0\_LO\_WIN4\_MMAP | 0 x3ff0\_2ba0 | HT0\_HI\_WIN4\_MMAP |
| 0 x3ff0\_2aa8 | HT0\_LO\_WIN5\_MMAP | 0 x3ff0\_2ba8 | HT0\_HI\_WIN5\_MMAP |
| 0 x3ff0\_2ab0 | HT0\_LO\_WIN6\_MMAP | 0 x3ff0\_2bb0 | HT0\_HI\_WIN6\_MMAP |
| 0 x3ff0\_2ab8 | HT0\_LO\_WIN7\_MMAP | 0 x3ff0\_2bb8 | HT0\_HI\_WIN7\_MMAP |
| 0 x3ff0\_2c00 | SE\_WIN0\_BASE | 0 x3ff0\_2d00 | MISC\_WIN0\_BASE |
| 0 x3ff0\_2c08 | SE\_WIN1\_BASE | 0 x3ff0\_2d08 | MISC\_WIN1\_BASE |
| 0 x3ff0\_2c10 | SE\_WIN2\_BASE | 0 x3ff0\_2d10 | MISC\_WIN2\_BASE |
| 0 x3ff0\_2c18 | SE\_WIN3\_BASE | 0 x3ff0\_2d18 | MISC\_WIN3\_BASE |
| 0 x3ff0\_2c20 | SE\_WIN4\_BASE | 0 x3ff0\_2d20 | MISC\_WIN4\_BASE |
| 0 x3ff0\_2c28 | SE\_WIN5\_BASE | 0 x3ff0\_2d28 | MISC\_WIN5\_BASE |
| 0 x3ff0\_2c30 | SE\_WIN6\_BASE | 0 x3ff0\_2d30 | MISC\_WIN6\_BASE |
| 0 x3ff0\_2c38 | SE\_WIN7\_BASE | 0 x3ff0\_2d38 | MISC\_WIN7\_BASE |
| 0 x3ff0\_2c40 | SE\_WIN0\_MASK | 0 x3ff0\_2d40 | MISC\_WIN0\_MASK |
| 0 x3ff0\_2c48 | SE\_WIN1\_MASK | 0 x3ff0\_2d48 | MISC\_WIN1\_MASK |
| 0 x3ff0\_2c50 | SE\_WIN2\_MASK | 0 x3ff0\_2d50 | MISC\_WIN2\_MASK |
| 0 x3ff0\_2c58 | SE\_WIN3\_MASK | 0 x3ff0\_2d58 | MISC\_WIN3\_MASK |
| 0 x3ff0\_2c60 | SE\_WIN4\_MASK | 0 x3ff0\_2d60 | MISC\_WIN4\_MASK |
| 0 x3ff0\_2c68 | SE\_WIN5\_MASK | 0 x3ff0\_2d68 | MISC\_WIN5\_MASK |
| 0 x3ff0\_2c70 | SE\_WIN6\_MASK | 0 x3ff0\_2d70 | MISC\_WIN6\_MASK |
| 0 x3ff0\_2c78 | SE\_WIN7\_MASK | 0 x3ff0\_2d78 | MISC\_WIN7\_MASK |
| 0 x3ff0\_2c80 | SE\_WIN0\_MMAP | 0 x3ff0\_2d80 | MISC\_WIN0\_MMAP |
| 0 x3ff0\_2c88 | SE\_WIN1\_MMAP | 0 x3ff0\_2d88 | MISC\_WIN1\_MMAP |
| 0 x3ff0\_2c90 | SE\_WIN2\_MMAP | 0 x3ff0\_2d90 | MISC\_WIN2\_MMAP |
| 0 x3ff0\_2c98 | SE\_WIN3\_MMAP | 0 x3ff0\_2d98 | MISC\_WIN3\_MMAP |
| 0 x3ff0\_2ca0 | SE\_WIN4\_MMAP | 0 x3ff0\_2da0 | MISC\_WIN4\_MMAP |
| 0 x3ff0\_2ca8 | SE\_WIN5\_MMAP | 0 x3ff0\_2da8 | MISC\_WIN5\_MMAP |
| 0 x3ff0\_2cb0 | SE\_WIN6\_MMAP | 0 x3ff0\_2db0 | MISC\_WIN6\_MMAP |
| 0 x3ff0\_2cb8 | SE\_WIN7\_MMAP | 0 x3ff0\_2db8 | MISC\_WIN7\_MMAP |
| 0 x3ff0\_2e00 | HT1\_LO\_WIN0\_BASE | 0 x3ff0\_2f00 | HT1\_HI\_WIN0\_BASE |
| 0 x3ff0\_2e08 | HT1\_LO\_WIN1\_BASE | 0 x3ff0\_2f08 | HT1\_HI\_WIN1\_BASE |
| 0 x3ff0\_2e10 | HT1\_LO\_WIN2\_BASE | 0 x3ff0\_2f10 | HT1\_HI\_WIN2\_BASE |
| 0 x3ff0\_2e18 | HT1\_LO\_WIN3\_BASE | 0 x3ff0\_2f18 | HT1\_HI\_WIN3\_BASE |
| 0 x3ff0\_2e20 | HT1\_LO\_WIN4\_BASE | 0 x3ff0\_2f20 | HT1\_HI\_WIN4\_BASE |
| 0 x3ff0\_2e28 | HT1\_LO\_WIN5\_BASE | 0 x3ff0\_2f28 | HT1\_HI\_WIN5\_BASE |
| 0 x3ff0\_2e30 | HT1\_LO\_WIN6\_BASE | 0 x3ff0\_2f30 | HT1\_HI\_WIN6\_BASE |
| 0 x3ff0\_2e38 | HT1\_LO\_WIN7\_BASE | 0 x3ff0\_2f38 | HT1\_HI\_WIN7\_BASE |
| 0 x3ff0\_2e40 | HT1\_LO\_WIN0\_MASK | 0 x3ff0\_2f40 | HT1\_HI\_WIN0\_MASK |
| 0 x3ff0\_2e48 | HT1\_LO\_WIN1\_MASK | 0 x3ff0\_2f48 | HT1\_HI\_WIN1\_MASK |
| 0 x3ff0\_2e50 | HT1\_LO\_WIN2\_MASK | 0 x3ff0\_2f50 | HT1\_HI\_WIN2\_MASK |
| 0 x3ff0\_2e58 | HT1\_LO\_WIN3\_MASK | 0 x3ff0\_2f58 | HT1\_HI\_WIN3\_MASK |
| 0 x3ff0\_2e60 | HT1\_LO\_WIN4\_MASK | 0 x3ff0\_2f60 | HT1\_HI\_WIN4\_MASK |
| 0 x3ff0\_2e68 | HT1\_LO\_WIN5\_MASK | 0 x3ff0\_2f68 | HT1\_HI\_WIN5\_MASK |
| 0 x3ff0\_2e70 | HT1\_LO\_WIN6\_MASK | 0 x3ff0\_2f70 | HT1\_HI\_WIN6\_MASK |
| 0 x3ff0\_2e78 | HT1\_LO\_WIN7\_MASK | 0 x3ff0\_2f78 | HT1\_HI\_WIN7\_MASK |
| 0 x3ff0\_2e80 | HT1\_LO\_WIN0\_MMAP | 0 x3ff0\_2f80 | HT1\_HI\_WIN0\_MMAP |
| 0 x3ff0\_2e88 | HT1\_LO\_WIN1\_MMAP | 0 x3ff0\_2f88 | HT1\_HI\_WIN1\_MMAP |
| 0 x3ff0\_2e90 | HT1\_LO\_WIN2\_MMAP | 0 x3ff0\_2f90 | HT1\_HI\_WIN2\_MMAP |
| 0 x3ff0\_2e98 | HT1\_LO\_WIN3\_MMAP | 0 x3ff0\_2f98 | HT1\_HI\_WIN3\_MMAP |
| 0 x3ff0\_2ea0 | HT1\_LO\_WIN4\_MMAP | 0 x3ff0\_2fa0 | HT1\_HI\_WIN4\_MMAP |
| 0 x3ff0\_2ea8 | HT1\_LO\_WIN5\_MMAP | 0 x3ff0\_2fa8 | HT1\_HI\_WIN5\_MMAP |
| 0 x3ff0\_2eb0 | HT1\_LO\_WIN6\_MMAP | 0 x3ff0\_2fb0 | HT1\_HI\_WIN6\_MMAP |
| 0 x3ff0\_2eb8 | HT1\_LO\_WIN7\_MMAP | 0 x3ff0\_2fb8 | HT1\_HI\_WIN7\_MMAP |

The secondary Xbar is mainly connected to two memory controllers and Io-ring as slave devices, with four Scache (4, representing 0x3ff0\_4XXX, the same as below, 5, 6, 7) and Io-Ring (9) as master devices for window mapping. These Windows can be used to configure registers (4, 5, 6, 7, 9) for memory window configuration and address translation.

Each address window is composed of three 64-bit registers, BASE, MASK and MMAP. BASE is aligned with K byte, while MASK adopts a format similar to network MASK with the high order of 1. MMAP contains the converted address, route selection and enabling control alleles, as shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | [47:10] | [17] | [3-0] |
|  | Converted address | The window can make | From the device number |

Where, the equipment corresponding to the device number is shown in the following table:

Table 3-7 correspondence between the slave device number and the module

|  |  |
| --- | --- |
| From the device number | Purpose equipment |
| 0-3 | Scache0-3 |
| 4 to 5 | MC0-1 |
| a. | HT0\_lo |
| b | HT0\_hi |
| c | SE |
| d | MISC |
| e | HT1\_lo |
| f | HT1\_hi |

The window enabling bits have the following meanings:

Table 3-8 MMAP field corresponding to the space access properties

|  |  |  |  |
| --- | --- | --- | --- |
| [7] | [6] | [5] | [4] |
| The window can make | Allows interleaving access to DDR, valid when the slave device number is 0, and routes requests to hit window addresses in a "interleaving select bit" configuration. Interleaving enablement is required  More than 10 | Allow the block read | Allowed to take to |

It should be noted that the window configuration cannot translate the address of the Cache consistency request, otherwise the address at SCache will be inconsistent with the address at the processor level Cache, resulting in a Cache consistency maintenance error.

Window hit formula :(IN\_ADDR & MASK) == BASE

New address conversion formula :OUT\_ADDR = (IN\_ADDR & ~MASK) | {MMAP[63:10],10 'h0}

According to the default register configuration, CPU 0x00000000-0x0FFFFfff address interval after chip startup

Map to the address interval of DDR 0x00000000-0x0FFFFfff, 0x10000000- 0x17FFffff map to the PCI\_MEM space of the bridge slice, 0x18000000- 0x19FFffff map to the PCI\_IO space of the bridge slice, 0x1A000000 -0x1affffff map to the PCI configuration space of the bridge slice (Type0),0x1B000000-0x1bFFffff maps to the PCI configuration space of the bridge slice (Type1), 0x40000000- 0x7FFfffff maps to the PCI\_MEM space of the bridge slice. Software can modify the corresponding configuration registers to achieve the new address space routing and transformation.

In addition, when a read access to an illegal address occurs due to CPU guessing execution, none of the 8 address Windows will hit and random data will be returned to prevent CPU from dying, etc.

## Chip configuration register

The chip configuration register in the Loong Chip 3A4000 provides a mechanism for reading and writing configuration of various functions of the chip. The individual configuration registers are detailed below.

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

## Version 4.1 Register (0x0000)

The base address is 0x1fe00000 and the offset address is 0x0000.

Table 4-1 version registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| away | The Version | R | 8 'h10 | Configure the register version number |

## 4.2 Chip Feature Register (0x0008)

This register identifies some software-related processor features for the software to view before enabling a specific function. The base address of the register is 0x1fe00000 and the offset address is 0x0008.

Table 4-2 Chip feature registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | Centigrade | R | 1 'b1 | Is 1, indicating that CSR[0x428] is valid |
| 1 | The Node counter | R | 1 'b1 | Is 1, indicating that CSR[0x408] is valid |
| 2 | MSI | R | 1 'b1 | When is 1, means MSI is available |
| 3 | EXT\_IOI | R | 1 'b1 | Is 1, indicating EXT\_IOI is available |
| 4 | IPI\_percore | R | 1 'b1 | When is 1, IPI is sent through the CSR private address |
| 5 | Freq\_percore | R | 1 'b1 | When is 1, the frequency is adjusted through the CSR private address |
| 6 | Freq\_scale | R | 1 'b0 | Is 1, indicating that the dynamic frequency division function is available |
| 7 | DVFS\_v1 | R | 1 'b0 | When is 1, means dynamic FM V1 is available |
| 8 | Tsensor | R | 1 'b0 | When is 1, means the temperature sensor is available |

## 4.3 Manufacturer name (0x0010)

This register is used to identify the manufacturer name. The base address is 0x1fe00000 and the offset address is 0x0010.

Table 4-3 Manufacturer name register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 63:0 | Vendor | R | 0 x6e6f7367\_6e6f6f4c | The string "Loongson" |

## 4.4 Chip Name (0x0020)

This register is used to identify the chip name. The base address is 0x1fe00000 and the offset address is 0x0020.

Table 4-4 Chip name register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 63:0 | ID | R | 0 x00003030\_30344133 | The string "3 a4000" |

## Function Setting register (0x0180)

The base address is 0x1fe00000 and the offset address is 0x0180.

Table 4-5 Function Settings register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 |  | RW | 1 'b0 |  |
| 1 |  | RW | 1 'b0 |  |
| 3:2 |  | RW | 2 b0 ' | reserve |
| 4 | MC0\_disable\_confspace | RW | 1 'b0 | Disable the MC0 DDR configuration space |
| 5 | MC0\_defult\_confspace | RW | 1 'b1 | Route all memory access to the configuration space |
| 6 | MCA0 clock en | RW | 1 'b1 | MCA0 clock enablement |
| 7 | MC0\_resetn | RW | 1 'b1 | MC0 Software Reset (low efficiency) |
| 8 | MC0\_clken | RW | 1 'b1 | Whether to enable MC0 |
| 9 | MC1\_disable\_confspace | RW | 1 'b0 | Disable the MC1 DDR configuration space |
| 10 | MC1\_defult\_confspace | RW | 1 'b1 | Route all memory access to the configuration space |
| 11 | MCA1 clock en | RW | 1 'b1 | MCA1 clock enablement |
| 12 | MC1\_resetn | RW | 1 'b1 | MC1 Software Reset (low efficiency) |
| 13 | MC1\_clken | RW | 1 'b1 | Whether to enable MC1 |
| they | HT0\_freq\_scale\_ctrl | RW | 3 'b011 | HT controller 0 frequency division |
| 27 | HT0\_clken | RW | 1 'b1 | Whether or not I enabled HT0 |
| he | HT1\_freq\_scale\_ctrl | RW | 3 'b011 | HT controller 1 frequency division |
| 31 | HT1\_clken | RW | 1 'b1 | Whether I enabled HT1 |
| 42:40 | Node\_freq\_ CTRL | RW | 3 'b111 | Nodal points frequency |
| 43 | - | RW | 1 'b1 |  |
| 63:56 | Cpu\_version | R | 2 'h3B | The CPU version |

## Pin Drive Setup register (0x0188)

Base address 0x1fe00000, offset address 0x0188.

Table 4-6 Pin drive setup register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 |  |  |  | (empty) |
| 63:32 | Pad1v8\_ctrl | RW | 32 'h4f0000 | 1 v8 control pad |

## Functional Sampling register (0x0190)

The base address is 0x1fe00000 and the offset address is 0x0190.

Table 4-7 Functional sampling registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | Compcode\_core | R |  |  |
| Go forth | Chip\_config | R |  | Mainboard configuration control |
| 47:38 | Sys\_clkseli | R |  | On - board frequency multiplication Settings |
| 55:48 | Bad\_ip\_core | R |  | Core7 - core0 is bad |
| 57:56 | Bad\_ip\_ddr | R |  | Are 2 DDR controllers broken |
| 61:60 | Bad\_ip\_ht | R |  | Is 2 HT controllers broken |

## Temperature sampling register (0x0198)

The base address is 0x1fe00000 and the offset address is 0x0198.

Table 4-8 Temperature sampling registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 15:0 |  | R |  |  |
| He hath | Compcode\_ok | R |  |  |
| 20 | dotest | R |  |  |
| 21 | iccc\_en | R |  |  |
| " |  | R |  |  |
| 24 | Thsens0\_overflow | R |  | Temperature sensor 0 overflows |
| 25 | Thsens1\_overflow | R |  | Temperature sensor 1 overflows |
| Upon this |  |  |  |  |
| 47:32 | Thsens0\_out | R |  | Temperature sensor 0 ℃ |
|  |  |  |  | Node temperature =Thens0\_out  \* 731/0 x4000-273  Temperature range -40-125 degrees |
| 63:48 | Thsens1\_out | R |  | Temperature sensor 1 ℃  Node temperature =Thens1\_out  \* 731/0 x4000-273  Temperature range -40-125 degrees |

## Bias configuration register (0x01A0)

The 3A4000 has built-in bias generation modules. The following registers are used for the control of these bias modules. Base address for

0x1fe00000, offset 0x1a0.

Table 4-9 Bias setting registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | BBGEN\_enable | RW | 0 x0 | Bias can make |
| 1 | BBMUX\_first | RW | 0 x0 | Set to switch voltage mode first |
| 3:2 |  | RW | 0 x0 |  |
| The log | BBGEN\_feedback | RW | 0 x0 | Disable BBGEN feedback signals |
| and | BBGEN\_vbbp\_val | RW | 0 x0 | Settings for Vbbp |
| " | BBGEN\_vbbn\_val | RW | 0 x0 | Set value for Vbbn |
| " | BBMUX\_SEL\_0 | RW | 0 x0 | The setting of the BBMUX\_SEL\_0 |
| 7 | BBMUX\_SEL\_1 | RW | 0 x0 | The setting of the BBMUX\_SEL\_1 |
| mark | BBMUX\_SEL\_2 | RW | 0 x0 | The setting of the BBMUX\_SEL\_2 |
| " | BBMUX\_SEL\_3 | RW | 0 x0 | The setting of the BBMUX\_SEL\_3 |
| came |  | RW | 0 x0 | reserve |
| 40:32 | BBGEN\_sm | RO | 0 x0 | BBGEN state machine current state |
| other | - | RW |  | reserve |

## Frequency configuration register (0x01B0)

The following sets of software doubler setting registers are used to set the operating frequency of the chip master clock and memory controller clock under the CLKSEL configuration as software control mode (see the CLKSEL setting method in Section 2.2). Where, MEM CLOCK is configured to correspond to the CLOCK frequency of the memory controller. The bus operating frequency is 2 times that of the CLOCK, and the bus operating rate is 4 times that of the CLOCK. NODE CLOCK corresponds to the CLOCK frequency of the processor core, on-chip network, and high-speed Shared cache.

Each clock configuration typically takes three parameters, DIV\_REFC, DIV\_LOOPC, and DIV\_OUT. The final clock frequency is

(See clock /DIV\_REFC \* DIV\_LOOPC)/DIV\_OUT.

In the software control mode, the default corresponding clock frequency is the frequency of external reference clock (100MHz or 25MHz), and the clock needs to be set in the process of processor startup. Each clock should be set in the following manner:

* + 1. Set registers other than SEL\_PLL\_\* and SOFT\_SET\_PLL, which are written to 0 during the set.
    2. Set SOFT\_SET\_PLL to 1 with the other register values unchanged.
    3. Wait register lock signal LOCKED\_\* for 1;
    4. Set SEL\_PLL\_\* to 1 and the corresponding clock frequency will switch to the frequency set by the software.

The following register is the configuration register for Main CLOCK, which is used to generate the highest working frequency of Node CLOCK, core CLOCK, and so on. Its base address is 0x1fe00000 and its offset address is 0x1b0:

Table 4-10 Clock software frequency doubling registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | SEL\_PLL\_NODE | RW | 0 x0 | Clock output selection  1: Select PLL output for Node clock  0: Node CLOCK selects SYS CLOCK |
| 1 |  | RW | 0 x0 |  |
| 2 | SOFT\_SET\_PLL | RW | 0 x0 | Allows software to set PLL |
| 3 | BYPASS\_L1 | RW | 0 x0 | Bypass L1 PLL |
| Indeed, | - | RW | 0 x0 | - |
| 16 | LOCKED\_L1 | R | 0 x0 | Is L1 PLL locked |
| That is | - | R | 0 x0 | - |
| 19 | PD\_L1 | RW | 0 x0 | Close the L1 PLL |
| At a |  | RW | 0 x0 |  |
| Upon this | L1\_DIV\_REFC | RW | 0 x1 | L1 PLL input parameter |
| 40:32 | L1\_DIV\_LOOPC | RW | 0 x1 | L1 PLL input parameter |
| 41 |  |  |  |  |
| 47:42 | L1\_DIV\_OUT | RW | 0 x1 | L1 PLL input parameter |
|  |  |  |  |  |
| other | - | RW |  | reserve |

PLL ouput = (clk\_ref /div\_refc \* div\_loopc)/div\_out.

The RESULT of THE PLL clk\_ref/ div\_REFc should be 25-50mhz and the VCO frequency (the part in brackets above) must be within the range of 1.2ghz to 3.2ghz. This requirement also applies to memory PLL.

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

Table 4-11 Memory clock software frequency multiplier setting register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | SEL \_MEM\_PLL | RW | 0 x0 | Clock output selection  1: MEM clock selects PLL output  0: SYS CLOCK is selected for MEM CLOCK |
| 1 | SOFT\_SET\_MEM\_PLL | RW | 0 x0 | Allows software to set MEM PLL |
| 2 | BYPASS\_MEM\_PLL | RW | 0 x0 | Bypass MEM\_PLL |
| o |  |  |  |  |
| 6 | LOCKED\_MEM\_PLL | R | 0 x0 | Is MEM\_PLL locked |
| 7 | PD\_MEM\_PLL | RW | 0 x0 | Close the MEM PLL |
| Will you | MEM\_PLL\_DIV\_REFC | RW | 0 x1 | MEM PLL input parameter  When NODE clock is selected (NODE\_CLOCK\_SEL is 1), it is used as a frequency divider input |
| brake | MEM\_PLL\_DIV\_LOOPC | RW | 0 x41 | MEM PLL input parameter |
| A partner | MEM\_PLL\_DIV\_OUT | RW | 0 x0 | MEM PLL input parameter |
| 30 | NODE\_CLOCK\_SEL | RW | 0 x0 | 0: Use MEM\_PLL as the MEM clock  1: Use NODE\_CLOCK as the divider input |
| other |  | RW |  | reserve |

## Processor core frequency division setting register (0x01D0)

The following registers are used for dynamic frequency division of the processor core. This register can be used for frequency modulation setting of the processor core to complete frequency conversion operation in 100ns without any additional overhead. Base address 0x1fe00000, offset address 0x01d0.

Table 4-12 Register of frequency division of processor core software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| The 2-0 | core0\_freqctrl | RW | 0 x7 | Nuclear 0 frequency division control value |
| 3 | core0\_en | RW | 0 x1 | Nuclear 0 clock enablement |
| 6:4 | core1\_freqctrl | RW | 0 x7 | Nuclear 1 frequency control value |
| 7 | core1\_en | RW | 0 x1 | Nuclear 1 clock enable |
| 10:8 | core2\_freqctrl | RW | 0 x7 | Nuclear 2 frequency control value |
| 11 | core2\_en | RW | 0 x1 | Nuclear 2 clock enablement |
| then | core3\_freqctrl | RW | 0 x7 | Nuclear 3 - frequency control value |
| 15 | core3\_en | RW | 0 x1 | Nuclear 3 clock enablement |
|  |  |  | Note: | The value of clock frequency after software frequency division is equal to original  Of (frequency division control value +1) /8 |

## Processor core reset Control Register (0x01D8)

The following registers are used for processor core software control reset use. When reset is needed, first set the resetn of the corresponding core to 0, and then set resetn\_pre to 0. After waiting for 500 microseconds, set resetn\_PRE to 1, and then set resetn to 1 to complete the whole reset process. The register is base address 0x1fe00000 and offset address 0x01d8.

Table 4-13 Register of frequency division of processor core software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | Core0\_resetn\_pre | RW | 0 x1 | Nuclear 0 reset auxiliary control |
| 1 | Core0\_resetn | RW | 0 x1 | Nuclear zero reset |
| 2 | Core1\_resetn\_pre | RW | 0 x1 | Nuclear 1 reset auxiliary control |
| 3 | Core1\_resetn | RW | 0 x1 | Nuclear 1 reset |
| 4 | Core2\_resetn\_pre | RW | 0 x1 | Nuclear 2 reset auxiliary control |
| 5 | Core2\_resetn | RW | 0 x1 | Nuclear 2 reset |
| 6 | Core3\_resetn\_pre | RW | 0 x1 | Nuclear 3 reset auxiliary control |
| 7 | Core3\_resetn | RW | 0 x1 | Nuclear 3 reset |

## Route Setup register (0x0400)

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

Table 4-14 Chip routing setup register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 3-0 | scid\_sel | RW | 0 x0 | Shared cache hash control |
| 6:4 | Node\_mask | RW | 0 x7 | Node mask to avoid no response when guessing the address of unused nodes |
| 7 |  | RW | 0 x0 | reserve |
| 8 | xrouter\_en | RW | 0 x0 | HT1 inter - chip routing enabling control |
| 9 | disable\_0x3ff0 | RW | 0 x0 | Routing to configuration register space through base address 0x3ff0\_0000 is prohibited |
| 12 | McC\_en | RW | 0 x0 | MCC mode enablement |
| He hath | ccsd\_id | RW | 0 x0 |  |
| 24 | ccsd\_en | RW | 0 x0 |  |
| charm | Mc\_en | RW | 0 x3 | Enable routing control for both MCS |
| Go forth | interleave\_bit | RW | 0 x0 | Memory hash control |
| 39 | interleave\_en | RW | 0 x0 | Memory hashing enablement |
| 43:40 | ht\_control | R |  | Ht related configuration pins |
| 47:44 | ht\_reg\_disable | RW | 0 x0 | Close HT space for consistency mode to avoid ROUTING HT spatial addresses to HT |

## Other Functions Set register (0x0420)

The following registers are used to control some function enablement in the chip. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 4-15 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | disable\_jtag | RW | 0 x0 | Disable the JTAG interface completely |
| 1 | disable\_ejtag | RW | 0 x0 | Disable the EJTAG interface completely |
| 2 | disable\_gs132 | RW | 0 x0 | Disable GS132 completely |
| 3 | disable\_ejtag132 | RW | 0 x0 | Disable the GS132 EJTAG interface completely |
| 4 | Disable\_antifuse0 | RW | 0 x0 |  |
| 5 | Disable\_antifuse1 | RW | 0 x0 |  |
| 6 | Disable\_ID | RW | 0 x0 |  |
| 8 | resetn\_gs132 | RW | 0 x0 | GS132 reset control |
| 9 | sleeping\_gs132 | R | 0 x0 | GS132 goes to sleep |
| 10 | soft\_int\_gs132 | RW | 0 x0 | GS132 intercore interrupt register |
| " | core\_int\_en\_gs132 | RW | 0 x0 | GS132 corresponds to IO interrupt enablement for each core |
| thou | freqscale\_gs132 | RW | 0 x0 | GS132 frequency division control |
| 19 | clken\_gs132 | RW | 0 x0 | GS132 clock enablement |
| 21 | stable\_resetn | RW | 0 x0 | Stable clock reset control |
| 22 | freqscale\_percore | RW | 0 x0 | Enable each core private FM register |
| 23 | clken\_percore | RW | 0 x0 | Enable each nuclear private clock enable |
| he | confbus\_timeout | RW | By 8 0 | Configure the bus timeout setting, which is actually a power of 2 |
| then | HT\_softresetn | RW | 0 x3 | HT controller software reset control |
| has | freqscale\_mode\_core | RW | 0 x0 | FM mode selection for each core  0: (n + 1) / 8  1:1 / (n + 1) |
| 36 | freqscale\_mode\_node | RW | 0 x0 | FM mode selection of nodes |
| 37 | freqscale\_mode\_gs132 | RW | 0 x0 | FM mode selection for GS132 |
| 39:38 | freqscale\_mode\_HT | RW | 0 x0 | Frequency modulation mode selection for each HT |
| 40 | freqscale\_mode\_stable | RW | 0 x0 | Frequency modulation selection of Stable Clock |
| 46:44 | freqscale\_stable | RW | 0 x0 | Stable Clock frequency modulation register |
| 47 | clken\_stable | RW | 0 x0 | Stable Clock enable |
| 48 | EXT\_INT\_en | RW | 0 x0 | Extend IO interrupt enablement |
| 57:56 | thsensor\_sel | RW | 0 x0 | Temperature sensor selection |
| 62:60 | Auto\_scale | R | 0 x0 | Automatic frequency modulation current value |
| 63 | Auto\_scale\_doing | R | 0 x0 | Automatic FM is now active flag |

## Celsius temperature register (0x0428)

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

Table 4-16 Temperature observation registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| away | Centigrade temperature | RO | 0 x0 | Celsius |
| 63:8 |  | RW | 0 x0 |  |

## SRAM Adjustment register (0x0430)

The following registers are used to adjust the operating frequency of the Sram inside the processor core. Base address 0x1fe00000, offset address 0x0430.

Table 4-17 Processor core SRAM adjustment register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | sram\_ctrl | RW | 0 x0 | The internal Sram configuration register |
| 63:32 |  | RW | 0 x0 |  |

## FUSE0 Observation register (0x0460)

The following registers are used to observe the Fuse0 values visible to part of the software. Base address 0x1fe00000, offset address 0x0460.

Table 4-18 FUSE observation registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 127:0 | Fuse\_0 | RW | 0 x0 |  |

## FUSE1 Observation register (0x0470)

The following registers are used to observe the Fuse1 values visible to some of the software. The base address is 0x1fe00000 and the offset address is 0x0470.

Table 4-19 FUSE observation registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 127:0 | Fuse\_1 | RW | 0 x0 |  |

## Chip clock frequency division and enable control

Longson 3A4000 can use a single external reference clock SYS\_CLOCK. Each clock can depend on SYS\_CLOCK, which are described in the following sections.

The frequency division mechanism is set for processor core, on-chip network and Shared cache, HT controller and GS132 core respectively. Compared with the original frequency division mechanism of 3A3000, the version implemented in 3A4000 adds a new frequency division mode, which can support the frequency division value of 1/n.

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

## Chip module clock introduction

The chip reference clock SYS\_CLOCK usually USES 100MHz crystal oscillator input or 25MHz crystal oscillator input. Different crystal frequencies need to be selected by CLKSEL[4].

In addition to USING SYS CLOCK, HT PHY's reference CLOCK can also use 200MHz differential reference inputs for each PHY. Use CLKSEL[8] for selection. When SYS CLOCK was selected as the reference CLOCK and 25MHz crystal input was used, HT PHY could not operate at the frequency of 3.2ghz.

The clock used in the Loong Chip 3A4000 and its control mode are shown in the table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **The clock** | **The clock source** | **Frequency doubling method** | **Frequency control** | **Can make control** | **The clock description** |
| The Boot Clock | SYS\_CLOCK | \* 1 | Does not support | Does not support | SPI, UART, I2C controller clock |
| The Main Clock | SYS PLL | PLL configuration | Does not support | Does not support | SYS PLL output.  Node Clock, Core Clock, HTcore Clock, GS132 Clock source Mem Clock, Stable Clock optional  Zhong Yuan |
| The Node Clock | The Main Clock | \* 1 | support | Does not support | On-chip network, Shared cache, node clock,  HT controller clock source |
| Core0 Clock | The Main Clock | \* 1 | support | support | Core0 clock |
| Core1 Clock | The Main Clock | \* 1 | support | support | Core1 clock |
| Core2 Clock | The Main Clock | \* 1 | support | support | Core2 clock |
| Core3 Clock | The Main Clock | \* 1 | support | support | Core3 clock |
| HTcore0 Clock | The Node Clock | \* 1 | support | support | HT0 controller clock, the software needs to ensure that the frequency division is less than 1GHz |
| HTcore1 Clock | The Node Clock | \* 1 | support | support | HT1 controller clock, the software needs to ensure that the frequency division is less than 1GHz |
| GS132 Clock | The Main Clock | \* 1 | support | support | GS132 clock, the software needs to ensure that after the frequency division  Less than 1 GHZ. |
| Stable Clock | The Main Clock | \* 1 | support | support | Processor core constant counter clock |
| Mem Clock | MEM PLL | PLL configuration | Does not support | support | Memory controller clock |
| The Main Clock | PI over 2, PI over 4, PI over 8 | Does not support | support | Memory controller alternative clock |

## Processor core frequency division and enabling control

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

#### Visit by address

It is compatible with 3A3000 processor by address access mode. Register is set by frequency division of processor core software and the same address is used for setting.

This register can be used to set the frequency of processor core, and the frequency conversion operation can be completed in 100ns without any additional overhead. Base address 0x1fe00000, offset address 0x01d0.

Table 5-1 Register of frequency division of processor core software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| The 2-0 | core0\_freqctrl | RW | 0 x7 | Nuclear 0 frequency division control value |
| 3 | core0\_en | RW | 0 x1 | Nuclear 0 clock enablement |
| 6:4 | core1\_freqctrl | RW | 0 x7 | Nuclear 1 frequency control value |
| 7 | core1\_en | RW | 0 x1 | Nuclear 1 clock enable |
| 10:8 | core2\_freqctrl | RW | 0 x7 | Nuclear 2 frequency control value |
| 11 | core2\_en | RW | 0 x1 | Nuclear 2 clock enablement |
| then | core3\_freqctrl | RW | 0 x7 | Nuclear 3 - frequency control value |
| 15 | core3\_en | RW | 0 x1 | Nuclear 3 clock enablement |
|  |  |  | Note: | The clock frequency value after the software frequency division is equal to the original (frequency division control value +1) /8 |

In addition to the frequency division configuration compatible with the 3A3000 processor, 3A4000 can also adjust the clock frequency after frequency division from the original "(frequency division control value +1) /8" to "1/ (frequency division control value +1)" by setting the register. This register is located in the Other Function Settings Register. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 5-2 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| has | freqscale\_mode\_core | RW | 0 x0 | FM mode selection for each core  0: (n + 1) / 8  1:1 / (n + 1) |

#### Configure register instruction access

In addition to the traditional address-by-address access mode, the 3A4000 also supports access to private divider configuration registers using configuration register instructions.

It should be noted that the private frequency division configuration register control and the original processor core software frequency division set register control are mutually exclusive, the two can only be used in one. The method of selection is controlled by the corresponding bit advance on the Other Function Set Register. The register is base address 0x1fe00000 and offset address 0x0420.

Table 5-3 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 22 | freqscale\_percore | RW | 0 x0 | Enable each core private FM register |
| 23 | clken\_percore | RW | 0 x0 | Enable each nuclear private clock enable |

When freqscale\_Percore is set to 1, use the freqScale bit in the private split frequency configuration register to split the clock (including Freqscale\_mode). When clken\_percore is set to 1, the clock enable is controlled using the clken bits in the private divider configuration register.

The configurator is defined as follows. The offset is 0x1050.

Table 5-4 Processor core private divider registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 4 | freqscale\_mode | RW | 0 x0 | Frequency division mode selection for the current processor core |
| 3 | clken | RW | 0 x0 | Clock enable for the current processor core |
| The 2-0 | freqscale | RW | 0 x0 | The split frequency setting of the current processor core |

## Node clock frequency division and enabling control

Node clock is the clock used by on-chip network and Shared cache. There are two different control modes: software setting mode and hardware automatic frequency division setting mode.

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

#### The software Settings

The software setting method is compatible with 3A3000 processor. The function is used to set the node frequency division bit in register. The same is used

Set the address of.

The register is base address 0x1fe00000 and offset address 0x0180.

Table 5-5 Function Settings register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 42:40 | Node0\_freq\_ CTRL | RW | 3 'b111 | Node 0 frequency division |

Consistent with the frequency division control of processor core, the node clock can also adjust the clock frequency after frequency division from the original "(frequency division control value +1) /8" to "1/ (frequency division control value +1)" by setting the register. This register is located in the Other Function Settings Register. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 5-6 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 36 | freqscale\_mode\_node | RW | 0 x0 | FM mode selection of nodes |

#### Hardware automatic setting

In addition to the active setting by software, the node clock also supports the automatic frequency division setting triggered by the temperature sensor. Automatic frequency division setting is preset by the software for different temperatures. When the temperature of the temperature sensor reaches the corresponding preset value, the corresponding automatic frequency division setting will be triggered.

In order to ensure the operation of the chip in the high temperature environment, the high temperature can be set to automatically reduce the frequency, so that the chip actively performs clock frequency division when the preset range is exceeded, so as to reduce the chip turnover rate.

There are four sets of control registers to set the behavior for the high temperature frequency reduction function. Each set of registers contains the following four control bits:

GATE: Sets the threshold for high or low temperature. When the input temperature is higher than the high temperature threshold or lower than the low temperature threshold, the frequency division operation will be triggered.

EN: Enabling control. The set of registers is only effective after setting 1.

SEL: Input temperature selection. The 3A4000 currently has four temperature sensors integrated into it, and this register is used to configure which sensor's temperature to select as input.

FREQ: Frequency. When the frequency division operation is triggered, the frequency is also affected by freqscale\_mode\_node. When it is 0, the frequency is adjusted to be (FREQ+1)/8 times of the current clock frequency. When is 1, adjust the frequency to 1/(FREQ+1) times of the current clock frequency.

Its base address is 0x1fe00000 or 0x3ff00000.

Table 5-7 High temperature and frequency drop control register description

|  |  |  |  |
| --- | --- | --- | --- |
| register | address | control | instructions |
| The Thsens\_freq\_scale high temperature drop control register | 0 x1480 | RW | Four groups set the priority from high to low  [7:0] : Scale\_gate0: High temperature threshold value 0, above this temperature will reduce frequency [8:8] : Scale\_en0: high temperature reduce frequency enable 0  [11:10] : Scale\_Sel0: Temperature sensor input source [14:12] : Scale\_freq0: Frequency dividing value when reducing frequency [23:16] : Scale\_gate1: high temperature threshold value 1, exceeding which will reduce frequency [24:24] : Scale\_en1: high temperature reducing frequency enable 1  [27:26] : Scale\_Sel1: Temperature sensor input source [30:28] : Scale\_freq1: Frequency partition value for frequency reduction [399:32] : Scale\_gate2: high temperature threshold value 2, above which the frequency will be reduced [40:40] : Scale\_en2: high temperature frequency reduction enable 2  [43:42] : Scale\_Sel2: Temperature sensor input source [46:44] : Scale\_freq2: Frequency partition value [55:48] : Scale\_gate3: high temperature threshold value 3, above which the high temperature will be reduced [56:56] : Scale\_en3: High temperature reduced frequency enable 3  [59:58] : Scale\_Sel3: Select the input source of temperature sensor with high temperature drop frequency 3  [62:60] : Scale\_freq3: The frequency division value when the frequency is reduced |
| Thsens\_freq\_scale\_up | 0 x1490 | RW | Temperature sensor control register high [7:0] Scale\_Hi\_gate0 high 8 bits  [15:8] Scale\_Hi\_gate1 is 8 bits high  [23:16] Scale\_Hi\_gate2 is 8 bits high  [31:24] Scale\_Hi\_gate3 is 8 bits high  [39:32] Scale\_Lo\_gate0 has a high 8-bit height  [47:40] Scale\_Lo\_gate1 is 8 bits high  [55:48] Scale\_Lo\_gate2 is 8 bits high  [63:56] Scale\_Lo\_gate3 has a high 8-bit height |

## HT controller frequency division and enabling control

The frequency division mechanism of HT controller is similar to others. The two HT controllers can be controlled separately. Set using the corresponding bit in the function Settings register. The base address is 0x1fe00000 and the offset address is 0x0180.

Table 5-8 Function setting registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| they | HT0\_freq\_scale\_ctrl | RW | 3 'b111 | HT controller 0 frequency division |
| 27 | HT0\_clken | RW | 1 'b1 | Whether or not I enabled HT0 |
| he | HT1\_freq\_scale\_ctrl | RW | 3 'b111 | HT controller 1 frequency division |
| 31 | HT1\_clken | RW | 1 'b1 | Whether I enabled HT1 |

Consistent with other frequency division control, HT controller clock can also adjust the clock frequency after frequency division from the original "(frequency division control value +1) /8" to "1/ (frequency division control value +1)" by setting the register. This register is located in the Other Function Settings Register. The base address is 0x1fe00000 and the offset address is 0x0420.

It is important to note that because HT core clock originates from Node Clock, it is also affected by the Node clock frequency.

Table 5-9 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 39:38 | freqscale\_mode\_HT | RW | 0 x0 | Selection of FREQUENCY modulation mode for HT controller |

## Stable Counter split frequency and enable control

The frequency division mechanism of Stable Counter is similar to others. Use other functions to set the corresponding bit in the register. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 5-10 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 21 | stable\_reset | RW | 0 x0 | Stable clock reset control  1: Set to reset state  0: Remove software reset |
| 40 | freqscale\_mode\_stable | RW | 0 x0 | Frequency modulation selection of Stable Clock |
| 46:44 | freqscale\_stable | RW | 0 x0 | Stable Clock frequency modulation register |
| 47 | clken\_stable | RW | 0 x0 | Stable Clock enable |

Note that after stable\_reset is set to 0, the software reset is only unreset. At this point, if GPIO\_FUNC\_en[13] is 1, the reset of stable Counter is also controlled by GPIO[13] (low effective).

GPIO output enable register base address 0x1fe00000, offset address 0x0500.

Table 5-11 GPIO output enable register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | GPIO\_OEn | RW | 32 'HFFFFFFFF | GPIO output enablement (low efficiency) |
| 63:32 | GPIO\_FUNC\_En | RW | 32 'hffff0000 | GPIO functional enablement (low efficiency) |

## Software clock system

The clock in the 3A4000 processor defines several different levels of usage for the system software. Inside the processor core are the traditional Counter/Compare register, the new Stable Counter register, and the chip-level Node Counter register.

Below is an introduction to Stable Counter and Node Counter.

## Stable Counter

Longson 3A4000 introduces a new constant clock source called Stable Counter. The Stable Counter is a separate master clock from the processor core and node clock.

The processor core clock and the node clock are both derived from the master clock, but both can freely control the frequency division (see the previous chapter). The clock of stable Counter is also derived from the master clock, and can also be divided independently without changing with the frequency division of other clocks.

According to the clock source, a timer and a timer are implemented. Please refer to chapter 13 (Timing equipment) of Instruction System Manual of Longson 3A4000 for the use of timers and timers. This chapter focuses on the registers associated with Stable Couter.

### Configuration address of Stable Timer

Using Stable Counter clock source, it implements a counter that increases monotonously and a timer that decreases from set value. Each processor core has its own Stable Counter and Stable Timer. When the processor accesses the timer, it can only access through RDHWR, DRDTIME and other specific instructions. When the processor accesses the timer, it can be accessed through the address, load/ Store, or through the CSR configuration register instruction.

Table 6-1 Address access method

|  |  |  |  |
| --- | --- | --- | --- |
| **The name of the** | **offset** | **permissions** | **describe** |
| Core0\_timer\_config | 0 x1060 | RW | The processor core 0 timer configuration register |
| Core0\_timer\_ticks | 0 x1070 | R | The remaining value of the processor core 0 timer |
| Core1\_timer\_config | 0 x1160 | RW | The processor core 1 timer configuration register |
| Core1\_timer\_ticks | 0 x1170 | R | The remaining value of the processor core 1 timer |
| Core2\_timer\_config | 0 x1260 | RW | Processor core 2's timer configuration register |
| Core2\_timer\_ticks | 0 x1270 | R | The remaining value of the processor core 2 timer |
| Core3\_timer\_config | 0 x1360 | RW | Processor core 3 timer configuration registers |
| Core3\_timer\_ticks | 0 x1370 | R | The remaining value of the processor core 3 timer |

Table 6-2 Configuration register instruction access mode

|  |  |  |  |
| --- | --- | --- | --- |
| **The name of the** | **offset** | **permissions** | **describe** |
| percore\_timer\_config | 0 x1060 | RW | The timer configuration register for the current processor core |
| percore\_timer\_ticks | 0 x1070 | R | The remaining value of the timer for the current processor core |

Table 6-3 The meanings of registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| timer\_config | | | | |
| 63 | 1 | RW | 0 x1 | Reset to 1, write to 1 |
| 62 | Periodic | RW | 0 x0 | Loop counting enablement. When the bit is 1, the timer will be reset to 0 automatically  The value of the InitVal field in timer\_config. |
| 61 | The Enable | RW | 0 x0 | Can always make. When the bit is 1, the timer takes effect. |
| 47:0 | InitVal | RW | 0 x0 | The initial value of the countdown |
| timer\_ticks | | | | |
| 63:48 | 0 | R | 0 x0 | Zero value |
| 47:0 | Ticks | R | 0 x0 | The remaining value of the countdown. When in an acyclic count, when the count is complete, the value stays at 48 'hffff\_ffFF\_FFFF. |

#### Clock control by Stable Counter

Stable Counter USES a master clock and is controlled by a software frequency division mechanism.

The following is the clock control register of Stable Counter. This register is located in the control chip's other function Settings register. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 6-4 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 21 | stable\_reset | RW | 0 x0 | Stable clock reset control  1: Set to reset state  0: Remove software reset |
| 40 | freqscale\_mode\_stable | RW | 0 x0 | Frequency modulation selection of Stable Clock |
| 46:44 | freqscale\_stable | RW | 0 x0 | Stable Clock frequency modulation register |
| 47 | clken\_stable | RW | 0 x0 | Stable Clock enable |

After the BIOS has configured the Stable Counter clock source, the MCSR portion of each processor core needs to be updated to control the values of cpUCfg.0x4 and CPUCFg.0x5. As described in Section 8.1, the crystal oscillator clock frequency in Hz should be filled in cpUCFg.0x4; Cpucfg.0x5 [31:16] should be filled in with the frequency division coefficient; Cpucfg. 0x5[15:0] should be filled in with the frequency doubling factor. The latter two are filled in with the help of BIOS for calculation, so that the result of CCFreq\*CFM/CFD is equal to the actual frequency of Stable Counter.

#### Calibration of Stable Counter

In the case of single chip, the Counter difference of each core is within 2 cycles, no special calibration is required. In the case of multiple chips, there will be a big difference between different chips, so a set of special hardware and software calibration mechanism is needed to reserve the counter difference of each core below 100ns.

First, in order to ensure that the master clock of each chip will not be deviated during use, the same crystal oscillator is used to drive the SYS\_CLK of the chip.

Second, in order to ensure that the Stable Counter of each chip starts timing at the same time, the multiplexing function of two GPIO pins is needed on the hardware. Node 0 USES GPIO12 to output the reset signal, and all other nodes (including node 0) use GPIO13 to input the reset signal (need to be configured as Stable Counter). Buffers are needed on the motherboard to ensure the reset timing sequence (mainly the signal slope). The better the reset timing sequence, the smaller the clock difference between different chips.

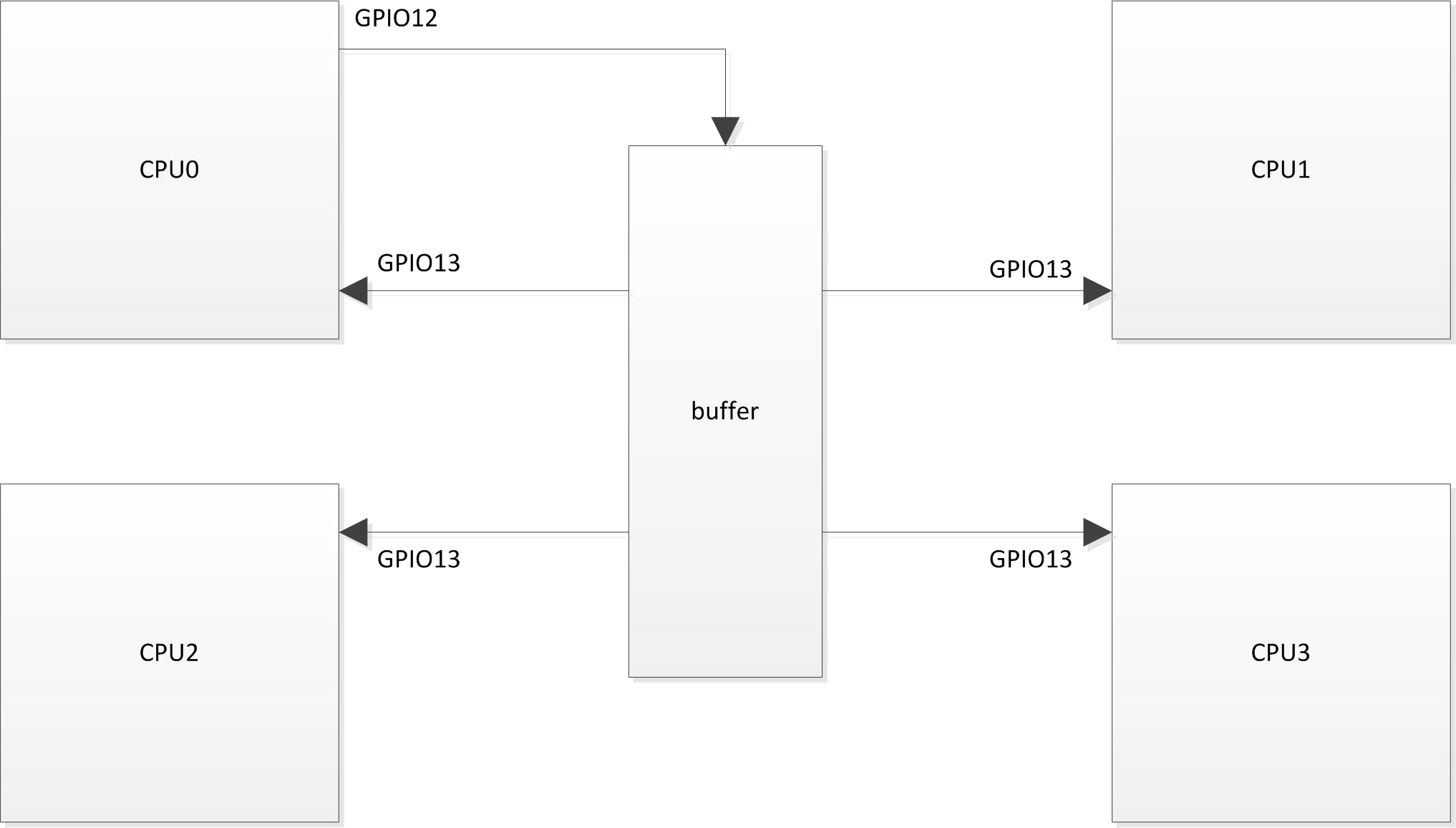
 Before using Stable Counter, the software must reset the global Stable Counter through GPIO12. Before the reset, the clock selection of each chip should be consistent and the reset of each chip has been lifted. This is usually done by the BIOS. The connection scheme of the system is shown in the figure below.

Figure 6-1 Stable reset control for multi-chip interconnection

## The Node Counter

The Node counter address in Loong Chip 3A4000 is the same as that in chip 3A3000 and before, but the original problem requiring software correction is avoided, and register instruction can also be used for access. It is also important to note that, and

The chip of 3A3000 and before is the same. The counting frequency of Node Counter is exactly the same as that of Node Clock. If you want to use Node Counter as the basis for clock calculation, you should avoid frequency conversion of Node Clock.

#### Visit by address

The address access mode is compatible with the 3A3000 processor and is set using the same address. The base address of the configuration register is 0x1fe00000 or 0x3FF00000, as shown in the table below.

Table 6-5 Node Counter registers

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| The Node counter | 0 x0408 | R | 64 bit node clock count |

#### Configure register instruction access

Node Counter USES the configuration register instruction to access it in a slightly different way than the other configuration registers. The use of Node Counter requires all processor cores to visit the same counter, rather than counter on each chip (multiple chips). Thus, even in a multiway system, each chip accesses the CSR[0x408] by configuring the register instruction to access the NODE Counter on NODE 0.

Please refer to the processor core manual for specific access address and register definitions.

## Summary of clock system

The Stable Counter added in Loongson 3A4000 has more advantages than Node Counter and CP0 counter in terms of stability. It will not change with the frequency division of other nodes (Node clock and core Clock).

In terms of ease of use, Stable Counter is also easier to access. With the RDHWR instruction, both user mode and Guest mode can be obtained directly. Stable Counter is the preferred solution for software reference clock systems.

Node Clock is more of a design consideration for traditional compatibility and is a backup solution for clock systems.

## GPIO control

Longson 3A4000 provides up to 32 GPIO for the system, most of which are reused with other functions. With register Settings, GPIO can also be configured to interrupt the input function and can be set to break the level.

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

## Output enable register (0x0500)

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

Table 7-1 Output enable register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | GPIO\_OEn | RW | 32 'HFFFFFFFF | GPIO output enablement (low efficiency) |
| 63:32 | GPIO\_FUNC\_En | RW | 32 'hffff0000 | GPIO functional enablement (low efficiency) |

## Input and Output register (0x0508)

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

Table 7-2 I/O register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | GPIO\_O | RW | 32 'h0 | GPIO output Settings |
| 63:32 | GPIO\_I | RO | 32 'h0 | GPIO input status |

## Interrupt Control register (0x0510)

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

Table 7-3 Interrupt control register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | GPIO\_INT\_Pol | RW | 32 'h0 | GPIO interrupts effective level setting   1. - Low level effective 2. - High level effective |
| 63:32 | GPIO\_INT\_en | RW | 32 'h0 | GPIO interrupts enable control, high efficiency |

## GPIO pin function multiplexing table

The GPIO pins in 3A4000 are heavily reused with other functions. The following list is a selection of pin functions for chip function pins.

In particular, GPIO00 -- GPIO15 chip reset is GPIO function, default is input state, do not drive IO.

GPIO16 -- GPIO31 is the control pin of multiplexing HT, which is reset as HT function. In order to prevent internal logic from driving the corresponding IO, the corresponding HT0/1\_HI /Lo\_Hostmode can be pulled under the lead. At this time, although the default function is still HT, IO pin will not be driven and the external device will not be affected. It is only necessary to set the function to GPIO mode before the software USES THE GPIO function.

Table 7-4 GPIO functional reuse table

|  |  |  |  |
| --- | --- | --- | --- |
| **GPIO registers** | **The name of the pin** | **Reuse function** | **The default function** |
| 0 | GPIO00 | SPI\_CSn1 | GPIO |
| 1 | GPIO01 | SPI\_CSn2 | GPIO |
| 2 | GPIO02 | UART1\_RXD | GPIO |
| 3 | GPIO03 | UART1\_TXD | GPIO |
| 4 | GPIO04 | UART1\_RTS | GPIO |
| 5 | GPIO05 | UART1\_CTS | GPIO |
| 6 | GPIO06 | UART1\_DTR | GPIO |
| 7 | GPIO07 | UART1\_DSR | GPIO |
| 8 | GPIO08 | UART1\_DCD | GPIO |
| 9 | GPIO09 | UART1\_RI | GPIO |
| 10 | GPIO10 | - | GPIO |
| 11 | GPIO11 | - | GPIO |
| 12 | GPIO12 | - | GPIO |
| 13 | GPIO13 | SCNT\_RSTn | GPIO |
| 14 | GPIO14 | PROCHOTn | GPIO |
| 15 | GPIO15 | THERMTRIPn | GPIO |
| 16 | HT0\_LO\_POWEROK | GPIO16 | HT0\_LO\_POWEROK |
| 17 | HT0\_LO\_RSTn | GPIO17 | HT0\_LO\_RSTn |
| 18 | HT0\_LO\_LDT\_REQn | GPIO18 | HT0\_LO\_LDT\_REQn |
| 19 | HT0\_LO\_LDT\_STOPn | GPIO19 | HT0\_LO\_LDT\_STOPn |
| 20 | HT0\_HI\_POWEROK | GPIO20 | HT0\_HI\_POWEROK |
| 21 | HT0\_HI\_RSTn | GPIO21 | HT0\_HI\_RSTn |
| 22 | HT0\_HI\_LDT\_REQn | GPIO22 | HT0\_HI\_LDT\_REQn |
| 23 | HT0\_HI\_LDT\_STOPn | GPIO23 | HT0\_HI\_LDT\_STOPn |
| 24 | HT1\_LO\_POWEROK | GPIO24 | HT1\_LO\_POWEROK |
| 25 | HT1\_LO\_RSTn | GPIO25 | HT1\_LO\_RSTn |
| 26 | HT1\_LO\_LDT\_REQn | GPIO26 | HT1\_LO\_LDT\_REQn |
| 27 | HT1\_LO\_LDT\_STOPn | GPIO27 | HT1\_LO\_LDT\_STOPn |
| 28 | HT1\_HI\_POWEROK | GPIO28 | HT1\_HI\_POWEROK |
| 29 | HT1\_HI\_RSTn | GPIO29 | HT1\_HI\_RSTn |
| 30 | HT1\_HI\_LDT\_REQn | GPIO30 | HT1\_HI\_LDT\_REQn |
| 31 | HT1\_HI\_LDT\_STOPn | GPIO31 | HT1\_HI\_LDT\_STOPn |

## GPIO interrupt control

GPIO pins in 3A4000 can be used as interrupt inputs.

GPIO00, GPIO08, GPIO16, GPIO24 share interrupt controller no. 0 break. GPIO01, GPIO09, GPIO17, GPIO25 share interrupt controller no. 1 break. GPIO02, GPIO10, GPIO18, GPIO26 share interrupt controller no. 2 break. GPIO03, GPIO11, GPIO19, GPIO27 share interrupt controller no. 3 break. GPIO04, GPIO12, GPIO20, GPIO28 share interrupt controller no. 4 break. GPIO05, GPIO13, GPIO21, GPIO29 share interrupt controller no. 5 break. GPIO06, GPIO14, GPIO22, GPIO30 share interrupt controller no. 6 break. GPIO07, GPIO15, GPIO23, GPIO31 share interrupt controller no. 7 break.

The interrupt enable of each GPIO is controlled by the configuration register GPIO\_INT\_en, and the interrupt level is controlled by GPIO\_INT\_POL. The register is as follows:

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

Table 7-5 Interrupt control register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 31:0 | GPIO\_INT\_Pol | RW | 32 'h0 | GPIO interrupts effective level setting  0 - Low level in effect |
|  |  |  |  | 1 - High level in effect |
| 63:32 | GPIO\_INT\_en | RW | 32 'h0 | GPIO interrupts enable control, high efficiency |

When each break on the interrupt controller enables only one bit of GPIO, the edge trigger mode can be used to trigger the interrupt fixed on a certain edge (POL set to 0 on the falling edge, POL set to 1 on the rising edge) and recorded in the interrupt controller.

## GS464V processor core

The GS464V is a quad-emitting 64-bit high-performance processor core. The processor core can be used as a single core for high-end embedded applications and desktop applications, or as a basic processor core to form an on-chip multi-core system for server and high-performance computer applications. The GS464V cores in the loongson 3A4000 and the Shared Cache module form a multi-core structure of the final Cache on the distributed Shared chip through AXI interconnection network. The main features of GS464V are as follows:

* MIPS64 compatible, support godson expansion instruction set;
* Four transmitting superscalar structures, four fixed points, two vectors and two accessors;
* Each vector part is 256bit wide, and each part supports up to 8 double-32-bit floating-point multiplication and addition operations.
* The access part supports 256 bit storage access, the virtual address is 64 bit, and the physical address is 48 bit.
* Support register renaming, dynamic scheduling, transfer prediction and other out-of-order execution techniques;
* 64 items fully connected plus 8 groups connected 2048 items, a total of 2112 TLB, 64 instruction TLB, variable page size;
* The size of the first-level instruction Cache and the data Cache is 64KB each, and the 4-way group is linked.
* The Victim Cache is a private secondary Cache, 256KB in size, connected to a 16-way block.
* Support non-blocking access, load-Speculation and other access optimization technologies;
* Supports Cache consistency protocol, which can be used for on-chip multi-core processors.
* The first-level Cache realizes parity, and the second-level and last-level Cache realizes ECC.
* Support EJTAG debugging standard, convenient for software and hardware debugging; The structure of GS464V is shown in the figure below.

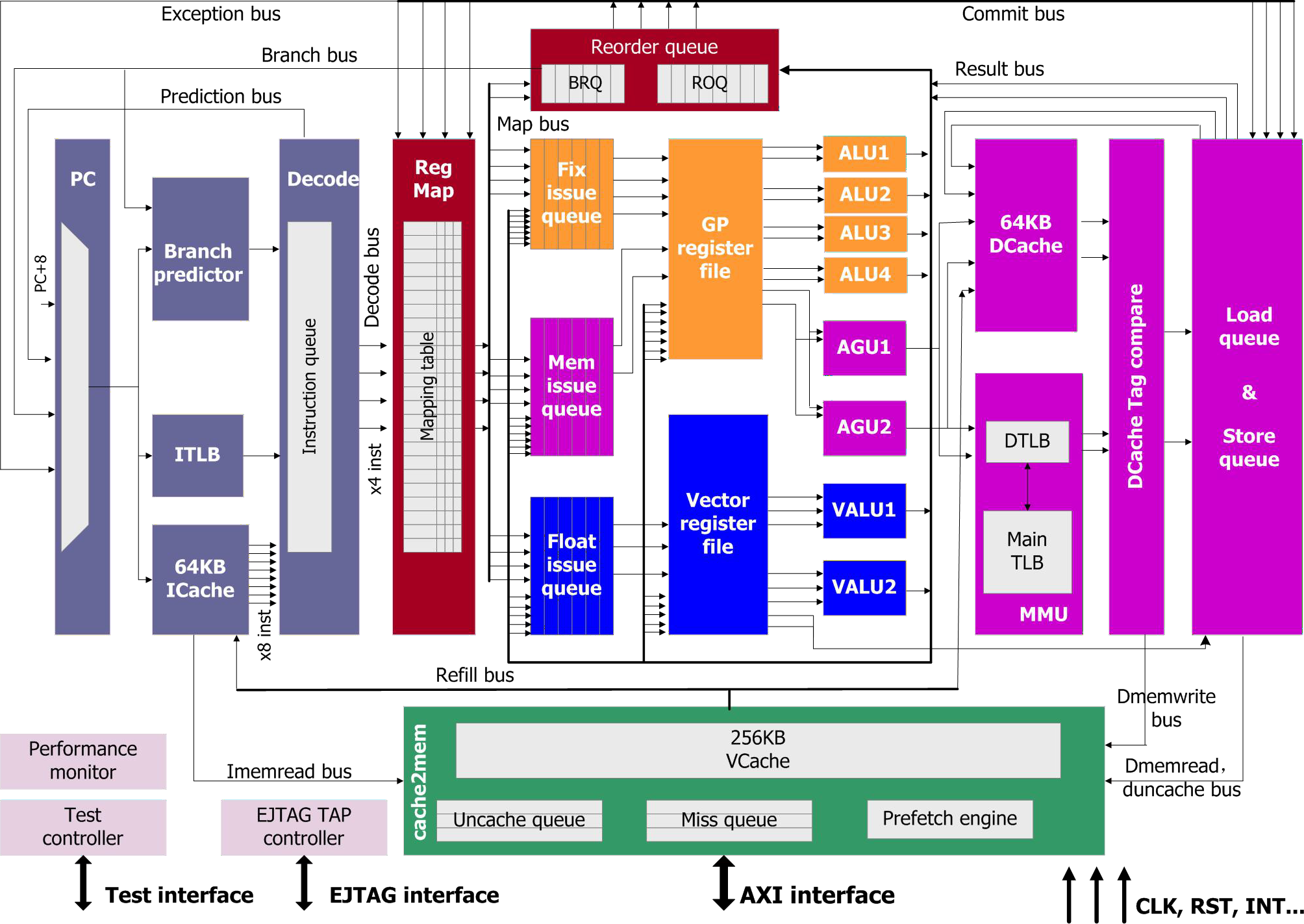


Figure 8-1 GS464V structure diagram

## 3A4000 implements instruction set features

The specific functional characteristics of instruction set of the godson 3A4000 can be identified by means of the defined method in MIPS specification and can also be dynamically confirmed by the instruction set attribute identification mechanism.

The recommended software of Loongson 3A4000 USES customized CPUCFG instructions to identify the loongson instruction set attributes (common information can also be obtained by executing RDCSR to read the relevant CSR, but RDCSR can only be executed in the system state).

CPUCFG instruction is user mode instruction, and its usage is CPUCFG RD, RS, where the register number of the configuration information word to be accessed is stored in the source operand RS register, the configuration word information returned is written into the RD register, and each configuration information word contains up to 32 bits of configuration information. For example, the configuration word No. 1 contains information related to MIPS compatibility in the godson instruction set, where the 0 bit indicates whether hardware floating-point coprocessor is supported or not, the configuration information is represented as CPUCfg.0x1.fp [bit0], where 0x1 represents the configuration information word with the font size of no. 1, and FP represents the configuration information field

The mnemonic name given by mnemonic is FP. Bit0 indicates that the field is located at the 0th bit of the configuration word. If configuration information requires multi-bit representation, its location information will be recorded as bitAA:BB, representing the continuous (AA-BB+1) bit from the AA bit of configuration word to the BB bit.

The following table shows the list of configuration information of 3A4000 implemented instruction set functions. The last column, "possible value," represents the value that is likely to be read from this register, but does not mean that it is read from the 3A4000 processor. The specific read out value shall be subject to the result of the instruction read out by the actual hardware, and the subsequent software judgment shall be made according to the actual read out value. Try not to directly determine whether a certain 3A4000 chip supports or does not support a certain function according to the last column of this table.

Table 8-1 List of instruction set function configuration information implemented by 3A4000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The register no. | A domain | The field name | describe | Possible value |
| 0 x0 | 31:0 | PRId | CP0. PRId | 32 'h14\_8001 |
| 0 x1 | 0 | FP | Equivalent to CP0. Config1. FP (bit0) | 1 'b1 |
| 3:1 | FPRev | The version number of the longson FPU floating point operation that follows the specification | 3 'h2 |
| 4 | MMI | Represents the realization of multimedia instruction extension of the loong chip | 1 'b1 |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 | LSX1 | Represents support for SIMD extension I | 1 'b1 |
| 10 | LSX2 | For 1 means support for SIMD extension II | 1 'b1 |
| 11 | LASX | A 1 indicates support for the advanced SIMD extension | 1 'b1 |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 | CNT64 | A value of 1 indicates that CP0.Count is 64 bits | 1 'b1 |
| 16 | LSLDR0 | A value of 1 indicates that load to R0 is equivalent to prefetching | 1 'b1 |
| 17 | LSPREF | 1 indicates that the PREF instruction has a prefetch effect | 1 'b1 |
| 18 | LSPREFX | Where 1 means that the PREFX directive has a pre-fetched effect | 1 'b1 |
| 19 | LSSYNCI | Where 1 indicates that a SYNCI instruction is implemented as a serialization instruction | 1 'b1 |
| 20 | LSUCA | A value of 1 indicates that partial CACHE execution is supported in user mode  instruction | 1 'b1 |
| 21 | LLSYNC | A 1 indicates the need to add the SYNC 0 instruction before LL | 1 'b0 |
|  | 22 | TGTSYNC | Is 1 indicates that the branch between LL and SC needs to jump on its target  Add the SYNC 0 command | 1 'b0 |
| 23 | LLEXC | A 1 represents the ability to enable the LL directive to make exclusive requests | 1 'b1 |
| 24 | SCRAND | A value of 1 indicates that the supported directory adds a random delay to the LL/SC exclusive request  The function of the late | 1 'b1 |
| 25 | MUALP | A value of 1 indicates support for unaligned access | 1 'b1 |
| 26 | KMUALEn | Is 1 indicates that the non-aligned access memory function has been used in non-user mode  After the opening | 1 'b0 |
| 27 | ITLBT | A value of 1 indicates that ITLB is software transparent | 1 'b1 |
| 28 | LSUPERF | For 1 means that access is allowed in user mode with (D)MFC0  The Performance Counter | 1 'b1 |
| 29 | SFBP | 1 means Store Fill Buffer is supported | 1 'b1 |
| 30 | CDMAP | A value of 1 indicates support for Cache DMA | 1 'b1 |
| 0 x2 | 0 | LEXT1 | 1 represents the realization of the universal extension I of the godson | 1 'b1 |
| 1 | LEXT2 | 1 represents the implementation of the godson universal Extension II | 1 'b1 |
| 2 | LEXT3 | 1 represents the realization of the universal extension III of the godson | 1 'b1 |
| 3 | LSPW | 1 represents the realization of the instruction extension of the loongson page table traversal | 1 'b1 |
| 4 | LBT1 | Where, 1 represents the implementation of loong-son binary translation accelerated extension I  version | 1 'b1 |
| 5 | LBT2 | As 1 represents the implementation of the loong-son binary translation accelerated expansion  II version | 1 'b1 |
| 6 | LBT3 | As 1 represents the implementation of the loong-son binary translation accelerated expansion  III version | 1 'b1 |
| 7 | LBTMMU | The 1 represents the realization of the longson binary translation address conversion  Acceleration mechanisms | 1 'b1 |
| 8 | LPMP | 1 represents the realization of the performance counter of the loong chip  CP0. Config1.PC[bit4] must be 1 | 1 'b1 |
| 11:9 | LPMRev | Loong chip performance counter implementation version number | 3 'h2 |
|  |  |  |  |
| 13 | LPIXU | 1 means support for enabling user-mode loong-son position-independent extension | 1 'b1 |
| 14 | LPIXNU | For 1, support for enabling non-user mode loong-son position independent extension | 1 'b1 |
| 15 | LVZP | A 1 represents the implementation of the loongson virtualization extension | 1 'b1 |
| thou | LVZRev | Version number of the loongson virtualization specification | 3 'h2 |
| 19 | LGFTP | 1 represents the realization of a global constant frequency timing device | 1 'b1 |
|  | Lift up | LGFTPRev | The version number of the global constant frequency timing device | 3 'h2 |
| 23 | LLFTP | 1 represents the realization of a local constant frequency timing device | 1 'b1 |
| they | LLFTPRev | The version number of the local constant frequency timing device | 3 'h2 |
| 27 | LCSRP | A 1 indicates that the status register for the loongson control is supported | 1 'b1 |
| 28 | LDISBLIKELY | A value of 1 indicates support for disabling the likely branch instruction | 1 'b1 |
| 0 x3 | 0 | LCAMP | 1 represents the realization of the hardware lookup table function | 1 'b1 |
| 3:1 | LCAMRev | The version number of the hardware lookup table feature | 3 'h2 |
| 4 | LCAMNUM | Hardware lookup table entries -1 | 8 'h3f |
| then | LCAMKW | Hardware lookup table Key field bit width -1 | 8 'h2f |
| " | LCAMVW | Hardware lookup table Data field bit width -1 | 8 'h3f |
| 0 x4 | 31:0 | CCFreq | Processor core crystal vibration frequency, unit Hz | N/A |
| 0 x 5 | 15:0 | CFM | Processor core frequency doubling factor | N/A |
| Caused the | CFD | Frequency division coefficient of processor core | N/A |
| 0 x6 | 31:0 | Safe | Safety expansion parameters of the loong chip | N/A |
| 0 x7 | 0 | GCCAEQRP | A value of 1 indicates support for Guest CCA as roots-only | 1 'b1 |
| 1 | UCAWINP | A 1 indicates support for non-cache acceleration properties configured by the address window | 1 'b1 |

## 3A4000 configuration status register access

The 3A4000 supports configurable status register space access, and the CSR USES a new independent addressing space called the CSR space, which does not overlap with the existing register space, memory space, and EJTAG Dseg space.

CSR reads and writes through custom RDCSR and WRCSR directives. RDCSR is used as RDCSR RD, RS, where the address of the ACCESSED CSR is stored in the source operand RS register, and the CSR read back is written into the RD register. WRCSR is used as WRCSR RD, RS, where the address of the ACCESSED CSR is stored in the RS register of the source operand, and the value to be written into the CSR is stored in the RD register of the source operand. RDCSR and WRCSR are only allowed to run with a nuclear mindset.

The RDCSR/WRCSR instruction can be used instead of the original address map to configure registers, namely 0x1fe00000 and 0x3FF00000 Spaces. Please refer to the relevant section for the specific access mode.

In addition, the core supports a set of CSR registers that are unique to each processor core, as described below. The following registers cannot be accessed using space 0x3ff00000 and 0x1fe00000.

Table 8-2 List of internal configuration status registers

|  |  |  |
| --- | --- | --- |
| The name of the | address | describe |
| GFTOffset | 0 xfffffffffffffff8 | Fixed frequency timer offset in Guest mode |
| TimerID | 0 xfffffffffffffff0 | The ID number of the local fixed frequency timer |
| CSRffe8 | 0 xffffffffffffffe8 | Please refer to instruction system of Longson 3A4000 for details  Manual" |
| ucacc\_win0\_lo | 0 xfffffffffffffef8 | Low order of non-cache acceleration window 0 |
| ucacc\_win1\_lo | 0 xfffffffffffffef0 | Low order of non-cache acceleration window 1 |
| ucacc\_win2\_lo | 0 xfffffffffffffee8 | Low order of non-cache acceleration window 2 |
| ucacc\_win3\_lo | 0 xfffffffffffffee0 | Low order of non-cache acceleration window 3 |
| ucacc\_win0\_hi | 0 xfffffffffffffeb8 | High level of non-cache acceleration window 0 |
| ucacc\_win1\_hi | 0 xfffffffffffffeb0 | High level of non-cache acceleration window 1 |
| ucacc\_win2\_hi | 0 xfffffffffffffea8 | High level of non-cache acceleration window 2 |
| ucacc\_win3\_hi | 0 xfffffffffffffea0 | High level of non-cache acceleration window 3 |
| MCSRWG | 0 xffffffffffff0000 | MCSR write control |

## Shared Cache (SCache)

SCache module is the three-level Cache Shared by all processor cores in the longson 3A4000 processor. The main features of SCache module include:

* Adopt 128-bit AXI interface.
* The 16-item Cache access queue.
* Keywords first.
* Support for Cache consistency protocol through directories.
* It can be used for on-chip multi-core structure, and can also be directly connected to a single processor IP.
* The 16-channel group linkage structure is adopted.
* Support for ECC validation.
* Support DMA consistent reads and writes and prefetch reads.
* Supports 16 Shared Cache hashes.
* Support sharing Cache by window lock.
* Ensures that read data returns atomicity.

Shared Cache module includes Shared Cache management module Scachemanage and Shared Cache access module ScacheAccess. The Scachemanage module is responsible for the processor's access requests from the processor and DMA, while information such as tags, directories, and data from the Shared Cache is stored in the ScacheAccess module. To reduce power consumption, the tags, directories and DATA of the Shared Cache can be accessed separately. The Shared Cache status bits and w bits are stored together with the TAG, which is stored in TAG RAM, the directory in DIR RAM, and the DATA in DATA RAM. The invalidated request accesses the Shared Cache, reads out the TAG and directory of all channels at the same time, selects the directory according to the TAG, and reads the data according to the hit situation. Replace requests, refill requests, and write back requests only work with tags, directories, and data along the way.

To improve performance for certain computing tasks, Shared Cache adds a locking mechanism. A Shared Cache block in a locked area is locked and will not be replaced (unless the 16-way Shared Cache is full of locked blocks). The chip configuration register space can be used to dynamically configure four sets of lock window registers within the Shared Cache module, but it must ensure that one of the 16 Shared Cache lines is not locked. In addition, when a Shared Cache receives a DMA write request, if the region being written is hit and locked in the Shared Cache, DMA writes are written directly to the Shared Cache rather than to memory.

Table 9-1 Configuration of the Shared Cache lock window register

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | address | A domain | describe |
| Slock0\_valid | 0 x3ff00200 | [63-63] | Lock window 0 valid bit |
| Slock0\_addr | 0 x3ff00200 | [47:0] | Lock window lock address |
| Slock0\_mask | 0 x3ff00240 | [47:0] | Lock window mask 0 |
| Slock1\_valid | 0 x3ff00208 | [63-63] | Lock window 1 valid bit |
| Slock1\_addr | 0 x3ff00208 | [47:0] | Lock address of lock window 1 |
| Slock1\_mask | 0 x3ff00248 | [47:0] | Lock 1 window mask |
| Slock2\_valid | 0 x3ff00210 | [63-63] | Lock window # 2 valid bit |
| Slock2\_addr | 0 x3ff00210 | [47:0] | Lock 2 window lock address |
| Slock2\_mask | 0 x3ff00250 | [47:0] | Lock 2 window mask |
| Slock3\_valid | 0 x3ff00218 | [63-63] | Lock window 3 valid bit |
| Slock3\_addr | 0 x3ff00218 | [47:0] | Lock address for lock window no.3 |
| Slock3\_mask | 0 x3ff00258 | [47:0] | Lock window mask number 3 |

For example, when an address addr makes slock0\_valid & ((addr & slock0\_mask) == (slock0\_addr & slock0\_mask) 1, the address is locked by the lock window 0.

The four scache use the same configuration register with base address 0x1fe00000 and offset address 0x0280. Table 9-2 Shared Cache configuration Register (SC\_CONFIG)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | LRU en | RW | 1 'b1 | Scache LRU replacement algorithm enablement |
| 16 | Prefetch En | RW | 1 'b1 | Scache prefetch function enables |
| Lift up | Prefetch config | RW | 3 'h1 | Stop scache prefetching when it exceeds the address range of the configured size  0-4 KB  1-16 KB  2-64 KB  3-1 MB  7 -- No restrictions  (Note: valid when SCID\_SEL==0) |
| they | Prefetch lookahead | RW | 3 'h2 | Scache prefetch step size  0 - reserve  1-0 x100  2-0 x200  3-0 x300  4-0 x400  5-0 x500  6-0 x600 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 7-0 x700  (Note: valid when SCID\_SEL==0) |
| he | Sc stall dirq cycle | RW | 3 'h2 | SC instruction blocks the number of clock cycles of DIRQ   1. -- 1 cycle (Nonstall) 2. 16-31 - cycle is random 3. Cycle - 32-63 random   3-64-127 cycle the random  4 -- 128-255 cycle Random  Others - Invalid values |
| 31 | MCC storefill en | RW | 1 'b0 | MCC StoreFill function enables |

## Interrupt communication between processor cores

The Loongson 3A4000 implements eight inter-core interrupt registers (IPI) for each processor core to support interrupt and communication between processor cores during BIOS startup and operating system runtime.

Two different access modes are supported in the Loongson 3A4000, one is address access mode compatible with processors such as 3A3000, and the other is to support direct private access to processor register space. The following sections are explained separately.

## Access mode by address

For the loong chip 3A4000, the following registers can be accessed using the base address 0x3FF0\_0000 or 0x1fe0\_0000. Where, the base address 0x3ff0\_0000 can be closed by the DISABle\_0x3FF0 control bit in the routing setting register. The detailed register description and address are shown in Tables 10-1 to 10-5.

Table 10-1 Register and function description of interrupt-related between processor cores

|  |  |  |
| --- | --- | --- |
| The name of the | Read and write access | describe |
| IPI\_Status | R | The 32-bit status register. When any bit is set to 1 and the corresponding bit enables, the processor core INT4 disconnects. |
| IPI\_Enable | RW | The 32-bit enable register, which controls whether the corresponding interrupt bit is valid |
| IPI\_Set | W. | Write 1 to the corresponding bit, then the corresponding STATUS register  Bit is set 1 |
| IPI\_Clear | W. | 32 bit clear register, write 1 to the corresponding bit, then the corresponding STATUS register bit is cleared 0 |
| MailBox0 | RW | The cache register, used to pass parameters at startup, presses 64 or 32 bits  Access by uncache. |
| MailBox01 | RW | The cache register, used to pass parameters at startup, presses 64 or 32 bits  Access by uncache. |
| MailBox02 | RW | The cache register, used to pass parameters at startup, presses 64 or 32 bits  Access by uncache. |
| MailBox03 | RW | The cache register, used to pass parameters at startup, presses 64 or 32 bits  Access by uncache. |

The registers and functions related to intercore interrupt of processor core are described as follows:

Table 10-2 lists of intercore interrupt and communication registers of processor core No. 0

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| Core0\_IPI\_Status | 0 x1000 | R | The IPI\_Status register for the no. 0 processor core |

|  |  |  |  |
| --- | --- | --- | --- |
| Core0\_IPI\_Enalbe | 0 x1004 | RW | The IPI\_Enalbe register for processor core no. 0 |
| Core0\_IPI\_Set | 0 x1008 | W. | The IPI\_Set register for the no. 0 processor core |
| Core0\_IPI\_Clear | 0 x100c | W. | The IPI\_Clear register for the no. 0 processor core |
| Core0\_MailBox0 | 0 x1020 | RW | The IPI\_MailBox0 register for the no. 0 processor core |
| Core0\_ MailBox1 | 0 x1028 | RW | The IPI\_MailBox1 register for the no. 0 processor core |
| Core0\_ MailBox2 | 0 x1030 | RW | The IPI\_MailBox2 register for the no. 0 processor core |
| Core0\_ MailBox3 | 0 x1038 | RW | The IPI\_MailBox3 register for processor core no. 0 |

Table 10-3 List of intercore interrupt and communication registers for processor core No. 1

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| Core1\_IPI\_Status | 0 x1100 | R | The IPI\_Status register for processor core No. 1 |
| Core1\_IPI\_Enalbe | 0 x1104 | RW | The IPI\_Enalbe register for processor core 1 |
| Core1\_IPI\_Set | 0 x1108 | W. | The IPI\_Set register of processor core 1 |
| Core1\_IPI\_Clear | 0 x110c | W. | The IPI\_Clear register of processor core no. 1 |
| Core1\_MailBox0 | 0 x1120 | R | The IPI\_MailBox0 register for processor core 1 |
| Core1\_ MailBox1 | 0 x1128 | RW | The IPI\_MailBox1 register of processor core 1 |
| Core1\_ MailBox2 | 0 x1130 | W. | The IPI\_MailBox2 register of processor core 1 |
| Core1\_ MailBox3 | 0 x1138 | W. | The IPI\_MailBox3 register of processor core 1 |

Table 10-4 List of intercore interrupt and communication registers for processor cores No. 2

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| Core2\_IPI\_Status | 0 x | R | IPI\_Status register for no. 2 processor core |
| Core2\_IPI\_Enalbe | 0 x1204 | RW | The IPI\_Enalbe register for processor core 2 |
| Core2\_IPI\_Set | 0 x1208 | W. | The IPI\_Set register for processor core 2 |
| Core2\_IPI\_Clear | 0 x120c | W. | IPI\_Clear register for no. 2 processor core |
| Core2\_MailBox0 | 0 x1220 | R | The IPI\_MailBox0 register for processor no.2 core |
| Core2\_ MailBox1 | 0 x1228 | RW | The IPI\_MailBox1 register for processor no.2 core |
| Core2\_ MailBox2 | 0 x1230 | W. | The IPI\_MailBox2 register for processor no.2 core |
| Core2\_ MailBox3 | 0 x1238 | W. | The IPI\_MailBox3 register for processor no.2 core |

Table 10-5 List of intercore interrupt and communication registers for No. 3 processor core

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| Core3\_IPI\_Status | 0 x1300 | R | IPI\_Status register for no. 3 processor core |
| Core3\_IPI\_Enalbe | 0 x1304 | RW | The IPI\_Enalbe register for processor core 3 |
| Core3\_IPI\_Set | 0 x1308 | W. | The IPI\_Set register for processor core 3 |
| Core3\_IPI\_Clear | 0 x130c | W. | The IPI\_Clear register of processor core 3 |
| Core3\_MailBox0 | 0 x1320 | R | The IPI\_MailBox0 register for processor core 3 |
| Core3\_ MailBox1 | 0 x1328 | RW | The IPI\_MailBox1 register for processor core 3 |
| Core3\_ MailBox2 | 0 x1330 | W. | The IPI\_MailBox2 register for processor core 3 |
| Core3\_ MailBox3 | 0 x1338 | W. | The IPI\_MailBox3 register for processor core 3 |

The list above is a list of intercore interruption-related registers for a single-node multiprocessor system consisting of a single loong chip 3A4000. When multi-chip Longshon 3A4000 interconnect is used to form multi-node CC-NUMa system, the node in each chip corresponds to the global node number of the system, and the IPI register address of processor core in the node is fixed offset relation according to the address of base of the node in the table above. For example, the IPI\_Status address of node no. 0 processor core is 0x3FF01000, and the 0 processor address of node No. 1 is 0x10003FF01000, and so on.

## Configure register instruction access

In the Loongson 3A4000, the processor core has direct register access instructions, which can be accessed through private space to the configuration register. In order to make it easier to use the interkernel interrupt register, some adjustments are made to the interkernel interrupt register definition in this mode.

Table 10-6 List of interrupt and communication registers between current processor cores

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| perCore\_IPI\_Status | 0 x1000 | R | The IPI\_Status register for the current processor core |
| perCore\_IPI\_Enalbe | 0 x1004 | RW | The IPI\_Enalbe register for the current processor core |
| perCore\_IPI\_Set | 0 x1008 | W. | The IPI\_Set register for the current processor core |
| perCore\_IPI\_Clear | 0 x100c | W. | The IPI\_Clear register for the current processor core |
| perCore\_MailBox0 | 0 x1020 | RW | The IPI\_MailBox0 register for the current processor core |
| PerCore\_ MailBox1 | 0 x1028 | RW | The IPI\_MailBox1 register for the current processor core |
| PerCore\_ MailBox2 | 0 x1030 | RW | The IPI\_MailBox2 register for the current processor core |
| PerCore\_ MailBox3 | 0 x1038 | RW | The IPI\_MailBox3 register for the current processor core |

To make an interrupt request and MailBox communication to other cores, access is made through the following registers.

Table 10-7 Processor inter-core communication registers

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| IPI\_Send | 0 x1040 | send | 32-bit interrupt distribution register  [31] Wait for the completion mark, and wait for the interruption to take effect when set to 1  [there]  [25:16] Processor core  [language]  [4:0] interrupt vector sign, corresponding to the vector in IPI\_Status |
| Mail\_Send | 0 x1048 | send | The 64-bit MailBox cache register  63: [32] MailBox data  [31] Wait for the completion flag, and wait for the write to take effect when set to 1  [there]  [25:16] Processor core  [language]  [and] the MailBox   1. -Mailbox0 low 32 bits 2. -Mailbox0 is 32 bits high 3. -Mailbox1 low 32 bits 4. -Mailbox1 is 32 bits high 5. - MailBox2 low 32 bits 6. -Mailbox2 is 32 bits high 7. -Mailbox3 low 32 bits 8. - MailBox4 is 32 bits high   [1:0] |
| FREQ\_Send | 0 x1058 | send | 32-bit frequency enable register  [31] Wait for the completion mark, and wait for the setting to take effect when set to 1  [there]  [25:16] Processor core  [language]  Writes to the corresponding processor core private frequency configuration register.  CSR x1050 [0] |

Note that since the Mail\_Send register can only send 32 bits of data at a time, when sending 64 bits of data it must be split into two sends. Therefore, while the target core is waiting for the Mail\_Box content, other software measures are needed to ensure the integrity of the transport. For example, after the Mail\_Box data has been sent, an intercore interrupt indicates that it has been sent.

## I/O interrupt

The Loong Chip 3A4000 supports two different interrupt modes. The first is the traditional interrupt mode, which is compatible with 3A3000 and other processors. The second is a new extended IO interrupt mode to support the interrupt chipping and dynamic distribution functions of HT controller. The two modes of interruption are described below.

## Traditional I/O interrupt

The traditional interrupt of The Loongson 3A4000 chip supports 32 interrupt sources and is managed in a unified manner, as shown in Figure 7-1 below. Any IO interrupt source can be configured to enable, fire, and route the target processor core interrupt pin. Traditional interrupts do not support interrupted cross-chip distribution and can only interrupt processor cores on the same processor chip.

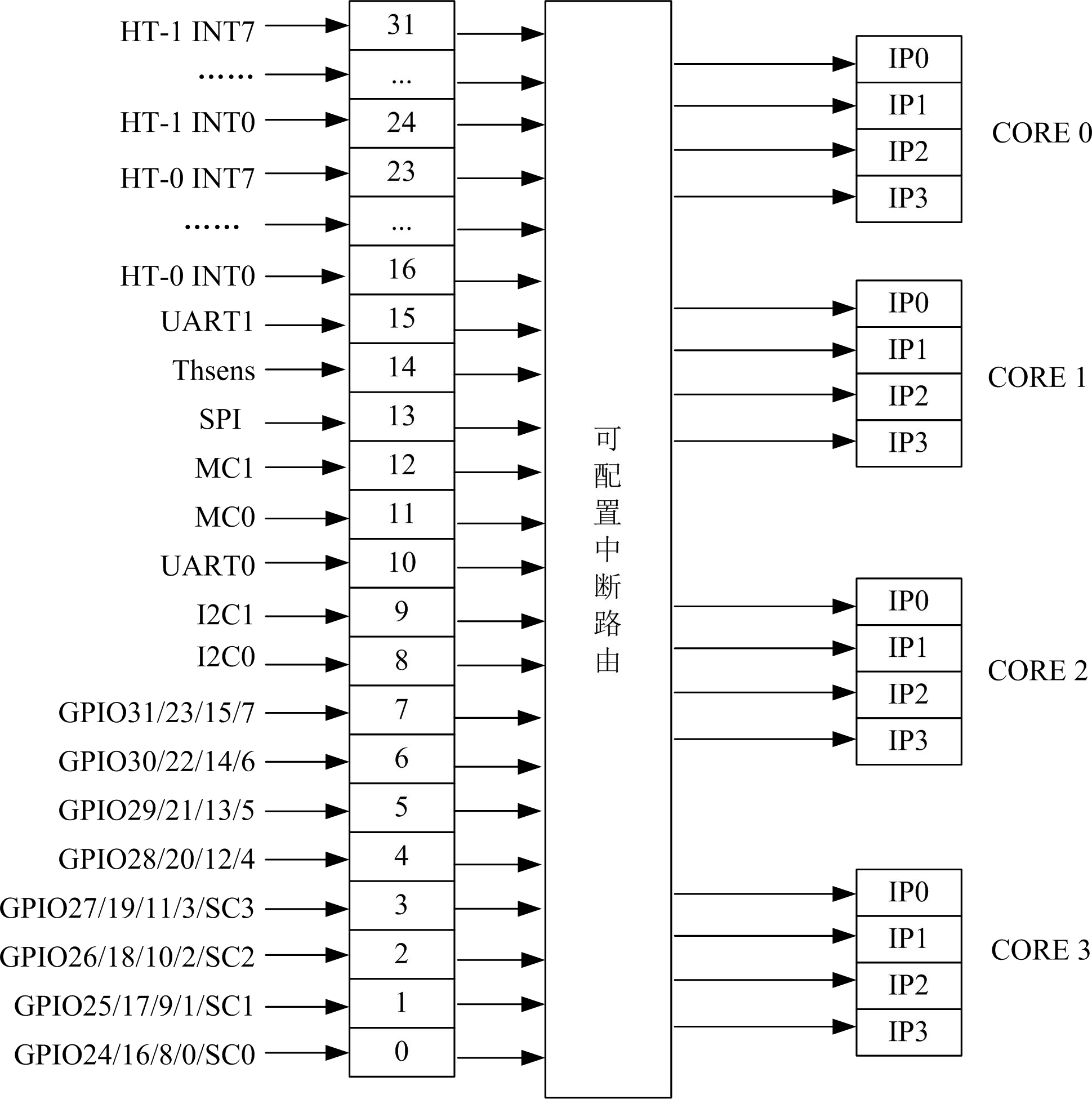


FIG. 11-1 Schematic diagram of interrupt routing for longshon 3A4000 processor

Interrupt-related configuration registers are used to control the corresponding broken wires in the form of bits. Interrupt control bit connection and property configuration are shown in the table below.

The interrupt Enable configuration has three registers: Intenset, Intenclr, and Inten. The Intenset sets the interrupt enabled, and the Intenset register writes 1 to the interrupt enabled. The Intenclr clears the interrupt enable, and the interrupt corresponding to the write 1 bit in the Intenclr register is cleared. The Inten register reads the current state of each interrupt enable.

The interrupt signal of edge trigger is selected by Intedge configuration register. Write 1 means edge trigger and write 0 means level trigger. The interrupt handler can clear the interrupt record with the corresponding bit in Intenclr, and the interrupt enablement is also cleared.

Table 11-1 Interrupt control register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A domain | Access properties/default values | | | | |
| Intedge | Inten | Intenset | Intenclr | The interrupt source |
| 0 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO24/16/8/0 / SC0 |
| 1 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO25/17/9/1 / SC1 |
| 2 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO26 18/10/2 / SC 2 |
| 3 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO27 19/11/3 / SC / 3 |
| 4 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO28/20/12/4 |
| 5 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO29/21/13/5 |
| 6 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO30/22/14/6 |
| 7 | RW / 0 | R / 0 | RW / 0 | RW / 0 | GPIO31/23/15/7 |
| 8 | RW / 0 | R / 0 | RW / 0 | RW / 0 | I2C0 |
| 9 | RW / 0 | R / 0 | RW / 0 | RW / 0 | I2C1 |
| 10 | RW / 0 | R / 0 | RW / 0 | RW / 0 | UART0 |
| 11 | RW / 0 | R / 0 | RW / 0 | RW / 0 | MC0 |
| 12 | RW / 0 | R / 0 | RW / 0 | RW / 0 | MC1 |
| 13 | RW / 0 | R / 0 | RW / 0 | RW / 0 | SPI |
| 14 | RW / 0 | R / 0 | RW / 0 | RW / 0 | Thsens |
| 15 | RW / 0 | R / 0 | RW / 0 | RW / 0 | UART1 |
| 23:16 | RW / 0 | R / 0 | RW / 0 | RW / 0 | HT0 [away] |
| 31:24 | RW / 0 | R / 0 | RW / 0 | RW / 0 | HT1 [away] |

Similar to intercore interrupts, the base address of an IO interrupt can also be accessed using 0x1fe00000 or 0x3FF00000, or through the processor core's special register configuration instruction.

#### Visit by address

This access is compatible with processors such as 3A3000, and the base address can be 0x1fe00000 or 0x3FF00000. The base address of 0x3ff00000 can be disabled by the DISABle\_0x3FF0 control bit in the routing configuration register.

Table 11-2 IO control register address

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| Intisr | 0 x1420 | 32-bit interrupt status register |
| Inten | 0 x1424 | The 32-bit interrupt enabled status register |
| Intenset | 0 x1428 | The 32-bit setup enable register |
| Intenclr | 0 x142c | The 32-bit clear enable register |
| Intedge | 0 x1434 | 32 bit trigger mode register |
| CORE0\_INTISR | 0 x1440 | 32-bit interrupt status routed to CORE0 |
| CORE1\_INTISR | 0 x1448 | 32-bit interrupt status routed to CORE1 |
| CORE2\_INTISR | 0 x1450 | 32-bit interrupt status routed to CORE2 |
| CORE3\_INTISR | 0 x1458 | 32-bit interrupt status routed to CORE3 |

Four processor cores are integrated into the Loongson 3A4000, and the 32-bit interrupt source above can be configured to select the desired interrupt target processor core. Further, the interrupt source can optionally route to either of the processor core interrupts INT0 to INT3, IP2 to IP5 corresponding to CP0\_Status. Each of the 32 I/O interrupt sources corresponds to an 8-bit routing controller, and its format and address are shown in Tables 11-3 and 11-4. The routing register USES vector routing, such as 0x48, to route to the INT2 of processor 3.

Table 11-3 Description of interrupt routing register

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

Table 11-4 Interrupt routing register addresses

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name offset address description Name offset address description | | | | | |
| Entry0 | 0 x1400 | GPIO24/16/8/0 | Entry16 | 0 x1410 | HT0 - int0 |
| Entry1 | 0 x1401 | GPIO25/17/9/1 | Entry17 | 0 x1411 | HT0 - int1 |
| Entry2 | 0 x1402 | GPIO26/18/10/2 | Entry18 | 0 x1412 | HT0 - int2 |
| Entry3 | 0 x1403 | GPIO27/19/11/3 | Entry19 | 0 x1413 | HT0 - int3 |
| Entry4 | 0 x1404 | GPIO28/20/12/4 | Entry20 | 0 x1414 | HT0 - int4 |
| Entry5 | 0 x1405 | GPIO29/21/13/5 | Entry21 | 0 x1415 | HT0 - int5 |
| Entry6 | 0 x1406 | GPIO30/22/14/6 | Entry22 | 0 x1416 | HT0 - int6 |
| Entry7 | 0 x1407 | GPIO31/23/15/7 | Entry23 | 0 x1417 | HT0 - int7 |
| Entry8 | 0 x1408 | I2C0 | Entry24 | 0 x1418 | HT1 - int0 |
| Entry9 | 0 x1409 | I2C1 | Entry25 | 0 x1419 | HT1 - int1 |
| Entry10 | 0 x140a | UART0 | Entry26 | 0 x141a | HT1 - int2 |
| Entry11 | 0 x140b | MC0 | Entry27 | 0 x141b | HT1 - int3 |
| Entry12 | 0 x140c | MC1 | Entry28 | 0 x141c | HT1 - int4 |
| Entry13 | 0 x140d | SPI | Entry29 | 0 x141d | HT1 - int5 |
| Entry14 | 0 x140e | Thsens | Entry30 | 0 x141e | HT1 - int6 |
| Entry15 | 0 x140f | UART1 | Entry31 | 0 x141f | HT1 - int7 |

#### Configure register instruction access

In the loong chip 3A4000, the configuration register can also be accessed through private space through the access method of configuration register instruction. The offset address used by the instruction is the same as that accessed through the address. In addition, for the convenience of users, a dedicated private interrupt status register is set for each core for different current interrupt states, as shown in the table below.

Table 11-5 Processor core private interrupt status register

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| perCore\_INTISR | 0 x1010 | 32-bit interrupt status routed to the current processor core |

## Extend I/O interrupt

In addition to compatibility with the original traditional IO interrupt mode, the 3A4000 began to support extended I/O interrupts, which were used to distribute the 256-bit interrupts on the HT bus directly to each processor core instead of forwarding them through the HT interrupts, improving the flexibility of IO interrupt usage.

The kernel needs to enable the corresponding bits in the "other function Settings register" before it can interrupt with the extension IO. The register is base address 0x1fe00000 and offset address 0x0420.

Table 11-6 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 48 | EXT\_INT\_en | RW | 0 x0 | Extend IO interrupt enablement |

In extended IO interrupt mode, HT interrupt can be directly carried out across the chip forwarding and rotation distribution operations. The current version supports up to 256 extended interrupt vectors.

#### Visit by address

Here are the related extended IO interrupt registers. As with other configuration registers, the base address can be accessed using 0x1fe00000 or 0x3FF00000, or through the processor core's special register configuration instruction.

Table 11-7 Extended IO interrupt enable register

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| EXT\_IOIen [63:0] | 0 x1600 | Extend the interrupt enable configuration for IO interrupt [63:0] |
| EXT\_IOIen [127, 64] | 0 x1608 | Extend the interrupt enable configuration for IO interrupts [127:64] |
| EXT\_IOIen [191, 128] | 0 x1610 | Extend the interrupt enablement configuration for IO interrupts [191:128] |
| EXT\_IOIen [255, 192] | 0 x1618 | Extension IO interrupt [255:192] interrupt enable configuration |

Table 11-8 Extension IO interrupt Automatic wheel enable register

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| EXT\_IOIbounce [63:0] | 0 x1680 | Extend IO interrupt [63:0] automatic rotation enabled configuration |
| EXT\_IOIbounce [127, 64] | 0 x1688 | Extended IO interrupt [127:64] automatic rotation enabled configuration |
| EXT\_IOIbounce [191, 128] | 0 x1690 | Extend the automatic rotation enable configuration for IO interrupt [191:128] |
| EXT\_IOIbounce [255, 192] | 0 x1698 | Extend IO interrupt [255:192] automatic rotation enablement configuration |

Table 11-9 Extended IO interrupt status register

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| EXT\_IOIsr [63:0] | 0 x1700 | Extends the interrupt state of IO interrupt [63:0] |
| EXT\_IOIsr [127, 64] | 0 x1708 | Extends the interrupt state of IO interrupt [127:64] |
| EXT\_IOIsr [191, 128] | 0 x1710 | Extends the interrupt state of IO interrupt [191:128] |
| EXT\_IOIsr [255, 192] | 0 x1718 | Extends the interrupt state of IO interrupt [255:192] |

Table 11-10 Extended IO interrupt status registers for each processor core

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| CORE0\_EXT\_IOIsr [63:0] | 0 x1800 | Interrupt status of extended IO interrupt [63:0] routed to processor core 0 |
| CORE0\_EXT\_IOIsr [127, 64] | 0 x1808 | Interrupt status of extended IO interrupt routed to processor core 0 [127:64] |
| CORE0\_EXT\_IOIsr [191, 128] | 0 x1810 | Interrupt status of extended IO interrupt [191:128] routed to processor core 0 |
| CORE0\_EXT\_IOIsr [255, 192] | 0 x1818 | Interrupt status of extended IO interrupt [255:192] routed to processor core 0 |
| CORE1\_EXT\_IOIsr [63:0] | 0 x1900 | Interrupt status of extended IO interrupt [63:0] routed to processor core 1 |
| CORE1\_EXT\_IOIsr [127, 64] | 0 x1908 | Interrupt status of extended IO interrupt routed to processor core 1 [127:64] |
| CORE1\_EXT\_IOIsr [191, 128] | 0 x1910 | Interrupt status of extended IO interrupt [191:128] routed to processor core 1 |
| CORE1\_EXT\_IOIsr [255, 192] | 0 x1918 | Interrupt status of extended IO interrupt routed to processor core 1 [255:192] |
| CORE2\_EXT\_IOIsr [63:0] | 0 x1a00 | Interrupt status of extended IO interrupt [63:0] routed to processor core 2 |
| CORE2\_EXT\_IOIsr [127, 64] | 0 x1a08 | Interrupt status of extended IO interrupt routed to processor core 2 [127:64] |
| CORE2\_EXT\_IOIsr [191, 128] | 0 x1a10 | Interrupt status of extended IO interrupt routed to processor core 2 [191:128] |
| CORE2\_EXT\_IOIsr [255, 192] | 0 x1a18 | Interrupt status of extended IO interrupt routed to processor core 2 [255:192] |
| CORE3\_EXT\_IOIsr [63:0] | 0 x1b00 | Interrupt status of extended IO interrupt [63:0] routed to processor core 3 |
| CORE3\_EXT\_IOIsr [127, 64] | 0 x1b08 | Interrupt status of extended IO interrupt routed to processor core 3 [127:64] |
| CORE3\_EXT\_IOIsr [191, 128] | 0 x1b10 | Interrupt status of extended IO interrupt [191:128] routed to processor core 3 |
| CORE3\_EXT\_IOIsr [255, 192] | 0 x1b18 | Interrupt status of extended IO interrupt routed to processor core 3 [255:192] |

Similar to traditional IO interrupts, the 256-bit interrupt source for extended IO interrupts can also select the target processor core for desired interrupts through software configuration.

However, the interrupt source can not be selected separately to route to any one of the processor core interrupts INT0 to INT3. Instead, INT interrupts are routed in groups to interrupt IP2 to IP5 corresponding to CP0\_Status. Below is the interrupt pin routing register configured by group.

Table 11-11 Description of the interrupt pin routing register

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

Table 11-12 Interrupt routing register addresses

|  |  |  |
| --- | --- | --- |
| Name offset address description | | |
| EXT\_IOImap0 | 0 x14c0 | Pin routing for EXT\_IOI[31:0] |
| EXT\_IOImap1 | 0 x14c1 | Pin routing for EXT\_IOI[63:32] |
| EXT\_IOImap2 | 0 x14c2 | Pin routing for EXT\_IOI[95:64] |
| EXT\_IOImap3 | 0 x14c3 | Pin routing for EXT\_IOI[127:96] |

|  |  |  |
| --- | --- | --- |
| EXT\_IOImap4 | 0 x14c4 | Pin routing for EXT\_IOI[159:128] |
| EXT\_IOImap5 | 0 x14c5 | Pin routing for EXT\_IOI[191:160] |
| EXT\_IOImap6 | 0 x14c6 | Pin routing for EXT\_IOI[223:192] |
| EXT\_IOImap7 | 0 x14c7 | Pin routing for EXT\_IOI[255:224] |

Each interrupt source corresponds to an 8-bit routing controller, whose format and address are shown in Table 11-13 and table below

11 minus 14. Where [7:4] is used to select the real node routing vector in Table 11-5. The routing register USES a vector for routing selection, such as 0x48, indicating the processor core no. 3 of the node indicated by EXT\_IOI\_node\_type4.

Table 11-13 Description of the interrupt target processor core routing register

|  |  |
| --- | --- |
| A domain | Said Ming |
| 3-0 | Routing processor kernel vector number |
| The log | Node mapping mode selection of routing (as shown in Table 11-15) |

It is important to note that when using the cycle distribution mode (which corresponds to EXT\_IOIbounce 1), the cycle is rotated on a fully mapped node number to the processor core number. EXT\_IOIbounce should be set after the associated routing map configuration.

When using fixed distribution mode (which corresponds to 0 EXT\_IOIbounce), only one bit or all zeros on the bitmap of node Numbers are allowed, corresponding to the local trigger.

Table 11-14 Interrupt target processor core routing register addresses

|  |  |  |
| --- | --- | --- |
| Name offset address description | | |
| EXT\_IOImap\_Core0 | 0 x1c00 | EXT\_IOI[0] processor core routing |
| EXT\_IOImap\_Core1 | 0 x1c01 | EXT\_IOI[1] 's processor core routing |
| EXT\_IOImap\_Core2 | 0 x1c02 | EXT\_IOI[2] 's processor core routing |
| ... |  |  |
| EXT\_IOImap\_Core254 | 0 x1cfe | EXT\_IOI[254] processor core routing |
| EXT\_IOImap\_Core255 | 0 x1cff | EXT\_IOI[255] processor core routing |

Table 11-15 Interrupt target node mapping mode configuration

|  |  |  |
| --- | --- | --- |
| Name offset address description | | |
| EXT\_IOI\_node\_type0 | 0 x14a0 | Mapping vector 0 of 16 nodes (Software configuration) |
| EXT\_IOI\_node\_type1 | 0 x14a2 | Mapping vector 1 of 16 nodes (Software configuration) |
| EXT\_IOI\_node\_type2 | 0 x14a4 | Mapping vector 2 of 16 nodes (Software configuration) |
| ... |  |  |
| EXT\_IOI\_node\_type15 | 0 x14be | Mapping vector 15 of 16 nodes (Software configuration) |

#### Configure register instruction access

When accessing the processor core's configuration register instruction, the biggest difference is that access to the processor core's interrupt status register becomes private, and each core only needs to make a query request to the same address to get the current core's interrupt status.

Table 11-16 Extended IO interrupt status register for current processor core

|  |  |  |
| --- | --- | --- |
| The name of the | offset | describe |
| PerCore\_EXT\_IOIsr [63:0] | 0 x1800 | Interrupt status of extended IO interrupt [63:0] routed to the current processor core |
| PerCore\_EXT\_IOIsr [127, 64] | 0 x1808 | Interrupt status of extended IO interrupt routed to current processor core [127:64] |
| PerCore\_EXT\_IOIsr [191, 128] | 0 x1810 | Interrupt status of extended IO interrupt [191:128] routed to the current processor core |
| PerCore\_EXT\_IOIsr [255, 192] | 0 x1818 | Interrupt status of extended IO interrupt [255:192] routed to the current processor core |

#### Extends IO interrupt trigger register

To support dynamic distribution of extended IO interrupts, an extended IO interrupt trigger register was added to the configuration register to set the corresponding IO interrupt. You can usually use this register to debug or test interrupts.

The description of this register is as follows:

Table 11-17 Extended IO interrupt trigger register

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| EXT\_IOI\_send | 0 x1140 | send | Extends the IO interrupt setup register  [7:0] the interrupt vector set for the expectation |

#### Differences between extended IO interrupts and traditional HT interrupt handling

In the traditional HT interrupt processing mode, HT interrupts are processed internally by HT controller, which is directly mapped to 256 interrupt vectors on HT configuration register, and then grouped by 256 interrupt vectors to generate 4 or 8 interrupts, and then routed to various processor cores. Due to the traditional disconnection, cross-chip interrupt cannot be generated directly, so all HT IO interrupts can only be processed directly by a single chip. On the other hand, the interrupt of in-chip hardware distribution is only at the final 4

The interrupts are in units of 8 or 8 and cannot be processed in bits, resulting in poor hardware interrupt distribution.

Extended IO interrupt mode, HT interrupt is processed by HT controller directly sent to the interrupt controller of the chip, the interrupt controller can directly get 256 bits interrupt, instead of the previous 4 or 8 bits interrupt, each of these 256 bits interrupt can be unique

Vertical routing, independent distribution, and can achieve cross-chip distribution and rotation.

After an interrupt with extended IO, the software handles it slightly differently than it does with a traditional HT interrupt.

In traditional HT interrupt processing, the kernel looks directly at the interrupt vector (typically 0x90000EFdFB000080) of the HT controller, and then processes it bitwise. At this point, all interrupts on the HT controller are read directly regardless of the routing mode configuration.

After an interrupt with extended IO, the kernel reads the interrupt state directly to the extended IO state register (configuration space 0x1800) for processing, and each core only reads the interrupt's own interrupt state and processes it, with no interference between different cores.

## Temperature sensor

## Real-time temperature sampling

The longson 3A4000 is internally integrated with two temperature sensors, which can be observed through the sampling register at 0x1FE00198. At the same time, it can be controlled with flexible high-low temperature interrupt alarm or automatic frequency modulation function. The corresponding bits of the temperature sensor in the sampling register are as follows (base address 0x1FE00000, offset address 0x0198) :

Table 12-1 Description of temperature sampling registers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **The field name** | | **access** | **Reset value** | **describe** |
| 24 | | Thsens0\_overflow | R |  | Temperature sensor 0 overflows |
| 25 | | Thsens1\_overflow | R |  | Temperature sensor 1 overflows |
| 47:32 | | Thsens0\_out | R |  | Temperature sensor 0 ℃  Node temperature =Thens0\_out  \* 731/0 x4000-273  Temperature range -40-125 degrees |
| 65:48 | | Thsens1\_out | R |  | Temperature sensor 1 ℃  Node temperature =Thens1\_out  - \* 731/0 x4000-273  Temperature range -40-125 degrees |

By setting the control register, the function of interruption above the preset temperature, interruption below the preset temperature and automatic frequency reduction at high temperature can be realized.

In addition, the new Celsius temperature register can be used to read the current Celsius temperature directly. This register can also be accessed using 0x1FE00000 or 0x3FF00000 for base address reads, or directly using the configuration register instruction at offset 0x0428. The register is described as follows:

Table 12-2 Extended IO interrupt trigger register

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| Thsens\_Temperature | 0 x0428 | R | Temperature sensor Celsius temperature |

## High and low temperature interrupt triggers

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

GATE: Sets the threshold for high or low temperature. When the input temperature is higher than the high temperature threshold or lower than the low temperature threshold, it will produce

Interrupt;

EN: Interrupt enable control. The set of registers is only effective after setting 1.

SEL: Input temperature selection. Two temperature sensors are currently integrated into the 3A4000, and this register is used to configure which sensor's temperature to select as input. You could use a 0 or a 1.

The high temperature interrupt control register contains 4 sets of setting bits used to control the high temperature interrupt trigger. Low temperature interrupt control register

The device contains four sets of Settings for controlling the low temperature interrupt trigger. There is also a set of registers for displaying interrupt status, corresponding to high-temperature and low-temperature interrupts, and any write to this register will clear the interrupted status.

These registers are described as follows, and their base addresses are 0x1fe00000 or 0x3FF00000:

Table 12-3 Description of high and low temperature interrupt registers

|  |  |  |  |
| --- | --- | --- | --- |
| register | address | control | instructions |
| The high temperature interrupt control register Thsens\_int\_ctrl\_Hi | 0 x1460 | RW | [7:0] : Hi\_gate0: High temperature threshold, above which interrupt will be generated [8:8] : Hi\_en0: high temperature interrupt enable 0  [11:10] : Hi\_Sel0: Select high temperature interrupt 0 input source of temperature sensor [23:16] : Hi\_gate1: high temperature threshold 1, exceeding which will generate interrupt [24:24] : Hi\_en1: high temperature interrupt enable 1  [27:26] : Hi\_Sel1: select the input source of temperature sensor for HTS interrupt 1 [39:32] : Hi\_gate2: HTS threshold 2, exceeding which will generate interrupt [40:40] : Hi\_en2: HTS interrupt enable 2  [43:42] : Hi\_Sel2: Select the temperature sensor input source of HTS interrupt 2 [55:48] : Hi\_gate3: HTS threshold 3, exceeding which will generate interrupt [56:56] : Hi\_en3: HTS interrupt enable 3  [59:58] : Hi\_Sel3: Select the input source of temperature sensor for high-temperature interrupt 3 |
| The cryogenic interrupt control register Thsens\_int\_ctrl\_Lo | 0 x1468 | RW | [7:0] : Lo\_gate0: low temperature threshold below which interrupts will be generated [8:8] : Lo\_en0: low temperature interrupt enabled 0  [11:10] : Lo\_Sel0: Select the temperature sensor input source of low temperature interrupt 0 [23:16] : Lo\_gate1: low temperature threshold 1, below which an interrupt will occur [24:24] : Lo\_en1: low temperature interrupt enabling 1  [27:26] : Lo\_Sel1: select the temperature sensor input source of low temperature interrupt 1 [39:32] : Lo\_gate2: low temperature threshold 2, below which an interrupt [40:40] : Lo\_en2: low temperature interrupt enabling 2  [43:42] : Lo\_Sel2: Select the temperature sensor input source of low temperature interrupt 2 [55:48] : Lo\_gate3: low temperature threshold 3, below which an interrupt will occur [56:56] : Lo\_en3: low temperature interrupt enabling 3  [59:58] : Lo\_Sel3: Select the temperature sensor input source of low temperature interrupt 3 |
| The interrupt status register Thsens\_int\_status/ CLR | 0 x1470 | RW | Interrupt status register, write any value clear interrupt [0] : high temperature interrupt trigger  [1] : Low temperature interrupt trigger |

## High temperature automatic frequency reduction setting

In order to ensure the operation of the chip in the high temperature environment, the high temperature can be set to automatically reduce the frequency, so that the chip actively performs clock frequency division when the preset range is exceeded, so as to reduce the chip turnover rate.

There are four sets of control registers to set the behavior for the high temperature frequency reduction function. Each set of registers contains the following four control bits:

GATE: Sets the threshold for high or low temperature. When the input temperature is higher than the high temperature threshold or lower than the low temperature threshold, the frequency division operation will be triggered.

EN: Enabling control. The set of registers is only effective after setting 1.

SEL: Input temperature selection. The 3A4000 currently has four temperature sensors integrated into it, and this register is used to configure which sensor's temperature to select as input.

FREQ: Frequency. When the frequency division operation is triggered, the clock is divided using the preset FREQ, and the frequency division mode is controlled by freqscale\_mode\_node.

Its base address is 0x1fe00000 or 0x3ff00000.

Table 12-4 Description of high temperature and frequency drop control register

|  |  |  |  |
| --- | --- | --- | --- |
| register | address | control | instructions |
|  |  |  | Four groups set the priority from high to low |
|  |  |  | [7:0] : Scale\_gate0: High temperature threshold value 0, beyond which frequency will be reduced |
|  |  |  | [8:8] : Scale\_en0: High temperature frequency drop enables 0 |
|  |  |  | [11:10] : Scale\_Sel0: Select the input source of temperature sensor with high temperature drop frequency 0 |
|  |  |  | [14:12] : Scale\_freq0: Frequency division value at frequency reduction |
|  |  |  | [23:16] : Scale\_gate1: High temperature threshold value 1, beyond which the frequency will drop |
|  |  |  | [24:24] : Scale\_en1: High temperature drop frequency enable 1 |
|  |  |  | [27:26] : Scale\_Sel1: Select the input source of temperature sensor with high temperature drop frequency 1 |
|  |  |  | [30:28] : Scale\_freq1: Frequency division value at frequency reduction |
|  |  |  | [39:32] : Scale\_gate2: high temperature threshold 2, beyond which the frequency will drop |
|  |  |  | [40:40] : Scale\_en2: High temperature drop frequency enable 2 |
|  |  |  | [43:42] : Scale\_Sel2: Select the input source of temperature sensor with high temperature and frequency reduction 2 |
|  |  |  | [46:44] : Scale\_freq2: The frequency division value when the frequency is reduced |
|  |  |  | [55:48] : Scale\_gate3: High temperature threshold 3, beyond which frequency will be lowered |
|  |  |  | [56:56] : Scale\_en3: High temperature drop frequency enable 3 |
| High temperature drop frequency control register |  |  | [59:58] : Scale\_Sel3: Select the input source of temperature sensor with high temperature drop frequency 3 |
| Thsens\_freq\_scale | 0 x1480 | RW | [62:60] : Scale\_freq3: The frequency division value when the frequency is reduced |
| Thsens\_freq\_scale\_up | 0 x1490 | RW | Temperature sensor control register high [7:0] Scale\_Hi\_gate0 high 8 bits  [15:8] Scale\_Hi\_gate1 is 8 bits high  [23:16] Scale\_Hi\_gate2 is 8 bits high  [31:24] Scale\_Hi\_gate3 is 8 bits high  [39:32] Scale\_Lo\_gate0 has a high 8-bit height  [47:40] Scale\_Lo\_gate1 is 8 bits high  [55:48] Scale\_Lo\_gate2 is 8 bits high  [63:56] Scale\_Lo\_gate3 has a high 8-bit height |

## Temperature status detection and control

PROCHOTn and THERMTRIPn are used for temperature status detection and control. The two signals are reused with GPIO14 and GPIO15 respectively. PROCHOTn can be used as both input and output, while THERMTRIPn has only output function.

When PROCHOTn is used as the input, the chip is controlled by the external temperature detection circuit. When the external temperature detection circuit needs to reduce the chip temperature, PROCHOTn can be set to 0. When the chip receives the low level, it will take frequency reduction measures. When PROCHOTn is output, the chip can output high-temperature interrupt. Through prochotn\_O\_SEL register, one of the four interrupts set in the high-temperature interrupt control register is selected as the high-temperature interrupt emitted from outside.

THERMTRIPn is the output, and the chip selects one of the four interrupts set in the high-temperature interrupt control register as the outgoing high-temperature interrupt by thermtripN\_O\_SEL register.

Although THERMTRIPn and PROCHOTn are external high temperature interruption, THERMTRIPn is more urgent than PROCHOTn. When PROCHOTn is set, the external temperature control circuit can also take certain measures, such as increasing the fan speed. When THERMTRIPn is set, the external power control circuit should take emergency power outage measures directly.

The specific control registers are as follows:

Table 12-5 Description of temperature status detection and control register

|  |  |  |  |
| --- | --- | --- | --- |
| register | address | control | instructions |
| Temperature status detection and control register  Thsens\_hi\_ctrl | 0 x1498 | RW | [0:0] : Prochotn\_OE PROCHOTn pin output enable control, 0 for output, 1 for input  [5:4] : Prochotn\_o\_sel PROCHOTn high-temperature Interrupt Output selection [10:8] : Prochotn\_freq\_scale: The frequency division value when PROCHOTn input is effective  [17:16] : Thermtripn\_o\_sel THERMTRIPn high temperature interrupt output selection |

# Temperature sensor control

3A4000 integrates four temperature sensors, which can adjust temperature/voltage monitoring through register configuration, monitoring point configuration and monitoring frequency configuration, etc. The output content of each temperature sensor can also be directly observed for debugging. (Base address 0x1FE00000, offset address of temperature sensor configuration register 0x01580+ vtSENsor\_id <<4, offset address of temperature sensor data register 0x01588+ Vtsensor\_id <<4)

Table 12-6 Temperature sensor configuration register description

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **The field name** | | **access** | **Reset value** | **describe** |
| 0 | | Thsens\_trigger | RW | 0 | Sens\_mode and Thsens\_Cluster can be used to select the monitoring mode and monitoring point. 0 is the default temperature monitoring mode, and the monitoring point is set by  Temp\_cluster configuration. |
| 2 | | Thsens\_mode | RW | 0 | 0: Temperature mode; 1: Voltage mode |
| 3 | | Thsens\_datarate | RW | 0 | Monitoring frequency:  0-10 to 20 hz  1-325 ~ 650 hz |
| 6:4 | | Thsens\_cluster | RW | 0 | Sensor monitoring point configuration: 0 for local monitoring  Point 1~7 are remote monitoring points |
| 8 | | Temp\_valid | RW | 0 | The values of Thsens0\_out and Thsens0\_overflow in CSR[0x198] are the temperature monitoring values of the temperature sensor. |
| 11:9 | | Temp\_cluster | RW | 0 | Temperature sensor output monitoring point selection,  Thsens\_trigger causes energy to be ineffective |

Table 12-7 Description of temperature sensor data register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **The field name** | | **access** | **Reset value** | **describe** |
| 3 | | Out\_mode | R | 0 | Sensor configuration monitoring mode  0: Temperature mode; 1: Voltage mode |
| 6:4 | | Out\_cluster | R | 0 | Sensor configuration monitoring point |
| 7 | | Overflow | R | 0 | Sensor monitoring value overflow |
| Was a | | The Data | R | 0 | The monitoring value read by the sensor |

Calculation method of readout value:

Node temperature =data\*731/ 0x4000-273 (temperature range -40 ° ~ 125 °)

The voltage =data\*1.226/0x1000 monitoring points are configured as follows

Table 12-8 Description of temperature sensor monitoring points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **The sensor** | **Cluster** | **Monitoring stations** | **The sensor** | **Cluster** | **Monitoring stations** |
| 0 | 0 | Reserved | 2 | 0 | Reserved |
| 1 | Core0 monitoring point 0 | 1 | Core2 monitoring point 0 |
| 2 | Core0 monitoring point 1 | 2 | Core2 monitoring point 1 |
| 3 | Core0 monitoring point 2 | 3 | Scache2 |
| 4 | Core0 monitoring point 3 | 4 | Mc1-phy monitoring point 0 |
| 5 | SCache0 | 5 | Mc0-phy monitoring point 0 |
| 6 | HT0 | 6 | Mc0 CTRL - |
| 7 | Reserved | 7 | Reserved |
| 1 | 0 | Reserved | 3 | 0 | Reserved |
| 1 | Core1 monitoring point 0 | 1 | Core3 monitoring point 2 |
| 2 | Core1 monitoring point 1 | 2 | Core3 monitoring point 3 |
| 3 | Core1 monitoring point 2 | 3 | Scache3 |
| 4 | SCache1 | 4 | Mc0-phy monitoring point 1 |
| 5 | L1X | 5 | Mc1-phy monitoring point 1 |
| 6 | HT1 | 6 | Mc1 CTRL - |
| 7 | NOC - VERT | 7 | L2X |

## Ddr3/4 SDRAM controller configuration

The internal integrated memory controller of the Loongson 3A4000 processor is designed to comply with the DDR3/4 SDRAM industry standard

Jesd79-3 and JESD79-4. In the Godson 3A4000 processor, all memory read/write operations implemented comply with the provisions of JESD79-3 and JESD79-4.

## Ddr3/4 SDRAM controller features Overview

The Longson 3A4000 processor supports DDP and 3DS package modes. DDP supports up to 8 CS (by 8 DDR3/DDR4 SDRAM chips, namely 4 double-sided memory chips), and 3DS supports up to 4 CS (by 8 DDR4 SDRAM chips, namely 32 logical ranks). A total of 22 bit address buses are included (i.e., 18 bit column address bus, 2 bit logical Bank bus and 2 bit logical Bank Group bus, where the column address bus is associated with RASn, CASn and Wen reuse).

The parameters of the DDR3/4 controller can be adjusted to support the longson 3A4000 processor when different memory chip types are selected. Where, the supported maximum block selection (CS\_n) is 8, the logical RANK (CHIP ID) number is 8, the ROW address (ROW) number is 18, the column address (COL) number is 12, the logical body selection (BANK) number is 2 (DDR4) or 3 (DDR3), and the logical body Group is 2 (DDR4 only). DDR3 and DDR4 pins have a multiplexing relationship, as shown in the table below. In addition, the multiplexing relationship between CS\_n and Chip ID can be matched, please refer to section 13.4 for details.

Table 13-1 DDR3/4 Address control signal multiplexing

|  |  |  |
| --- | --- | --- |
| **The name of the PAD** | **DDR3** | **DDR4** |
| DDR\_ACTn | DDR\_A15 | DDR\_ACTn |
| DDR\_RASn | DDR\_RASn | DDR\_RASn/DDR\_A16 |
| DDR\_CASn | DDR\_CASn | DDR\_CASn/DDR\_A15 |
| DDR\_WEn | DDR\_WEn | DDR\_WEn/DDR\_A14 |
| DDR\_BG [1] | DDR\_A14 | DDR\_BG1 |
| DDR\_BG [0] | DDR\_BA [2] | DDR\_BG0 |

The physical address of the memory request sent by the CPU can be mapped to many different addresses, depending on the configuration within the controller

Shoot.

The memory control circuit integrated with the Loongson 3A4000 processor only accepts memory reads/from the processor or external devices

Write request, in all memory read/write operations, memory control circuit is in from device State (Slave State). The memory controller in the Godson 3A4000 processor has the following characteristics:

* + - Interface command, read and write data full flow operation;
    - Memory command consolidation and sorting improve the overall bandwidth;
    - The configuration register reads and writes the port, may modify the memory device basic parameter;
    - Built-in dynamic delay compensation circuit (DCC) for reliable sending and receiving of data;
    - ECC function can detect 1-bit and 2-bit errors on the data path and automatically correct 1-bit errors.
    - Ddr3/4 SDRAM is supported, and x4, X8 and X16 particles are supported in parameter configuration.
    - The frequency ratio of controller to PHY is 1/2;
    - Support data transfer rate range from 800Mbps to 3200Mbps.

## Ddr3/4 SDRAM read operation protocol

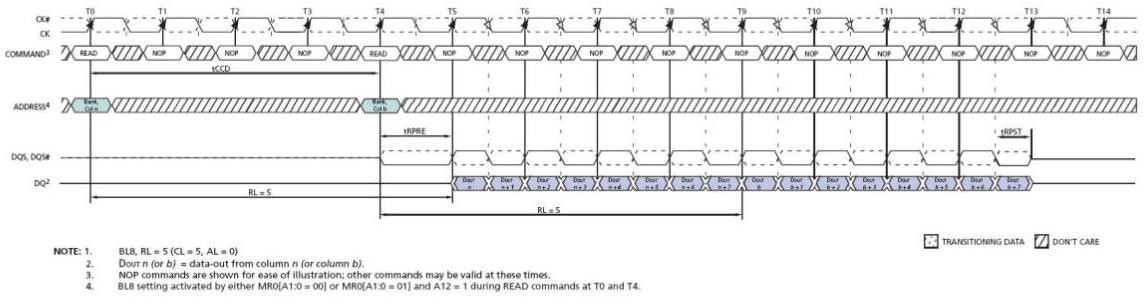
 The protocol for DDR3 SDRAM read operations is shown in Figure 13-1. In the figure, the Command (CMD) consists of RAS\_n, CAS\_n and WE\_n signals. For read operations, RAS\_n=1, CAS\_n=0, and WE\_n=1.

Figure 13- 1 DDR3 SDRAM read operation protocol

In the figure above, Cas Latency (CL) =5, Read Latency (RL) =5, and Burst Length = 8. DDR4 SDRAM read operation protocol is similar. In the figure, the command CMD consists of ACT\_n, RAS\_n, CAS\_n, and WE\_n

Signal composition. For read operations, ACT\_n=1, RAS\_n=1, CAS\_n=0, WE\_n=1.

## Ddr3/4 SDRAM write operation protocol

The protocol for DDR3 SDRAM write operations is shown in Figure 13-2. In the figure, the command CMD consists of RAS\_n, CAS\_n and WE\_n. For write operations, RAS\_n=1, CAS\_n=0, WE\_n=0. In addition, different from read operation, write operation can identify the data mask of write operation through DQM, that is, the number of bytes to be written. DQM is synchronized with DQS signal in the figure.

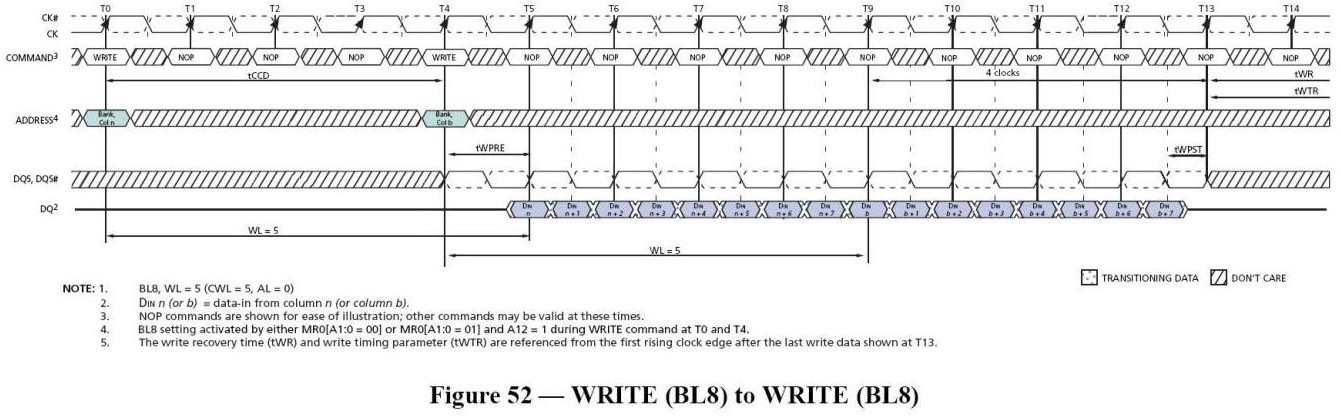


Figure 13-2 DDR3 SDRAM write operation protocol

In the figure above, Cas Latency (CL) =5, Wead Latency (WL) =5, and Burst Length = 8. DDR4 SDRAM write operation protocol is similar. In the figure, the command CMD consists of ACT\_n, RAS\_n, CAS\_n, and WE\_n

Signal composition. For read operations, ACT\_n=1, RAS\_n=1, CAS\_n=0, WE\_n=0.

## Ddr3/4 SDRAM parameter configuration format

13.4.1 Memory controller parameter list

Table 13-2 List of parameters visible to memory controller software

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Offset** | **63:55** | **55:48** | **47:40** | **39:32** | **came** | **Ephron;** | **"** | **away** |
| **PHY** | | | | | | | | |
| 0 x0000 |  |  |  |  |  |  | Version (RD) | |
| 0 x0008 |  |  | x4\_mode | ddr3\_mode |  |  | Capability (RD) | |
| 0 x0010 |  |  |  |  |  |  | Dram\_init (RD) | init\_start |
| 0 x0018 |  |  |  |  |  |  |  |  |
| 0 x0020 |  |  |  |  |  |  | preamble2 | rdfifo\_valid |
| 0 x0028 |  | Rdfifo\_empty (RD) | | |  | Overflow (RD) | | |
| 0 x0030 |  | Dll\_value (RD) | Dll\_init\_done (RD) | dll\_lock\_mode | dll\_bypass | dll\_adjj\_cnt | dll\_increment | dll\_start\_point |
| 0 x0038 |  |  |  | dll\_dbl\_fix |  |  | dll\_close\_disable | dll\_ck |
| 0 x0040 |  |  |  | dbl\_ctrl\_ckca |  |  |  | dll\_dbl\_ckca |
| 0 x0048 | pll\_ctrl\_ckca | | | | Pll\_lock\_ckca (RD) | Dll\_lock\_ckca (RD) | clken\_ckca | clksel\_ckca |
| 0 x0050 |  |  |  | dbl\_ctrl\_ds\_0 |  |  |  | dll\_dbl\_ds\_0 |
| 0 x0058 | pll\_ctrl\_ds\_0 | | | | Pll\_lock\_ds\_0 (RD) | Dll\_lock\_ds\_0 (RD) | clken\_ds\_0 | clksel\_ds\_0 |
| 0 x0060 |  |  |  | dbl\_ctrl\_ds\_1 |  |  |  | dll\_dbl\_ds\_1 |
| 0 x0068 | pll\_ctrl\_ds\_1 | | | | Pll\_lock\_ds\_1 (RD) | Dll\_lock\_ds\_1 (RD) | clken\_ds\_1 | clksel\_ds\_1 |
| 0 x0070 |  |  |  | dbl\_ctrl\_ds\_2 |  |  |  | dll\_dbl\_ds\_2 |
| 0 x0078 | pll\_ctrl\_ds\_2 | | | | Pll\_lock\_ds\_2 (RD) | Dll\_lock\_ds\_2 (RD) | clken\_ds\_2 | clksel\_ds\_2 |
| 0 x0080 |  |  |  | dbl\_ctrl\_ds\_3 |  |  |  | dll\_dbl\_ds\_3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x0088 | pll\_ctrl\_ds\_3 | | | | Pll\_lock\_ds\_3 (RD) | Dll\_lock\_ds\_3 (RD) | clken\_ds\_3 | clksel\_ds\_3 |
| 0 x0090 |  |  |  | dbl\_ctrl\_ds\_4 |  |  |  | dll\_dbl\_ds\_4 |
| 0 x0098 | pll\_ctrl\_ds\_4 | | | | Pll\_lock\_ds\_4 (RD) | Dll\_lock\_ds\_4 (RD) | clken\_ds\_4 | clksel\_ds\_4 |
| 0 x00a0 |  |  |  | dbl\_ctrl\_ds\_5 |  |  |  | dll\_dbl\_ds\_5 |
| 0 x00a8 | pll\_ctrl\_ds\_5 | | | | Pll\_lock\_ds\_5 (RD) | Dll\_lock\_ds\_5 (RD) | clken\_ds\_5 | clksel\_ds\_5 |
| 0 x00b0 |  |  |  | dbl\_ctrl\_ds\_6 |  |  |  | dll\_dbl\_ds\_6 |
| 0 x00b8 | pll\_ctrl\_ds\_6 | | | | Pll\_lock\_ds\_6 (RD) | Dll\_lock\_ds\_6 (RD) | clken\_ds\_6 | clksel\_ds\_6 |
| 0 x00c0 |  |  |  | dbl\_ctrl\_ds\_7 |  |  |  | dll\_dbl\_ds\_7 |
| 0 x00c8 | pll\_ctrl\_ds\_7 | | | | Pll\_lock\_ds\_7 (RD) | Dll\_lock\_ds\_7 (RD) | clken\_ds\_7 | clksel\_ds\_7 |
| 0 x00d0 |  |  |  | dbl\_ctrl\_ds\_8 |  |  |  | dll\_dbl\_ds\_8 |
| 0 x00d8 | pll\_ctrl\_ds\_8 | | | | Pll\_lock\_ds\_8 (RD) | Dll\_lock\_ds\_8 (RD) | clken\_ds\_8 | clksel\_ds\_8 |
| 0 x00e0 |  |  | vrefclk\_inv | vref\_sample | | vref\_num | vref\_dly | dll\_vref |
| ... |  |  |  |  |  |  |  |  |
| 0 x0100 |  |  |  |  | dll\_1xdly\_0 | dll\_1xgen\_0 | dll\_wrdqs\_0 | dll\_wrdq\_0 |
| 0 x0108 |  |  |  |  |  | dll\_gate\_0 | dll\_rddqs1\_0 | dll\_rddqs0\_0 |
| 0 x0110 | rdodt\_ctrl\_0 | rdgate\_len\_0 | rdgate\_mode\_0 | rdgate\_ctrl\_0 |  |  | dqs\_oe\_ctrl\_0 | dq\_oe\_ctrl\_0 |
| 0 x0118 |  |  |  |  |  | dly\_2x\_0 | redge\_sel\_0 | Rddqs\_phase\_0 (RD) |
| 0 x0120 | W\_bdly0\_0 [would] | W\_bdly0\_0 [he] | W\_bdly0\_0 [thou wouldest | W\_bdly0\_0 [when] | W\_bdly0\_0 [even] | W\_bdly0\_0 [8] | W\_bdly0\_0 [17] | W\_bdly0\_0 [3-0] |
| 0 x0128 |  | W\_bdly0\_0 [59:56] | W\_bdly0\_0 [55:52] | W\_bdly0\_0 [spoilers] | W\_bdly0\_0 [47:44] | W\_bdly0\_0 [43:40] | W\_bdly0\_0 [39:36] | W\_bdly0\_0 [has] |
| 0 x0130 | W\_bdly1\_0 [so] | W\_bdly1\_0 [formerly] | W\_bdly1\_0 [he] | W\_bdly1\_0 [then] | W\_bdly1\_0 [9] | W\_bdly1\_0 [therefore] | W\_bdly1\_0 [53] | W\_bdly1\_0 [2-0] |
| 0 x0138 |  |  |  |  |  |  |  | W\_bdly1\_0 [but] |
| 0 x0140 |  |  |  |  |  |  | Rg\_bdly\_0 [17] | Rg\_bdly\_0 [3-0] |
| 0 x0148 |  |  |  |  |  |  |  |  |
| 0 x0150 | Rdqsp\_bdly\_0 [would] | Rdqsp\_bdly\_0 [27:  24] | Rdqsp\_bdly\_0 [23:  20] | Rdqsp\_bdly\_0 [when] | Rdqsp\_bdly\_0 [even] | Rdqsp\_bdly\_0 [8] | Rdqsp\_bdly\_0 [17] | Rdqsp\_bdly\_0 [3-0] |
| 0 x0158 |  |  |  |  |  |  |  | Rdqsp\_bdly\_0 [has] |
| 0 x0160 | Rdqsn\_bdly\_0 [would] | Rdqsn\_bdly\_0 [27:  24] | Rdqsn\_bdly\_0 [23:  20] | Rdqsn\_bdly\_0 [when] | Rdqsn\_bdly\_0 [even] | Rdqsn\_bdly\_0 [8] | Rdqsn\_bdly\_0 [17] | Rdqsn\_bdly\_0 [3-0] |
| 0 x0168 |  |  |  |  |  |  |  | Rdqsn\_bdly\_0 [has] |
| 0 x0170 | Rdq\_bdly\_0 [so] | Rdq\_bdly\_0 [formerly] | Rdq\_bdly\_0 [he] | Rdq\_bdly\_0 [then] | Rdq\_bdly\_0 [9] | Rdq\_bdly\_0 [therefore] | Rdq\_bdly\_0 [53] | Rdq\_bdly\_0 [2-0] |
| 0 x0178 |  |  |  |  |  |  |  | Rdq\_bdly\_0 [but] |
| 0 x0180 |  |  |  |  | dll\_1xdly\_1 | dll\_1xgen\_1 | dll\_wrdqs\_1 | dll\_wrdq\_1 |
| 0 x0188 |  |  |  |  |  | dll\_gate\_1 | dll\_rddqs1\_1 | dll\_rddqs0\_1 |
| 0 x0190 | rdodt\_ctrl\_1 | rdgate\_len\_1 | rdgate\_mode\_1 | rdgate\_ctrl\_1 |  |  | dqs\_oe\_ctrl\_1 | dq\_oe\_ctrl\_1 |
| 0 x0198 |  |  |  |  |  | dly\_2x\_1 | redge\_sel\_1 | Rddqs\_phase\_1 (RD) |
| 0 x01a0 | W\_bdly0\_1 [would] | W\_bdly0\_1 [he] | W\_bdly0\_1 [thou wouldest | W\_bdly0\_1 [when] | W\_bdly0\_1 [even] | W\_bdly0\_1 [8] | W\_bdly0\_1 [17] | W\_bdly0\_1 [3-0] |
| 0 x01a8 |  | W\_bdly0\_1 [59:56] | W\_bdly0\_1 [55:52] | W\_bdly0\_1 [spoilers] | W\_bdly0\_1 [47:44] | W\_bdly0\_1 [43:40] | W\_bdly0\_1 [39:36] | W\_bdly0\_1 [has] |
| 0 x01b0 | W\_bdly1\_1 [so] | W\_bdly1\_1 [formerly] | W\_bdly1\_1 [he] | W\_bdly1\_1 [then] | W\_bdly1\_1 [9] | W\_bdly1\_1 [therefore] | W\_bdly1\_1 [53] | W\_bdly1\_1 [2-0] |
| 0 x01b8 |  |  |  |  |  |  |  | W\_bdly1\_1 [but] |
| 0 x01c0 |  |  |  |  |  |  | Rg\_bdly\_1 [17] | Rg\_bdly\_1 [3-0] |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x01c8 |  |  |  |  |  |  |  |  |
| 0 x01d0 | Rdqsp\_bdly\_1 [would] | Rdqsp\_bdly\_1 [27:  24] | Rdqsp\_bdly\_1 [23:  20] | Rdqsp\_bdly\_1 [when] | Rdqsp\_bdly\_1 [even] | Rdqsp\_bdly\_1 [8] | Rdqsp\_bdly\_1 [17] | Rdqsp\_bdly\_1 [3-0] |
| 0 x01d8 |  |  |  |  |  |  |  | Rdqsp\_bdly\_1 [has] |
| 0 x01e0 | Rdqsn\_bdly\_1 [would] | Rdqsn\_bdly\_1 [27:  24] | Rdqsn\_bdly\_1 [23:  20] | Rdqsn\_bdly\_1 [when] | Rdqsn\_bdly\_1 [even] | Rdqsn\_bdly\_1 [8] | Rdqsn\_bdly\_1 [17] | Rdqsn\_bdly\_1 [3-0] |
| 0 x01e8 |  |  |  |  |  |  |  | Rdqsn\_bdly\_1 [has] |
| 0 x01f0 | Rdq\_bdly\_1 [so] | Rdq\_bdly\_1 [formerly] | Rdq\_bdly\_1 [he] | Rdq\_bdly\_1 [then] | Rdq\_bdly\_1 [9] | Rdq\_bdly\_1 [therefore] | Rdq\_bdly\_1 [53] | Rdq\_bdly\_1 [2-0] |
| 0 x01f8 |  |  |  |  |  |  |  | Rdq\_bdly\_1 [but] |
| 0 x0200 |  |  |  |  | dll\_1xdly\_2 | dll\_1xgen\_2 | dll\_wrdqs\_2 | dll\_wrdq\_2 |
| 0 x0208 |  |  |  |  |  | dll\_gate\_2 | dll\_rddqs1\_2 | dll\_rddqs0\_2 |
| 0 x0210 | rdodt\_ctrl\_2 | rdgate\_len\_2 | rdgate\_mode\_2 | rdgate\_ctrl\_2 |  |  | dqs\_oe\_ctrl\_2 | dq\_oe\_ctrl\_2 |
| 0 x0218 |  |  |  |  |  | dly\_2x\_2 | redge\_sel\_2 | Rddqs\_phase\_2 (RD) |
| 0 x0220 | W\_bdly0\_2 [would] | W\_bdly0\_2 [he] | W\_bdly0\_2 [thou wouldest | W\_bdly0\_2 [when] | W\_bdly0\_2 [even] | W\_bdly0\_2 [8] | W\_bdly0\_2 [17] | W\_bdly0\_2 [3-0] |
| 0 x0228 |  | W\_bdly0\_2 [59:56] | W\_bdly0\_2 [55:52] | W\_bdly0\_2 [spoilers] | W\_bdly0\_2 [47:44] | W\_bdly0\_2 [43:40] | W\_bdly0\_2 [39:36] | W\_bdly0\_2 [has] |
| 0 x0230 | W\_bdly1\_2 [so] | W\_bdly1\_2 [formerly] | W\_bdly1\_2 [he] | W\_bdly1\_2 [then] | W\_bdly1\_2 [9] | W\_bdly1\_2 [therefore] | W\_bdly1\_2 [53] | W\_bdly1\_2 [2-0] |
| 0 x0238 |  |  |  |  |  |  |  | W\_bdly1\_2 [but] |
| 0 x0240 |  |  |  |  |  |  | Rg\_bdly\_2 [17] | Rg\_bdly\_2 [3-0] |
| 0 x0248 |  |  |  |  |  |  |  |  |
| 0 x0250 | Rdqsp\_bdly\_2 [would] | Rdqsp\_bdly\_2 [27:  24] | Rdqsp\_bdly\_2 [23:  20] | Rdqsp\_bdly\_2 [when] | Rdqsp\_bdly\_2 [even] | Rdqsp\_bdly\_2 [8] | Rdqsp\_bdly\_2 [17] | Rdqsp\_bdly\_2 [3-0] |
| 0 x0258 |  |  |  |  |  |  |  | Rdqsp\_bdly\_2 [has] |
| 0 x0260 | Rdqsn\_bdly\_2 [would] | Rdqsn\_bdly\_2 [27:  24] | Rdqsn\_bdly\_2 [23:  20] | Rdqsn\_bdly\_2 [when] | Rdqsn\_bdly\_2 [even] | Rdqsn\_bdly\_2 [8] | Rdqsn\_bdly\_2 [17] | Rdqsn\_bdly\_2 [3-0] |
| 0 x0268 |  |  |  |  |  |  |  | Rdqsn\_bdly\_2 [has] |
| 0 x0270 | Rdq\_bdly\_2 [so] | Rdq\_bdly\_2 [formerly] | Rdq\_bdly\_2 [he] | Rdq\_bdly\_2 [then] | Rdq\_bdly\_2 [9] | Rdq\_bdly\_2 [therefore] | Rdq\_bdly\_2 [53] | Rdq\_bdly\_2 [2-0] |
| 0 x0278 |  |  |  |  |  |  |  | Rdq\_bdly\_2 [but] |
| 0 x0280 |  |  |  |  | dll\_1xdly\_3 | dll\_1xgen\_3 | dll\_wrdqs\_3 | dll\_wrdq\_3 |
| 0 x0288 |  |  |  |  |  | dll\_gate\_3 | dll\_rddqs1\_3 | dll\_rddqs0\_3 |
| 0 x0290 | rdodt\_ctrl\_3 | rdgate\_len\_3 | rdgate\_mode\_3 | rdgate\_ctrl\_3 |  |  | dqs\_oe\_ctrl\_3 | dq\_oe\_ctrl\_3 |
| 0 x0298 |  |  |  |  |  | dly\_2x\_3 | redge\_sel\_3 | Rddqs\_phase\_3 (RD) |
| 0 x02a0 | W\_bdly0\_3 [would] | W\_bdly0\_3 [he] | W\_bdly0\_3 [thou wouldest | W\_bdly0\_3 [when] | W\_bdly0\_3 [even] | W\_bdly0\_3 [8] | W\_bdly0\_3 [17] | W\_bdly0\_3 [3-0] |
| 0 x02a8 |  | W\_bdly0\_3 [59:56] | W\_bdly0\_3 [55:52] | W\_bdly0\_3 [spoilers] | W\_bdly0\_3 [47:44] | W\_bdly0\_3 [43:40] | W\_bdly0\_3 [39:36] | W\_bdly0\_3 [has] |
| 0 x02b0 | W\_bdly1\_3 [so] | W\_bdly1\_3 [formerly] | W\_bdly1\_3 [he] | W\_bdly1\_3 [then] | W\_bdly1\_3 [9] | W\_bdly1\_3 [therefore] | W\_bdly1\_3 [53] | W\_bdly1\_3 [2-0] |
| 0 x02b8 |  |  |  |  |  |  |  | W\_bdly1\_3 [but] |
| 0 x02c0 |  |  |  |  |  |  | Rg\_bdly\_3 [17] | Rg\_bdly\_3 [3-0] |
| 0 x02c8 |  |  |  |  |  |  |  |  |
| 0 x02d0 | Rdqsp\_bdly\_3 [would] | Rdqsp\_bdly\_3 [27:  24] | Rdqsp\_bdly\_3 [23:  20] | Rdqsp\_bdly\_3 [when] | Rdqsp\_bdly\_3 [even] | Rdqsp\_bdly\_3 [8] | Rdqsp\_bdly\_3 [17] | Rdqsp\_bdly\_3 [3-0] |
| 0 x02d8 |  |  |  |  |  |  |  | Rdqsp\_bdly\_3 [has] |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x02e0 | Rdqsn\_bdly\_3 [would] | Rdqsn\_bdly\_3 [27:  24] | Rdqsn\_bdly\_3 [23:  20] | Rdqsn\_bdly\_3 [when] | Rdqsn\_bdly\_3 [even] | Rdqsn\_bdly\_3 [8] | Rdqsn\_bdly\_3 [17] | Rdqsn\_bdly\_3 [3-0] |
| 0 x02e8 |  |  |  |  |  |  |  | Rdqsn\_bdly\_3 [has] |
| 0 x02f0 | Rdq\_bdly\_3 [so] | Rdq\_bdly\_3 [formerly] | Rdq\_bdly\_3 [he] | Rdq\_bdly\_3 [then] | Rdq\_bdly\_3 [9] | Rdq\_bdly\_3 [therefore] | Rdq\_bdly\_3 [53] | Rdq\_bdly\_3 [2-0] |
| 0 x02f8 |  |  |  |  |  |  |  | Rdq\_bdly\_3 [but] |
| 0 x0300 |  |  |  |  | dll\_1xdly\_4 | dll\_1xgen\_4 | dll\_wrdqs\_4 | dll\_wrdq\_4 |
| 0 x0308 |  |  |  |  |  | dll\_gate\_4 | dll\_rddqs1\_4 | dll\_rddqs0\_4 |
| 0 x0310 | rdodt\_ctrl\_4 | rdgate\_len\_4 | rdgate\_mode\_4 | rdgate\_ctrl\_4 |  |  | dqs\_oe\_ctrl\_4 | dq\_oe\_ctrl\_4 |
| 0 x0318 |  |  |  |  |  | dly\_2x\_4 | redge\_sel\_4 | Rddqs\_phase\_4 (RD) |
| 0 x0320 | W\_bdly0\_4 [would] | W\_bdly0\_4 [he] | W\_bdly0\_4 [thou wouldest | W\_bdly0\_4 [when] | W\_bdly0\_4 [even] | W\_bdly0\_4 [8] | W\_bdly0\_4 [17] | W\_bdly0\_4 [3-0] |
| 0 x0328 |  | W\_bdly0\_4 [59:56] | W\_bdly0\_4 [55:52] | W\_bdly0\_4 [spoilers] | W\_bdly0\_4 [47:44] | W\_bdly0\_4 [43:40] | W\_bdly0\_4 [39:36] | W\_bdly0\_4 [has] |
| 0 x0330 | W\_bdly1\_4 [so] | W\_bdly1\_4 [formerly] | W\_bdly1\_4 [he] | W\_bdly1\_4 [then] | W\_bdly1\_4 [9] | W\_bdly1\_4 [therefore] | W\_bdly1\_4 [53] | W\_bdly1\_4 [2-0] |
| 0 x0338 |  |  |  |  |  |  |  | W\_bdly1\_4 [but] |
| 0 x0340 |  |  |  |  |  |  | Rg\_bdly\_4 [17] | Rg\_bdly\_4 [3-0] |
| 0 x0348 |  |  |  |  |  |  |  |  |
| 0 x0350 | Rdqsp\_bdly\_4 [would] | Rdqsp\_bdly\_4 [27:  24] | Rdqsp\_bdly\_4 [23:  20] | Rdqsp\_bdly\_4 [when] | Rdqsp\_bdly\_4 [even] | Rdqsp\_bdly\_4 [8] | Rdqsp\_bdly\_4 [17] | Rdqsp\_bdly\_4 [3-0] |
| 0 x0358 |  |  |  |  |  |  |  | Rdqsp\_bdly\_4 [has] |
| 0 x0360 | Rdqsn\_bdly\_4 [would] | Rdqsn\_bdly\_4 [27:  24] | Rdqsn\_bdly\_4 [23:  20] | Rdqsn\_bdly\_4 [when] | Rdqsn\_bdly\_4 [even] | Rdqsn\_bdly\_4 [8] | Rdqsn\_bdly\_4 [17] | Rdqsn\_bdly\_4 [3-0] |
| 0 x0368 |  |  |  |  |  |  |  | Rdqsn\_bdly\_4 [has] |
| 0 x0370 | Rdq\_bdly\_4 [so] | Rdq\_bdly\_4 [formerly] | Rdq\_bdly\_4 [he] | Rdq\_bdly\_4 [then] | Rdq\_bdly\_4 [9] | Rdq\_bdly\_4 [therefore] | Rdq\_bdly\_4 [53] | Rdq\_bdly\_4 [2-0] |
| 0 x0378 |  |  |  |  |  |  |  | Rdq\_bdly\_4 [but] |
| 0 x0380 |  |  |  |  | dll\_1xdly\_5 | dll\_1xgen\_5 | dll\_wrdqs\_5 | dll\_wrdq\_5 |
| 0 x0388 |  |  |  |  |  | dll\_gate\_5 | dll\_rddqs1\_5 | dll\_rddqs0\_5 |
| 0 x0390 | rdodt\_ctrl\_5 | rdgate\_len\_5 | rdgate\_mode\_5 | rdgate\_ctrl\_5 |  |  | dqs\_oe\_ctrl\_5 | dq\_oe\_ctrl\_5 |
| 0 x0398 |  |  |  |  |  | dly\_2x\_5 | redge\_sel\_5 | Rddqs\_phase\_5 (RD) |
| 0 x03a0 | W\_bdly0\_5 [would] | W\_bdly0\_5 [he] | W\_bdly0\_5 [thou wouldest | W\_bdly0\_5 [when] | W\_bdly0\_5 [even] | W\_bdly0\_5 [8] | W\_bdly0\_5 [17] | W\_bdly0\_5 [3-0] |
| 0 x03a8 |  | W\_bdly0\_5 [59:56] | W\_bdly0\_5 [55:52] | W\_bdly0\_5 [spoilers] | W\_bdly0\_5 [47:44] | W\_bdly0\_5 [43:40] | W\_bdly0\_5 [39:36] | W\_bdly0\_5 [has] |
| 0 x03b0 | W\_bdly1\_5 [so] | W\_bdly1\_5 [formerly] | W\_bdly1\_5 [he] | W\_bdly1\_5 [then] | W\_bdly1\_5 [9] | W\_bdly1\_5 [therefore] | W\_bdly1\_5 [53] | W\_bdly1\_5 [2-0] |
| 0 x03b8 |  |  |  |  |  |  |  | W\_bdly1\_5 [but] |
| 0 x03c0 |  |  |  |  |  |  | Rg\_bdly\_5 [17] | Rg\_bdly\_5 [3-0] |
| 0 x03c8 |  |  |  |  |  |  |  |  |
| 0 x03d0 | Rdqsp\_bdly\_5 [would] | Rdqsp\_bdly\_5 [27:  24] | Rdqsp\_bdly\_5 [23:  20] | Rdqsp\_bdly\_5 [when] | Rdqsp\_bdly\_5 [even] | Rdqsp\_bdly\_5 [8] | Rdqsp\_bdly\_5 [17] | Rdqsp\_bdly\_5 [3-0] |
| 0 x03d8 |  |  |  |  |  |  |  | Rdqsp\_bdly\_5 [has] |
| 0 x03e0 | Rdqsn\_bdly\_5 [would] | Rdqsn\_bdly\_5 [27:  24] | Rdqsn\_bdly\_5 [23:  20] | Rdqsn\_bdly\_5 [when] | Rdqsn\_bdly\_5 [even] | Rdqsn\_bdly\_5 [8] | Rdqsn\_bdly\_5 [17] | Rdqsn\_bdly\_5 [3-0] |
| 0 x03e8 |  |  |  |  |  |  |  | Rdqsn\_bdly\_5 [has] |
| 0 x03f0 | Rdq\_bdly\_5 [so] | Rdq\_bdly\_5 [formerly] | Rdq\_bdly\_5 [he] | Rdq\_bdly\_5 [then] | Rdq\_bdly\_5 [9] | Rdq\_bdly\_5 [therefore] | Rdq\_bdly\_5 [53] | Rdq\_bdly\_5 [2-0] |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x03f8 |  |  |  |  |  |  |  | Rdq\_bdly\_5 [but] |
| 0 x0400 |  |  |  |  | dll\_1xdly\_6 | dll\_1xgen\_6 | dll\_wrdqs\_6 | dll\_wrdq\_6 |
| 0 x0408 |  |  |  |  |  | dll\_gate\_6 | dll\_rddqs1\_6 | dll\_rddqs0\_6 |
| 0 x0410 | rdodt\_ctrl\_6 | rdgate\_len\_6 | rdgate\_mode\_6 | rdgate\_ctrl\_6 |  |  | dqs\_oe\_ctrl\_6 | dq\_oe\_ctrl\_6 |
| 0 x0418 |  |  |  |  |  | dly\_2x\_6 | redge\_sel\_6 | Rddqs\_phase\_6 (RD) |
| 0 x0420 | W\_bdly0\_6 [would] | W\_bdly0\_6 [he] | W\_bdly0\_6 [thou wouldest | W\_bdly0\_6 [when] | W\_bdly0\_6 [even] | W\_bdly0\_6 [8] | W\_bdly0\_6 [17] | W\_bdly0\_6 [3-0] |
| 0 x0428 |  | W\_bdly0\_6 [59:56] | W\_bdly0\_6 [55:52] | W\_bdly0\_6 [spoilers] | W\_bdly0\_6 [47:44] | W\_bdly0\_6 [43:40] | W\_bdly0\_6 [39:36] | W\_bdly0\_6 [has] |
| 0 x0430 | W\_bdly1\_6 [so] | W\_bdly1\_6 [formerly] | W\_bdly1\_6 [he] | W\_bdly1\_6 [then] | W\_bdly1\_6 [9] | W\_bdly1\_6 [therefore] | W\_bdly1\_6 [53] | W\_bdly1\_6 [2-0] |
| 0 x0438 |  |  |  |  |  |  |  | W\_bdly1\_6 [but] |
| 0 x0440 |  |  |  |  |  |  | Rg\_bdly\_6 [17] | Rg\_bdly\_6 [3-0] |
| 0 x0448 |  |  |  |  |  |  |  |  |
| 0 x0450 | Rdqsp\_bdly\_6 [would] | Rdqsp\_bdly\_6 [27:  24] | Rdqsp\_bdly\_6 [23:  20] | Rdqsp\_bdly\_6 [when] | Rdqsp\_bdly\_6 [even] | Rdqsp\_bdly\_6 [8] | Rdqsp\_bdly\_6 [17] | Rdqsp\_bdly\_6 [3-0] |
| 0 x0458 |  |  |  |  |  |  |  | Rdqsp\_bdly\_6 [has] |
| 0 x0460 | Rdqsn\_bdly\_6 [would] | Rdqsn\_bdly\_6 [27:  24] | Rdqsn\_bdly\_6 [23:  20] | Rdqsn\_bdly\_6 [when] | Rdqsn\_bdly\_6 [even] | Rdqsn\_bdly\_6 [8] | Rdqsn\_bdly\_6 [17] | Rdqsn\_bdly\_6 [3-0] |
| 0 x0468 |  |  |  |  |  |  |  | Rdqsn\_bdly\_6 [has] |
| 0 x0470 | Rdq\_bdly\_6 [so] | Rdq\_bdly\_6 [formerly] | Rdq\_bdly\_6 [he] | Rdq\_bdly\_6 [then] | Rdq\_bdly\_6 [9] | Rdq\_bdly\_6 [therefore] | Rdq\_bdly\_6 [53] | Rdq\_bdly\_6 [2-0] |
| 0 x0478 |  |  |  |  |  |  |  | Rdq\_bdly\_6 [but] |
| 0 x0480 |  |  |  |  | dll\_1xdly\_7 | dll\_1xgen\_7 | dll\_wrdqs\_7 | dll\_wrdq\_7 |
| 0 x0488 |  |  |  |  |  | dll\_gate\_7 | dll\_rddqs1\_7 | dll\_rddqs0\_7 |
| 0 x0490 | rdodt\_ctrl\_7 | rdgate\_len\_7 | rdgate\_mode\_7 | rdgate\_ctrl\_7 |  |  | dqs\_oe\_ctrl\_7 | dq\_oe\_ctrl\_7 |
| 0 x0498 |  |  |  |  |  | dly\_2x\_7 | redge\_sel\_7 | Rddqs\_phase\_7 (RD) |
| 0 x04a0 | W\_bdly0\_7 [would] | W\_bdly0\_7 [he] | W\_bdly0\_7 [thou wouldest | W\_bdly0\_7 [when] | W\_bdly0\_7 [even] | W\_bdly0\_7 [8] | W\_bdly0\_7 [17] | W\_bdly0\_7 [3-0] |
| 0 x04a8 |  | W\_bdly0\_7 [59:56] | W\_bdly0\_7 [55:52] | W\_bdly0\_7 [spoilers] | W\_bdly0\_7 [47:44] | W\_bdly0\_7 [43:40] | W\_bdly0\_7 [39:36] | W\_bdly0\_7 [has] |
| 0 x04b0 | W\_bdly1\_7 [so] | W\_bdly1\_7 [formerly] | W\_bdly1\_7 [he] | W\_bdly1\_7 [then] | W\_bdly1\_7 [9] | W\_bdly1\_7 [therefore] | W\_bdly1\_7 [53] | W\_bdly1\_7 [2-0] |
| 0 x04b8 |  |  |  |  |  |  |  | W\_bdly1\_7 [but] |
| 0 x04c0 |  |  |  |  |  |  | Rg\_bdly\_7 [17] | Rg\_bdly\_7 [3-0] |
| 0 x04c8 |  |  |  |  |  |  |  |  |
| 0 x04d0 | Rdqsp\_bdly\_7 [would] | Rdqsp\_bdly\_7 [27:  24] | Rdqsp\_bdly\_7 [23:  20] | Rdqsp\_bdly\_7 [when] | Rdqsp\_bdly\_7 [even] | Rdqsp\_bdly\_7 [8] | Rdqsp\_bdly\_7 [17] | Rdqsp\_bdly\_7 [3-0] |
| 0 x04d8 |  |  |  |  |  |  |  | Rdqsp\_bdly\_7 [has] |
| 0 x04e0 | Rdqsn\_bdly\_7 [would] | Rdqsn\_bdly\_7 [27:  24] | Rdqsn\_bdly\_7 [23:  20] | Rdqsn\_bdly\_7 [when] | Rdqsn\_bdly\_7 [even] | Rdqsn\_bdly\_7 [8] | Rdqsn\_bdly\_7 [17] | Rdqsn\_bdly\_7 [3-0] |
| 0 x04e8 |  |  |  |  |  |  |  | Rdqsn\_bdly\_7 [has] |
| 0 x04f0 | Rdq\_bdly\_7 [so] | Rdq\_bdly\_7 [formerly] | Rdq\_bdly\_7 [he] | Rdq\_bdly\_7 [then] | Rdq\_bdly\_7 [9] | Rdq\_bdly\_7 [therefore] | Rdq\_bdly\_7 [53] | Rdq\_bdly\_7 [2-0] |
| 0 x04f8 |  |  |  |  |  |  |  | Rdq\_bdly\_7 [but] |
| 0 x0500 |  |  |  |  | dll\_1xdly\_8 | dll\_1xgen\_8 | dll\_wrdqs\_8 | dll\_wrdq\_8 |
| 0 x0508 |  |  |  |  |  | dll\_gate\_8 | dll\_rddqs1\_8 | dll\_rddqs0\_8 |
| 0 x0510 | rdodt\_ctrl\_8 | rdgate\_len\_8 | rdgate\_mode\_8 | rdgate\_ctrl\_8 |  |  | dqs\_oe\_ctrl\_8 | dq\_oe\_ctrl\_8 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x0518 |  |  |  |  |  | dly\_2x\_8 | redge\_sel\_8 | Rddqs\_phase\_8 (RD) |
| 0 x0520 | W\_bdly0\_8 [would] | W\_bdly0\_8 [he] | W\_bdly0\_8 [thou wouldest | W\_bdly0\_8 [when] | W\_bdly0\_8 [even] | W\_bdly0\_8 [8] | W\_bdly0\_8 [17] | W\_bdly0\_8 [3-0] |
| 0 x0528 |  | W\_bdly0\_8 [59:56] | W\_bdly0\_8 [55:52] | W\_bdly0\_8 [spoilers] | W\_bdly0\_8 [47:44] | W\_bdly0\_8 [43:40] | W\_bdly0\_8 [39:36] | W\_bdly0\_8 [has] |
| 0 x0530 | W\_bdly1\_8 [so] | W\_bdly1\_8 [formerly] | W\_bdly1\_8 [he] | W\_bdly1\_8 [then] | W\_bdly1\_8 [9] | W\_bdly1\_8 [therefore] | W\_bdly1\_8 [53] | W\_bdly1\_8 [2-0] |
| 0 x0538 |  |  |  |  |  |  |  | W\_bdly1\_8 [but] |
| 0 x0540 |  |  |  |  |  |  | Rg\_bdly\_8 [17] | Rg\_bdly\_8 [3-0] |
| 0 x0548 |  |  |  |  |  |  |  |  |
| 0 x0550 | Rdqsp\_bdly\_8 [would] | Rdqsp\_bdly\_8 [27:  24] | Rdqsp\_bdly\_8 [23:  20] | Rdqsp\_bdly\_8 [when] | Rdqsp\_bdly\_8 [even] | Rdqsp\_bdly\_8 [8] | Rdqsp\_bdly\_8 [17] | Rdqsp\_bdly\_8 [3-0] |
| 0 x0558 |  |  |  |  |  |  |  | Rdqsp\_bdly\_8 [has] |
| 0 x0560 | Rdqsn\_bdly\_8 [would] | Rdqsn\_bdly\_8 [27:  24] | Rdqsn\_bdly\_8 [23:  20] | Rdqsn\_bdly\_8 [when] | Rdqsn\_bdly\_8 [even] | Rdqsn\_bdly\_8 [8] | Rdqsn\_bdly\_8 [17] | Rdqsn\_bdly\_8 [3-0] |
| 0 x0568 |  |  |  |  |  |  |  | Rdqsn\_bdly\_8 [has] |
| 0 x0570 | Rdq\_bdly\_8 [so] | Rdq\_bdly\_8 [formerly] | Rdq\_bdly\_8 [he] | Rdq\_bdly\_8 [then] | Rdq\_bdly\_8 [9] | Rdq\_bdly\_8 [therefore] | Rdq\_bdly\_8 [53] | Rdq\_bdly\_8 [2-0] |
| 0 x0578 |  |  |  |  |  |  |  | Rdq\_bdly\_8 [but] |
| ... |  |  |  |  |  |  |  |  |
| 0 x0700 |  |  |  |  | leveling\_cs | tLVL\_DELAY | Leveling\_req (WR) | leveling\_mode |
| 0 x0708 |  |  |  |  |  |  | Leveling\_done (RD  ) | Leveling\_ready (RD) |
| 0 x0710 | leveling\_resp\_7 | leveling\_resp\_6 | leveling\_resp\_5 | leveling\_resp\_4 | leveling\_resp\_3 | leveling\_resp\_2 | leveling\_resp\_1 | leveling\_resp\_0 |
| 0 x0718 |  |  |  |  |  |  |  | leveling\_resp\_8 |
| 0 x0720 |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |
| 0 x0800 | dfe\_ctrl\_ds | pad\_ctrl\_ds | | |  | pad\_ctrl\_ck | | |
| 0 x0808 |  | pad\_reset\_po | pad\_oplen\_ca | pad\_opdly\_ca |  | pad\_ctrl\_ca | | |
| 0 x0810 | vref\_ctrl\_ds\_3 | | vref\_ctrl\_ds\_2 | | vref\_ctrl\_ds\_1 | | vref\_ctrl\_ds\_0 | |
| 0 x0818 | vref\_ctrl\_ds\_7 | | vref\_ctrl\_ds\_6 | | vref\_ctrl\_ds\_5 | | vref\_ctrl\_ds\_4 | |
| 0 x0820 |  |  |  |  |  |  | vref\_ctrl\_ds\_8 | |
| 0 x0828 |  |  |  |  |  |  |  |  |
| 0 x0830 |  |  | Pad\_comp\_o (RD) | |  |  | pad\_comp\_i | |
| 0 x0838 |  |  |  |  |  |  |  |  |
| **CTL** | | | | | | | | |
| 0 x1000 |  | tRP | tWLDQSEN | tMOD | tXPR | | tCKE | tRESET |
| 0 x1008 |  |  |  |  |  |  |  | tODTL |
| 0 x1010 | tREFretention | | | | tRFC | | tREF | |
| 0 x1018 | tCKESR | tXSRD | tXS | | tRFC\_dlr | |  | tREF\_IDLE |
| 0 x1020 |  |  |  |  | tRDPDEN | tCPDED | tXPDLL | tXP |
| 0 x1028 |  |  |  |  | tZQperiod | tZQCL | tZQCS | tZQ\_CMD |
| ... |  |  |  |  |  |  |  |  |
| 0 x1040 | tRCD | tRRD\_S\_slr | tRRD\_L\_slr | tRRD\_dlr |  |  |  | tRAS\_min |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x1048 |  |  |  | tRTP | tWR\_CRC\_DM | tWR | tFAW\_slr | tFAW |
| 0 x1050 | tWTR\_S\_CRC\_DM | tWTR\_L\_CRC\_D  M | tWTR\_S | tWTR |  | tCCD\_dlr | tCCD\_S\_slr | tCCD\_L\_slr |
| 0 x1058 |  |  |  |  |  |  |  |  |
| 0 x1060 |  |  | tPHY\_WRLAT | these |  | tRDDATA | tPHY\_RDLAT | tRL |
| 0 x1068 |  |  |  | tCAL |  |  |  | The tPL |
| 0 x1070 |  |  | tW2P\_sameba | tW2W\_sameba | tW2R\_sameba | tR2P\_sameba | tR2W\_sameba | tR2R\_sameba |
| 0 x1078 |  |  | tW2P\_samebg | tW2W\_samebg | tW2R\_samebg | tR2P\_samebg | tR2W\_samebg | tR2R\_samebg |
| 0 x1080 |  |  | tW2P\_samec | tW2W\_samec | tW2R\_samec | tR2P\_samec | tR2W\_samec | tR2R\_samec |
| 0 x1088 |  |  |  |  |  |  |  |  |
| 0 x1090 |  |  | tW2P\_samecs | tW2W\_samecs | tW2R\_samecs | tR2P\_samecs | tR2W\_samecs | tR2R\_samecs |
| 0 x1098 |  |  |  | tW2W\_diffcs | tW2R\_diffcs |  | tR2W\_diffcs | tR2R\_diffcs |
| ... |  |  |  |  |  |  |  |  |
| 0 x1100 |  |  | cs\_ref | cs\_resync | cs\_zqcl | cs\_zq | cs\_mrs | cs\_enable |
| 0 x1108 | cke\_map | | | | cs\_map | | | |
| 0 x1110 |  |  |  | cs2cid |  |  |  | cid\_map |
| 0 x1118 |  |  |  |  |  |  |  |  |
| 0 x1120 | Mrs\_done (RD) | Mrs\_req (WR) | Pre\_all\_done (RD) | Pre\_all\_req (WR) | cmd\_cmd | Status\_cmd (RD) | Cmd\_req (WR) | command\_mode |
| 0 x1128 | cmd\_cke | cmd\_a | | | cmd\_ba | cmd\_bg | cmd\_c | cmd\_cs |
| 0 x1130 |  |  |  |  |  |  |  | cmd\_pda |
| 0 x1138 |  |  |  |  |  | cmd\_dq0 | | |
| 0 x1140 | mr\_3\_cs\_0 | | mr\_2\_cs\_0 | | mr\_1\_cs\_0 | | mr\_0\_cs\_0 | |
| 0 x1148 | mr\_3\_cs\_1 | | mr\_2\_cs\_1 | | mr\_1\_cs\_1 | | mr\_0\_cs\_1 | |
| 0 x1150 | mr\_3\_cs\_2 | | mr\_2\_cs\_2 | | mr\_1\_cs\_2 | | mr\_0\_cs\_2 | |
| 0 x1158 | mr\_3\_cs\_3 | | mr\_2\_cs\_3 | | mr\_1\_cs\_3 | | mr\_0\_cs\_3 | |
| 0 x1160 | mr\_3\_cs\_4 | | mr\_2\_cs\_4 | | mr\_1\_cs\_4 | | mr\_0\_cs\_4 | |
| 0 x1168 | mr\_3\_cs\_5 | | mr\_2\_cs\_5 | | mr\_1\_cs\_5 | | mr\_0\_cs\_5 | |
| 0 x1170 | mr\_3\_cs\_6 | | mr\_2\_cs\_6 | | mr\_1\_cs\_6 | | mr\_0\_cs\_6 | |
| 0 x1178 | mr\_3\_cs\_7 | | mr\_2\_cs\_7 | | mr\_1\_cs\_7 | | mr\_0\_cs\_7 | |
| 0 x1180 | mr\_3\_cs\_0\_ddr4 | | mr\_2\_cs\_0\_ddr4 | | mr\_1\_cs\_0\_ddr4 | | mr\_0\_cs\_0\_ddr4 | |
| 0 x1188 |  |  | mr\_6\_cs\_0\_ddr4 | | mr\_5\_cs\_0\_ddr4 | | mr\_4\_cs\_0\_ddr4 | |
| 0 x1190 | mr\_3\_cs\_1\_ddr4 | | mr\_2\_cs\_1\_ddr4 | | mr\_1\_cs\_1\_ddr4 | | mr\_0\_cs\_1\_ddr4 | |
| 0 x1198 |  |  | mr\_6\_cs\_1\_ddr4 | | mr\_5\_cs\_1\_ddr4 | | mr\_4\_cs\_1\_ddr4 | |
| 0 x11a0 | mr\_3\_cs\_2\_ddr4 | | mr\_2\_cs\_2\_ddr4 | | mr\_1\_cs\_2\_ddr4 | | mr\_0\_cs\_2\_ddr4 | |
| 0 x11a8 |  |  | mr\_6\_cs\_2\_ddr4 | | mr\_5\_cs\_2\_ddr4 | | mr\_4\_cs\_2\_ddr4 | |
| 0 x11b0 | mr\_3\_cs\_3\_ddr4 | | mr\_2\_cs\_3\_ddr4 | | mr\_1\_cs\_3\_ddr4 | | mr\_0\_cs\_3\_ddr4 | |
| 0 x11b8 |  |  | mr\_6\_cs\_3\_ddr4 | | mr\_5\_cs\_3\_ddr4 | | mr\_4\_cs\_3\_ddr4 | |
| 0 x11c0 | mr\_3\_cs\_4\_ddr4 | | mr\_2\_cs\_4\_ddr4 | | mr\_1\_cs\_4\_ddr4 | | mr\_0\_cs\_4\_ddr4 | |
| 0 x11c8 |  |  | mr\_6\_cs\_4\_ddr4 | | mr\_5\_cs\_4\_ddr4 | | mr\_4\_cs\_4\_ddr4 | |
| 0 x11d0 | mr\_3\_cs\_5\_ddr4 | | mr\_2\_cs\_5\_ddr4 | | mr\_1\_cs\_5\_ddr4 | | mr\_0\_cs\_5\_ddr4 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x11d8 |  |  | mr\_6\_cs\_5\_ddr4 | | mr\_5\_cs\_5\_ddr4 | | mr\_4\_cs\_5\_ddr4 | |
| 0 x11e0 | mr\_3\_cs\_6\_ddr4 | | mr\_2\_cs\_6\_ddr4 | | mr\_1\_cs\_6\_ddr4 | | mr\_0\_cs\_6\_ddr4 | |
| 0 x11e8 |  |  | mr\_6\_cs\_6\_ddr4 | | mr\_5\_cs\_6\_ddr4 | | mr\_4\_cs\_6\_ddr4 | |
| 0 x11f0 | mr\_3\_cs\_7\_ddr4 | | mr\_2\_cs\_7\_ddr4 | | mr\_1\_cs\_7\_ddr4 | | mr\_0\_cs\_7\_ddr4 | |
| 0 x11f8 |  |  | mr\_6\_cs\_7\_ddr4 | | mr\_5\_cs\_7\_ddr4 | | mr\_4\_cs\_7\_ddr4 | |
| 0 x |  |  | nc16\_map | nc | channel\_width | ba\_xor\_row\_offset | addr\_new | cs\_place |
| 0 x1208 |  |  |  |  |  | bg\_xor\_row\_offset |  | addr\_mirror |
| 0 x1210 | addr\_base\_1 | | | | addr\_base\_0 | | | |
| 0 x1218 |  |  |  |  |  |  |  |  |
| 0 x1220 | addr\_mask\_1 | | | | addr\_mask\_0 | | | |
| 0 x1228 |  |  |  |  |  |  |  |  |
| 0 x1230 |  |  | cs\_diff | c\_diff | bg\_diff | ba\_diff | row\_diff | col\_diff |
| 0 x1238 |  |  |  | CF\_confbus\_timeout |  |  |  |  |
| 0 x1240 | WRQthreshold | tRDQidle | wr\_pkc\_num | rwq\_rb | retry | no\_dead\_inorder | placement\_en | Stb\_en/pbufs |
| 0 x1248 |  |  |  |  |  |  |  | tRWGNTidle |
| 0 x1250 |  |  |  |  |  |  | rfifo\_age | |
| 0 x1258 | prior\_age3 | | prior\_age2 | | prior\_age1 | | prior\_age0 | |
| 0 x1260 | Retry\_cnt (RD) | | | |  | Rbuffer\_max (RD) | rdfifo\_depth | stat\_en |
| 0 x1268 |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |
| 0 x1280 | aw\_512\_align |  | rd\_before\_wr | ecc\_enable |  | Int\_vector (RD) | Int\_trigger (RD) | int\_enable |
| 0 x1288 |  |  |  |  |  |  |  |  |
| 0 x1290 |  |  |  |  |  | Int\_cnt\_fatal (RD) | Int\_cnt\_err (RD) | int\_cnt |
| 0 x1298 | Ecc\_cnt\_cs\_7 (RD) | Ecc\_cnt\_cs\_6 (R  D) | Ecc\_cnt\_cs\_5 (RD  ) | Ecc\_cnt\_cs\_4 (RD) | Ecc\_cnt\_cs\_3 (RD) | Ecc\_cnt\_cs\_2 (RD) | Ecc\_cnt\_cs\_1 (RD  ) | Ecc\_cnt\_cs\_0 (RD) |
| 0 x12a0 | Ecc\_data\_dir (RD) | Ecc\_code\_dir (RD  ) | Ecc\_code\_256 (RD) | |  |  |  | Ecc\_code\_64 (RD) |
| 0 x12a8 | Ecc\_addr (RD) | | | | | | | |
| 0 x12b0 | Ecc\_data [63:0] (RD) | | | | | | | |
| 0 x12b8 | Ecc\_data [127-64] (RD) | | | | | | | |
| 0 x12c0 | Ecc\_data [191-128] (RD) | | | | | | | |
| 0 x12c8 | Ecc\_data [255-192] (RD) | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x1300 |  |  |  |  |  |  | ref\_num | ref\_sch\_en |
| 0 x1308 |  |  |  |  |  |  | Status\_sref (RD) | srefresh\_req |
| ... |  |  |  |  |  |  |  |  |
| 0 x1340 | hardware\_pd\_7 | hardware\_pd\_6 | hardware\_pd\_5 | hardware\_pd\_4 | hardware\_pd\_3 | hardware\_pd\_2 | hardware\_pd\_1 | hardware\_pd\_0 |
| 0 x1348 | Power\_sta\_7 (RD) | Power\_sta\_6 (RD  ) | Power\_sta\_5 (RD) | Power\_sta\_4 (RD) | Power\_sta\_3 (RD) | Power\_sta\_2 (RD) | Power\_sta\_1 (RD) | Power\_sta\_0 (RD) |
| 0 x1350 | selfref\_age | | slowpd\_age | | fastpd\_age | | active\_age | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x1358 |  |  |  | power\_up |  |  |  | Age\_step |
| 0 x1360 | tCONF\_IDLE | | | | tLPMC\_IDLE | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x1380 |  |  |  |  |  |  |  | zq\_overlap |
| 0 x1388 |  |  |  |  |  |  |  | zq\_stat\_en |
| 0 x1390 | Zq\_cnt\_1 (RD) | | | | Zq\_cnt\_0 (RD) | | | |
| 0 x1398 | Zq\_cnt\_3 (RD) | | | | Zq\_cnt\_2 (RD) | | | |
| 0 x13a0 | Zq\_cnt\_5 (RD) | | | | Zq\_cnt\_4 (RD) | | | |
| 0 x13a8 | Zq\_cnt\_6 (RD) | | | | Zq\_cnt\_6 (RD) | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x13c0 |  |  |  |  | odt\_wr\_cs\_map | | | |
| 0 x13c8 |  |  |  |  |  |  | odt\_wr\_length | odt\_wr\_delay |
| 0 x13d0 |  |  |  |  | odt\_rd\_cs\_map | | | |
| 0 x13d8 |  |  |  |  |  |  | odt\_rd\_length | odt\_rd\_delay |
| ... |  |  |  |  |  |  |  |  |
| 0 x1400 |  |  |  | tRESYNC\_length | tRESYNC\_delay | tRESYNC\_shift | tRESYNC\_max | tRESYNC\_min |
| ... |  |  |  |  |  |  |  |  |
| 0 x1440 |  |  |  |  | pre\_predict | | tm\_cmdq\_num | burst\_length |
| 0 x1448 |  |  |  |  |  |  |  | ca\_timing |
| 0 x1450 |  |  |  |  |  | Wr/rd\_dbi\_en | ca\_par\_en | crc\_en |
| 0 x1458 |  |  |  |  |  |  | tCA\_PAR | tWR\_CRC |
| 0 x1460 | bit\_map\_7 | bit\_map\_6 | bit\_map\_5 | bit\_map\_6 | bit\_map\_3 | bit\_map\_2 | bit\_map\_1 | bit\_map\_0 |
| 0 x1468 | bit\_map\_15 | bit\_map\_14 | bit\_map\_13 | bit\_map\_12 | bit\_map\_11 | bit\_map\_10 | bit\_map\_9 | bit\_map\_8 |
| 0 x1470 |  |  |  |  |  |  | bit\_map\_17 | bit\_map\_16 |
| 0 x1478 |  |  |  |  |  |  |  | bitmap\_mirror |
| 0 x1480 |  |  |  | Alertn\_misc (RD) |  |  | alertn\_cnt | alertn\_clr |
| 0 x1488 | Alertn\_addr (RD) | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x1500 | win0\_base | | | | | | | |
| 0 x1508 | win1\_base | | | | | | | |
| 0 x1510 | win2\_base | | | | | | | |
| 0 x1518 | win3\_base | | | | | | | |
| 0 x1520 | win4\_base | | | | | | | |
| 0 x1528 | win5\_base | | | | | | | |
| 0 x1530 | win6\_base | | | | | | | |
| 0 x1538 | win7\_base | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x1580 | win0\_mask | | | | | | | |
| 0 x1588 | win1\_mask | | | | | | | |
| 0 x1590 | win2\_mask | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x1598 | win3\_mask | | | | | | | |
| 0 x15a0 | win4\_mask | | | | | | | |
| 0 x15a8 | win5\_mask | | | | | | | |
| 0 x15b0 | win6\_mask | | | | | | | |
| 0 x15b8 | win7\_mask | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x1600 | win0\_mmap | | | | | | | |
| 0 x1608 | win1\_mmap | | | | | | | |
| 0 x1610 | win2\_mmap | | | | | | | |
| 0 x1618 | win3\_mmap | | | | | | | |
| 0 x1620 | win4\_mmap | | | | | | | |
| 0 x1628 | win5\_mmap | | | | | | | |
| 0 x1630 | win6\_mmap | | | | | | | |
| 0 x1638 | win7\_mmap | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x1700 |  |  |  |  |  |  | acc\_hp | acc\_en |
| 0 x1708 | acc\_fake\_b | | | | acc\_fake\_a | | | |
| 0 x1710 |  |  |  |  |  |  |  |  |
| 0 x1718 |  |  |  |  |  |  |  |  |
| 0 x1720 | addr\_base\_acc\_1 | | | | addr\_base\_acc\_0 | | | |
| 0 x1728 |  |  |  |  |  |  |  |  |
| 0 x1730 | addr\_mask\_acc\_1 | | | | addr\_mask\_acc\_0 | | | |
| 0 x1738 |  |  |  |  |  |  |  |  |
| **MON** | | | | | | | | |
| 0 x2000 |  |  |  |  |  |  |  | cmd\_monitor |
| 0 x2008 |  |  |  |  |  |  |  |  |
| 0 x2010 | Cmd\_fbck [63:0] (RD) | | | | | | | |
| 0 x2018 | Cmd\_fbck [127-64] (RD) | | | | | | | |
| 0 x2020 |  | | | | Rw\_switch\_cnt (RD) | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x2100 |  |  |  |  |  |  |  | scheduler\_mon |
| 0 x2108 |  |  |  |  |  |  |  |  |
| 0 x2110 | Sch\_cmd\_num (RD) | | | | | | | |
| 0 x2118 | Ba\_conflict\_all (RD) | | | | | | | |
| 0 x2120 | Ba\_conflict\_last1 (RD) | | | | | | | |
| 0 x2128 | Ba\_conflict\_last2 (RD) | | | | | | | |
| 0 x2130 | Ba\_conflict\_last3 (RD) | | | | | | | |
| 0 x2138 | Ba\_conflict\_last4 (RD) | | | | | | | |
| 0 x2140 | Ba\_conflict\_last5 (RD) | | | | | | | |
| 0 x2148 | Ba\_conflict\_last6 (RD) | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x2150 | Ba\_conflict\_last7 (RD) | | | | | | | |
| 0 x2158 | Ba\_conflict\_last8 (RD) | | | | | | | |
| 0 x2160 | Rd\_conflict (RD) | | | | | | | |
| 0 x2168 | Wr\_conflict (RD) | | | | | | | |
| 0 x2170 | Rtw\_conflict (RD) | | | | | | | |
| 0 x2178 | Wtr\_conflict (RD) | | | | | | | |
| 0 x2180 | Rd\_conflict\_last1 (RD) | | | | | | | |
| 0 x2188 | Wr\_conflict\_last1 (RD) | | | | | | | |
| 0 x2190 | Rtw\_conflict\_last1 (RD) | | | | | | | |
| 0 x2198 | Wtr\_conflict\_last1 (RD) | | | | | | | |
| 0 x21a0 | Wr\_rd\_turnaround (RD) | | | | | | | |
| 0 x21a8 | Cs\_turnaround (RD) | | | | | | | |
| 0 x21b0 | Bg\_conflict (RD) | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x2300 |  |  |  |  |  | sm\_leveling |  | sm\_init |
| 0 x2308 |  |  |  |  |  |  |  |  |
| 0 x2310 |  | sm\_rank\_03 |  | sm\_rank\_02 |  | sm\_rank\_01 |  | sm\_rank\_00 |
| 0 x2318 |  | sm\_rank\_07 |  | sm\_rank\_06 |  | sm\_rank\_05 |  | sm\_rank\_04 |
| 0 x2320 |  | sm\_rank\_11 |  | sm\_rank\_10 |  | sm\_rank\_09 |  | sm\_rank\_08 |
| 0 x2328 |  | sm\_rank\_15 |  | sm\_rank\_14 |  | sm\_rank\_13 |  | sm\_rank\_12 |
| 0 x2330 |  | sm\_rank\_19 |  | sm\_rank\_18 |  | sm\_rank\_17 |  | sm\_rank\_16 |
| 0 x2338 |  | sm\_rank\_23 |  | sm\_rank\_22 |  | sm\_rank\_21 |  | sm\_rank\_20 |
| 0 x2340 |  | sm\_rank\_27 |  | sm\_rank\_26 |  | sm\_rank\_25 |  | sm\_rank\_24 |
| 0 x2348 |  | sm\_rank\_31 |  | sm\_rank\_30 |  | sm\_rank\_29 |  | sm\_rank\_28 |
| ... |  |  |  |  |  |  |  |  |
| **TST** | | | | | | | | |
| 0 x3000 |  |  |  |  |  | lpbk\_mode | lpbk\_start | lpbk\_en |
| 0 x3008 | Lpbk\_correct (RD) | | | | Lpbk\_counter (RD) | |  | Lpbk\_error (RD) |
| 0 x3010 | Lpbk\_data\_en [63:0] | | | | | | | |
| 0 x3018 |  |  |  |  |  |  |  | Lpbk\_data\_en [71, 64] |
| 0 x3020 |  |  |  |  |  |  | lpbk\_data\_mask\_en | |
| 0 x3028 |  |  |  |  |  |  |  |  |
| 0 x3030 | Lpbk\_dat\_w0 [63:0] | | | | | | | |
| 0 x3038 | Lpbk\_dat\_w0 [127, 64] | | | | | | | |
| 0 x3040 | Lpbk\_dat\_w1 [63:0] | | | | | | | |
| 0 x3048 | Lpbk\_dat\_w1 [127, 64] | | | | | | | |
| 0 x3050 |  | lpbk\_ecc\_mask\_  w0 | lpbk\_dat\_mask\_w0 | |  |  | lpbk\_ecc\_w0 | |
| 0 x3058 |  | lpbk\_ecc\_mask\_  w1 | lpbk\_dat\_mask\_w1 | |  |  | lpbk\_ecc\_w1 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 x3060 |  |  |  |  |  |  |  | prbs\_23 |
| 0 x3068 |  |  |  |  |  | prbs\_init | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x3100 |  |  |  |  | fix\_data\_pattern\_ind  The ex | bus\_width | page\_size | test\_engine\_en |
| 0 x3108 |  |  | cs\_diff\_tst | c\_diff\_tst | bg\_diff\_tst | ba\_diff\_tst | row\_diff\_tst | col\_diff\_tst |
| 0 x3120 | addr\_base\_tst | | | | | | | |
| 0 x3128 |  |  |  |  |  |  |  |  |
| 0 x3130 | user\_data\_pattern | | | | | | | |
| 0 x3138 |  |  |  |  |  |  |  |  |
| 0 x3140 | Valid\_bits [63:0] | | | | | | | |
| 0 x3148 |  |  |  |  |  |  |  | Valid\_bits [71, 64] |
| 0 x3150 | CTRL [63:0] | | | | | | | |
| 0 x3158 | CTRL (127-64) | | | | | | | |
| 0 x3160 | Obs [63:0] (RD) | | | | | | | |
| 0 x3168 | Obs (127-64) (RD) | | | | | | | |
| 0 x3170 | Obs (191-128) (RD) | | | | | | | |
| 0 x3178 | Obs (255-192) (RD) | | | | | | | |
| 0 x3180 | Obs (319-256) (RD) | | | | | | | |
| 0 x3188 | Obs (383-320) (RD) | | | | | | | |
| 0 x3190 | Obs (447-384) (RD) | | | | | | | |
| 0 x3198 | Obs (511-448) (RD) | | | | | | | |
| 0 x31a0 | Obs (575-512) (RD) | | | | | | | |
| 0 x31a8 | Obs (639-576) (RD) | | | | | | | |
| 0 x31b0 |  |  |  |  | Obs (671-640) (RD) | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x3200 |  |  |  |  |  |  |  |  |
| 0 x3208 |  |  |  |  |  |  |  |  |
| 0 x3220 | tud\_i0 | | | | | | | |
| 0 x3228 | tud\_i1 | | | | | | | |
| 0 x3230 | Tud\_o (RD) | | | | | | | |
| ... |  |  |  |  |  |  |  |  |
| 0 x3300 | tst\_300 | | | | | | | |
| 0 x3308 | tst\_308 | | | | | | | |
| 0 x3310 | tst\_310 | | | | | | | |
| 0 x3318 | tst\_318 | | | | | | | |
| 0 x3320 | tst\_320 | | | | | | | |
| 0 x3328 | tst\_328 | | | | | | | |
| 0 x3330 | tst\_330 | | | | | | | |
| 0 x3338 | tst\_338 | | | | | | | |

|  |  |
| --- | --- |
| 0 x3340 | tst\_340 |
| 0 x3348 | tst\_348 |
| 0 x3350 | tst\_350 |
| 0 x3358 | tst\_358 |
| 0 x3360 | tst\_360 |
| 0 x3368 | tst\_368 |
| 0 x3370 | tst\_370 |
| 0 x3378 | tst\_378 |

## Software Programming Guide

#### Initialization operation

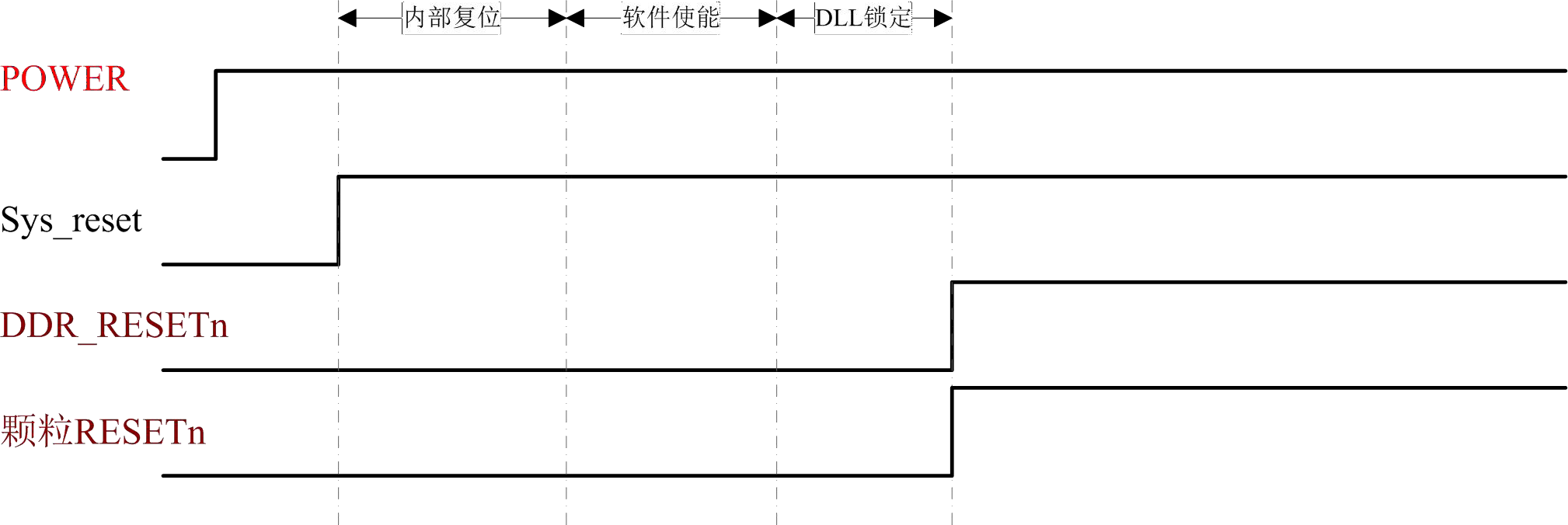
Initialization begins when the software writes 0x2 to register Init\_start (0x010). All other registers must be set to the correct value before the Init\_start signal is set.

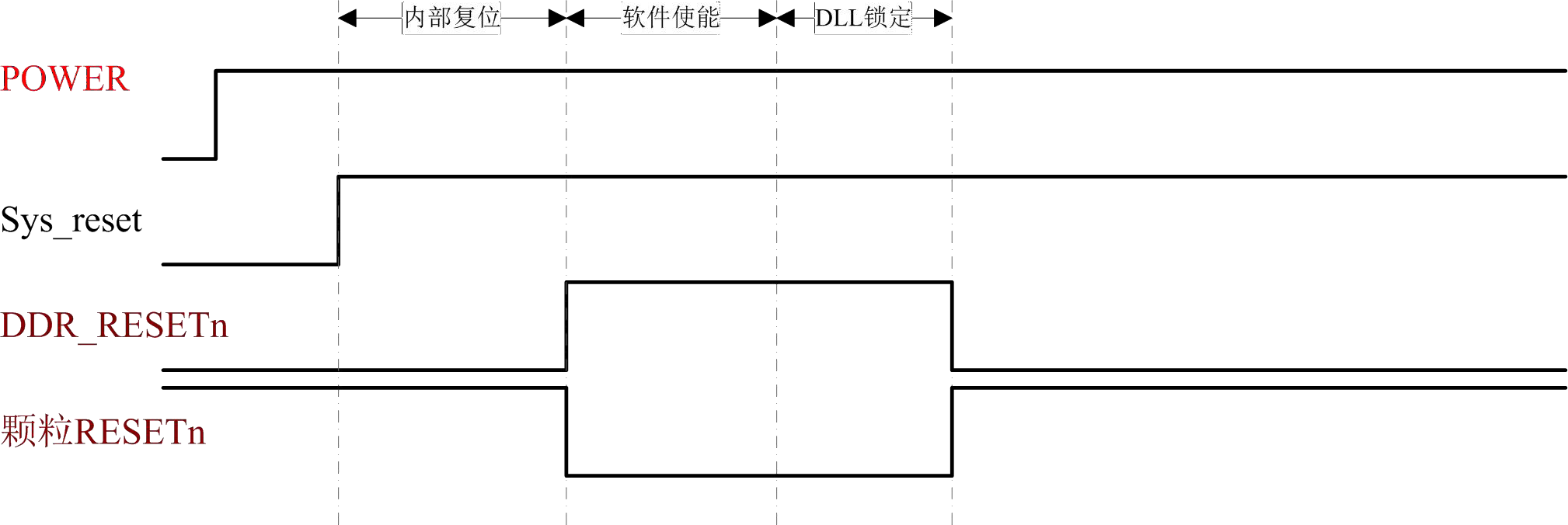
The DRAM initialization process of software and hardware collaboration is as follows:

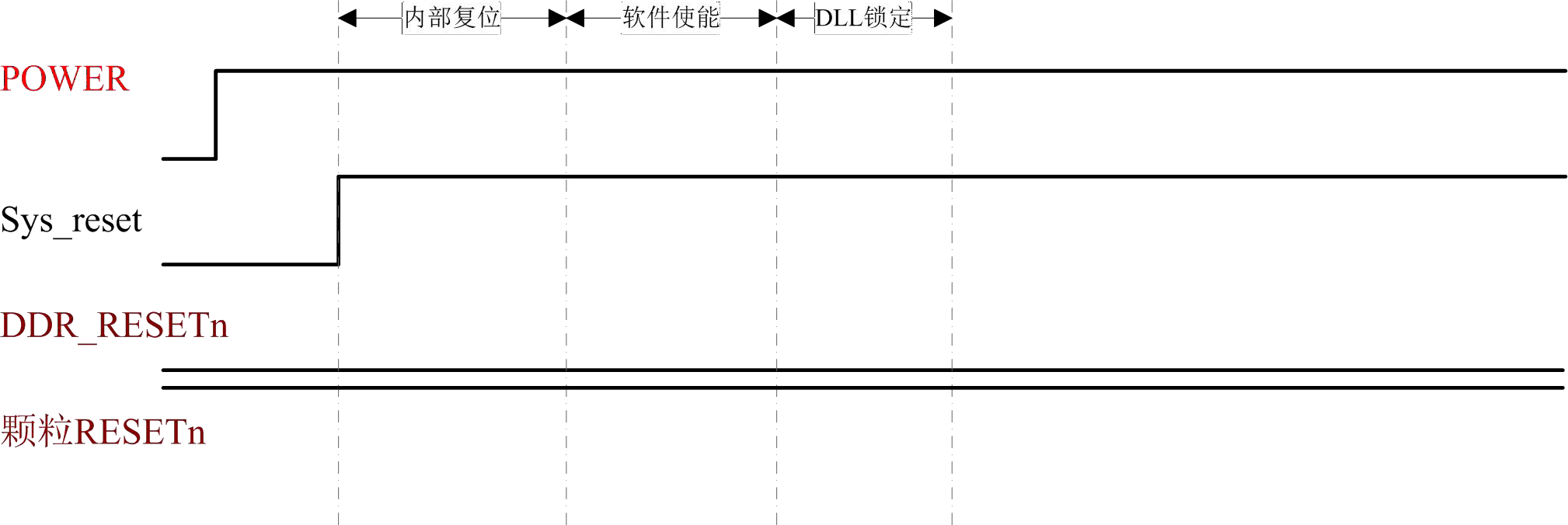
* + - 1. Set PM\_clk\_SEL\_CKCA and PM\_clk\_SEL\_DS
      2. Set PM\_phy\_init\_START to 1 to initiate the PHY
      3. Wait for DLL master module lock, that is, PM\_DLL\_init\_done is 1
      4. Wait for pm\_DLL\_lock\_\* or PM\_pll\_lock\_\* of all clock generation modules to become 1
      5. Enable all pm\_clken\_\*
      6. With PM\_init\_start set to 1, the memory controller begins to initialize
      7. Wait for the memory controller to initialize, that is, pm\_DRAM\_init has the same value as PM\_CS\_enable.

#### Control of the reset pin

To make it easier to control the reset pin in STR and other states, you can use the pad\_reset\_PO (0x808) register for special reset pin (DDR\_RESETn) control. There are two main control modes:

* + - 1. In general mode, reset\_ctrl[1:0] == 2 'b00. In this mode, the behavior of the reset signal pin is compatible with the general control mode. DDR\_RESETn is directly connected to the corresponding pin on the memory slot on the motherboard. The behavior of the pin is:
* When not powered on: pin state is low;
* When power on: pin state is low;
  + - 1. When the controller is initialized, the pin state is high;
      2.  In normal operation, the pin state is high. The timing sequence is shown in the figure below:
      3. In reverse mode, reset\_ctrl[1:0] == 2 'b10. In this mode, the effective level of the reset signal pin is reversed when the actual memory control is carried out. So DDR\_RESETn needs to be connected to the corresponding pin on the memory slot via the reverter on the motherboard. The behavior of the pin is:
* When not powered on: pin state is low;
* When power on: pin state is low;
* When the controller is configured: the pin state is high;
* When the controller is initialized: the pin state is low;
* Normal operation: pin condition is low. The timing sequence is shown in the figure below:



* + - 1. Reset disable mode, PM\_PAD\_reset\_o [1:0] == 2 'b01. In this mode, the reset signal pin remains low throughout the memory operation. So DDR\_RESETn needs to be connected to the corresponding pin on the memory slot via the reverter on the motherboard. The behavior of the pin is:
*  It's always low; The timing sequence is shown in the figure below:

Combined with the latter two reset modes, STR control can be implemented directly using the reset signal of the memory controller. When the entire system is started from the closed state, use the method in (2) to reset normally and start working with the memory stick. When the system recovers from STR, use the method in (3) to reconfigure the memory bar so that it can start working properly again without breaking the original state of the memory bar.

#### Leveling

The operation came with Leveling in DDR3/4, which was used to intelligent configure the phase relationship between various signals in the read and write operation of memory controller. Usually it includes the Write Leveling, Read Leveling and Gate Leveling. In this controller, only the Write Leveling and Gate Leveling were made, and the Read Leveling was not made. The software needed to make the function of the Read Leveling come to pass by judging the correctness of the Read Leveling. In addition to the DQS phase and GATE phase that were handled during the Leveling process, the configuration methods of writing DQ phase and reading DQ phase could also be calculated according to the final phase Leveling. In addition, the design supports bit-deskew to compensate for the delay difference between different bits within a dataslice.

##### The Write Leveling

To configure the phase relationship between Write DQS and the clock, the software program needs to refer to the following steps.

* + - * 1. Complete the controller initialization, see the previous section;
        2. Put Dll\_wrdqs\_x (x = 0... 8) Set to 0x20;
        3. Put Dll\_wrdq\_x (x = 0... 8) Set to 0x0;
        4. Set Lvl\_mode to 2 'b01;
        5. The Lvl\_ready register, if it is 1, indicates that the Write Leveling request can be started.
        6. Set Lvl\_req to 1;
        7. The Lvl\_done register, if it is 1, represents the completion of the Write Leveling request.
        8. If Lvl\_resp\_x is 0, increase the corresponding Dll\_wrdq\_x[6:0] and DLL\_1Xdly [6:0] by 1, and repeat 5-7 until Lvl\_resp\_x is 1, then turn to 9. If it is 1, increase the corresponding Dll\_wrdq\_x[6:0] and DLL\_1xdly [6:0] by 1, and repeat 5-7 until LVl\_WRdQ\_x [6:0] and DLL\_1xdly [6:0] by 1, and repeat 5-7 until Lvl\_resp\_x is 1, then turn to 9.
        9. Subtract 0x40 from the Dll\_wrdq\_x and dLL\_1xdly value, in which case the Dll\_wrdq\_x and DLL\_1xdly value should be the correct set value.
        10. Set PM\_dly\_2X according to the DIMM type, and increase the pm\_dly\_2X value by 0x010101 for the particle to the right of the 0x0 boundary.
        11. Set Lvl\_mode (0x700) to 2 'B00 and exit the Write Leveling mode.

##### Gate Leveling

Gate Leveling was used to configure the timing of making the DQS window available for sampling and reading. The software programming was referred to the following steps

Flash.

* + - * 1. Complete the controller initialization, see the previous section;
        2. Complete the Write Leveling, see the previous section;
        3. Will Dll\_gate\_x (x = 0... 8) Set to 0;
        4. Set Lvl\_mode to 2 'b10;
        5. The Lvl\_ready register, if it is 1, means that the Gate Leveling request can be started.
        6. Set Lvl\_req to 1;
        7. The Lvl\_done register, if it is 1, represents the completion of a Gate Leveling request.
        8. Sample Lvl\_resp\_x[0] register. If the first sampling finds Lvl\_resp\_x[0] to be 1, increment the corresponding Dll\_gate\_x[6:0] by 1 and repeat 6-8 until the sampling result is 0, otherwise proceed to the next step.
        9. If the sample result is 0, increment the corresponding Dll\_gate\_x[6:0] by 1 and repeat 6-9; If it is 1, then the Gate Leveling operation has been successful.
        10. Set pm\_rdedge\_sel (11) to Dll\_gate\_x (x = 0... 8) minus 0x20;

1. After adjusting, do two more Lvl\_req operations. Watch for changes in the values of Lvl\_resp\_x[7:5] and Lvl\_resp\_x[4:2]. If it's not 4, you might need to add or subtract one to Rd\_oe\_begin\_x, and if it's greater than Burst\_length/2, you'll probably need to do some fine-tuning of the value of Dll\_gate\_x;
2. Set Lvl\_mode (0x700) to 2 'b00 and quit the Gate Leveling mode.
3. Thus the Gate Leveling operation came to an end.

#### Power control configuration flow

First you need to set PM\_PAD\_ctrl\_CA [0] to 1 and wait for memory initialization to complete

Pm\_pad\_ctrl\_ca [0] is 0. CAL Mode is only enabled in DDR4 Mode.

#### Initiate MRS command alone

In DDR3 mode, the MRS commands issued by the memory controller to the memory are in the following order:

MR2\_CS0, MR2\_CS1, MR2\_CS2, MR2\_CS3, MR2\_CS4, MR2\_CS5, MR2\_CS6, MR2\_CS7, MR3\_CS0, MR3\_CS1, MR3\_CS2, MR3\_CS3, MR3\_CS4, MR3\_CS5, MR3\_CS6, MR3\_CS7, MR1\_CS0, MR1\_CS1, MR1\_CS2, MR1\_CS3, MR1\_CS4, MR1\_CS5, MR1\_CS6, MR1\_CS7, MR0\_CS0,MR1\_CS1, MR1\_CS2, MR1\_CS3, MR0\_CS4, MR0\_CS5, MR0\_CS6, MR0\_CS7.

In addition, in the case of DDR4 mode, the MRS commands issued by the memory controller to the memory are in the following order:MR3\_CS0, MR3\_CS1, MR3\_CS2, MR3\_CS4, MR3\_CS5, MR3\_CS6, MR3\_CS7, MR6\_CS0, MR6\_CS2, MR6\_CS4, MR6\_CS6, MR6\_CS7, MR5\_CS0, MR5\_CS1, MR5\_CS7, MR4\_CS0, MR1\_CS1, MR1\_CS2, MR1\_CS3, MR4\_CS4, MR4\_CS5, MR4\_CS6, MR4\_CS7, MR2\_CS0, MR2\_CS1, MR2\_CS2, MR2\_CS3, MR2\_CS4, MR2\_CS5, MR2\_CS6, MR2\_CS7, MR1\_CS0, MR1\_CS1, MR1\_CS2, MR1\_CS3, MR1\_CS4, MR1\_CS5, MR1\_CS6, MR1\_CS7, MR0\_CS0, MR1\_CS1, MR1\_CS2, MR1\_CS3, MR0\_CS4, MR0\_CS5, MR0\_CS6, MR0\_CS7.

Where, the validity of MRS command corresponding to CS is determined by Cs\_mrs. Only when the selected bit of Cs\_mrs corresponding to each slice is valid, the MRS command will be issued to DRAM. The value of each corresponding MR is determined by the register MR \*\_cs\*. These values are also used for the MRS command when initializing memory.

Specific operations are as follows:

* + - 1. Set registers Cs\_mrs (0x1101), Mr\* \_CS \* (0x1140 -- 0x11f8) to the correct values;
      2. Set Command\_mode (0x0x1120) to 1 to make the controller enter command send mode;
      3. Sample Status\_cmd (0x1122). If it is 1, the controller has entered the command sending mode and can proceed to the next step. If it is 0, it needs to continue to wait.
      4. Write Mrs\_req (0x1126) as 1 and send MRS command to DRAM;
      5. Sampling Mrs\_done (0x1127). If it is 1, it means that the MRS command has been sent and can exit. If it is 0, it needs to continue to wait.
      6. Set Command\_mode (0x1120) to 0 to cause the controller to exit command send mode.

#### Any operation controls the bus

The memory controller can issue any combination of commands to the DRAM through command sending mode, and the software can set Cmd\_cs, Cmd\_cmd, Cmd\_ba, Cmd\_a (0x1128) to issue to the DRAM in command sending mode.

Specific operations are as follows:

* + - 1. Set registers Cmd\_cs, Cmd\_cmd, Cmd\_ba, Cmd\_a (0x1128) to the correct values;
      2. Set Command\_mode (0x1120) to 1 to make the controller enter command send mode;
      3. Sample Status\_cmd (0x1122). If it is 1, the controller has entered the command sending mode and can proceed to the next step. If it is 0, it needs to continue to wait.
      4. Write Cmd\_req (0x1121) as 1 and send the command to DRAM;
      5. Set Command\_mode (0x1120) to 0 to cause the controller to exit command send mode.

#### Self-circulating test mode control

Since the cycle test pattern can be respectively in test mode or normal mode using the function, therefore, this memory control device, the two sets of independent control interface is implemented, a set of used in the test mode directly controlled by the test port, another set of used in normal function mode by the register allocation module configuration can make the test.

The reuse of these two sets of interfaces is controlled by port test\_PHY. When test\_PHY is effective, the controller's test\_\* port is used for control. At this time, the self-test is completely controlled by hardware. When test\_PHY does not work, the pm\_\* parameter of the software program is used for control. The specific signal meaning of using the test port can be referred to in the register parameters with the same name.

These two sets of interfaces are basically the same in terms of control parameters, only different access points. This paper introduces the control method of software programming. Specific operations are as follows:

* + - 1. Set all parameters of the memory controller correctly.
      2. Wait for the clock reset to be stable according to the initialization process;
      3. Set register Lpbk\_en to 1;
      4. Set register Lpbk\_start to 1; At this point, loop testing begins.
      5. Since the cycle test has started, the software needs to frequently detect whether there is an error. The specific operation is as follows:
      6. Sample register Lpbk\_error. If the value is 1, it means that an error has occurred. At this time, the error data and correct data in the first error can be observed with the register through observation such as Lpbk\_\*. If the

A value of 0 indicates that no data error has occurred.

#### ECC function usage control

ECC functionality is available only in 64-bit mode. Ecc\_enable includes the following two control bits:

Ecc\_enable[0] controls whether to enable the ECC function, which will only be enabled if the effective bit is set. Ecc\_enable[1] controls whether an error is reported through the read response path inside the processor to enable the ECC two digits to occur

A wrong read access can cause an immediate exception to the processor core.

In addition, AN ECC error can notify the processor core by means of an interrupt. This interrupt is controlled by Int\_enable. The interrupt consists of two vectors. Int\_vector[0] indicates an ECC error (including 1 and 2 bits), and Int\_vecotr[1] indicates an ECC two-bit error. The purge of Int\_vector is achieved by writing 1 to the corresponding bit.

#### Error condition observation

After an error occurs in the memory controller, the corresponding error information can be obtained by accessing the corresponding system configuration register, and a simple debugging operation can be carried out. The register base address is 0x1fe00000 or 0x3ff00000 and can also be accessed using the configuration register instruction, the register and its corresponding bits are shown below.

Table 13- 30 0 memory controller error state observation register

|  |  |  |  |
| --- | --- | --- | --- |
| register | offset | control | instructions |
| Memory controller no. 0 ECC set register  Mc0\_ecc\_set | 0 x0600 | RW | Memory controller ECC set register [5:0] : MC0 int\_enable, interrupt enable [8]: MC0 int\_trigger, interrupt trigger configuration  [21:16]: MC0 int\_vector(RO), interrupt vector(read-only) [33:32]: MC0 ecc\_enable, ECC related function enable  [40]: MC0 RD\_before\_WR, read and write function enabled |
|  | 0 x0608 | RW | reserve |
| Memory controller no. 0 ECC counting register  Mc0\_ecc\_cnt | 0 x0610 | RW | Memory controller no. 0 ECC counting register  [7:0]: MC0 int\_Cnt, configure the threshold value of ECC check triggered interrupt times [15:8]: MC0 int\_CNT\_ERR (RO), ECC check bit error times statistics  (read-only)  [23:16]: MC0 int\_CNT\_fatal (RO), ECC check two-digit error count (read-only) |
| Memory controller 0 ECC error count register Mc0\_ecc\_cs\_cnt | 0 x0618 | RO | Memory controller no. 0 ECC error count register  [7:0]: MC0 ECC\_CNT\_CS\_0, CS0 ECC error statistics [15:8]: MC0 ECC\_CNT\_CS\_1, CS1 ECC error statistics [23:16]: MC0 ECC\_CNT\_CS\_2, CS2 ECC error statistics [31:24]: MC0 ECC\_CNT\_CS\_3, CS3 ECC error statistics [39:32]:MC0 ECC\_CNT\_CS\_4, CS4 ECC error statistics [47:40]: MC0 ECC\_CNT\_CS\_5, CS5 ECC error statistics [55:48]: MC0 ECC\_CNT\_CS\_6, CS6 ECC error statistics [63:56]: MC0 ECC\_CNT\_CS\_7, CS7 ECC error statistics |
| Memory controller ZERO ECC check code register  Mc0\_ecc\_code | 0 x0620 | RO | Memory controller ZERO ECC check code register  [7:0]: MC0 ECC\_code\_64, ECC check code for 64-bit ECC check, which makes the memory directory function invalid  [41:32]: MC0 ECC\_code\_256, 256-bit ECC check code, so that the memory directory function can be valid  [52:48]: MC0 ECC\_code\_dir, ECC check code, valid only if enabled to memory directory function  [60/56]: MC0 ECC\_DATa\_DIR, memory directory ECC data, only enabled  Memory directory function when valid |
| Memory controller 0 ECC error address register  Mc0\_ecc\_addr | 0 x0628 | RO | Memory controller 0 ECC error address register  MC0 ECC\_ADDr, ECC check error address information |
| Memory controller ECC error data register 0  Mc0\_ecc\_data0 | 0 x0630 | RO | ECC error data depositor 0 [63:0]:Mc0\_ecc\_data0, ECC check error data information, 64 bits  Data in ECC mode, data in 256-bit ECC mode [63:0] |
| Memory controller 0 ECC error data register 1  Mc0\_ecc\_data1 | 0 x0638 | RO | ECC error data depositor 1 [63:0]:Mc0\_ecc\_data1, ECC check error data information, 256 bits  Data in ECC mode [127:64] |
| Memory controller 0 ECC error data register 2  Mc0\_ecc\_data2 | 0 x0640 | RO | ECC error data depositor 2 [63:0]:Mc0\_ecc\_data2, ECC check error data information, 256 bits  Data in ECC mode [191:128] |
| Memory controller 0 ECC error data register 3  Mc0\_ecc\_data3 | 0 x0648 | RO | ECC error data depositor 3 [63:0]:Mc0\_ecc\_data3, ECC check error data information, 256 bits  Data in ECC mode [255:192] |

Table 13- 4 1 Memory controller error state observation register

|  |  |  |  |
| --- | --- | --- | --- |
| register | address | control | instructions |
| Memory controller no. 1 ECC sets registers  Mc1\_ecc\_set | 0 x0700 | RW | 1 memory controller ECC set register [5:0] : MC1 int\_enable, interrupt enable [8]: MC1 int\_trigger, interrupt trigger configuration  [21:16]: MC1 int\_vector(RO), interrupt vector(read-only) [33:32]: MC1 ecc\_enable, ECC related function enable  [40]: MC1 RD\_before\_WR, read and write function enabled |
|  | 0 x0708 | RW | reserve |
| Memory controller 1 ECC count register  Mc1\_ecc\_cnt | 0 x0710 | RW | Memory controller 1 ECC count register  [7:0]: MC1 int\_Cnt, configure the threshold value of ECC check trigger interrupt times [15:8]: MC1 int\_CNT\_ERR (RO), ECC check bit error times statistics  (read-only)  [23:16]: MC1 int\_CNT\_fatal (RO), ECC check two-digit error count (read-only) |
| Memory controller 1 ECC error count register Mc1\_ecc\_cs\_cnt | 0 x0718 | RO | Memory controller no. 1 ECC error count register  [7:0]: MC1 ECC\_CNT\_CS\_0, CS0 ECC error statistics [15:8]: MC1 ECC\_CNT\_CS\_1, CS1 ECC error statistics [23:16]: MC1 ECC\_CNT\_CS\_2, CS2 ECC error statistics [31:24]: MC1 ECC\_CNT\_CS\_3, CS3 ECC error statistics [39:32]:MC1 ECC\_CNT\_CS\_4, CS4 ECC error statistics [47:40]: MC1 ECC\_CNT\_CS\_5, CS5 ECC error statistics [55:48]: MC1 ECC\_CNT\_CS\_6, CS6 ECC error statistics [63:56]: MC1 ECC\_CNT\_CS\_7, CS7 ECC error statistics |
| Memory controller 1 ECC check code register  Mc1\_ecc\_code | 0 x0720 | RO | Memory controller 1 ECC check code register  [7:0]: MC1 ECC\_code\_64, ECC check code for 64-bit ECC check, which makes the memory directory function invalid  [41:32]: MC1 ECC\_code\_256, 256-bit ECC check code, so that the memory directory function can be valid  [52:48]: MC1 ECc\_code\_dir, ECC check code, valid only if enabled to memory directory function  MC1 ECc\_DATa\_DIR, memory directory ECC data, only enabled  Memory directory function when valid |
| Memory controller 1 ECC error address register  Mc1\_ecc\_addr | 0 x0728 | RO | Memory controller 1 ECC error address register  MC1 ECC\_ADDr, ECC check error address information |
| Memory controller 1 ECC error data register 0  Mc1\_ecc\_data0 | 0 x0730 | RO | ECC error data depositor 0 [63:0]:Mc1\_ecc\_data0, ECC check error data information, 64 bits  Data in ECC mode, data in 256-bit ECC mode [63:0] |
| Memory controller 1 ECC error data register 1  Mc1\_ecc\_data1 | 0 x0738 | RO | ECC error data depositor 1 [63:0]:Mc1\_ecc\_data1, ECC check error data information, 256 bits  Data in ECC mode [127:64] |
| Memory controller 1 ECC error data register 2  Mc1\_ecc\_data2 | 0 x0740 | RO | ECC error data depositor 2 [63:0]:Mc1\_ecc\_data2, ECC check error data information, 256 bits  Data in ECC mode [191:128] |
| Memory controller 1 ECC error data register 3  Mc1\_ecc\_data3 | 0 x0748 | RO | ECC error data depositor 3 [63:0]:Mc1\_ecc\_data3, ECC check error data information, 256 bits  Data in ECC mode [255:192] |

## HyperTransport controller

In the Loong Chip 3A4000, the HyperTransport bus is used for external device connection and multi-chip interconnection. When used for connecting peripherals, but by the user program free to choose whether to support the IO Cache consistency (through the address window Uncache Settings, see section 14.5.14) : when configured to support the Cache consistency model, IO device internal DMA access for transparent Cache levels, namely by the hardware, automatically maintain consistency without software Cache instructions for maintenance by the program; When the HyperTransport bus is used for multi-chip interconnections, the HT0 controller (starting at 0x0C00\_0000\_0000 -- 0x0DFFFF\_FFFF) can be configured to support inter-chip Cache consistency, while the HT1 controller (starting at 0x0E00\_0000\_0000 -- 0x0FFF\_FFFF\_FFFF) can be configured to support inter-chip Cache consistency, as detailed in Section 14.7. In the 8-chip interconnection structure, the consistent pattern of THE HT1\_HI controller is configured through the pins in CHIP\_CONFIG.

The HyperTransport controller supports a maximum two-way 16-bit width and 2.4GHz operation frequency. After the connection is automatically initialized by the system, the user program can change the width and running frequency by modifying the corresponding configuration registers in the protocol, and then reinitialize, as detailed in Section 14.1.

The main features of the Longcore 3A4000 HyperTransport controller are as follows:

* Support HT1.0/HT3.0 protocol
* Support for 200/400/800/1600/2000/2400 MHZ operating frequency
* The maximum frequency of the controller is 1GHz
* HT1.0 supports 8-bit widths
* HT3.0 supports 8/16 bit widths
* Each HT controller (HT0/HT1) can be configured as two 8-bit HT controllers
* The bus control signal (including PowerOK, Rstn, LDT\_Stopn) direction is configurable
* Peripheral DMA space Cache/Uncache can be configured
* The Cache consistency mode can be configured for multi-chip interconnections

## HyperTransport hardware setup and initialization

HyperTransport bus is composed of transmission signal bus and control signal pin, etc. The following table gives the pins related to HyperTransport bus and its function description.

Table 14-1 HyperTransport bus related pin signals

|  |  |  |
| --- | --- | --- |
| **pin** | **The name of the** | **describe** |
| HT0\_8x2 | Bus width configuration | 1. The 16-bit HyperTransport bus is configured as two independent 8-bit buses, which are controlled by two independent controllers respectively. The address space is divided as follows  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, address space is the address of HT0\_Lo, namely address[40]  = 0; All HT0\_Hi signals are invalid. |
| HT0\_Lo\_mode | Master device mode | 1: Set HT0\_Lo as the main device mode. In this mode, bus control signals are driven by HT0\_Lo, including HT0\_Lo\_Powerok, HT0\_Lo\_Rstn, HT0\_Lo\_Ldt\_Stopn. In this mode, the control signals can also be bidirectional. At the same time, this pin determines the initial value of the register "Act as Slave". When this register is 0, the Bridge bit in the package on the HyperTransport bus is 1, otherwise it is 0. In addition, when this register is 0, if the request address on the HyperTransport bus does not hit the receive window of the controller, it will be sent back to the bus as a P2P request; if this register is 1 and it does not hit, it will be responded as an error request.  0: Set HT0\_Lo to slave device mode, in which bus control signals such as HT0\_Lo\_Powerok, HT0\_Lo\_Rstn, HT0\_Lo\_Ldt\_Stopn are driven by the other device. In this mode, these control signals are driven by the other device, if not driven correctly, then THE HT bus  Not working correctly. |
| HT0\_Lo\_Powerok | Bus Powerok | When HT0\_Lo\_Mode is 1, it is controlled by HT0\_Lo. When HT0\_Lo\_Mode is 0, it is controlled by the other 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 other device. |
| HT0\_Lo\_Ldt\_Stopn | Bus Ldt\_Stopn | When HT0\_Lo\_Mode is 1, it is controlled by HT0\_Lo. When HT0\_Lo\_Mode is 0, it is controlled by the other device. |
| HT0\_Lo\_Ldt\_Reqn | Bus Ldt\_Reqn | HyperTransport bus Ldt\_Reqn signal, |
| HT0\_Hi\_mode | Master device mode | 1: Set HT0\_Hi as the main device mode. In this mode, bus control signals are driven by HT0\_Hi, including HT0\_Hi\_Powerok, HT0\_Hi\_Rstn, HT0\_Hi\_Ldt\_Stopn. In this mode, the control signals can also be bidirectional. At the same time, this pin determines the initial value of the register "Act as Slave". When this register is 0, the Bridge bit in the package on the HyperTransport bus is 1, otherwise it is 0. In addition, when this register is 0, if the request address on the HyperTransport bus does not hit the receive window of the controller, it will be sent back to the bus as a P2P request; if this register is 1 and it does not hit, it will be done |
|  |  | Respond to an incorrect request.  0: Set HT0\_Hi to slave device mode, in which bus control signals such as HT0\_Hi\_Powerok, HT0\_Hi\_Rstn, HT0\_Hi\_Ldt\_Stopn are driven by the other device. In this mode, these control signals are driven by the other device, if not driven correctly, the HT bus will not work correctly. |
| HT0\_Hi\_Powerok | Bus Powerok | When HT0\_Lo\_Mode is 1, it is controlled by HT0\_Hi. When HT0\_Lo\_Mode is 0, it is controlled by the other device. When HT0\_8x2 is 1, the high 8-bit bus is controlled. If HT0\_8x2 is 0, it is invalid. |
| HT0\_Hi\_Rstn | Bus Rstn | When HT0\_Lo\_Mode is 1, it is controlled by HT0\_Hi. When HT0\_Lo\_Mode is 0, it is controlled by the other device. When HT0\_8x2 is 1, the high 8-bit bus is controlled. If HT0\_8x2 is 0, it is invalid. |
| HT0\_Hi\_Ldt\_Stopn | Bus Ldt\_Stopn | When HT0\_Lo\_Mode is 1, it is controlled by HT0\_Hi. When HT0\_Lo\_Mode is 0, it is controlled by the other device. When HT0\_8x2 is 1, the high 8-bit bus is controlled. If HT0\_8x2 is 0, it is invalid. |
| HT0\_Hi\_Ldt\_Reqn | Bus Ldt\_Reqn | HyperTransport bus Ldt\_Reqn signal,  When HT0\_8x2 is 1, the high 8-bit bus is controlled.  If HT0\_8x2 is 0, it is invalid. |
| HT0\_Rx\_CLKp HT0\_Rx\_CLKn [1:0] [1:0] HT0\_Tx\_CLKp [1:0] HT0\_Tx\_CLKp (1-0) | CLK [1:0] | HyperTransport bus CLK signal  When HT0\_8x2 is 1, CLK[1] is controlled by HT0\_Hi  CLK[0] is controlled by HT0\_Lo  When HT0\_8x2 is 0, CLK[1:0] is controlled by HT0\_Lo |
| HT0\_Rx\_CTLp HT0\_Rx\_CTLn [1:0] [1:0] HT0\_Tx\_CTLp [1:0] HT0\_Tx\_CTLn (1-0) | CTL (1-0) | HyperTransport Bus CTL signal  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 HT0\_Rx\_CADn [15:0] [15:0] HT0\_Tx\_CADp HT0\_Tx\_CADn [15:0] [15:0] | CAD [15:0] | HyperTransport Bus CAD signals  When HT0\_8x2 is 1, CAD[15:8] is controlled by HT0\_Hi  CAD[7:0] is controlled by HT0\_Lo  When HT0\_8x2 is 0, CAD[15:0] is controlled by HT0\_Lo |

The initialization of HyperTransport starts automatically after each reset. After cold start, the HyperTransport bus will automatically work at the lowest frequency (200MHz) and the minimum width (8BIT), and try to do the bus initialization handshake. Whether the initialization is Complete can be read out by the register "Init Complete" (see section 14.5.2). After initialization, the bus Width can be read Out from registers "Link Width Out" and "Link Width In" (see section 14.5.2).

After initialization, the user can rewrite registers "Link Width Out", "Link Width In" and "Link Freq", and also configure corresponding registers of the other device. After configuration, the user needs to reinitialize the bus or through "HT\_Ldt\_Stopn" signal to make the rewritten register value effective. After the reinitialization is complete the HyperTransport bus will work at the new frequency and width. It is important to note that the configuration of the devices on both ends of HyperTransport needs to be one-to-one, otherwise the HyperTransport interface will not work properly.

## HyperTransport protocol support

The HyperTransport bus of the Loongson 3A4000 supports most of the commands in the 1.03/3.0 protocol and includes some extension instructions in the extended Conformance protocol that supports multi-chip interconnection. In both modes, the commands that the HyperTransport receiver can receive are shown in the following table. It is important to note that atomic operation commands are not supported for the HyperTransport bus.

Table 14-2 Commands that the HyperTransport receiver can receive

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **coding** | **channel** | **The command** | **The standard model** | **Extension (consistency)** |
| 000000 | - | The NOP | Empty packet or flow control |  |
| 000001 | NPC | FLUSH | No operation |  |
| x01xxx | NPC  The or PC | The Write | Bit 5:0 - Nonposted  1 - Posted bit 2:0 - Byte  1 -  Doubleword  Bit 1: Don't Care bit 0: Don't Care | Bit 5: Must be 1, POSTED  Bit 2:0 - Byte  1 -- Doubleword bit 1: Don't Care  Bit 0: Must be 1 |
| 01 XXXX | NPC | The Read | Bit 3: Don't Care bit 2:0 -- Byte  1 -  Doubleword  Bit 1: Don't Care bit 0: Don't Care | Bit 3: Don't Care bit 2:0 -- Byte  1 -- Doubleword bit 1: Don't Care  Bit 0: Must be 1 |
| 110000 | R | RdRespons  e | Read operation return |  |
| 110011 | R | TgtDone | Write operation return |  |
| 110100 | The PC | WrCoherent | ---- | Write command extension |
| 110101 | The PC | WrAddr | ---- | Write address extension |
| 111000 | R | RespCohere nt | ---- | Read response extension |
| 111001 | NPC | RdCoherent | ---- | Read command extension |
| 111010 | The PC | Broadcast | No operation |  |
| 111011 | NPC | RdAddr | ---- | Read address extension |
| 111100 | The PC | A FENCE | Guaranteed order relation |  |
| 111111 | - | The Sync/Error | The Sync/Error |  |

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

Table 14-3 Commands that will be sent out in two modes

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

## HyperTransport interrupt support

The HyperTransport controller provides 256 interrupt vectors that can support types of interrupts like Fix, Arbiter, etc., but has no support for hardware automatic EOI. For the above two types of interrupts, the controller will automatically write to the interrupt register after receiving, and the system interrupt controller will be informed of the interrupt according to the setting of the interrupt mask register. For the specific interrupt control, see the interrupt control register description in Section 14.5.7.

#### PIC interrupt

PIC interrupts are specifically supported by the controller to speed up this type of interrupt handling.

A typical PIC interrupt is accomplished by the following steps: (1) PIC controller sends PIC interrupt request to the system; The system sends interrupt vector query to PIC controller; PIC controller sends interrupt vector number to the system; The system clears the corresponding interrupt on PIC controller. PIC controller will issue the next interrupt to the system only after the above 4 steps are completed. For longson 3A4000 HyperTransport controller, the first 3 steps will be automatically processed and PIC interrupt vector will be written into the corresponding position in 256 interrupt vectors. After the software system has processed the interrupt, it needs to carry out the fourth step, that is, send the clear interrupt to PIC controller. Then the processing of the next interrupt begins.

#### Local interrupt handling

In the traditional interrupt processing mode, all interrupts are stored by the interrupt vector inside the HT controller, and then distributed through the interrupt router on the chip connected to the interrupt line of the HT controller. In this case, HT interrupts only by a finite amount

The CPU core can be interrupted in several connection modes, and it cannot be distributed across slices, so the usage scenario is limited.

In this HT interrupt mode, during interrupt processing, the interrupt router on the chip is transparent to the software, and the kernel finds directly on the interrupt vector of HT controller (generally 0x90000EFDFB000080), and then processes by bit. At this time, no matter how the routing mode is configured, all interrupts on HT controller are directly read.

#### Extended interrupt processing

Extended IO interrupts implemented in 3A4000 can greatly increase the flexibility of interrupt distribution and interrupt handling.

In THE INTERRUPT extension mode of HT, interrupts other than PIC interrupts are directly written into the newly added extended interrupt register on the chip interrupt router, and then routed or distributed according to the relevant configuration of the extended interrupt register.

After using the extended IO interrupt, the HT controller is transparent to the software for interrupt processing, and the kernel reads the interrupt state directly to the extended IO state register (configuration space 0x1800) for processing. Each core only reads its interrupt state and processes it, without interference between different cores.

Interrupt forwarding is performed on HT controller by enabling external interrupt transformation configuration register. As stated in 14.5.34, the software needs to set HT\_int\_trans to the target address of the extended IO interrupt trigger register. The register address in 3A4000 is 0x1fe01140, or 0x10000\_00001140.

The kernel needs to enable the corresponding bits in the "other function set register" before it can use extended interrupt processing. The register is base address 0x1fe00000 and offset address 0x0420.

Table 14-4 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| spoilers | EXT\_INT\_en | RW | 0 x0 | Extend IO interrupt enablement |

## HyperTransport address window

#### HyperTransport space

In the Loongson 3A4000 processor, the address window distribution of the default four HyperTransport interfaces is as follows: Table 14-5

|  |  |  |  |
| --- | --- | --- | --- |
| **Base address** | **End address** | **The size of the** | **define** |
| 0 x0a00\_0000\_0000 | 0 x0aff\_ffff\_ffff | One Tbytes | HT0\_LO window |
| 0 x0b00\_0000\_0000 | 0 x0bff\_ffff\_ffff | One Tbytes | HT0\_HI window |
| 0 x0e00\_0000\_0000 | 0 x0eff\_ffff\_ffff | One Tbytes | HT1\_LO window |
| 0 x0f00\_0000\_0000 | 0 x0fff\_ffff\_ffff | One Tbytes | HT1\_HI window |

By default (the system address window is not configured separately), the software accesses each HyperTransport interface according to the address space mentioned above. In addition, the software can also access each HyperTransport interface with other address Spaces by configuring the address window on the cross-switch (see section 3.3 for details). The distribution of address Windows in the internal 40-bit address space of each HyperTransport interface is shown in the table below.

Table 14-6 Address window distribution in the HyperTransport interface of Loongson 3 processor

|  |  |  |  |
| --- | --- | --- | --- |
| **Base address** | **End address** | **The size of the** | **define** |
| 0 x00\_0000\_0000 | 0 xfc\_ffff\_ffff | 1012 Gbytes | MEM space |
| 0 xfd\_0000\_0000 | 0 xfd\_f7ff\_ffff | 3968 Mbytes | reserve |
| 0 xfd\_f800\_0000 | 0 xfd\_f8ff\_ffff | 16 Mbytes | interrupt |
| 0 xfd\_f900\_0000 | 0 xfd\_f90f\_ffff | 1 Mbyte | PIC interrupt response |
| 0 xfd\_f910\_0000 | 0 xfd\_f91f\_ffff | 1 Mbyte | System information |
| 0 xfd\_f920\_0000 | 0 xfd\_faff\_ffff | 30 Mbytes | reserve |
| 0 xfd\_fb00\_0000 | 0 xfd\_fbff\_ffff | 16 Mbytes | HT controller configuration space |
| 0 xfd\_fc00\_0000 | 0 xfd\_fdff\_ffff | 32 Mbytes | I/O space |
| 0 xfd\_fe00\_0000 | 0 xfd\_ffff\_ffff | 32 Mbytes | HT bus configuration space |
| 0 xfe\_0000\_0000 | 0 xff\_ffff\_ffff | 8 Gbytes | reserve |

#### HyperTransport Controller internal window configuration

The HyperTransport interface of Loongson 3A4000 processor provides a variety of rich address Windows for users to use. The functions and functions of these address Windows are described in the following table.

Table 14-7 Address window provided in longson 3A4000 processor HyperTransport interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **The address window** | **Window number** | **Accept the bus** | **role** | **note** |
| Receiving window  See window configuration  14.5.10) | 3 | HyperTransport | Decide whether to accept an access issued on a HyperTransport bus. | When in master bridge mode (that is, act\_as\_slave in the configuration register is 0), only those accesses falling into these address Windows will be responded to by the internal bus, and other accesses will be considered as P2P accesses re-sent back to the HyperTransport bus; When in device mode (that is, act\_AS\_slave is 1 in the configuration register), only accesses falling into these address Windows are received and processed by the internal bus, and other accesses are given error returns according to the protocol. |
| The Post window  See window configuration  14.5.12) | 2 | Inside the bus | Determines whether Write access from the internal bus to the HyperTransport bus should be treated as Post Write | Outgoing calls that fall into these address Spaces will be treated as Post writes.  Post Write: In the HyperTransport protocol, this Write access does not require waiting for a Write response, that is, after the controller issues this Write access to the bus, the Write access to the processor completes the response. |

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prefetch window is available  See window configuration  14.5.13) | 2 | Inside the bus | Determines whether to receive internal Cache access fetcher. | When the processor core is executing out of order, some guess read or point access is made to the bus, which is wrong for some IO Spaces. By default, this  , access to the HT controller is returned directly without access to the HyperTransport bus. These Windows enable such access to the HyperTransport bus. |
| Uncache window  See window configuration  14.5.14) | 2 | HyperTransport | Decide whether to treat an access on a HyperTransport bus as an Uncache access to an inner part | IO DMA access within the Loongson 3A4000 processor will, by default, be accessed as Cache mode via SCache to determine whether a hit has been made, thus maintaining IO consistency. The configuration of these Windows enables access hit in these Windows to access memory directly in the manner of Uncache without maintaining its IO consistency information through hardware. |

## Configuration register

The configuration register module is mainly used to control the access request of configuration registers arriving from AXI SLAVE end or HT RECEIVER end, carry out external interrupt processing, and save the configuration registers visible to a large number of software for controlling various working modes of the system.

Firstly, the access and storage of configuration registers used to control various behaviors of HT controller are in this module. The access offset address of this module is 0xFD\_FB00\_0000 to 0xFD\_FBFF\_FFFF at the HT controller side. All visible registers of software in HT controller are shown in the following table:

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 x38 | | | Reserved | | | | | | | | | | |
| 0 x3c | | | Bridge Control | | | | | | Interrupt Pin | | | | Interrupt Line |
| Cap 0 PRI | 0 x40 | | | The Command | | | | | | "Capabilities Pointer | | | | Capability ID |
| 0 x44 | | | The Link Config 0 | | | | | | The Link Control 0 | | | | |
| 0 x48 | | | The Link Config 1 | | | | | | The Link Control 1 | | | | |
| 0 x4c | | | LinkFreqCap0 | | | | | | The Link Error0 / Link Freq  0 | | | | Revision ID |
| 0 x50 | | | LinkFreqCap1 | | | | | | The Link Error1 / Link Freq  1 | | | | Feature |
| 0 x54 | | | The Error Handling | | | | | | Enumeration  Scratchpad | | | |  |
| 0 x58 | | | Reserved | | | | | | Mem Limit Upper | | | | Mem Enable Upper |
|  |  | | |  | | |  | | |  | | | |  |
| Cap 1 Retry | 0 x60 | | | Capability Type | | | Reserved | | | Capability Pointer | | | | Capabiliter ID |
| 0 x64 | | | The Status of 1 | | | Control 1 | | | The Status of 0 | | | | The Control 0 |
| 0 x68 | | | Retry Count 1 | | | | | | Retry Count 0 | | | | |
| CAP 3 | 0 x6c | | | Capability Type | | | Revision ID | | | Capability Pointer | | | | Capabiliter ID |
| CAP 4  Interrupt | 0 x70 | | | Capability Type | | | The Index | | | Capability Pointer | | | | Capabiliter ID |
| 0 x74 | | | Dataport | | | | | | | | | | |
| 0 x78 | | | IntrInfo [31:0] | | | | | | | | | | |
| 0 x7c | | | IntrInfo [63:32] | | | | | | | | | | |
| Int the Vector | 0 x80 | | | INT the Vector [31:0] | | | | | | | | | | |
| 0 x84 | | | INT the Vector [63:32] | | | | | | | | | | |
| 0 x88 | | | INT the Vector (95-64) | | | | | | | | | | |
| 0 x8c | | | INT the Vector (127-96) | | | | | | | | | | |
| 0 x90 | | | INT the Vector (159-128) | | | | | | | | | | |
| 0 x94 | | | INT the Vector (191-160) | | | | | | | | | | |
| 0 x98 | | | INT the Vector (223-192) | | | | | | | | | | |
| 0 x9c | | | INT the Vector (255-224) | | | | | | | | | | |
| 0 xa0 | | | INT Enable [31:0] | | | | | | | | | | |
| 0 xa4 | | | INT Enable [63:32] | | | | | | | | | | |
| 0 xa8 | | | INT the Enable (95-64) | | | | | | | | | | |
| 0 xac | | | INT the Enable (127-96) | | | | | | | | | | |
| 0 xb0 | | | INT the Enable (159-128) | | | | | | | | | | |
| 0 xb4 | | | INT the Enable (191-160) | | | | | | | | | | |
| 0 xb8 | | | INT the Enable (223-192) | | | | | | | | | | |
| 0 XBC | | | INT the Enable (255-224) | | | | | | | | | | |
| CAP 5 | 0 xc0 | | | Capability Type | | | Cap Enum/Index | | | Capability Pointer | | | | Capabiliter ID |
| Gen3 | 0 xc4 | | | Global Link Training | | | | | | | | | | |
| 0 xc8 | | | Transmitter Configuration 0 | | | | | | | | | | |
| 0 XCC | | | The Receiver Configuration 0 | | | | | | | | | | |
| 0 xd0 | | | The Link Training 0 | | | | | | | | | | |
| 0 xd4 | | | Frequency Extension | | | | | | | | | | |
| 0 xd8 | | | Transmitter Configuration 1 | | | | | | | | | | |
| 0 XDC | | | The Receiver Configuration 1 | | | | | | | | | | |
| 0 xe0-0xfc | | | The Link Training 1 | | | | | | | | | | |
| 0 xe4 | | | BIST Control | | | | | | | | | | |
| The Enable | | 0 x100 | | | The Device ID | | | | Vendor ID | | | | | |
| 0 x104 | | | The Status | | | | The Command | | | | | |
| 0 x108 | | | The Class Code | | | | | | | | Revision ID | |
| 0 x10c | | | BIST | | | The Header Type | | | Latency Timer | | The Cache Line Size | |
| 0 x110 | | |  | | | | | | | | | |
| 0 x114 | | |  | | | | | | | | | |
| 0 x118 | | |  | | | | | | | | | |
| 0 x11c | | |  | | | | | | | | | |
| 0 x120 measures how | | |  | | | | | | | | | |
| 0 x124 | | |  | | | | | | | | | |
| 0 x128 | | | Cardbus CIS Pointer | | | | | | | | | |
| 0 x12c | | | Subsystem ID | | | | Subsystem Vendor ID | | | | | |
| 0 x130 | | | Expansion ROM Enable the Address | | | | | | | | | |
| 0 x134 | | | Reserved | | | | | | | "Capabilities Pointer | | |
| 0 x138 | | | Reserved | | | | | | | | | |
| 0 x13c | | | Bridge Control | | | | Interrupt Pin | | | Interrupt Line | | |
| The Receive Windows | | 0 x140 | | | HT RX Enable 0 | | | | | | | | | |
| 0 x144 | | | HT RX Mask0 | | | | | | | | | |
| 0 x148 | | | HT RX Enable 1 | | | | | | | | | |
| 0 x14c | | | HT RX Mask1 | | | | | | | | | |
| 0 x150 | | | HT RX Enable 2 | | | | | | | | | |
| 0 x154 | | | HT RX Mask2 | | | | | | | | | |
| 0 x158 | | | HT RX Enable 3 | | | | | | | | | |
| 0 x15c | | | HT RX Mask3 | | | | | | | | | |
| 0 x160 | | | HT RX Enable 4 | | | | | | | | | |
| 0 x164 | | | HT RX Mask4 | | | | | | | | | |
| The Header Trans | | 0 x168 | | | HT RX Header Trans | | | | | | | | | |
|  | | 0 x16c | | | HT RX EXT Header Trans | | | | | | | | | |
| Post Windows | | 0 x170 | | | HT TX Post Enable 0 | | | | | | | | | |
| 0 x174 | | | HT TX Post Mask0 | | | | | | | | | |
| 0 x178 | | | HT TX Post Enable 1 | | | | | | | | | |
| 0 x17c | | | HT TX Post Mask1 | | | | | | | | | |
| Prefetchable Windows | | 0 x180 | | | HT TX Prefetchable Enable 0 | | | | | | | | | |
| 0 x184 | | | HT TX Prefetchable Mask0 | | | | | | | | | |
| 0 x188 | | | HT TX Prefetchable Enable 1 | | | | | | | | | |
| 0 x18c | | | HT TX Prefetchable Mask1 | | | | | | | | | |
| Uncache Windows | | 0 x190 | | | HT RX Uncache Enable 0 | | | | | | | | | |
| 0 x194 | | | HT RX Uncache Mask0 | | | | | | | | | |
| 0 x198 | | | HT RX Uncache Enable 1 | | | | | | | | | |
| 0 x19c | | | HT RX Uncache Mask1 | | | | | | | | | |
| 0 x1a0 | | | HT RX Uncache Enable 2 | | | | | | | | | |
| 0 x1a4 | | | HT RX Uncache Mask2 | | | | | | | | | |
| 0 x1a8 | | | HT RX Uncache Enable 3 | | | | | | | | | |
| 0 x1ac | | | HT RX Uncache Mask3 | | | | | | | | | |
| Peer-to-peer (P2P)  Windows | | 0 x1b0 | | | HT RX P2P Enable 0 | | | | | | | | | |
| 0 x1b4 | | | HT RX P2P Mask0 | | | | | | | | | |
| 0 x1b8 | | | HT RX P2P Enable 1 | | | | | | | | | |
| 0 x1bc | | | HT RX P2P Mask1 | | | | | | | | | |
| The APP  The Config | | 0 x1c0 | | | APP Configuration 0 | | | | | | | | | |
| 0 x1c4 | | | APP Configuration 1 | | | | | | | | | |
| 0 x1c8 | | | RX Bus Value | | | | | | | | | |
| 0 x1cc | | | PHY status | | | | | | | | | |
| Buffer | | 0 x1d0 | | | The TX Buffer 0 | | | | | | | | | |
| 0 x1d4 | | | TX Buffer 1 / Rx Buffer HI | | | | | | | | | |
| 0 x1d8 | | | The TX Buffer turning | | | | | | | | | |
| 0 x1dc | | | RX Buffer lo | | | | | | | | | |
| Training | | 0 x1e0 | | | Short Training 0 Counter | | | | | | | | | |
| 0 x1e4 | | | Long Training 0 Counter | | | | | | | | | |
| 0 x1e8 | | | Training 1 Counter | | | | | | | | | |
| 0 x1ec | | | Training 2 Counter | | | | | | | | | |
| 0 x1f0 | | | Training 3 Counter | | | | | | | | | |
| PLL | | 0 x1f4 | | | PLL Configuration | | | | | | | | | |
| PHY | | 0 x1f8 | | | IO Configuration | | | | | | | | | |
| 0 x1fc | | | PHY Configuration | | | | | | | | | |
| The DEBUG | | | 0 x240 | | | HT3 DEBUG 0 | | | | | | | | |
| 0 x244 | | | HT3 DEBUG 1 | | | | | | | | |
| 0 x248 | | | HT3 DEBUG 2 | | | | | | | | |
| 0 x24c | | | HT3 DEBUG 3 | | | | | | | | |
| 0 x250 | | | HT3 DEBUG 4 | | | | | | | | |
| 0 x254 | | | 5 HT3 DEBUG | | | | | | | | |
| 0 x258 | | | HT3 DEBUG 6 | | | | | | | | |
| POST ID WINDOWS | | | 0 x260 | | | HT TX POST ID WIN0 | | | | | | | | |
| 0 x264 | | | HT TX POST ID WIN1 | | | | | | | | |
| 0 x268 | | | HT TX POST ID WIN2 | | | | | | | | |
| 0 x26c | | | HT TX POST ID WIN3 | | | | | | | | |
| POST ID WINDOWS | | | 0 x270 | | | INT TRANS WIN lo | | | | | | | | |
| 0 x274 | | | INT TRANS WIN hi | | | | | | | | |

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

#### Bridge Control

Offset: 0x3C

Reset value: 0x00000000

Name: Bus Reset Control

Table 14-8 Definition of Bus Reset Control register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| For calamity | Reserved | 9 | 0 x0 |  | reserve |
| 22 | The Reset | 1 | 0 x0 | R/W | Bus reset control:   1. >1: HT\_RSTn set 0, bus reset 2. >0: HT\_RSTn set 1, bus unreset |
| 21:0 | Reserved | 22 | 0 x0 |  | reserve |

#### Capability Registers

Offset: 0x40

Reset value: 0x20010008

Name: Command, Capabilities Pointer, Capability ID

Table 14-9 Command, Capabilities Pointer, Capability ID register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| take | Slave/Pri | 3 | 0 x0 | R | The Command format is HOST/Sec |
| Yea, | Reserved | 2 | 0 x0 | R | reserve |
| " | The Unit Count | 5 | 0 x0 | R/W | Provide to the software to record the current number of units |
| " | The Unit ID | 5 | 0 x0 |  | HOST mode: can be used to record the number of ID used in SLAVE mode: record itself Unit ID HOST/SLAVE mode sent by act\_AS\_slave  Control register |
| 15:08 | "Capabilities Pointer | 8 | 0 x60 | R | The next Cap register offset address |
| away | Capability ID | 8 | 0 x08 | R | HyperTransport capability ID |

Offset: 0x44

Reset value: 0x00112000

Name: Link Config, Link Control

Table 14-10 Link Config, Link Control register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| he | The Link Width Out | 3 | 0 x0 | R/W | Sending end width  The value after a cold reset is the maximum width of the current connection. The value written into this register will take effect after the next hot reset or HT Disconnect in a 000:8-bit mode  001:16 bit mode |
| 27 | Reserved | 1 | 0 x0 |  | reserve |
| they | The Link Width In | 3 | 0 x0 | R/W | Receiver width  The value after cold reset is the maximum width of the current connection. The value written to this register will take effect after the next hot reset or HT Disconnect |
| 23 | Dw Fc out | 1 | 0 x0 | R | The sender does not support two-word streaming |
| Lift up | Max Link Width out | 3 | 0 x1 | R | Maximum width of HT bus sender: 16bits |
| 19 | Dw Fc In | 1 | 0 x0 | R | Two-word streaming is not supported on the receiving end |
| thou | Max Link Width In | 3 | 0 x1 | R | Maximum width of HT bus receiver: 16bits |
| The lowest | Reserved | 2 | 0 x0 |  | reserve |
| 13 | LDTSTOP#  Tristate Enable | 1 | 0 x1 | R/W | When the HT bus enters THE HT Disconnect state, does it close the HT PHY  1: closed  0: Not closed |
| 12:10 | Reserved | 3 | 0 x0 |  | reserve |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9 | CRC Error (hi) | 1 | 0 x0 | R/W | CRC errors occurred in high 8 bits |
| 8 | CRC Error (lo) | 1 | 0 x0 | R/W | CRC errors occurred in low 8 bits |
| 7 | Trans off | 1 | 0 x0 | R/W | HT PHY shutdown control  When in 16-bit bus mode  1: Closed HIGH/low 8-bit HT PHY  0: The lower 8-bit HT PHY and the higher 8-bit HT PHY are controlled by BIT 0 |
| 6 | The End of the Chain | 0 | 0 x0 | R | HT bus terminal |
| 5 | Init Complete | 1 | 0 x0 | R | HT bus initialization is complete |
| 4 | The Link Fail | 1 | 0 x0 | R | Indicates connection failure |
| 3:2 | Reserved | 2 | 0 x0 |  | reserve |
| 1 | CRC Flood Enable | 1 | 0 x0 | R/W | When CRC error occurs, whether flood HT bus |
| 0 | Trans off (hi) | 1 | 0 x0 | R/W | When the 16-bit HT bus is used to run the 8-bit protocol, the high-8-bit PHY is switched off  1: High 8-bit HT PHY was closed  0: Elevating 8-bit HT PHY |

Offset: 0x4C

Reset value: 0x80250023

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

Table 14-11 Revision ID, Link Freq, Link Error, Link Freq Cap register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | The Link Freq Cap | 16 | 0 x0000 | R | The supported HT bus frequency produces different values depending on the setting of the external PLL (when using software configuration PLL)  When (0x1F4), the bit is meaningless.  {3.2 G and 2.6 G, 2.4 G and 2.2 G, 2.0 G and 1.8 G, 1.6 G and 1.4 G, 1.2 G and 1.0 G, 800 M, 600 M, 500 M, 400 M, 300 M, 200 M} |
| The lowest | Reserved | 2 | 0 x0 |  | reserve |
| 13 | Over Flow Error | 1 | 0 x0 | R | HT bus package overflow |
| 12 | Protocol Error | 1 | 0 x0 | R/W | Protocol error when an unrecognized error is received on the HT bus  The command |
| and | The Link Freq | 4 | 0 x0 | R/W | HT bus working frequency. The value written into this register will take effect after the next hot reset or HT Disconnect. The set value corresponds to the Link Freq Cap bit  When using the software configuration PLL (0x1F4), the  Bit nonsense) |
| away | Revision ID | 8 | 0 x60 | R/W | Version number: 3.0 |

Offset: 0x50

Reset value: 0x00000002

Name: Feature Capability

Table 14-12 Definition of Feature Capability register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Is wasted | Reserved | 23 | 0 x0 |  | reserve |
| 8 | Extended the Register | 1 | 0 x0 | R | There is no |
| The log | Reserved | 3 | 0 x0 |  | reserve |
| 3 | Extended CTL Time | 1 | 0 x0 | R | Don't need |
| 2 | CRC Test Mode | 1 | 0 x0 | R | Does not support |
| 1 | LDTSTOP# | 1 | 0 x1 | R | Support LDTSTOP# |
| 0 | Isochronous Mode | 1 | 0 x0 | R | Does not support |

#### Error Retry controls the register

For error retransmission enable in HyerTransport 3.0 mode, configure the maximum number of Short Retry counts to indicate whether the Retry counter is flipped.

Offset: 0x64

Reset value: 0x00000000

Name: Error Retry control register

Table 14-13 Error Retry control register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| for | Reserved | 22 | 0 x0 | R | reserve |
| 9 | Retry Count Rollover | 1 | 0 x0 | R | The Retry counter flips its count |
| 8 | Reserved | 1 | 0 x0 | R | reserve |
| but | Short Retry Attempts | 2 | 0 x0 | R/W | Maximum number of Short retries allowed |
| 5-1 | Reserved | 5 | 0 x0 | R |  |
| 0 | The Link Retry the Enable | 1 | 0 x0 | R/W | Error reconnection enabled |

#### The Retry Count register

Used for error retransmission counting in HyerTransport 3.0 mode.

Name: Retry Count register

Table 14-14 Retry Count register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Reserved | 16 | 0 x0 | R | reserve |
| 15:0 | Retry Count | 16 | 0 x0 | R | Retry count |

#### Revision ID register

Used to configure the controller version to a new version number that takes effect via Warm Reset. Offset: 0x6C

Reset value: 0x00200000

Name: RevisionID register

Table 14-15 Revision ID register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| came | Reserved | 8 | 0 x0 | R | reserve |
| Ephron; | Revision ID | 8 | 0 x20 | R/W | Revision ID control register  0 x20: HyperTransport 1.00  0 x60: HyperTransport 3.00 |
| 15:0 | Reserved | 16 | 0 x0 | R | reserve |

#### Interrupt Discovery & Configuration

Offset: 0x70

Reset value: 0x80000008

Name: Interrupt Capability

Table 14-16 Interrupt register definitions

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

Offset: 0x74

Reset value: 0x00000000

Name: Dataport

Table 14-17 Dataport register definitions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Dataport | 32 | 0 x0 | R/W | This register reads and writes when the previous register Index is 0x10  The result is the 0xA8 register, otherwise 0xAC |

Offset: 0x78

Reset value: 0xF8000000

Name: IntrInfo [31:0]

Table 14-18 Definition of IntrInfo Register (1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| came | IntrInfo [came] | 8 | 0 xf8 | R | reserve |
| Isle; | IntrInfo [great] | 22 | 0 x0 | R/W | IntrInfo[23:2], when PIC interrupt is emitted, the value of IntrInfo is used to represent the interrupt vector |
| 1-0 | Reserved | 2 | 0 x0 | R | reserve |

Offset: 0x7c

Reset value: 0x00000000

Name: IntrInfo [63:32]

Table 14-19 Definition of IntrInfo Register (2)

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

#### Interrupt vector register

Interrupt vector register a total of 256, including removal of HT bus Fix, interrupt Arbiter and PIC 256 map directly to the interrupt vector, other plants, such as SMI, NMI, INIT, INTA, intb.br deal, a steady, 0 x50 [doth INTD can register mapped to any one of the eight interrupt vector, the order of the map for {INTD, steady, intb.br deal, INTA, 1 'b0, INIT, NMI, SMI}. At this point, the corresponding value of Interrupt vector is {Interrupt Index, internal vector [2:0]}.

By default, 256-bit interrupts can be distributed to 4-bit interrupts. Interrupt without using high 8 bit HT controller

, you can also distribute 256-bit interrupts to 8-bit interrupt lines by setting ht\_int\_8bit.

The 256 interrupt vectors are mapped to different interrupt lines by selecting different register configurations according to the interrupt routing mode. The specific mapping mode is as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The interrupt number | Strip | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4  X = [63:0] | 1 | [X] | [X + 64] | [X + 128] | [X + 192] | - | - | - | - |
| 2 | [2 x] | [2 x + 1] | [2 x + 128] | [2 x + 129] | - | - | - | - |
| 4 | (4 x) | [4 x + 1] | [4 + 2 x] | [4 + 3 x] | - | - | - | - |
| 8 | 1 | [X] | [Y] | [X + 64] | [Y + 64] | [X + 128] | [Y + 128] | [X + 192] | [Y + 192] |
| X = [31:0] | 2 | [2 x] | [2] y | [2 x + 1] | [2 + 1] y | [2 x + 128] | [2 + 128] y | [2 x + 129] | [2 + 129] y |
| Y = [63:32] | 4 | (4 x) | [4 x + 32] | [4 x + 1] | [33] 4 x + | [4 + 2 x] | [4 x + 34] | [4 + 3 x] | [4 x + 35] |

Take the example of using a 4-bit disconnection, the different mappings are as follows. Ht\_int\_stripe\_1:

[0,1,2,3... 63] corresponds to neutral 0 /HT HI corresponds to neutral 4

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

[128,129,130,131... 191] corresponds to median 2 /HT HI corresponds to median 6

[192,193,194,195... 255] median 3 /HT HI median 7 ht\_int\_stripe\_2:

[0,2,4,6... 126] corresponds to neutral 0 /HT HI corresponds to neutral 4

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

[128,130,132,134...... 254] corresponds to medium break 2 /HT HI corresponds to medium break 6

[129,131,133,135... 255] The median break 3 /HT HI the median break 7 ht\_int\_stripe\_4:

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

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

[2,6,10,14... 254] corresponding medium break line 2 /HT HI corresponding medium break line 6

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

The following interrupt vector description corresponds to HT\_int\_stripe\_1, and the other two modes can be obtained from the above description.

Offset: 0x80

Reset value: 0x00000000

HT Bus Interrupt Vector Register [31:0]

Table 14-20 HT Bus Interrupt Vector register Definition (1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [31:0] | 32 | 0 x0 | R/W | HT bus interrupt vector register [31:0],  So this is 0 over HT HI and this is 4 |

Offset: 0x84

Reset value: 0x00000000

HT Bus Interrupt Vector Register [63:32]

Table 14-21 HT Bus Interrupt Vector register Definition (2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [63:32] | 32 | 0 x0 | R/W | HT bus interrupt vector register [63:32],  So this is 0 over HT HI and this is 4 |

Offset: 0x88

Reset value: 0x00000000

HT Bus Interrupt Vector Register [95:64]

Table 14-22 HT Bus Interrupt Vector Register Definition (3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [95-64] | 32 | 0 x0 | R/W | HT bus interrupt vector register [95:64],  It's 1 over HT HI, it's 5 |

Offset: 0x8c

Reset value: 0x00000000

HT Bus Interrupt Vector Register [127:96]

Table 14-23 HT Bus Interrupt Vector register Definition (4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [127-96] | 32 | 0 x0 | R/W | HT bus interrupt vector register [127:96],  It's 1 over HT HI, it's 5 |

Offset: 0x90

Reset value: 0x00000000

HT Bus Interrupt Vector Register [159:128]

Table 14-31 HT Bus Interrupt Vector register Definition (5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [159-128] | 32 | 0 x0 | R/W | HT bus interrupt Vector register [159:128],  So this is 2 over HT HI and this is 6 |

Offset: 0x94

Reset value: 0x00000000

HT Bus Interrupt Vector Register [191:160]

Table 14-24 HT Bus Interrupt Vector register Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [191-160] | 32 | 0 x0 | R/W | HT bus interrupt vector register [191:160],  So this is 2 over HT HI and this is 6 |

Offset: 0x98

Reset value: 0x00000000

HT Bus Interrupt Vector Register [223:192]

Table 14-25 HT Bus Interrupt Vector Register Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [223-192] | 32 | 0 x0 | R/W | HT bus interrupt vector register [223:192],  It's 3 over HT HI, it's 7 |

Offset: 0x9c

Reset value: 0x00000000

HT Bus Interrupt Vector Register [255:224]

Table 14-26 HT Bus Interrupt Vector register Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_case  [255-224] | 32 | 0 x0 | R/W | HT bus interrupt vector register [255:224],  It's 3 over HT HI, it's 7 |

#### Interrupt enable register

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

The 256 interrupt vectors are mapped to different interrupt lines by selecting different register configurations according to the interrupt routing mode. The specific mapping mode is as follows:

Ht\_int\_stripe\_1:

[0,1,2,3... 63] corresponds to neutral 0 /HT HI corresponds to neutral 4

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

[128,129,130,131... 191] corresponds to median 2 /HT HI corresponds to median 6

[192,193,194,195... 255] median 3 /HT HI median 7 ht\_int\_stripe\_2:

[0,2,4,6... 126] corresponds to neutral 0 /HT HI corresponds to neutral 4

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

[128,130,132,134...... 254] corresponds to medium break 2 /HT HI corresponds to medium break 6

[129,131,133,135... 255] The median break 3 /HT HI the median break 7 ht\_int\_stripe\_4:

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

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

[2,6,10,14... 254] corresponding medium break line 2 /HT HI corresponding medium break line 6

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

The following interrupt vector description corresponds to HT\_int\_stripe\_1, and the other two modes can be obtained from the above description.

Offset: 0xa0

Reset value: 0x00000000

HT Bus Interrupt Enable Register [31:0]

Table 14-27 HT Bus Interrupt Enabled Register Definition (1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [31:0] | 32 | 0 x0 | R/W | HT bus interrupt enable register [31:0],  So this is 0 over HT HI and this is 4 |

Offset: 0xA4

Reset value: 0x00000000

HT Bus Interrupt Enable Register [63:32]

Table 14-28 HT Bus Interrupt Enabled Register Definition (2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [63:32] | 32 | 0 x0 | R/W | HT bus interrupt enable register [63:32],  So this is 0 over HT HI and this is 4 |

Offset: 0xa8

Reset value: 0x00000000

HT Bus Interrupt Enable Register [95:64]

Table 14-29 HT Bus Interrupt Enabled Register Definition (3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [95-64] | 32 | 0 x0 | R/W | HT bus interrupt enable register [95:64],  It's 1 over HT HI, it's 5 |

Offset: 0xAC

Reset value: 0x00000000

HT Bus Interrupt Enable Register [127:96]

Table 14-30 HT Bus Interrupt Enabled Register Definition (4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [127-96] | 32 | 0 x0 | R/W | HT bus interrupt enable register [127:96],  It's 1 over HT HI, it's 5 |

Offset: 0xb0

Reset value: 0x00000000

HT Bus Interrupt Enable Register [159:128]

Table 14-31 HT Bus Interrupt Enabled Register Definition (5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [159-128] | 32 | 0 x0 | R/W | HT bus interrupt enable register [159:128],  So this is 2 over HT HI and this is 6 |

Offset: 0xb4

Reset value: 0x00000000

HT Bus Interrupt Enable Register [191:160]

Table 14-32 HT Bus Interrupt Enabled Register Definition (6)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [191-160] | 32 | 0 x0 | R/W | HT bus interrupt enable register [191:160],  So this is 2 over HT HI and this is 6 |

Offset: 0xb8

Reset value: 0x00000000

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

Table 14-33 HT Bus Interrupt Enabled Register Definition (7)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [223-192] | 32 | 0 x0 | R/W | HT bus interrupt enable register [223:192],  It's 3 over HT HI, it's 7 |

Offset: 0xBC

Reset value: 0x00000000

HT Bus Interrupt Enable Register [255:224]

Table 14-34 HT Bus Interrupt Enabled Register Definition (8)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Interrupt\_mask  [255-224] | 32 | 0 x0 | R/W | HT bus interrupt enable register [255:224],  It's 3 over HT HI, it's 7 |

#### Link Train register

HyperTransport 3.0 Link initialization and Link Training Control Register. Offset: 0xD0

Reset value: 0x00000070

Name: Link Train register

Table 14-35 Link Train registers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| For calamity | Reserved | 9 | 0 x0 | R | reserve |
| "You | Transmitter LS the select | 2 | 0 x0 | R/W | Link state on the sending side in Disconnected or Inactive state:  2 'b00 LS1  2 'b01 LS0  2 'b10 LS2  2 'bl1 you |
| 14 | DsiableCmd Throttling | 1 | 0 x0 | R/W | In HyperTransport 3.0 mode, only one non-info CMD can appear in any four consecutive DWS by default.  1 'b0 enabled Cmd Throttling  1 'B1 prohibits the use of Cmd Throttling |
| " | Reserved | 4 | 0 x0 | R | reserve |
| " | Receiver LS the select | 2 | 0 x0 | R/W | Link state on the receiving end in a Disconnected or Inactive state:  2 'b00 LS1  2 'b01 LS0  2 'b10 LS2  2 'bl1 you |
| 6:4 | Long Retry Count | 3 | 0 x7 | R/W | Maximum number of times Long Retry |
| 3 | Scrambling the Enable | 1 | 0 x0 | R/W | (3) : to misuse or deprive of health care  1: can Scramble |
| 2 | 8 b10b Enable | 1 | 0 x0 | R/W | Whether to enable 8B10B 0: Ban 8B10B  1: can make 8 b10b |
| 1 | AC | 1 | 0 x0 | R | Is AC Mode detected  AC Mode 1: AC Mode was detected |
| 0 | Reserved | 1 | 0 x0 | R | reserve |

#### The receive address window configures registers

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

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

Addr\_out\_trans = TRANS\_EN? TRANS | ADDR & ~MASK: ADDR addr\_out = Multi\_node\_en?

Addr\_out\_trans addr\_out\_trans [39:37], [43:40], 3 'b0, addr\_out [36:0] : addr\_out\_trans;

It should be noted that when configuring the address window register, the MASK should be all 1 high and all 0 low. The actual number of zeros in MASK represents the size of the address window.

The address of the receiving address window is the address received on the HT bus. HT address in the P2P window will be forwarded back to THE HT bus as a P2P command, HT address in the normal receiving window and not in the P2P window will be sent to the CPU, and commands at other addresses will be forwarded back to the HT bus as a P2P command.

Offset: 0x140

Reset value: 0x00000000

HT Bus Receive Address Window 0 enable (external access)

Table 14-36 HT bus receive address window 0 enable (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_image0\_en | 1 | 0 x0 | R/W | HT bus receives address window 0, enabling signal |
| 30 | ht\_rx\_image0\_trans\_en | 1 | 0 x0 | R/W | HT bus receives address window 0, mapping enabled signal |
| 29 | ht\_rx\_image0\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus receives address window 0, multi-node address mapping enables [39:37] to be converted to [46:44] |
| 28 | ht\_rx\_image0\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus receives address window 0, protocol address hit enable  Must be s |
| 25:0 | Ht\_rx\_image0\_trans [prey] | 26 | 0 x0 | R/W | HT bus receives address window 0, mapped address [49:24] |

Offset: 0x144

Reset value: 0x00000000

HT Bus Receiving Address Window 0 Base address (external access)

Table 14-37 HT bus receive address window 0 base address (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_image0\_base [they] | 16 | 0 x0 | R/W | HT bus receives address window 0, address base address [39:24] |
| 15:0 | Ht\_rx\_image0\_mask [they] | 16 | 0 x0 | R/W | HT bus receiving address window 0, address shielded [39:24] |

Offset: 0x148

Reset value: 0x00000000

HT Bus Receive Address Window 1 enabling (external access)

Table 14-38 HT bus receiver address window 1 enables (externally accessible) register definitions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_image1\_en | 1 | 0 x0 | R/W | HT bus receives address window 1, enabling signal |
| 30 | ht\_rx\_image1\_trans\_en | 1 | 0 x0 | R/W | HT bus receives address window 1, mapping enabled signal |
| 29 | ht\_rx\_image1\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus receives address window 1, multi-node address mapping enables [39:37] to be converted to [46:44] |
| 28 | ht\_rx\_image1\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus receives address window 1, protocol address hit enable must be set 0 |
| 25:0 | Ht\_rx\_image1\_trans [prey] | 26 | 0 x0 | R/W | HT bus receives address window 1, mapped address [49:24] |

Offset: 0x14c

Reset value: 0x00000000

HT Bus Receiving Address Window 1 Base Address (external access)

Table 14-39 HT bus receiving Address window 1 Base address (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_image1\_base [they] | 16 | 0 x0 | R/W | HT bus receives address window 1, address base address [39:24] |
| 15:0 | Ht\_rx\_image1\_mask [they] | 16 | 0 x0 | R/W | HT bus receiving address window 1, address shielded [39:24] |

Offset: 0x150

Reset value: 0x00000000

HT Bus Receive Address Window 2 enable (external access)

Table 14-40 HT bus receive address window 2 enable (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_image2\_en | 1 | 0 x0 | R/W | HT bus receives address window 2, enabling signal |
| 30 | ht\_rx\_image2\_trans\_en | 1 | 0 x0 | R/W | HT bus receives address window 2, mapping enabled signal |
| 29 | ht\_rx\_image2\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus receives address window 2, multi-node address mapping enables [39:37] to be converted to [46:44] |
| 28 | ht\_rx\_image2\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus receives address window 2, protocol address hit enable must be set 0 |
| 25:0 | Ht\_rx\_image2\_trans [prey] | 26 | 0 x0 | R/W | HT bus receives address window 2, mapped address [49:24] |

Offset: 0x154

Reset value: 0x00000000

HT Bus Receiving Address Window 2 Base Address (external access)

Table 14-41 HT bus receiver address window 2 Base address (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_image2\_base [they] | 16 | 0 x0 | R/W | HT bus receives address window 2, address base address [39:24] |
| 15:0 | Ht\_rx\_image2\_mask [they] | 16 | 0 x0 | R/W | HT bus receiving address window 2, address shielded [39:24] |

Offset: 0x158

Reset value: 0x00000000

HT Bus Receive Address Window 3 enable (external access)

Table 14-42 HT bus receive address window 3 enable (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_image3\_en | 1 | 0 x0 | R/W | HT bus receives address window 3, enabling signal |
| 30 | ht\_rx\_image3\_trans\_en | 1 | 0 x0 | R/W | HT bus receives address window 3, mapping enabled signal |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 29 | ht\_rx\_image3\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus receives address window 3, enabling multi-node address mapping  Convert address [39:37] to [46:44] |
| 28 | ht\_rx\_image3\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus receives address window 3, protocol address hit enable must be set 0 |
| 25:0 | Ht\_rx\_image3\_trans [prey] | 26 | 0 x0 | R/W | HT bus receives address window 3, mapped address [49:24] |

Offset: 0x15C

Reset value: 0x00000000

HT Bus Receiving Address Window 3 Base Address (external access)

Table 14-43 HT bus receiver address window 3 Base address (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_image3\_base [they] | 16 | 0 x0 | R/W | HT bus receive address window 3, address base address [39:24] |
| 15:0 | Ht\_rx\_image3\_mask [they] | 16 | 0 x0 | R/W | HT bus receiving address window 3, address shielded [39:24] |

Offset: 0x160

Reset value: 0x00000000

HT Bus Receive Address Window 4 enable (external access)

Table 14-44 HT bus receive address window 4 enable (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_image4\_en | 1 | 0 x0 | R/W | HT bus receives address window 4, enabling signal |
| 30 | ht\_rx\_image4\_trans\_en | 1 | 0 x0 | R/W | HT bus receives address window 4, mapping enabled signal |
| 29 | ht\_rx\_image4\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus receives address window 4, enabling multi-node address mapping  Convert address [39:37] to [46:44] |
| 28 | ht\_rx\_image4\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus receives address window 4, protocol address hit enable must be set 0 |
| 25:0 | Ht\_rx\_image4\_trans [prey] | 26 | 0 x0 | R/W | HT bus receives address window 4, mapped address [49:24] |

Offset: 0x164

Reset value: 0x00000000

HT Bus Receiving Address Window 4 Base Address (external access)

Table 14-45 HT bus receiver address window 4 Base address (external access) register definition

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_image4\_base [they] | 16 | 0 x0 | R/W | HT bus receives address window 4, address base address [39:24] |
| 15:0 | Ht\_rx\_image4\_mask [they] | 16 | 0 x0 | R/W | HT bus receiving address window 4, address shielded [39:24] |

#### Configure the space conversion register

Used for various transformations to the HT configuration space.

Offset: 0x168

Reset value: 0x00000000

Name: Configuration space extension address translation

Table 14-46 Configuration space extension address translation register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_header\_trans\_ext | 1 | 0 x1 | R/W | Adjust the type1 flag bit from 24 to 28 after converting the configuration space (0xFD\_FE000000) for use with the EXT HEADER space  unified |
| 30 | ht\_rx\_header\_trans\_en | 1 | 0 x1 | R/W | Enable configuration space (0xFD\_FE000000)  The high address ([39:24]) is converted |
| 29:0 | Ht\_rx\_header\_trans [53:24] | 30 | 0 xfe00 | R/W | High address after configuration space conversion [53:24]  (Actually only [53:25] is available) |

Offset: 0x16C

Reset value: 0x00000000

Name: Extended address translation

Table 14-47 Extended address translation register definitions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 30 | ht\_rx\_ext\_header\_trans\_en | 1 | 0 x0 | R/W | Enable to expand the allocation of space  The high address of (0xFE\_00000000)  ([39:28]) conversion |
| 29:0 | Ht\_rx\_ext\_header\_trans [53:24] | 30 | 0 x0 | R/W | The extended configuration space is converted to a higher address  [53:24] (actually only [53:29] is available) |

#### The POST address window configures registers

See Section 14.5.10 for the hit formula of address window.

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

Offset: 0x170

Reset value: 0x00000000

HT Bus POST Address window 0 enabled (internal access)

Table 14-48 HT bus POST address window 0 enable (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_post0\_en | 1 | 0 x0 | R/W | HT bus POST address window 0, enabling signal |
| 30 | ht\_split0\_en | 1 | 0 x0 | R/W | HT access unpacking enable (corresponding to the external of the CPU core  Uncache ACC Operation Window) |
| throne | Reserved | 14 | 0 x0 |  | reserve |
| 15:0 | Ht\_post0\_trans [they] | 16 | 0 x0 | R/W | HT bus POST address window 0, translated address [39:24] |

Offset: 0x174

Reset value: 0x00000000

HT Bus POST Address window 0 Base address (internal access)

Table 14-49 HT bus POST Address window 0 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_post0\_base [they] | 16 | 0 x0 | R/W | HT bus POST address window 0, address base address [39:24] |
| 15:0 | Ht\_post0\_mask [they] | 16 | 0 x0 | R/W | HT bus POST address window 0, address shielded [39:24] |

Offset: 0x178

Reset value: 0x00000000

HT Bus POST Address Window 1 enable (internal access)

Table 14-50 HT bus POST Address Window 1 enable (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_post1\_en | 1 | 0 x0 | R/W | HT bus POST address window 1, enabling signal |
| 30 | ht\_split1\_en | 1 | 0 x0 | R/W | HT access unpacking enable (corresponding to the external of the CPU core  Uncache ACC Operation Window) |
| Was a | Reserved | 14 | 0 x0 |  | reserve |
| 15:0 | Ht\_post1\_trans [they] | 16 | 0 x0 | R/W | HT bus POST address window 1, translated address [39:24] |

Offset: 0x17c

Reset value: 0x00000000

HT Bus POST Address Window 1 Base address (internal access)

Table 14-51 HT Bus POST Address Window 1 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_post1\_base [they] | 16 | 0 x0 | R/W | HT bus POST address window 1, address base address [39:24] |
| 15:0 | Ht\_post1\_mask [they] | 16 | 0 x0 | R/W | HT bus POST address window 1, address shielded [39:24] |

#### The prefetch address window configures registers

See Section 14.5.10 for the hit formula of address window.

The address of this window is the address received on a AXI bus. The fetch instruction and CACHE access in this window will be sent to THE HT bus. Other fetch instruction or CACHE access will not be sent to the HT bus, but will be returned immediately. If it is a read command, the corresponding number of invalid read data will be returned.

Offset: 0x180

Reset value: 0x00000000

HT Bus Preaddressable window 0 enabling (internal access)

Table 14-52 HT Bus Prefetch Address Window 0 enable (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_prefetch0\_en | 1 | 0 x0 | R/W | HT bus can prefetch address window 0, enabling signal |
| and | Reserved | 15 | 0 x0 |  | reserve |
| 15:0 | Ht\_prefetch0\_trans [they] | 16 | 0 x0 | R/W | HT bus prefetchable address window 0, translated address [39:24] |

Offset: 0x184

Reset value: 0x00000000

HT Bus Preaddressable window 0 Base address (internal access)

Table 14-53 HT Bus Prefetch address window 0 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_prefetch0\_base [they] | 16 | 0 x0 | R/W | HT bus prefetchable address window 0, address base address of [39:24]  An address |
| 15:0 | Ht\_prefetch0\_mask [they] | 16 | 0 x0 | R/W | HT bus prefetchable address window 0, address shielded [39:24] |

Offset: 0x188

Reset value: 0x00000000

HT Bus Preaddressable Window 1 enabling (internal access)

Table 14-54 HT Bus Preaddressable window 1 enabling (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_prefetch1\_en | 1 | 0 x0 | R/W | HT bus can prefetch address window 1, enabling signal |
| and | Reserved | 15 | 0 x0 |  | reserve |
| 15:0 | Ht\_prefetch1\_trans [they] | 16 | 0 x0 | R/W | HT bus can prefetch address window 1, after translation address  [they] |

Offset: 0x18c

Reset value: 0x00000000

HT Bus Preaddressable window 1 Base address (internal access)

Table 14-55 HT Bus Prefetch Address Window 1 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_prefetch1\_base [they] | 16 | 0 x0 | R/W | HT bus prefetchable address window 1, address base address [39:24] |
| 15:0 | Ht\_prefetch1\_mask [they] | 16 | 0 x0 | R/W | HT bus prefetchable address window 1, address shielded [39:24] |

#### The UNCACHE Address window configures registers

See Section 14.5.10 for the hit formula of address window.

The address of 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 SCACHE, nor will they invalidate a primary CACHE, but will be sent directly to memory or other address space, meaning that the read and write commands in this window will not maintain IO CACHE consistency. This window is mainly aimed at some operations that will not hit in the CACHE, so it can improve the access efficiency, such as video memory access.

Offset: 0x190

Reset value: 0x00000000

Name: HT Bus Uncache Address window 0 enabled (internal access)

Table 14-56 HT Bus Uncache Address Window 0 enabled (internal access)

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_uncache0\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 0, enabling signal |
| 30 | ht\_uncache0\_trans\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 0, map enabling signal |
| 29 | ht\_uncache0\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 0, enabling multi-node address mapping |
| 28 | ht\_uncache0\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 0, protocol address  Hit can make |
| 25:0 | Ht\_uncache0\_trans [prey] | 26 | 0 x0 | R/W | HT Bus UNCache address window 0, address after translation [49:24] |

Offset: 0x194

Reset value: 0x00000000

Name: HT Bus Uncache Address window 0 Base address (internal access)

Table 14-57 HT Bus Uncache Address Window 0 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_uncache0\_base [they] | 16 | 0 x0 | R/W | HT bus uncache address window 0, address base address [39:24] |
| 15:0 | Ht\_uncache0\_mask [they] | 16 | 0 x0 | R/W | HT bus uncache address window 0, address shielded [39:24] |

Offset: 0x198

Reset value: 0x00000000

Name: HT Bus Uncache Address Window 1 enabled (internal access)

Table 14-58 HT Bus Uncache Address Window 1 enabling (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_uncache1\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 1, enabling signal |
| 30 | ht\_uncache1\_trans\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 1, map enabling signal |
| 29 | ht\_uncache1\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 1, enabling multi-node address mapping |
| 28 | ht\_uncache1\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 1, protocol address  Hit can make |
| 25:0 | Ht\_uncache1\_trans [prey] | 26 | 0 x0 | R/W | HT Bus UNCache Address window 1, address after translation [49:24] |

Offset: 0x19c

Reset value: 0x00000000

Name: HT Bus Uncache Address Window 1 Base address (internal access)

Table 14-59 HT Bus Uncache Address Window 1 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_uncache1\_base [they] | 16 | 0 x0 | R/W | HT bus uncache address window 1, address base address [39:24] |
| 15:0 | Ht\_uncache1\_mask [they] | 16 | 0 x0 | R/W | HT bus uncache address window 1, address shielded [39:24] |

Offset: 0x1A0

Reset value: 0x00000000

Name: HT Bus Uncache Address Window 2 enabled (internal access)

Table 14-60 HT Bus Uncache Address Window 2 Enable (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_uncache2\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 2, enabling signal |
| 30 | ht\_uncache2\_trans\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 2, map enabling signal |
| 29 | ht\_uncache2\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 2, enabling multi-node address mapping |
| 28 | ht\_uncache2\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 2, protocol address life  Can make |
| 25:0 | Ht\_uncache2\_trans [prey] | 26 | 0 x0 | R/W | HT Bus UNCache Address window 2, address after translation [49:24] |

Offset: 0x1A4

Reset value: 0x00000000

Name: HT Bus Uncache Address Window 2 Base address (internal access)

Table 14-61 HT Bus Uncache Address Window 2 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_uncache2\_base [they] | 16 | 0 x0 | R/W | HT bus uncache address window 2, address base address [39:24] |
| 15:0 | Ht\_uncache2\_mask [they] | 16 | 0 x0 | R/W | HT bus uncache address window 2, address shielded [39:24] |

Offset: 0x1A8

Reset value: 0x00000000

Name: HT Bus Uncache Address Window 3 enabled (internal access)

Table 14-62 HT Bus Uncache Address Window 3 Enable (internal access)

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 | ht\_uncache3\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 3, enabling signal |
| 30 | ht\_uncache3\_trans\_en | 1 | 0 x0 | R/W | HT bus UNCache address window 3, map enabling signal |
| 29 | ht\_uncache3\_multi\_node\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 3, enabling multi-node address mapping |
| 28 | ht\_uncache3\_conf\_hit\_en | 1 | 0 x0 | R/W | HT bus UNCache receives address window 3, protocol address hit enable |
| 25:0 | Ht\_uncache3\_trans [prey] | 26 | 0 x0 | R/W | HT Bus UNCache Address window 3, address after translation [49:24] |

Offset: 0x1AC

Reset value: 0x00000000

Name: HT Bus Uncache Address Window 3 Base address (internal access)

Table 14-63 HT Bus Uncache Address Window 3 Base address (internal access)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_uncache3\_base [they] | 16 | 0 x0 | R/W | HT bus uncache address window 3, address base address [39:24] |
| 15:0 | Ht\_uncache3\_mask [they] | 16 | 0 x0 | R/W | HT bus uncache address window 3, address shielded [39:24] |

#### The P2P address window configures registers

See Section 14.5.10 for the hit formula of address window.

The address of this window is the address received on the HT bus. The read and write command falling on the address of this window is directly forwarded back to the bus as a P2P command, which has the highest priority compared to the normal receive window and Uncache window.

Offset: 0x1B0

Reset value: 0x00000000

HT bus P2P address window 0 enable (external access)

Table 14-64 HT bus P2P address window 0 enable (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_p2p0\_en | 1 | 0 x0 | R/W | HT bus P2P address window 0, enabling signal |
| 29:0 | Ht\_rx\_p2p0\_trans [53:24] | 30 | 0 x0 | R/W | HT bus P2P address window 0, translated address [53:24] |

Offset: 0x1B4

Reset value: 0x00000000

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

Table 14-65 HT bus P2P address window 0 base address (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_p2p0\_base [they] | 16 | 0 x0 | R/W | HT bus P2P address window 1, address base address [39:24] |
| 15:0 | Ht\_rx\_p2p0\_mask [they] | 16 | 0 x0 | R/W | HT bus P2P address window 1, address shielded [39:24] |

Offset: 0x1B8

Reset value: 0x00000000

HT bus P2P address window 1 enabling (external access)

Table 14-66 HT bus P2P address window 1 enables (externally accessible) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | ht\_rx\_p2p1\_en | 1 | 0 x0 | R/W | HT bus P2P address window 1, enabling signal |
| 29:0 | Ht\_rx\_p2p1\_trans [53:24] | 30 | 0 x0 | R/W | HT bus P2P address window 1, translated address [53:24] |

Offset: 0x1BC

Reset value: 0x00000000

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

Table 14-67 HT bus P2P address window 1 base address (external access) register definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Ht\_rx\_p2p1\_base [they] | 16 | 0 x0 | R/W | HT bus P2P address window 1, address base address [39:24] |
| 15:0 | Ht\_rx\_p2p1\_mask [they] | 16 | 0 x0 | R/W | HT bus P2P address window 1, address shielded [39:24] |

#### Controller parameters configure registers

Offset: 0x1C0

Reset value: 0x00904321

Name: APP CONFIG 0

Table 14-68 Controller parameter configuration register 0 definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| charm | Reserved | 1 | 0 x0 |  | reserve |
| 29 | Ldt Stop Gen | 1 | 0 x0 | R/W | Allow the bus to enter THE LDT DISCONNECT mode  The correct way to do it is: 0 minus > 1 |
| 28 | Ldt the Req Gen | 1 | 0 x0 | R/W | Wake HT bus, set, from LDT DISCONNECT  Buy LDT\_REQ\_n |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | The right way to do it is to put 0 first and then 1:0 -> 1  In addition, a direct read/write request to the bus can also automatically wake up the bus |
| 27 | Rx sample en | 1 | 0 x0 | R/W | Enable sampling of input CAD and CTL sent at (0x1C8)  Display in memory for debugging |
| 26 | Dword Write | 1 | 0 x1 | R/W | For 32/64/128/256 MEM Write access, use Dword Write command format (Byte Write)  When writing is received, it will be converted to 128-bit MASK writing. |
| 25 | Dword Write CFG | 1 | 0 x1 | R/W | For Write access to configuration space, whether to use the Dword Write command format (Byte Write in receive)  Will be converted to 128-bit MASK) |
| 24 | Dword Write IO | 1 | 0 x1 | R/W | For Write access to IO space, use Dword Write  Command format (Byte Write is converted to 128-bit MASK when received) |
| 23 | Axi aw resize | 1 | 0 x0 | RW | Whether to write 128 bits with MASK size according to MASK  The reset of |
| 22 | Coherent Mode | 1 | 0 x0 | RW | Is it processor consistent or not? The initial value is given by  ICCC\_EN pin determines when reset takes effect |
| 21 | Coherent\_split | 1 | 0 x0 | RW | In consistent mode, all packages are processed as 32Byte |
| 20 | Not Care Seqid | 1 | 0 x0 | R/W | Does the receiver not care about the HT order relationship |
| He hath | Fixed Seqid | 4 | 0 x0 | R/W | When Not Axi2Seqid is valid, configure the HT bus to emit  The Seqid |
| " | Priority Nop | 4 | 0 x4 | R/W | HT bus Nop stream packet priority |
| and | Specify the NPC | 4 | 0 x3 | R/W | Non Post channel read/write priority |
| The log | Priority RC | 4 | 0 x2 | R/W | The Response channel reads and writes the first stage |
| 3-0 | Priority PC | 4 | 0 x1 | R/W | Post channel read/write priority  0x0: Highest priority  0xF: Lowest priority  For each channel priority is adopted according to the time change priority strategy, this storage is used to configure the initial priority of each channel |

Offset: 0x1C4

Reset value: 0x00904321

Name: APP CONFIG1

Table 14-69 Definition of controller parameter configuration register 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | Reset value | access | describe |
| 31 | Tx post split en | 1 | 0 x0 | R/W | Enable write and unpack when the TX POST ID window hits  All write requests that cross the 32-byte boundary are split into |
|  |  |  |  |  | Two consecutive write requests (Byte write) |
| 30 | Tx wr passPW PC | 1 | 0 x0 | R/W | Write all outgoing Post channels to the passPW of the request  The bit is set to 1 |
| 29 | Tx wr passPW NPCS | 1 | 0 x0 | R/W | Writes all outgoing Nonpost channels to the request  The passPW bit is set to 1 |
| 28 | Tx rd passPW | 1 | 0 x0 | R/W | Set the passPW bit for all issued read requests to 1 |
| 27 | Stop same id wr | 1 | 0 x0 | R/W | When the sending side encounters a write request with the same AXI ID, it stops  Send until the previous request with the same ID is returned |
| 26 | Stop same id rd | 1 | 0 x0 | R/W | When the sending side encounters a read request with the same AXI ID, it stops  Send until the previous request with the same ID is returned |
| 25 | The Not axi2seqid wr | 1 | 0 x0 | R/W | Do not write the request AXI ID to seQID conversion directly  Using fixed seqid |
| 24 | The Not axi2seqid rd | 1 | 0 x0 | R/W | Do not read the request for AXI ID to seQID conversion directly  Using fixed seqid |
| " | Reserved | 2 | 0 x0 | R/W | reserve |
| 21 | Act as a slave | 1 | 0 x1 | R/W | Set SLAVE mode |
| 20 | The Host hide | 1 | 0 x0 | R/W | Disable receiver access to the configuration register space |
| He hath | Rrequest delay | 4 | 0 x3 | R/W | Used to control the random delay range of Rrequest transmission in consistent mode  000:0 delay  001: Random delay 0-8  010: Random delay 8-15  011: Random delay 16-31  100: Random delay 32-63  101: Random delay 64-127  110: Random delay 128-255  111:0 delay |
| 15 | Crc Int en | 1 | 0 x0 | R/W | Enable an interrupt transmission in case of a CRC error |
| then | Crc Int the route | 3 | 0 x0 | R/W | Interrupt pin selection for CRC interrupts |
| 11 | Reserved |  |  |  |  |
| 10 | Ht int 8 bit | 1 | 0 x0 | R/W | Use 8 middle breaks |
| o | ht\_int\_stripe | 2 | 0 x0 | R/W | Corresponding to the three interrupt routing methods, see the interrupt vector register for specific description  0 x0: ht\_int\_stripe\_1  0x1: ht\_int\_stripe\_2 0x2: ht\_int\_stripe\_4 |
| 4:0! | Interrupt the Index | 5 | 0 x0 | R/W | Redirect interrupts other than standard interrupts to which interrupt vector (including SMI, NMI, INIT, INTA, INTB, INTC, INTD)  A total of 256 interrupt vectors are represented in this register |
|  |  |  |  |  | The high 5 bits of interrupt vector, the internal interrupt vector is as follows:  000: SMI  001: NMI  010: INIT  011: Reservered 100: INTA  101: intb.br deal  110: INTC  111: INTD |

#### Receiving diagnostic register

Offset: 0x1C8

Reset value: 0x00000000

Name: Receive diagnostic register

Table 14-70 receives the diagnostic register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | Reset value | access | describe |
| Caused the | rx\_cad\_phase\_0 | 16 | 0 x0 | R/W | Save the value of the sampled input CAD[15:0] |
| " | rx\_ctl\_catch | 8 | 0 x0 | R/W | Save the sampled input CTL  (0, 2, 4, 6) correspond to four phases of CTL0 sampling  (1, 3, 5, 7) correspond to four phases of CTL1 sampling |
| away |  |  |  |  |  |

#### PHY status register

For PHY related state observation, offset 0x1CC is used for debugging

Reset value: 0x83308000

Name: PHY status register

Table 14-71 PHY status register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | Reset value | access | describe |
| take | Reserved | 3 | 0 x0 | R | reserve |
| 28 | DLL locked hi | 1 | 0 x0 | R |  |
| 27 | DLL locked lo | 1 | 0 x0 | R |  |
| 26 | CDR locked hi | 1 | 0 x0 | R |  |
| 25 | CDR locked lo | 1 | 0 x0 | R |  |
| 24 | Phase locked | 1 | 0 x0 | R |  |
| Behold, | Phy state | 4 | 0 x0 | R |  |
| michal | Tx training status | 3 | 0 x0 | R |  |
| " | Rx training status | 3 | 0 x0 | R |  |
| Will you | Init done | 6 | 0 x0 | R |  |
| away | Reserved | 8 |  | R |  |

#### The command sends the cache size register

The command send cache size register is used to observe the number of caches available for each command channel at the sender. Offset: 0x1D0

Reset value: 0x00000000

Name: Command sends cache size register

Table 14-72 commands send the cache size register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| came | B\_CMD\_txbuffer | 8 | 0 x0 | R | Number of b-channel command caches at sending end |
| Ephron; | R\_CMD\_txbuffer | 8 | 0 x0 | R | Number of R channel command caches on the sending side |
| " | NPC\_CMD\_txbuffer | 8 | 0 x0 | R | Number of NPC channel command caches on the sending side |
| away | PC\_CMD\_txbuffer | 8 | 0 x0 | R | Number of caches of sending PC channel commands |

#### Data sent cache size register

The data send cache size register is used to observe the number of caches available for each data channel at the sending end. Offset: 0x1D4

Reset value: 0x00000000

Name: Data send cache size register

Table 14-73 Data send cache size register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | Reserved | 1 | 0 x0 | R | reserve |
| 30 | Rx\_buffer\_r\_data [4] | 1 | 0 x0 | R/W | Initialize read buffer for the receive buffer  The information bit [4] |
| 29 | Rx\_buffer\_npc\_data [4] | 1 | 0 x0 | R/W | Initial NPC data Buffer to receive buffer  Bit [4] |
| 28 | Rx\_buffer\_pc\_data [4] | 1 | 0 x0 | R/W | Initialize the PC data Buffer for the receive buffer  The information bit [4] |
| 27 | Rx\_buffer\_b\_cmd [4] | 1 | 0 x0 | R/W | Receive the BResponse order from the buffer zone |
|  |  |  |  |  | Buffer initializes the bit of information [4] |
| 26 | Rx\_buffer\_r\_cmd [4] | 1 | 0 x0 | R/W | Initialize the read command Buffer for the receive buffer  The information bit [4] |
| 25 | Rx\_buffer\_npc\_cmd [4] | 1 | 0 x0 | R/W | Receive buffer initial NPC command buffer  Bit [4] |
| 24 | Rx\_buffer\_pc\_cmd [4] | 1 | 0 x0 | R/W | Initialize the PC command Buffer for the receive buffer  The information bit [4] |
| Ephron; | R\_DATA\_txbuffer | 8 | 0 x0 | R | Number of R channel data caches at the sending end |
| " | NPC\_DATA\_txbuffer | 8 | 0 x0 | R | Number of NPC channel data caches on the sending side |
| away | PC\_DATA\_txbuffer | 8 | 0 x0 | R | Number of PC channel data caches at the sending end |

#### Send the cache debug register

The sending cache debug register is used to set the number of sending buffer of HT controller manually, and the number of sending buffer is adjusted by increasing or decreasing.

Offset: 0x1D8

Reset value: 0x00000000

Name: Send cache debug register

Table 14-74 sends the cache debug register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | b\_interleave | 1 | 0 x0 | R/W | Consistent mode enables the interleaving of Channel B with other channels  transmission |
| 30 | nop\_interleave | 1 | 0 x0 | R/W | Enables the interleaving of streaming packets with other virtual channels |
| 29 | Tx\_neg | 1 | 0 x0 | R/W | The sender cache debug symbol  0: Increase the corresponding number  1: Reduce (corresponding register number +1) |
| 28 | Tx\_buff\_adj\_en | 1 | 0 x0 | R/W | The sender cache debugging enablement register  0->1: causes the value of this register to increase or decrease once |
| he | R\_DATA\_txadj | 4 | 0 x0 | R/W | Number of increase or decrease of R channel data cache at sending end  When tx\_NEg is 0, increase R\_DATA\_txadj; When tx\_NEg is 1, reduce R\_DATA\_txadj+1 |
| Behold, | NPC\_DATA\_txadj | 4 | 0 x0 | R/W | The number of NPC channel data cache increases or decreases on the sending side  When tx\_NEg is 0, increase NPC\_DATA\_txadj  A;  When tx\_NEg is 1, reduce NPC\_DATA\_txadj+1  a |
| He hath | PC\_DATA\_txadj | 4 | 0 x0 | R/W | Number of increase or decrease of data cache on PC channel at sending end |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | When tx\_NEg is 0, increase PC\_DATA\_txadj; When tx\_NEg is 1, reduce PC\_DATA\_txadj+1  a |
| " | B\_CMD\_txadj | 4 | 0 x0 | R/W | Number of increase or decrease in the command cache of the sending side B channel  When tx\_NEg is 0, increase B\_CMD\_txadj; When tx\_NEg is 1, reduce B\_CMD\_txadj+1 |
| and | R\_CMD\_txadj | 4 | 0 x0 | R/W | Number of increase or decrease of R channel command cache on sending side  When tx\_NEg is 0, increase R\_CMD\_txadj; When tx\_NEg is 1, reduce R\_CMD\_txadj+1 |
| The log | NPC\_CMD\_txadj | 4 | 0 x0 | R/W | Number of NPC channel commands/data cache increases or decreases on the sending side  When tx\_NEg is 0, add NPC\_CMD\_txadj; When tx\_NEg is 1, reduce NPC\_CMD\_txadj+1  a |
| 3-0 | PC\_CMD\_txadj | 4 | 0 x0 | R/W | The number of increase or decrease of command cache on PC channel on sending side  When tx\_NEg is 0, increase PC\_CMD\_txadj; When tx\_NEg is 1, reduce PC\_CMD\_txadj+1  a |

#### Receive buffer initial register

Offset: 0x1DC

Reset value: 0x07778888

Name: Receive buffer initializes the configuration register

Table 14-75 receive buffer initial register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | Reset value | access | describe |
| he | rx\_buffer\_r\_data | 4 | 0 x0 | R/W | Read Buffer initialization information for the receive buffer |
| Behold, | rx\_buffer\_npc\_data | 4 | 0 x0 | R/W | The NPC data Buffer that receives the buffer initializes the information |
| He hath | rx\_buffer\_pc\_data | 4 | 0 x0 | R/W | Receive PC data Buffer initialization information for the buffer |
| " | rx\_buffer\_b\_cmd | 4 | 0 x0 | R/W | Receive bResponse buffer initialization information for the buffer |
| and | rx\_buffer\_r\_cmd | 4 | 0 x0 | R/W | Receives read buffer initialization information for the buffer |
| The log | rx\_buffer\_npc\_cmd | 4 | 0 x0 | R/W | The NPC command Buffer that receives the buffer initializes the information |
| 3-0 | rx\_buffer\_pc\_cmd | 4 | 0 x0 | R/W | Receive THE PC command Buffer initialization information for the buffer |

#### Training 0 short timeout register

It is used to configure the short time timeout threshold of Training 0 in HyerTransport 3.0 mode, and the counter clock frequency is 1/4 of the link bus clock frequency of HyperTransport3.0.

Offset: 0x1E0

Reset value: 0x00000080

Name: Short timeout count register for Training 0

Table 14-76 Short timeout register of Training 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | Gen3\_timing\_soft | 1 | 0 x0 | R/W |  |
| then | Retry\_nop\_num | 8 | 0 x0 | R/W |  |
| 22:0 | T0 time | 23 | 0 x80 | R/W | Training 0 short timeout register |

#### Training 0 timeout long timing register

Used for Training 0 long count timeout threshold in HyerTransport 3.0 mode, the counter clock frequency is 1/4 of the link bus clock frequency HyperTransport3.0.

Offset: 0x1E4

Reset value: 0x000fffff

Name: Training 0 timeout long count register

Table 14-77 Training 0 timeout long count register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | T0 time | 32 | 0 XFFFFF | R/W | Training 0 timeout long count register |

#### Training 1 counting register

Used for Training 1 counting threshold in HyerTransport 3.0 mode, the counter clock frequency is 1/4 of the link bus clock frequency HyperTransport3.0.

Offset: 0x1E8

Reset value: 0x0004fffff

Name: Training 1 counting register

Table 14-78 Training 1 counting register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | T1 time | 32 | 0 x4fffff | R/W | Training 1 counting register |

#### Training 2 counting register

For Training 2 counting threshold in HyerTransport 3.0 mode, the counter clock frequency is 1/4 of the link bus clock frequency HyperTransport3.0.

Offset: 0x1EC

Reset value: 0x0007fffff

Name: Training 2 counting register

Table 14-79 Training 2 counting register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | T2 time | 32 | 0 x7fffff | R/W | Training 2 counting register |

#### Training 3 counting register

For Training 3 counting threshold in HyerTransport 3.0 mode, the counter clock frequency is 1/4 of the link bus clock frequency of HyperTransport3.0.

Offset: 0x1F0

Name: Training 3 counting register

Table 14-80 Training 3 counting register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | T3 time | 32 | 0 x7fffff | R/W | Training 3 counting register |

#### Software frequency configuration registers

It is used to switch to the link frequency and controller frequency supported by any protocol and PLL during the operation. The specific switching method is as follows: under the premise of enabling software configuration mode, set the software frequency configuration register bit 1, and

Write the new clock-related parameters, including div\_REFc and div\_loop to determine the PLL output frequency, phy\_hi\_div and phy\_lo\_div on the link, and core\_div for the controller. After entering Warm Reset or LDT Disconnect, the controller will automatically reset the PLL and configure the new clock parameters.

PHY\_LINK\_CLK is the HT bus frequency. The calculation formula of clock frequency is:

When SYS\_CLOCK is used as the reference clock input and SYS\_CLOCK is 25MHz (CLKSEL[8] is 1 and CLKSEL[5] is 1), the frequency calculation method is:

HyperTransport 1.0:

PHY\_LINK\_CLK = 12.5mhz ×div\_loop /div\_refc /phy\_div HyperTransport 3.0:

In other cases, the frequency is calculated as follows:

HyperTransport 1.0:

PHY\_LINK\_CLK = 50MHz×div\_loop /div\_refc /phy\_div HyperTransport 3.0:

PHY\_LINK\_CLK = 100MHz×div\_loop /div\_refc /phy\_div

The waiting time for PLL re-lock is about 30US when the system CLK is 33M by default. You can also write a custom wait count upper limit in the register.

Note that in 3A4000, HT\_CORE\_CLK is no longer controlled by this configuration, but by the NODE clock divider.

Offset: 0x1F4

Reset value: 0x00000000

Name: Software frequency configuration register

Table 14-81 Software frequency configuration registers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| behold | PLL relock counter | 5 | 0 x0 | R/W | The upper limit of the counter configuration register, when setting the counter SELECT, is  {PLL\_relock\_counter,5 'h1f}, otherwise count  Up to 10 '3 ff |
| 26 | Counter the select | 1 | 0 x0 | R/W | Lock timer customization enabled: 1 'b0 USES default counting upper limit;  1 'b1 is calculated by PLL\_relock\_counter |
| Struggled together | Soft\_phy\_lo\_div | 4 | 0 x0 | R/W | Low LEVEL PHY frequency division coefficient |
| Lift up | Soft\_phy\_hi\_div | 4 | 0 x0 | R/W | High LEVEL PHY frequency division coefficient |
| " | Soft\_div\_refc | 2 | 0 x0 | R/W | PLL frequency division coefficient |
| Put no | Soft\_div\_loop | 7 | 0 x0 | R/W | PLL internal frequency multiplication factor |
| then | Soft\_core\_div | 4 | 0 x0 | R/W | Frequency division coefficient of controller clock |
| 4-2 | Reserved | 3 | 0 x0 | R | reserve |
| 1 | Soft cofig enable | 1 | 0 x0 | R/W | Software configuration enablement bit  1 'b0 disables software frequency configuration  1 'b1 enabled software frequency configuration |
| 0 | Reserved | 1 | 0 x0 | R | reserve |

#### PHY impedance matching control register

Impedance matching enablers are used to control the PHY. The offset of the impedance matching parameters at the sending and receiving ends is set: 0x1F8

Reset value: 0x00000000

Name: PHY Impedance matching Control Register

Table 14-82 Impedance matching control register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | Tx\_scanin\_en | 1 | 0 x0 | R/W | TX impedance matching enablement |
| 30 | Rx\_scanin\_en | 1 | 0 x0 | R/W | RX impedance matching enablement |
| he | Tx\_scanin\_ncode | 4 | 0 x0 | R/W | TX impedance matching scan input Ncode |
| Behold, | Tx\_scanin\_pcode | 4 | 0 x0 | R/W | TX impedance matching scan input pcode |
| then | Rx\_scanin\_code | 8 | 0 x0 | R/W | RX impedance matching scan input |

#### PHY configuration register

It is used to configure phY-related physical parameters. When the controller is two independent 8BIT controllers, the higher level PHY and the lower level PHY are controlled independently by the two controllers respectively. When the controller is a 16BIT controller, the configuration parameters of the high and low LEVEL PHYs are controlled by the low level controller.

Offset: 0x1FC

Reset value: 0x83308000

Name: PHY Configuration register

Table 14-83 PHY configuration registers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | Rx\_ckpll\_term | 1 | 0 x1 | R/W | Terminal impedance of PLL to RX terminal |
| 30 | Tx\_ckpll\_term | 1 | 0 x0 | R/W | Terminal impedance of PLL to TX terminal |
| 29 | Rx\_clk\_in\_sel\_ | 1 | 0 x0 | R/W | The clock PAD is provided with the clock selection of the data PAD, which is automatically selected as CLKPAD in HT1 mode:  1 'b0 external clock source  1 'b1 PLL clock |
| 28 | Rx\_ckdll\_sell | 1 | 0 x0 | R/W | Clock selection for locking DLLS:  1 'b0 PLL clock  1 'b1 external clock source |
| But after | Rx\_ctle\_bitc | 2 | 0 x0 | R/W | PAD EQD high frequency gain |
| Thus for | Rx\_ctle\_bitr | 2 | 0 x3 | R/W | PAD EQD low frequency gain |
| " | Rx\_ctle\_bitlim | 2 | 0 x0 | R/W | PAD EQD compensation restrictions |
| 21 | Rx\_en\_ldo | 1 | 0 x1 | R/W | They control  1 'b0 "disabled  1 'b1 can they make |
| 20 | Rx\_en\_by | 1 | 0 x1 | R/W | BandGap control  1 'b0 BandGap is disabled  1 'b1 BandGap can make |
| michal | Reserved | 3 | 0 x0 | R | reserve |
| then | Tx\_preenmp | 5 | 0 x08 | R/W | PAD pre-weighted control signal |
| 11:0 | Reserved | 12 | 0 x0 | R | reserve |

#### Link initializes the debug register

In HyperTransport 3.0 mode, it is used to configure whether THE CDR lock signal provided by the PHY is used as the symbol of completion of link CDR during link initialization. If the lock signal is ignored, the default CDR is completed after the controller counts for a certain amount of time.

Offset: 0x240

Reset value: 0x00000000

Name: Link initialization debug register

Table 14-84 Link initialization debug register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 15 | Cdr\_ignore\_enable | 1 | 0 x0 | R/W | Whether CRC lock is ignored during link initialization and wait is completed through counter counting:  1 'b0 wait for CDR lock  1 'b1 ignores the CDR lock signal and waits by accumulating through the counter |
| 14:0 | Cdr\_wait\_counter | 15 | 0 x0 | R/W | Wait for the upper limit of the counter count to complete based on the controller clock |

#### LDT debug register

When the software changes the controller frequency, the timing of LDT REconnect stage will be inaccurate. The counter shall be configured. As the frequency of software configuration, the time between the invalid LDT signal and the initialization of the controller start link is based on the controller clock.

Offset: 0x244

Reset value: 0x00000000

Name: LDT debug register 1

Table 14-85 LDT debug register 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | Rx\_wait\_time | 16 | 0 x0 | R/W | The RX end waits for the initial value of the counter |
| 15:0 | Tx\_wait\_time | 16 | 0 x0 | R/W | The TX terminal waits for the initial value of the counter |

Offset: 0x248

Reset value: 0x00000000

Name: LDT debug register 2

Table 14-86 LDT debug register 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| charm | Reserved | 16 | 0 x0 | R/W |  |
| 29:0 | Rx lane ts 0 | 16 | 0 x0 | R/W |  |

Offset: 0x24C

Reset value: 0x00000000

Name: LDT debug register 3

Table 14-87 LDT debug register 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| charm | Reserved | 16 | 0 x0 | R/W |  |
| 29:0 | Rx lane ts 1 | 16 | 0 x0 | R/W |  |

Offset: 0x250

Reset value: 0x00000000

Name: LDT debug register 4

Table 14-88 LDT debug register 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| charm | Reserved | 16 | 0 x0 | R/W |  |
| 29:0 | 2 rx lane ts | 16 | 0 x0 | R/W |  |

Offset: 0x254

Reset value: 0x00000000

Name: LDT debug register 5

Table 14-89 LDT debug register 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| then | Reserved | 10 | 0 x0 | R/W |  |
| Lift up | Wait CTL | 4 | 0 x0 | R/W |  |
| 17:0 | Phase lock | 18 | 0 x0 | R/W |  |

Offset: 0x258

Reset value: 0x00000000

Name: LDT debug register 5

Table 14-90 LDT debug register 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31:0 | Wait cad | 32 | 0 x0 | R/W |  |

#### HT TX POST ID window configures registers

The window sends the hit request out through the HT POST channel by comparing the ID of the internal write request with the default window.

Offset: 0x260

Reset value: 0x00000000

Name: HT TX POST ID WIN0

Table 14-91 HT TX POST ID WIN0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | HT TX POST ID0 MASK | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window is transmitted, the MASK bit of ID |
| 15:0 | HT TX POST ID0 BASE | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window carries the ID of the BASE bit |

Offset: 0x264

Reset value: 0x00000000

Name: HT TX POST ID WIN1

Table 14-92 HT TX POST ID WIN1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | HT TX POST ID1 MASK | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window is transmitted, the MASK bit of ID |
| 15:0 | HT TX POST ID1 BASE | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window carries the ID of the BASE bit |

Offset: 0x268

Reset value: 0x00000000

Name: HT TX POST ID WIN2

Table 14-93 HT TX POST ID WIN2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | HT TX POST ID2 MASK | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window is transmitted, the MASK bit of ID |
| 15:0 | HT TX POST ID2 BASE | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window carries the ID of the BASE bit |

Offset: 0x26C

Reset value: 0x00000000

Name: HT TX POST ID WIN3

Table 14-94 HT TX POST ID WIN3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| Caused the | HT TX POST ID3 MASK | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window is transmitted, the MASK bit of ID |
| 15:0 | HT TX POST ID3 BASE | 16 | 0 x0 | R/W | The AXI ID hit request USES POST  The window carries the ID of the BASE bit |

#### External interrupt transformation configuration

This setting converts the interrupts HT receives into a write to a specific address, writing directly to the extended IO interrupt vector inside the chip instead of generating interrupts inside the HT controller. In this way, advanced features such as direct slice distribution of IO interrupts can be used.

Offset: 0x270

Complex bit value: 0x00000000

HT RX INT TRANS Lo

Table 14-95 HT RX INT TRANS LO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| does | INT\_trans\_addr [also] | 28 | 0 x0 | R/W | Interrupt address conversion low order |
| 3-0 | Reserved | 4 | 0 x0 | R | reserve |

Offset: 0x274

Complex bit value: 0x00000000

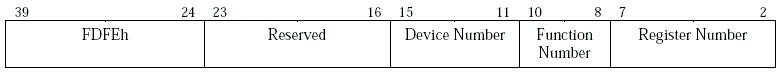
HT RX INT TRANS Hi

Table 14-96 HT RX INT TRANS Hi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | INT\_trans\_en | 1 | 0 x0 | R/W | Interrupt transformation enablement |
| 30 | INT\_trans\_allow | 1 | 0 x0 | R/W | Interrupt transformation enablement  After this bit is set, the INT\_trans\_en or EXT\_INT\_en of the chip takes effect. |
| then | INT\_trans\_cache | 4 | 0 x0 | R/W | Interrupt the conversion Cache field |
| 25:0 | INT\_trans\_addr [57:32] | 26 | 0 x0 | R/W | Interrupt address conversion to high level |

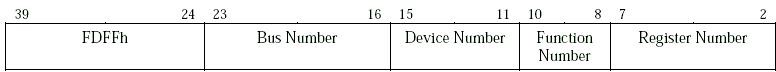
## The HyperTransport bus concodes access methods for Spaces

The protocol of the HyperTransport interface software layer is basically the same as PCI protocol, but the details of the access are slightly different because the access to the configuration space is directly related to the underlying protocol. As listed in Table 14-6, the address range of the HT bus configuration space is 0xFD\_FE00\_0000 ~ 0xFD\_FFFF\_FFFF. For configuration access in HT protocol, the following format is adopted in Longson 3A4000:



Type 0:

Type 1:

 Figure 14-1 Configuration access of HT protocol in Longson 3A4000

## HyperTransport multiprocessor support

The Loongson 3 processor USES the HyperTransport interface for multiprocessor interconnection, and the hardware can automatically maintain consistency requests between 2-8 chips.

###### Loongson no.3 interconnection route

There are two ways to route Loongson 3 interconnect. One is to adopt simple X-Y route. When routing, X is followed by Y. Take four chips as an example, ID number is 00,01,10,11 respectively. If a request is sent from 11 to 00, it is routed from 11 to 00, first in the X direction, then from 11 to 10, and then from 10 to 00 in the Y direction. When its response returns from 00 to 11, the route goes first in the X direction, from 00 to 01, and then in the Y direction, from 01 to 11. The other is diagonal direct access, which is achieved by connecting two diagonal chips by hardware, greatly reducing the access delay. This access mode needs to be enabled separately by software. Due to the characteristics of this algorithm, we can use many different methods to build multi-chip interconnections.

###### Four-chip Longshon 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 pieces, and HT1 HI to connect with diagonal chip, thus obtaining the interconnection structure as follows:

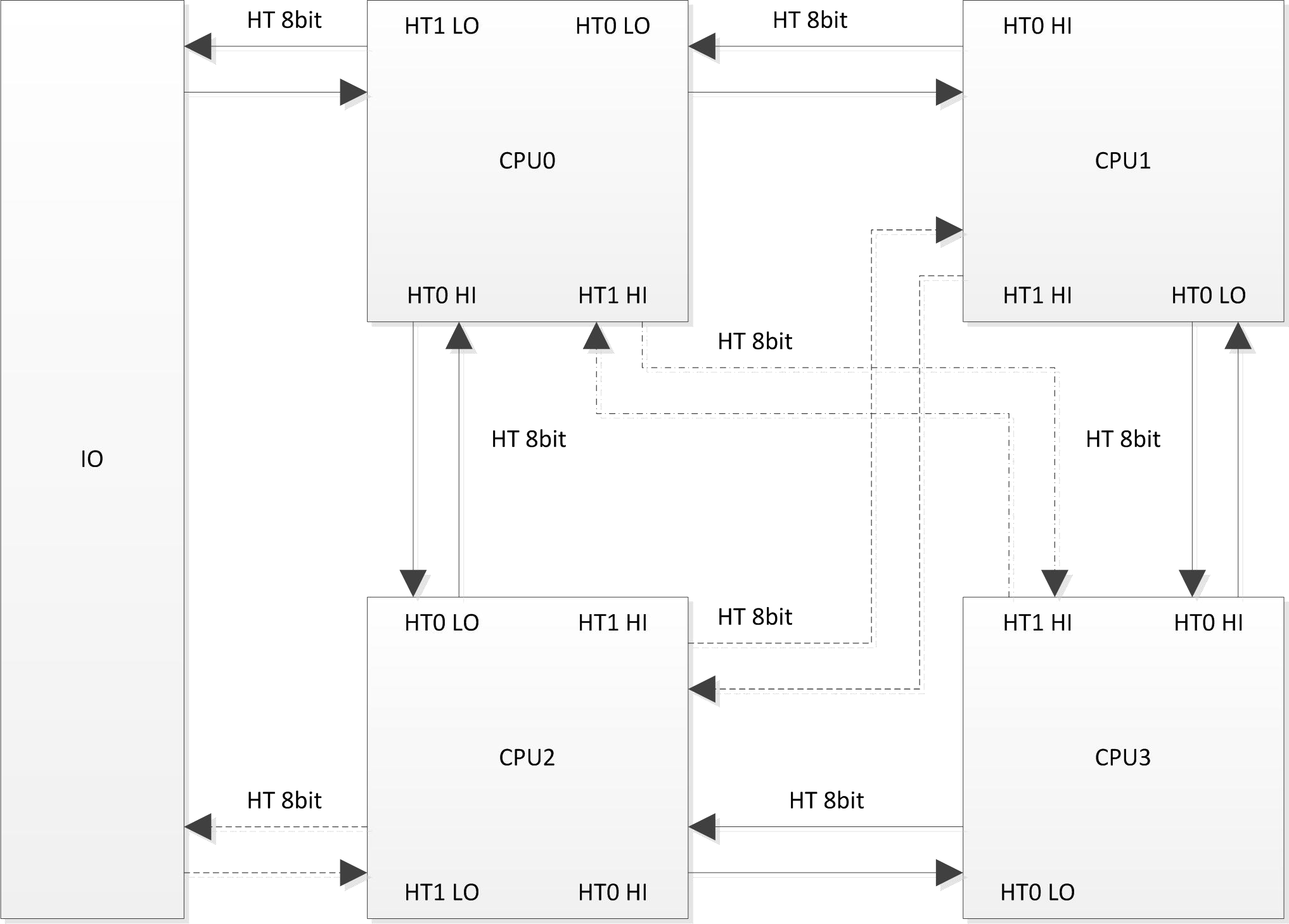


FIG. 14-2 Four-chip Loongson No.3 interconnection structure

###### Eight-chip Loongson no.3 interconnection structure

Eight Cpus make up the cube. Each CPU USES HT0's two 8-bit controllers to connect with the adjacent two pieces, thus obtaining the interconnection structure in the following figure by using HT1:

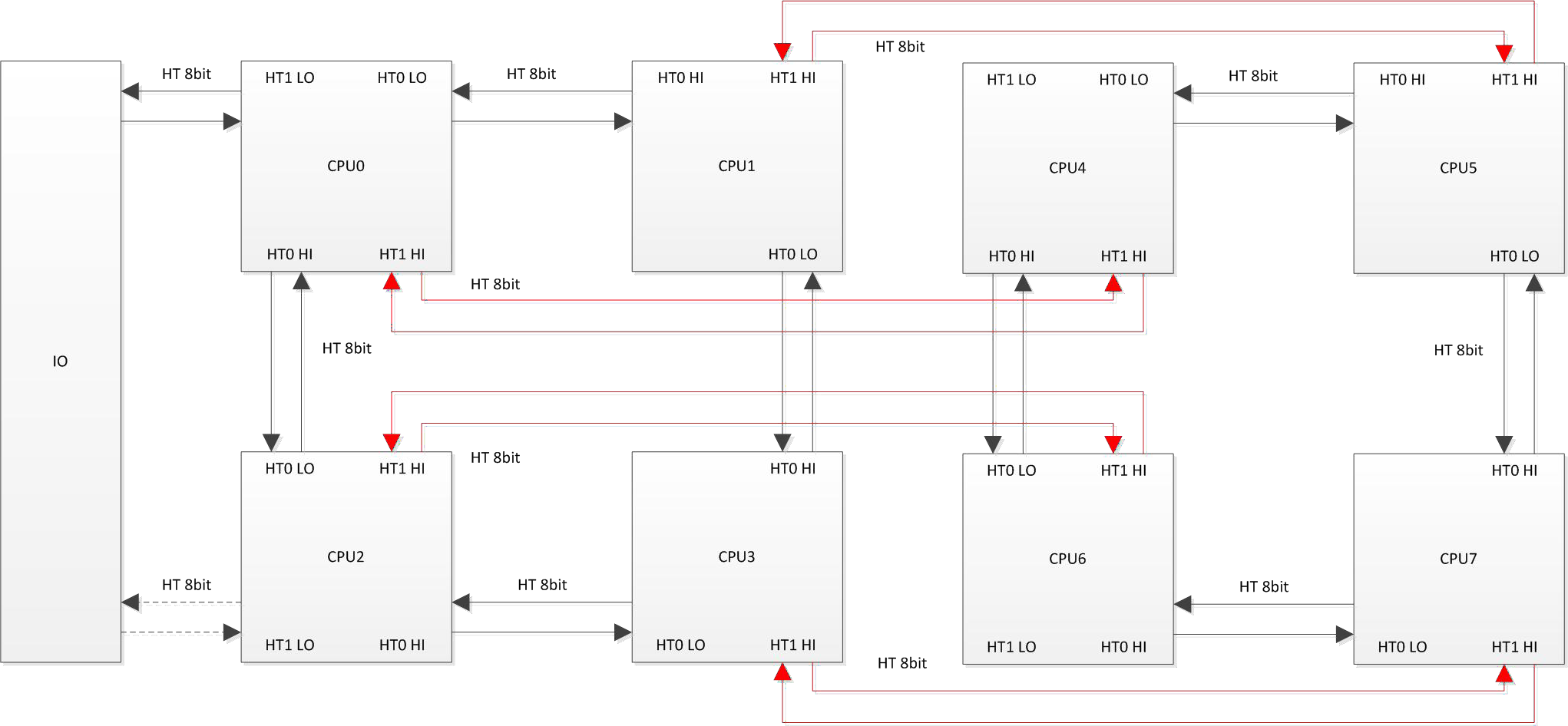


FIG. 14-3 Eight-chip Loongson no.3 interconnection structure

###### Two - piece Loong chip No. 3 interconnection structure

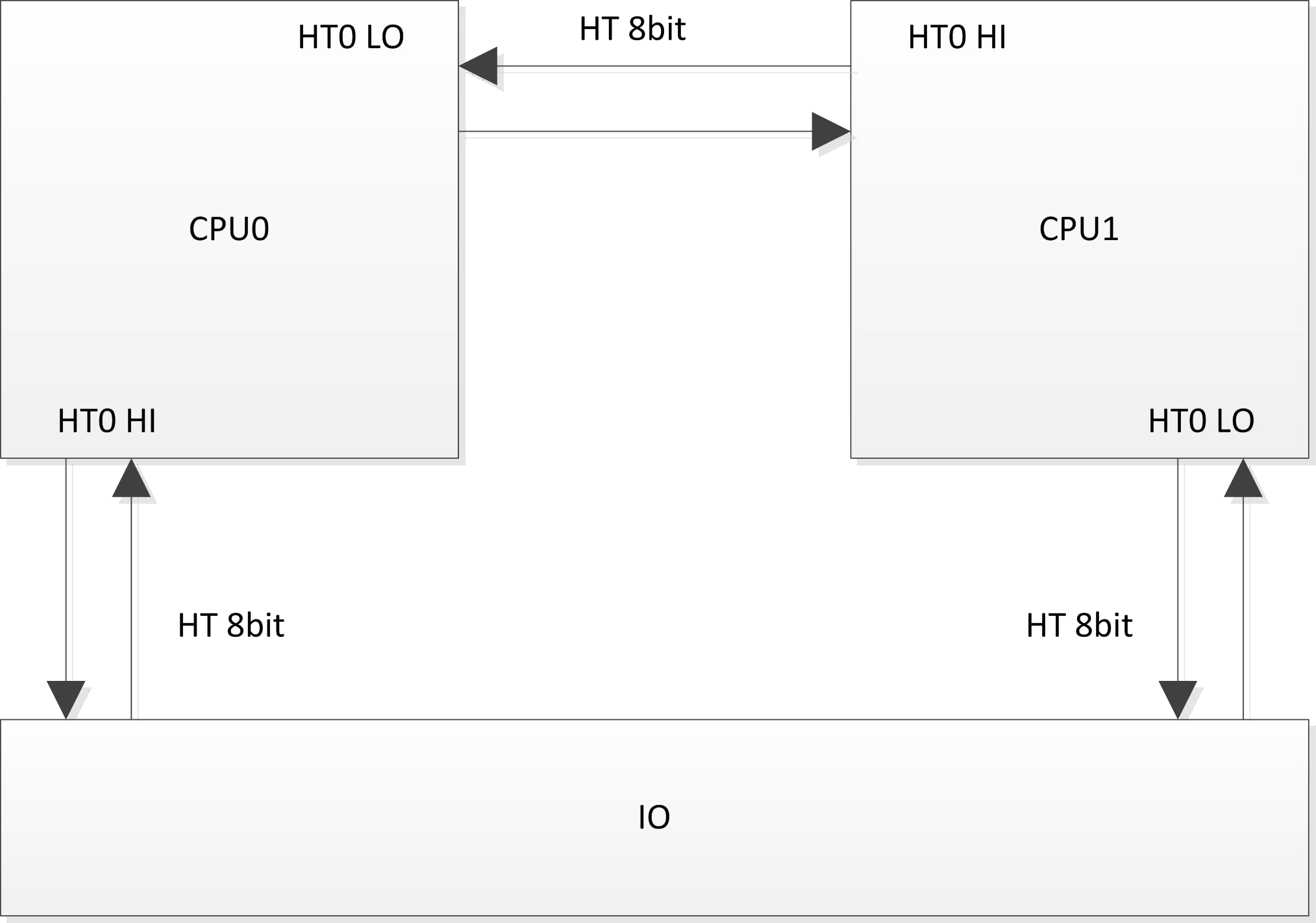
 Due to the nature of the fixed routing algorithm, we have two different approaches to building the two-chip interconnection. The first is the 8 bit HT bus interconnection. In this interconnection mode, only 8-bit HT interconnection can be used between two processors. The two chip Numbers are 00 and 01 respectively. From the routing algorithm, we can know that the two chips access each other through the 8-bit HT bus consistent with the four-chip interconnection. As shown below:

FIG. 14-4 Two-piece Loong chip No.3 8-bit interconnection structure

However, the widest HT bus can adopt the 16-bit mode, so the connection mode to maximize the bandwidth should adopt the 16-bit interconnection structure. In Loongson 3, as long as the HT0 controller is set to 16-bit mode, all commands sent to the HT0 controller will be sent to HT0\_LO, instead of to HT0\_HI or HT0\_LO according to the routing table, so that we can use the 16-bit bus when interconnecting. Therefore, we only need to configure the 16-bit mode of CPU0 and CPU1 correctly and connect the high-low bit bus correctly to use the 16-bit HT bus interconnection. The interconnection structure can also be accessed using 8-bit HT bus protocol. The resulting interconnection structure is as follows:

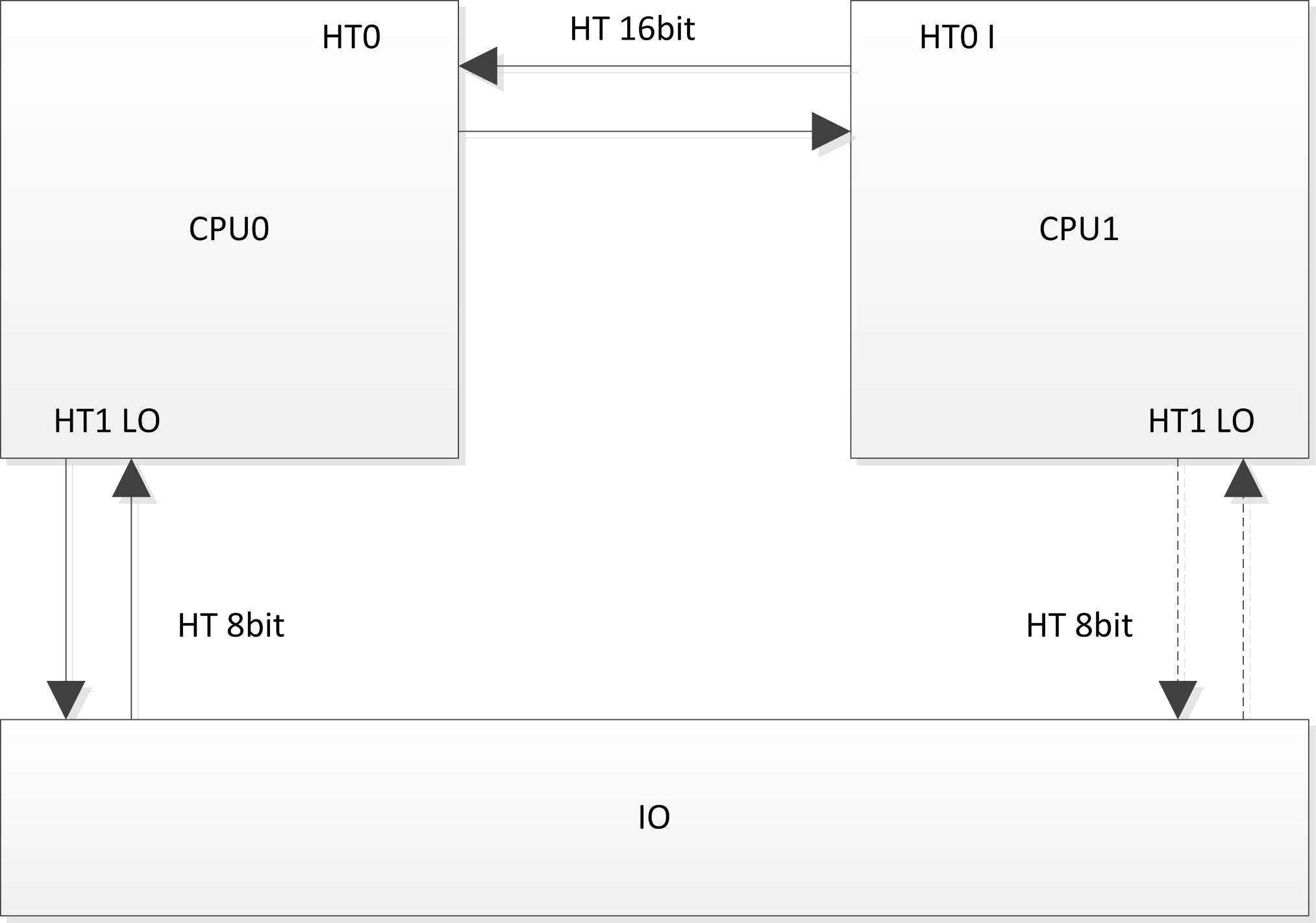


Figure 14-5 Two-chip Longson no.3 16-bit interconnection structure

## Low speed IO controller configuration

The Loongson 3 I/O controller includes UART controller, SPI controller, I2C and GPIO registers. These I/O controllers share a port where CPU requests are addressed and sent to the appropriate device.

## UART controller

The UART controller has the following characteristics

* Full duplex asynchronous data receiving/sending
* A programmable data format
* 16-bit programmable clock counter
* Support for receiving timeout detection
* Multiinterrupt system with arbitration
* Work only in FIFO mode
* NS16550A compatible register and function

Two UART interfaces are integrated inside the chip, with exactly the same functional registers, but with different access addresses. The physical address base of the UART0 register is 0x1FE001E0.

The physical address of the UART1 register is 0x1FE001E8.

A physical address of 0x1FE00100(UART0) and 0x1FE00110(UART1) was also provided for each of the two UArts. The two new registers, RFC and TFC, are accessed through this set of addresses.

#### Data Register (DAT)

Chinese name: Data Transmission Register Register width: [7:0]

Offset: 0x00

Reset value: 0x00

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

#### Interrupt enablement register (IER)

Interrupt enabled register Register width: [7:0]

Offset: 0x01

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| The log | Reserved | 4 | RW | reserve |
| 3 | IME | 1 | RW | Modem status interruption enabled '0' - close '1' - turn on |
| 2 | ILE | 1 | RW | Receiver line state interrupts to enable '0' - close '1' - open |
| 1 | ITxE | 1 | RW | The transfer save register enables' 0 '- close' 1 '- open for air break |
| 0 | IRxE | 1 | RW | Receive valid data interrupts enable '0' - close '1' - open |

#### Interrupt Identification Register (IIR)

Interrupt source Register Register width: [7:0]

Offset: 0x02

Reset value: 0xc1

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

Interrupt control menu

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bit 3 | 2 - | Bit 1 | priority | Interrupt type | The interrupt source | Interrupt reset control |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 st | Receiving line status | Parity, overflow, or frame error, or dozen  Break the interrupt | Read the LSR |
| 0 | 1 | 0 | 2 nd | A significant number is received  According to the | The number of characters in FIFO is up to  The level of the trigger | FIFO has a low number of characters  In the trigger value |
| 1 | 1 | 0 | 2 nd | Receive a timeout | At least one character in FIFO, but no character in four character time  What operations, including read and write operations | Read receive FIFO |
| 0 | 0 | 1 | 3 rd | Transfer save hosting  Device is empty | The transfer save register is empty | Write data to THR or  More IIR |
| 0 | 0 | 0 | 4 th | Modem state | CTS, DSR, RI or DCD. | Read MSR |

#### FIFO control Register (FCR)

Chinese name: FIFO control register Register width: [7:0]

Offset: 0x02

Reset value: 0xc0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| but | TL | 2 | W. | Trigger value of FIFO's interrupt request  '00' - 1 byte '01' - 4 bytes  '10' - 8 bytes' 11 '- 14 bytes |
| o | Reserved | 3 | W. | reserve |
| 2 | Txset | 1 | W. | '1' clears the contents of sending FIFO and resets its logic |
| 1 | Rxset | 1 | W. | '1' clears the contents of the RECEIVED FIFO and resets its logic |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | Reserved | 1 | W. | reserve |

#### Line Control Register (LCR)

Chinese name: Line Control register Register Width: [7:0]

Offset: 0x03

Reset value: 0x03

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | dlab | 1 | RW | Frequency divider latch access bit  '1' - Access the operation divider latch  '0' - Access the normal register |
| 6 | BCB | 1 | RW | Interrupt control bit  '1' - The output of the serial port is set to 0(interrupt state). '0' - Normal operation |
| 5 | .spb | 1 | RW | Specifies the parity bit  '0' - Do not specify parity bits  '1' - Transfer and check parity bit 0 if LCR[4] bit is 1. Transfer and check parity if LCR[4] bit is 0  The check position is 1. |
| 4 | eps | 1 | RW | Parity bit selection  '0' - An odd number of 1s in each character (including data and parity bits)  '1' - An even number of 1s in each character |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | PE | 1 | RW | Parity bit enable  '0' - No parity bits  '1' - Generates parity bits on output, and determines parity bits on input |
| 2 | sb | 1 | RW | Defines the number of bits that generate the stop bits  '0' - 1 stop bit  '1' - 1.5 stop bits in 5 character length, others  The length is 2 stop bits |
| 1-0 | bec | 2 | RW | Sets the number of digits per character  '00' - 5 '01' - 6 bits  '10' - 7 '11' - 8 bits |

#### MODEM Control Register (MCR)

Chinese name: Modem Control register

Offset: 0x04

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7:5 | Reserved | 3 | W. | reserve |
| 4 | Loop | 1 | W. | Loop mode control bit  '0' - Normal operation  '1' - loop mode. In loopback mode, TXD output is always 1, output shift register directly connected to the input shift register |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
|  |  |  |  | In the device. The other links are as follows.  DTR  DSR RTS  CTS  Out1  RI  Out2  DCD |
| 3 | OUT2 | 1 | W. | Connect to DCD input in loop mode |
| 2 | The OUT1 | 1 | W. | Connect to the RI input in loop mode |
| 1 | RTSC | 1 | W. | RTS signal control bit |
| 0 | DTRC | 1 | W. | DTR signal control bit |

#### Line Status Register (LSR)

Chinese name: Line Status register Register bit width: [7:0]

Offset: 0x05

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | The ERROR | 1 | R | Error representation bit  '1' - one with at least a parity bit error, frame error, or interrupt.  '0' - No errors |
| 6 | TE | 1 | R | The transfer is empty to represent a bit  '1' - Transfer FIFO and transfer shift register are null. to |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Zero out when transmitting FIFO write data  '0' - there is data |
| 5 | TFE | 1 | R | Transmit FIFO bit null for bit representation  '1' - The current transmission OF FIFO is null, and the data written to the transmission of FIFO is cleared  '0' - there is data |
| 4 | BI | 1 | R | Interrupts indicate bits  '1' - the start bit received + data + parity + stop bits are 0, that is, there is an interrupt  '0' - No interruptions |
| 3 | FE | 1 | R | A frame error represents a bit  '1' - Received data with no stop bits  '0' - No errors |
| 2 | PE | 1 | R | Parity bit errors represent bits  '1' - An even or odd error is currently receiving data  '0' - No parity errors |
| 1 | OE | 1 | R | Data overflow represents a bit  '1' -- Data overflow  '0' - No overflow |
| 0 | Dr. | 1 | R | The received data effectively represents a bit  '0' - No data in FIFO  '1' - Data in FIFO |

When this register is read, LSR[4:1] and LSR[7] are cleared, and LSR[6:5] is writing FIFO to the transmission

According to the time clearing, LSR[0] makes a judgment on receiving FIFO.

#### MODEM Status Register (MSR)

Chinese name: Modem Status register

Offset: 0x06

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | CDCD | 1 | R | DCD input value, or connect to Out2 in loopback mode |
| 6 | CRI | 1 | R | RI input value inverse, or connected to OUT1 in loopback mode |
| 5 | CDSR | 1 | R | The inverse of the DSR input value, or is connected to the DTR in loopback mode |
| 4 | CCTS | 1 | R | The inverse of the CTS input value, or is connected to the RTS in loopback mode |
| 3 | DDCD | 1 | R | DDCD indicating a |
| 2 | TERI | 1 | R | RI edge detection. RI goes from low to high |
| 1 | DDSR | 1 | R | DDSR indicating a |
| 0 | DCTS | 1 | R | DCTS indicating a |

#### Receive FIFO count values (RFC)

Receive FIFO number register bit width: [7:0]

Offset: 0x08 Reset value: 0x00

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| away | The RFC | 8 | R | Reflects the number of valid data in the current RECEIVED FIFO |

#### Send FIFO count values (TFC)

Send FIFO number register bit width: [7:0]

Offset: 0x09 Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| away | TFC | 8 | R | Reflects the number of valid data in the current FIFO |

#### Frequency division latch

Chinese name: frequency division latch 1

Register bit width: [7:0]

Offset: 0x00

Reset value: 0x00

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

Chinese name: frequency division latch 2

Register bit width: [7:0]

Offset: 0x01

Reset value: 0x00

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

Chinese name: frequency division latch 3 register bit width: [7:0] Offset: 0x02

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| away | D\_DIV | 8 | RW | Store the fractional frequency division of a frequency divider latch |

#### The use of new registers

The newly added RECEIVING FIFO counter (RFC) can be used by the CPU to detect the number of valid data in receiving FIFO, so that the CPU can continuously read multiple data after receiving a single interrupt to improve the CPU's ability to process data received by UART.

Sending FIFO counter (TFC) can be used by THE CPU to detect the number of valid data in sending FIFO. Accordingly, the CPU can continuously send multiple data under the premise of ensuring sending FIFO without overflow, so as to improve the CPU's ability to process DATA sent by UART.

Frequency division latch 3 is used to solve the problem that the required baud rate cannot be obtained accurately by integer division alone. With reference clock 100MHz divided by 16, divided by the baud rate, the obtained quotient integer part is assigned to the frequency divider latch MSB and LSB, the fractional part is multiplied by 256 and assigned to the frequency divider latch D\_DIV.

## SPI controller

The SPI controller has the following characteristics:

* Full duplex synchronous serial port data transmission
* Supports variable length byte transfers up to 4
* Main mode support
* A mode failure generates an error flag and issues an interrupt request
* Double-buffered receiver
* Polarity and phase programmable serial clock
* SPI can be controlled in wait mode
* Support for starting from SPI
* Dual/Quad Mode SPI Flash support

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

Table 15-1 SPI controller address space distribution

|  |  |  |
| --- | --- | --- |
| **Address name** | **Address range** | **The size of the** |
| SPI Boot | 0 x1fc0\_0000 x1fd0\_0000 0 | 1 mbyte |
| SPI Memory | 0 x1d00\_0000 x1e00\_0000 0 | 16 mbyte |
| SPI Register | 0 x1fe0\_01f0 x1fe0\_01ff 0 | 16 byte |

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

The SPI Memory space can also be accessed directly from the CPU's read request, with a minimum of 1 Megabyte overlapping the SPI BOOT space.

#### Control register (SPCR)

Chinese name: Control register Register width: [7:0]

Offset: 0x00

Reset value: 0x10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | Spie | 1 | RW | Interrupt output to make the energy signal highly efficient |
| 6 | The spe | 1 | RW | The system works to make the energy signal highly efficient |
| 5 | Reserved | 1 | RW | reserve |
| 4 | MSTR | 1 | RW | Master mode selects bits, which remain at 1 |
| 3 | cpol | 1 | RW | Clock polarity |
| 2 | cpha | 1 | RW | Clock phase 1 is opposite and 0 is the same |
| 1-0 | SPR | 2 | RW | Sclk\_o frequency setting, which needs to be used with the SPRE of the SPer |

#### Status Register (SPSR)

Status register Register width: [7:0]

Offset: 0x01

Reset value: 0x05

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | spif | 1 | RW | Interrupt flag bit 1 indicates an interrupt request, write 1 to clear |
| 6 | wcol | 1 | RW | Write register overflow flag bit 1 indicates overflow, write 1 is cleared |
| when | Reserved | 2 | RW | reserve |
| 3 | wffull | 1 | RW | Write register full flag 1 indicates full |
| 2 | wfempty | 1 | RW | Write register null flag 1 indicates null |
| 1 | rffull | 1 | RW | Read register full flag 1 indicates full |
| 0 | rfempty | 1 | RW | Read register null flag 1 indicates null |

#### Data Register (TxFIFO)

Chinese name: Data Transmission Register Register width: [7:0]

Offset: 0x02

Reset value: 0x00

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

#### External register (SPER)

Chinese name: External register Register width: [7:0]

Offset: 0x03

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| but | icnt | 2 | RW | Sends an interrupt request signal after how many bytes have been transmitted  00 -- 1 byte 01-2 bytes  10-3 bytes 11-3 bytes |
| 5-2 | Reserved | 4 | RW | reserve |
| 1-0 | spre | 2 | RW | Set the frequency division ratio with the Spr |

Frequency division coefficient:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| spre  SPR | 00  00 | 00  01 | 00  10 | 00  11 | 01  00 | 01  01 | 01  10 | 01  11 | 10  00 | 10  01 | 10  10 | 10  11 |
| Frequency division coefficient | 2 | 4 | 16 | 32 | 8 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |

#### Parameter control register (SFC\_PARAM)

SPI Flash Parameter Control register Register width: [7:0]

Offset: 0x04

Reset value: 0x21

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| The log | clk\_div | 4 | RW | Clock frequency division selection (the division coefficient is the same as {SPre, SPR} combination) |
| 3 | dual\_io | 1 | RW | Use dual I/O mode, with priority over fast read mode |
| 2 | fast\_read | 1 | RW | Use quick read mode |
| 1 | burst\_en | 1 | RW | Spi Flash supports sequential address reading mode |
| 0 | memory\_en | 1 | RW | Spi Flash read enable, when invalid CSN [0] can be controlled by software. |

#### Chip Selector control register (SFC\_SOFTCS)

SPI Flash Chip Selection control Register Register width: [7:0]

Offset: 0x05

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| The log | CSN | 4 | RW | CSN pin output value |
| 3-0 | csen | 4 | RW | When is 1, the corresponding cs line is controlled by 7:4 |

#### Timing Control Register (SFC\_TIMING)

SPI Flash Timing Control Register Register Width: [7:0]

Offset: 0x06

Reset value: 0x03

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| The log | Reserved | 4 | RW | reserve |
| 3 | quad\_io | 1 | RW | 4 wire mode enablement, 1 valid |
| 2 | tFast | 1 | RW |  |
| 1-0 | it | 2 | RW | The shortest invalid time of chip selection signal of SPI Flash is calculated with clock cycle T after frequency division  00:1 t  01:2 t  10:4 t  11:8 t |

#### Custom control Register (CTRL)

SPI Flash Custom Control Register Register width: [7:0]

Offset: 0x08

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| The log | nbyte | 4 | RW | The number of bytes transmitted at one time |
| 3:2 | reserve | 2 | RW | reserve |
| 1 | nbmode | 1 | RW | Multibyte transfer mode |
| 0 | start | 1 | RW | Start multi-byte transmission and reset automatically after completion |

#### Custom command register (CMD)

SPI Flash Custom command register Register width: [7:0]

Offset: 0x09

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| away | CMD | 8 | RW | Sets the command sent to SPI Flash |

#### Custom data Register 0 (BUF0)

SPI Flash custom data register 0

Register bit width: [7:0]

Offset: 0x0a

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| away | buf0 | 8 | RW | When a write command is sent to the SPI, the register is configured to send the first  Three bytes of data; When a read command is sent to the SPI, this register stores the first data read back. |

#### Custom Data Register 1 (BUF1)

SPI Flash Custom data Register 1

Register bit width: [7:0]

Offset: 0x0b

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| away | buf1 | 8 | RW | When a write command is sent to the SPI, the register configures the second byte of data sent; This register when a read command is sent to the SPI  Store the second read data. |

#### Custom timing register 0 (TIMER0)

SPI Flash custom timing register 0

Register bit width: [7:0]

Offset: 0x0c

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| away | time0 | 8 | RW | The lower 8 bits of the time value required by the custom command |

#### Custom Timing register 1 (TIMER1)

SPI Flash Custom Timing register 1

Register bit width: [7:0]

Offset: 0x0d

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| away | time1 | 8 | RW | The middle eight bits of the required time value for the custom command |

#### Custom Timing register 2 (TIMER2)

SPI Flash Custom Timing register 2

Register bit width: [7:0]

Offset: 0x0e

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **access** | **describe** |
| away | time2 | 8 | RW | The higher 8 bits of the required time value for the custom command |

#### SPI two wire and four wire guide

In addition to the traditional single-wire mode, SPI controller also supports dual mode and quad mode to boot from SPI Flash. Setting the DUal\_IO register enables the SPI controller to go into two-wire mode, while setting the Quad\_IO register enables the SPI controller to go into four-wire mode. Can be in the BIOS code in the first few instructions to increase the configuration code of the two registers, after the completion of the configuration of the controller is in accordance with the configuration of the corresponding working mode to take the finger, so as to improve the boot speed.

Note that some SPI FLASH does not enable four-wire mode by default, or you need to configure time-related parameters (such as dummy clocks) in four-wire mode. To increase the applicability of the SPI controller to various FLASH, this controller adds custom registers (0x8-0xe). The specific usage is as follows:

* + - 1. Set the custom command register (CMD) (0x9), which is the command sent to SPI FLASH;
      2. If SPI FLASH requires that the command sent take some time to complete, configure the wait time in the custom timing registers Timer0-Timer2 (0xC-0xe), otherwise the registers remain at the default value of 0;
      3. If configuration information is written to SPI FLASH, it needs to be written to the custom data register BUf0-BUF1 (0XA-0xb); If configuration information is read to SPI FLASH, these registers store the values read back.
      4. Configure the custom control register CTRL[7:1] where CTRL[1](NBMode) represents the multi-byte transmission mode, and the number of bytes transferred is given by CTRL[7:4](Nbyte);
      5. Configure the custom control register CTRL[0] to start the transfer.

In general, the registers that need to be configured are located in FLASH's non-volatile storage area, so the above configuration only needs to be configured once.

## I2C controller

This chapter gives the detailed description and configuration of I2C. The system chip integrates I2C interface, which is mainly used to realize data exchange between two devices. I2C bus is a serial bus composed of data line SDA and clock SCL, which can send and receive data. Bidirectional transmission between devices with a maximum transfer rate of 400kbps.

The integrated I2C controller in the Loongson 3A4000 can be used as either a master device or a slave device, switching between the two modes by configuring internal registers. When used as a slave device, it is only used to read the chip internal temperature, and the address of the slave device is specified by the register SLV\_CTRL[6:0].

The I2C0 controller register has a physical base address of 0x1FE00120. I2C1 controller register physical address base is 0x1FE00130. The specific internal registers are described below.

#### Divider latch Low byte Register (PRERlo)

Frequency division latch low byte register Register width: [7:0]

Offset: 0x00

Reset value: 0xFF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| away | PRERlo | 8 | RW | The lower 8 bits of the divider latch |

#### Divider latch High Byte Register (PRERhi)

Frequency division latch high byte register Register width: [7:0]

Offset: 0x01

Reset value: 0xFF

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

Assuming that the value of the frequency divider latch is Prescale and the input frequency from the LPB bus PCLK clock is clock\_A and the output frequency of the SCL bus is clock\_S, the following relation should be satisfied

Prcescale = clock\_a/(4 \* clock\_s) - 1

#### Control register (CTR)

Chinese name: Control register Register width: [7:0] Offset: 0x02

Reset value: 0x20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | EN | 1 | RW | The module works so that the energy bit is 1.  0 operates on the divider register |
| 6 | IEN | 1 | RW | Interrupts with an energy bit of 1 turn interrupts on |
| 5 | MST\_EN | 1 | RW | Module master and slave selection  0: Slave mode  1: Master mode |
| 4:0! | Reserved | 5 | RW | reserve |

#### Send data Register (TXR)

Chinese name: send register Register width: [7:0] Offset: 0x03

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7:1 | The DATA | 7 | W. | Store the next byte to be sent |
| 0 | DRW | 1 | W. | When data is transmitted, this bit holds the lowest bit of data  This bit indicates the read/write status when the address is transmitted |

#### Receive data Register (RXR)

Chinese name: receive register Register bit width: [7:0] Offset: 0x03

Complex bit value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| away | RXR | 8 | R | Store the last byte received |

#### Command control Register (CR)

Chinese name: command register Register bit width: [7:0] Offset: 0x04

Complex bit value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | The STA | 1 | W. | Generate the START signal |
| 6 | STO | 1 | W. | Generate STOP signal |
| 5 | RD | 1 | W. | Generate read signal |
| 4 | WR | 1 | W. | Generate write signal |
| 3 | ACK | 1 | W. | Generate an answer signal |
| 2:1 | Reserved | 2 | W. | reserve |
| 0 | IACK | 1 | W. | Generate an interrupt acknowledge signal |

The hardware will reset automatically after I2C sends the data. For these bit-reads, the controller always reads back '0' when bit 3 is 1, which means that the controller does not send an ACK at the end of this transmission, but sends an ACK at the end of this transmission.

#### Status Register (SR)

Chinese name: status register Register bit width: [7:0] Offset: 0x04

Complex bit value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | RxACK | 1 | R | Received reply bit  1 No answer |
|  |  |  |  | 0 Received reply bit |
| 6 | Busy | 1 | R | I2c bus busy flag bit  The bus is busy  0 bus idle |
| 5 | AL | 1 | R | Position 1 when the I2C core loses control of the I2C bus |
| 4-2 | Reserved | 3 | R | reserve |
| 1 | TIP # | 1 | R | Indicates the process of transmission  One indicates that data is being transmitted  Zero means that the data has been transferred |
| 0 | The IF | 1 | R | Interrupt flag bit, one data transmission over, or another device  Initiate data transfer at location 1 |

#### Control register from device (SLV\_CTRL)

Chinese name: From device control register Register width: [7:0]

Offset: 0x07

Reset value: 0x00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A domain | A domain name | A wide | access | describe |
| 7 | SLV\_EN | 1 | WR | From the mode enable, when MST\_EN is 0, can be used for  Reset from machine internal logic |
| lost | SLV\_ADDR | 7 | WR | From mode I2C address, available through software configuration |

## 3A3000 kernel compatibility

In order to achieve backward compatibility of the Linux kernel from 3A3000 onwards, some modifications to the existing kernel must be made according to the implementation specification of the chip.

In order to achieve compatibility with the 3A3000 kernel, the 3A4000 chip not only implements a set of configuration methods according to the new specification, but also needs to support the mechanism widely used in the current kernel.

The following is an introduction to the 3A4000 kernel in terms of kernel compatibility and new feature support.

## Compatible with 3A3000 kernel

In order to be compatible with the 3A3000 kernel, the following changes must be made to the kernel.

#### Processor feature recognition method

In the case of MIPS processors, the kernel does not use a common method to identify the different characteristics of the processor. Instead, it USES PRID to distinguish the processor model and then performs different processing according to the processor model in different situations. Because at present, the kernel only determines and processes the existing processor model, and there is no default processing method for the new processor that has not yet been implemented, which results in many underlying code that does not have corresponding implementation when running on the new processor.

To solve this problem, starting from 3A4000, a set of processor configuration instructions, as well as processor characteristic identification instructions, are implemented to standardize the hardware and software interface. It can be accessed through the processor configuration instructions or through the base address 0x3ff00000. Register CSR[offset address][bit].

This register identifies some software-related processor features for the software to view before enabling a specific function. The register of

Offset 0x0008. Remember to CSR [0 x08].

Table 16-1 Chip characteristic register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 0 | Centigrade | R | 1 'b1 | CSR x428 [0] is effective |
| 1 | The Node counter | R | 1 'b1 | CSR x408 [0] is effective |
| 2 | MSI | R | 1 'b1 | MSI available |
| 3 | EXT\_IOI | R | 1 'b1 | EXT\_IOI available |
| 4 | IPI\_percore | R | 1 'b1 | IPI is sent through the CSR private address |
| 5 | Freq\_percore | R | 1 'b1 | Adjust the frequency through the CSR private address |
| 6 | Freq\_scale | R | 1 'b0 | Dynamic frequency division is available |
| 7 | DVFS\_v1 | R | 1 'b0 | Dynamic FM V1 is available |
| 8 | Tsensor | R | 1 'b0 | Temperature sensor available |

#### Current kernel change method

In the current 3.10 kernel, there are six pieces of PRID code for feature recognition, five of which will need to be modified to support future processors.

The five functions are as follows:

|  |  |  |
| --- | --- | --- |
| **function** | **The path** | **describe** |
| cpu\_probe\_loongson | Arch/MIPS/kernel/CPU - probe. C | Carry out the identification of chip type |
| loonson\_cpu\_temp | Driver/platform/MIPS/cpu\_hwmon. C | Read the on-chip temperature sensor |
| play\_dead | The arch/MIPS/loongson/loongson - 3 / SMP. C | Dynamic switching core support |
| init\_node\_counter\_clocksource | The arch/MIPS/loongson/loongson - 3 / node\_counter. C | Enable on-chip clock source |
| ls7a\_init\_irq | Arch/MIPS/loongson/loongson - 3 / ls7a - irq. C | Enable MSI interrupt |

* + - 1. cpu\_probe\_loongson

This function is used for the identification of chip model. It is necessary to add the code to the original default condition that USES the processor configuration instruction to identify the manufacturer name, chip name and assign value to the corresponding data structure. The corresponding registers are shown below.

Vendor name register. CSR x0010 [0].

Table 16-2 The vendor name register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 63:0 | Vendor | R | 0 x6e6f7367\_6e6f6f4c | The string "Loongson" |

Chip name register. CSR x0020 [0].

Table 16-3 Chip Name register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 63:0 | ID | R | 0 x00003030\_30344133 | The string "3 a4000" |

* + - 1. loongson\_cpu\_temp

This function is used to read the temperature sensor on the chip, and a new processing should be added to the original default condition. First, determine whether there is an on-chip temperature sensor according to CSR[0x8][0] and decide whether to read the on-chip temperature register CSR[0x428] using the processor configuration instruction.

* + - 1. play\_dead

This function is used for dynamic switching cores and will need to be changed significantly for future processors. The original function calls different functions according to PRID to carry out targeted Cache brush operation and close the Cache. Instead, according to the specification of the second volume of MIPS manual, read the route and size configuration of ICACHE/DCACHE/VCACHE from the relevant CP0 register, carry out corresponding Cache brush operation and then close the Cache. It should be noted that the Secondary cache in CP0. Config2 refers to the final on-chip cache, or scache, when the cache configuration information is read through the CP0 register in all processors of the Loongson 2 and Loongson 3 series. External Cache can be represented by cp0.config2, an external Cache. When closing, it is necessary to decide whether to use CSR[0x1050][3] for closing or the corresponding bit of 0x3FF001D0 for closing according to whether CSR[0x420][23] is 1.

* + - 1. init\_node\_counter\_clocksource

This function is used to initialize the clock source on the chip, and a default condition is added to determine if there is node\_counter based on THE CSR[0x8][1]. You also need to add a parameter that does not modify certain values.

* + - 1. ls7a\_init\_irq

This function is used to determine whether or not to use MSI interrupts and requires an additional default condition to determine whether or not MSI support is available based on the CSR[0x8][2], in addition to the various known processors.

## New feature support

To use the new features offered by the 3A4000 processor in the kernel, you can identify or enable it in the following ways. Only the parts that can improve the performance of the system are described here, but new mechanisms such as sending inter-core interrupts through the CSR instruction, because of the requirement of compatibility with the 3A3000 processor, actually have to use the existing specific address access mode, there is no need for CSR support, but increase the software overhead.

#### Processor characteristic recognition

The new features are identified by the CSR register directive, and in order to support the new features, you need to consider not supporting the processor

Configure how the instructions are processed by the old processor. One is to increase the judgment of PRID at all locations that need to be recognized for characteristics, and to process all existing Prids; the other is to increase the processing of abnormal instructions. When the reserved instruction is recognized as processor configuration instruction, the correct return value can be built according to the instruction content and PRID.

#### Extended interrupt mode

To enable extension of interrupt mode in the kernel, you need to set this up in the following order.

* + - 1. Extended interrupt mode support is identified by CSR[0x8][3].
      2. The external interrupt transform register of the HT controller that is expected to support extended interrupt mode needs to be configured with the correct value in PMON. Its registers are defined as follows and set to the following values:

INT\_trans\_en = 0// enables control using the CSR register. Both CSR[0x420][48] enable extended interrupt mode, which is not enabled by default in PMON and turned on by the kernel configuration CSR[0x420][48]

INT\_trans\_allow = 1// allows external enablement to interrupt the conversion function

INT\_trans\_addr = 0x1000000001140// Extend interrupt register address, see 14.3.3. INT\_trans\_cache = 0//Uncache mode

Offset: 0x270

Reset value: 0x00000000

HT RX INT TRANS Lo

Table 16-4 HT RX INT TRANS LO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| does | INT\_trans\_addr [also] | 28 | 0 x0 | R/W | Interrupt address conversion low order |
| 3-0 | Reserved | 4 | 0 x0 | R | reserve |

Offset: 0x274

Reset value: 0x00000000

Name: HT RX INT TRANS Hi

Table 16-5 HT RX INT TRANS Hi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A domain** | **A domain name** | **A wide** | **Reset value** | **access** | **describe** |
| 31 | INT\_trans\_en | 1 | 0 x0 | R/W | Interrupt transformation enablement |
| 30 | INT\_trans\_allow | 1 | 0 x0 | R/W | Interrupt transformation allows |
| then | INT\_trans\_cache | 4 | 0 x0 | R/W | Interrupt the conversion Cache field |
| 25:0 | INT\_trans\_addr [58:32] | 26 | 0 x0 | R/W | Interrupt address conversion to high level |

* + - 1. The kernel first identifies the extended interrupt mode support through CSR[0x8][3] and then enables it to extend the interrupt mode through the register CSR[0x420][48]. The base address is 0x1fe00000 and the offset address is 0x0420.

Table 16-6 Register Settings for other functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A domain** | **The field name** | **access** | **Reset value** | **describe** |
| 48 | EXT\_INT\_en | RW | 0 x0 | Extend IO interrupt enablement |

* + - 1. Sets the appropriate routing and internal control for extended interrupt mode.

## Configure register instruction debugging support

In principle, the register instruction is not accessible across the chip when it is used, but in order to meet the need for debugging and other functions, it is supported by multiple register addresses. It is worth noting that such registers can only be written, not read.

In addition to the original intercore interrupt and other registers that can be accessed across the chip, all such registers and addresses are as follow

|  |  |  |  |
| --- | --- | --- | --- |
| The name of the | offset | permissions | describe |
| IPI\_Send | 0 x1040 | send | 32-bit interrupt distribution register  [31] Wait for the completion mark, and wait for the interruption to take effect when set to 1  [there]  [25:16] Processor core  [language]  [4:0] interrupt vector sign, corresponding to the vector in IPI\_Status |
| Mail\_Send | 0 x1048 | send | The 64-bit MailBox cache register  63: [32] MailBox data  [31] Waits for the completion mark. When set 1, it waits for the mask in which the data is written [30:27]. Each bit means that the byte corresponding to the 32-bit data is not really written to the target address  Reserve the processor core number |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | [language]  [and] the MailBox   1. -Mailbox0 low 32 bits 2. -Mailbox0 is 32 bits high 3. -Mailbox1 low 32 bits 4. -Mailbox1 is 32 bits high 5. - MailBox2 low 32 bits 6. -Mailbox2 is 32 bits high 7. -Mailbox3 low 32 bits 8. - MailBox4 is 32 bits high   [1:0] |
| FREQ\_Send | 0 x1058 | send | 32-bit frequency enable register  [31] Wait for the completion mark, and wait for the interruption to take effect when set to 1  [there]  [25:16] Processor core  [language]  Writes to the corresponding processor core private frequency configuration register.  CSR x1050 [0] |
| ANY\_Send | 0 x1158 | send | 64-bit registers access registers  [63:32] Writes data  [31] Waits for the completion flag. When set 1, it waits for the mask whose interrupt takes effect [30:27] and writes the data. Each bit means that the byte corresponding to the 32-bit write data will not really write the target address  [26] Preserves [25:16] the target processor core number  [15:0] write the register offset address |