LOONGSON

# 龙芯 3B1500 处理器用户手册

## 上册　多核处理器架构与寄存器描述

V1.7

***2016 年 4 月***

*龙芯中科技术有限公司*



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

# 版权声明



**免责声明**

本文档仅提供阶段性信息，所含内容可根据产品的实际情况随时更新，恕不另行通知。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

**龙芯中科技术有限公司**

Loongson Technology Corporation Limited

地址：北京市海淀区中关村环保科技示范园龙芯产业园 2 号楼

Building No.2, Loongson Industrial Park,

Zhongguancun Environmental Protection Park, Haidian District, Beijing

电话(Tel)：010-62546668

传真(Fax)：010-62600826

# 阅读指南

《龙芯 3B1500 处理器用户手册》分为上册和下册。

《龙芯 3B1500 处理器用户手册》上册，介绍龙芯 3B1500 多核处理器架构，主要包括多核处理器架构与寄存器描述；

《龙芯 3B1500 处理器用户手册》下册，从系统软件开发者角度详细介绍龙芯 3B1500 所采用的 GS464 高性能处理器核。

# 修订历史

| 文档更新记录 | 文档名： | 龙芯 3B1500 处理器用户手册——上册 |
|---|---|---|
| | 版本号： | V1.7 |
| | 创建人： | 芯片研发部 |
| | 创建日期： | 2016-04-01 |

## 更新历史

| 序号. | 更新日期 | 版本号 | 更新内容 |
|---|---|---|---|
| 1 | 2012-05-14 | V1.0 | 初稿完成 |
| 2 | 2012-08-02 | V1.1 | 完善部分寄存器描述 |
| 3 | 2012-10-02 | V1.2 | 完善部分寄存器描述 |
| 4 | 2012-11-11 | V1.3 | 修改图表目录格式<br>增加第八章 全局时钟 |
| 5 | 2013-05-09 | V1.4 | 增加 HT 寄存器描述<br>更新温度传感器描述<br>修改矩阵加速器寄存器地址描述 |
| 6 | 2013-06-16 | V1.5 | 增加时钟软件配置说明 |
| 7 | 2013-10-28 | V1.6 | 增加 LS3B1500F 内容 |
| 8 | 2016-04-01 | V1.7 | 根据 LS3B1500G 调整部分处理器核相关描述 |

手册信息反馈: service@loongson.cn

# 目 录

# 图目录

# 表目录

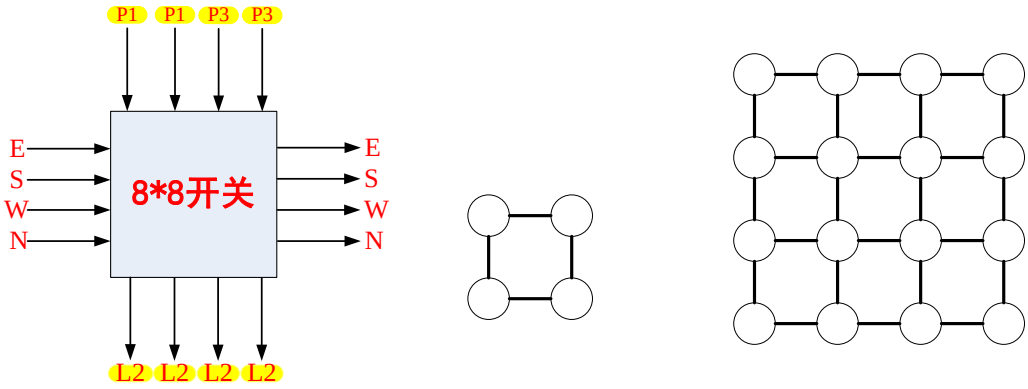
# 1 概述

## 1.1 龙芯系列处理器介绍

龙芯处理器主要包括三个系列。龙芯 1 号处理器及其 SOC 系列主要面向嵌入式应用，龙芯 2 号超标量处理器及其 SOC 系列主要面向桌面应用，龙芯 3 号多核处理器系列主要面向服务器和高性能机应用。根据应用的需要，其中部分龙芯 2 号也可以面向部分高端嵌入式应用，部分低端龙芯 3 号也可以面向部分桌面应用。上述三个系列将并行地发展。

龙芯 3 号多核系列处理器基于可伸缩的多核互连架构设计，在单个芯片上集成多个高性能处理器核以及大量的 SCache，还通过高速 I/O 接口实现多芯片的互连以组成更大规模的系统。

龙芯 3 号采用的可伸缩互连结构如下图 1-1 所示。龙芯 3 号片内及多片系统均采用二维 mesh 互连结构，其中每个结点由 8*8 的交叉开关组成，每个交叉开关连接四个处理器核以及四个 SCache，并与东（E）南（N）西（W）北（N）四个方向的其他结点互连。因此，2*2 的 mesh 可以连接 16 个处理器核，4*4 的 mesh 可以连接 64 个处理器核。




龙芯3号结点及二维互连结构，（a）结点结构，（b）2*2的mesh网络连接16个处理器，（c）4*4的mesh网络连接64个处理器。

图 1-1 龙芯 3 号系统结构

龙芯 3 号的结点结构如下图 1-2 所示。每个结点有两级 AXI 交叉开关连接处理器、片上共享高速缓存（简称 SCache）、内存控制器以及 IO 控制器。其中第一级 AXI 交叉开关（称为 X1 Switch，简称 X1）连接处理器和 SCache。第二级交叉开关（称为 X2 Switch，简称 X2）连接 SCache 和内存控制器。

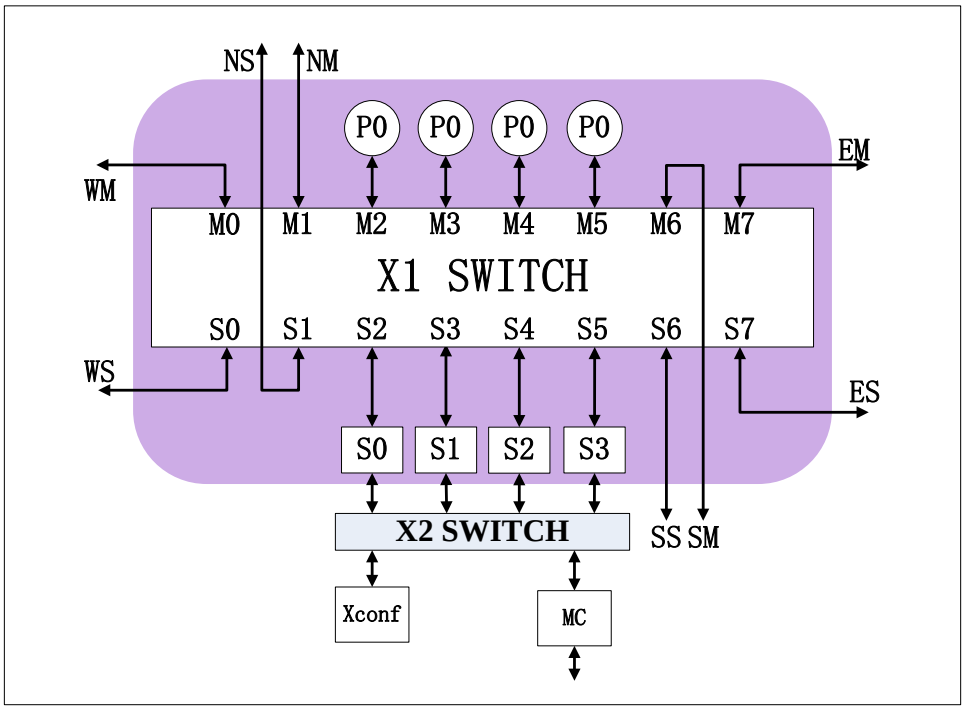



图 1-2 龙芯 3 号节点结构

在每个结点中，最多 8*8 的 X1 交叉开关通过四个 Master 端口连接四个 GS464 处理器核（图中 P0、P1、P2、P3），通过四个 Slave 端口连接统一编址的四个 SCache 块（图中 S0、S1、S2、S3），通过四对 Master/Slave 端口连接东、南、西、北四个方向的其他结点或 IO 结点（图中 EM/ES、SM/SS、WM/WS、NM/NS）。

X2 交叉开关通过四个 Master 端口连接四个 SCache，至少一个 Slave 端口连接一个内存控制器，至少一个 Slave 端口连接一个交叉开关的配置模块（Xconf）用于配置本结点的 X1 和 X2 的地址窗口等。还可以根据需要连接更多的内存控制器和 IO 端口等。

龙芯 3 号的互连系统只是定义上层协议，不对传输协议的实现做具体规定，因此，结点之间的互连即可以采用片上网络进行实现，也可以通过 I/O 控制链路实现多芯片的互连。以一个 4 结点 16 核系统为例，既可以通过 4 片 4 核芯片组成，也可以通过 2 片 8 核芯片，或基于一个单芯片 4 节点 16 核芯片组成。由于互连系统的物理实现对软件透明，上述 3 种配置的系统可以运行相同的操作系统进行。

## 1.2 龙芯 3B1500 简介

龙芯 3B1500 是龙芯 3 号多核处理器系列的第三款产品，是一个配置为双节点的 8 核处理器，采用 32nm 工艺制造，最高工作主频为 1.2GHz（低电压）/1.5GHz（高电压），主要技术特征如下：

- 片内集成 8 个 64 位的四发射超标量 GS464 高性能处理器核；
- 每个处理器核频率单独可设；

- 片内集成 8 MB 的分体共享共享高速缓存（SCache，由 8 个体模块组成，每个体模块容量为 1M Byte）；
- 通过目录协议维护多核及 I/O DMA 访问的 Cache 一致性；
- 片内集成 2 个 64 位 667MHz 的 DDR2/3 控制器，支持 DDR3-1333；
- 片内集成 2 个 16 位 800MHz 的 HyperTransport 控制器，最高支持 1600MHz 总线；
- 每个 16 位的 HT 端口可以拆分成两个 8 路的 HT 端口使用；
- 片内集成 32 位 33MHz PCI（非标准电压）；
- 片内集成 1 个 LPC（非标准电压）、2 个 UART、1 个 SPI、16 路 GPIO 接口；

龙芯 3B1500 号芯片整体架构基于两级互连实现，结构如下图 1-3 所示。

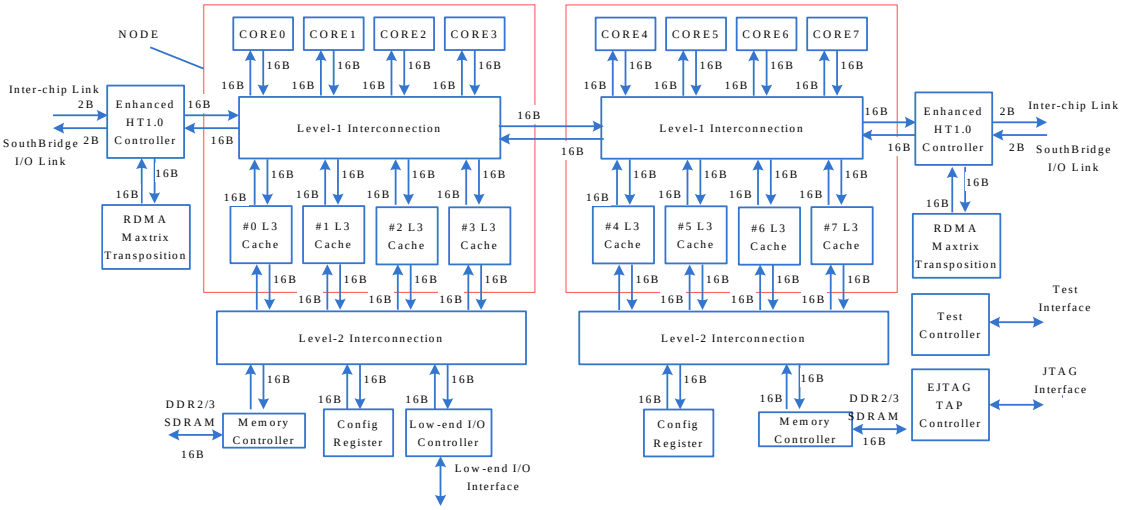



图 1-3 龙芯 3B1500 芯片结构

第一层互连采用两个相连的 6*6 交叉开关，分别用于连接四个 GS464 处理器核（作为主设备）、四个 SCache 模块（作为从设备）、一个 HT 端口（每个端口使用一个 Master 和一个 Slave）以及相邻的节点。一级互连开关连接的 16 位 HT 控制器还可以作为两个 8 位的 HT 端口使用。HT 控制器和一级互联开关相连，其中的 DMA 控制器负责 HT 总线 IO 请求的 DMA 控制并负责片间一致性的维护。龙芯 3B1500 的 DMA 控制器还可以通过配置实现预取、矩阵转置及数据搬运功能。

第二级互连采用两个独立的 5*4 交叉开关，分别连接 4 个 SCache 模块（作为主设备），DDR2 内存控制器、低速高速 I/O（包括 PCI、LPC、SPI 等）（节点二没有低速 IO）以及芯片内部的控制寄存器模块。

上述两级互连开关都采用读写分离的数据通道，数据通道宽度为 128bit，工作在与处理器核相同的频率，用以提供高速的片上数据传输。

基于龙芯 3B1500 可扩展互联架构，两片 8 核龙芯 3B1500 可以通过 HT 端口连接构成两芯片 16 核的 SMP 结构。

# 2 芯片配置与控制

## 2.1 芯片工作模式

龙芯 3B1500 有两种工作模式：

- 单芯片模式。系统只集成 1 片龙芯 3B1500，是一个两节点的非均匀访存多处理器系统（CC-NUMA）。
- 多芯片互连模式。系统中包含 2 片龙芯 3B1500，通过龙芯 3B1500 的 HT0 端口进行互连，是一个四节点的非均匀访存多处理器系统（CC-NUMA）。

## 2. 2 控制引脚说明

龙芯 3B1500 的控制引脚包括 DO_TEST、ICCC_EN、NODE_ID[1:0]、CLKSEL[15:0]、PCI_CONFIG。

表 2-1 控制引脚说明

<table>
<tr><th>信号</th><th>上下拉</th><th>作用</th></tr>
<tr><td>DO_TEST</td><td>上拉</td><td>1’b1 表示功能模式<br>1’b0 表示测试模式</td></tr>
<tr><td>ICCC_EN</td><td>下拉</td><td>1’b1 表示多芯片一致性互联模式<br>1’b0 表示单芯片模式</td></tr>
<tr><td>NODE_ID[1:0]</td><td></td><td>在多芯片一致性互连模式下表示处理器号<br>主处理器设为 2’b00；<br>从处理器设为 2’b10</td></tr>
<tr><td colspan="3">上电时钟控制</td></tr>
<tr><td>CLKSEL[15:0]</td><td colspan="2">HT 时钟控制<br><table><tr><th>信号</th><th>作用</th></tr><tr><td>CLKSEL[15]</td><td>1’b1：表示 HT 控制器时钟采用 CLKSEL[14:10]控制<br>1’b0：初始倍频为 1 倍频，可由软件进行重新配置</td></tr><tr><td>CLKSEL[14:13]</td><td>2’b00：控制器时钟为 PHY 时钟频率（PHY 时钟 bypass 时为 3.2G）除以 2<br>2’b01：控制器时钟为 PHY 时钟频率（PHY 时钟 bypass 时为 3.2G）除以 4<br>2’b10：控制器时钟为 PHY 时钟频率（PHY 时钟 bypass</td></tr></table></td></tr>
</table>

| | |
|---|---|
| | 时为 3.2G）除以 8<br>2’b11：控制器时钟取决于 PCICONF[7]： 1’b1 普通输入时钟 100MHz，1’b0 差分输入时钟 200MHz |
| CLKSEL[12:10] | 3’b000：PHY 时钟为 800M（HT 总线 200/400）<br>3’b001：PHY 时钟为 1.2G（HT 总线 200/300/600）<br>3’b010：PHY 时钟为 1.6G（HT 总线 200/400/800）<br>3’b011：PHY 时钟为 2.0G（HT 总线 200/500/1000）<br>3’b100：PHY 时钟为 2.4G（HT 总线 200/300/400/600/1200）<br>3’b101：PHY 时钟为 2.8G（HT 总线 200/1400）<br>3’b110：PHY 时钟为 3.2G（HT 总线 200/400/800/1600）<br>3’b111：PHY 时钟取决于 PCICONF[7]： 1’b1 普通输入时钟 100MHz，1’b0 差分输入时钟 200MHz |

（PCICONF[7]：1’b1 普通输入时钟 100MHz，
1’b0 差分输入时钟 200MHz）

**DDR 的时钟频率要求**

MEMCLK（主板输入的时钟） 必须在 25MHz ~ 200MHz 之间 mem_clock 是最后内存的运行频率，可以参考下表的倍频系数。

**内存控制器时钟控制**

| CLKSEL[9:6] | 倍频系数 | CLKSEL[9:6] | 倍频系数 |
|---|---|---|---|
| 4’b1111 | 1 | 4’b1110 | 26 |
| 4’b1101 | 12 | 4’b1100 | 24 |
| 4’b1011 | 11 | 4’b1010 | 22 |
| 4’b1001 | 10 | 4’b1000 | 20 |
| 4’b0xxx | 初始倍频为 1 倍频，可由软件进行重新配置 | | |

**处理器核及节点的时钟频率要求**

SYSCLK（主板输入的时钟）必须在 25MHz ~ 200MHz 之间 Core_clock 是最后处理器核的运行频率，可以参考下表的倍频系数。

**处理器核及节点时钟控制**

| CLKSEL[5:0] | 处理器核倍频系数 | CLKSEL[5:0] | 节点倍频系数 |
|---|---|---|---|
| 6’b110xx0 | 40 | 6’b1xx100 | 40 |

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>6’b101xx0</td><td>32</td><td>6’b1xx010</td><td>32</td></tr>
<tr><td>6’b100xx0</td><td>24</td><td>6’b1xx000</td><td>24</td></tr>
<tr><td>6’b110xx1</td><td>18</td><td>6’b1xx101</td><td>18</td></tr>
<tr><td>6’b101xx1</td><td>16</td><td>6’b1xx011</td><td>16</td></tr>
<tr><td>6’b100xx1</td><td>12</td><td>6’b1xx001</td><td>12</td></tr>
<tr><td>6’b111xx0</td><td>2</td><td>6’b1xx110</td><td>2</td></tr>
<tr><td>6’b111xx1</td><td>1</td><td>6’b1xx111</td><td>1</td></tr>
<tr><td>6’b0xxxxx</td><td colspan="3">初始倍频为 1 倍频，可由软件进行重新配置</td></tr>
</table>
</td></tr>
<tr><td>PCI_CONFIG[7:0]</td><td>IO 配置控制<br>7 1’b1 普通输入时钟 100MHz（HTCLK），<br>1’b0 差分输入时钟 200MHz（HTnCLKp/n）<br>6:5 PCIX 总线速度选择*<br>4 PCIX 总线模式选择<br>3 PCI 主设备模式<br>2 1’b1 系统从 PCI 空间启动<br>1’b0 由 GPIO[0]决定启动空间<br>1 使用外部 PCI 仲裁<br>注*:
<table>
<tr><td>6</td><td>5</td><td>4</td><td>PCIX 总线模式</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCI 33</td></tr>
</table>
</td></tr>
<tr><td>GPIO[1:0]</td><td>1 芯片内时钟延迟控制使能（调试模式），默认情况需要下拉<br>0 芯片启动设置，上拉表示从 SPI Flash 取指，下拉表示从 LPC Flash 取指</td></tr>
</table>

## 2.3 Cache 一致性

龙芯 3B1500 由硬件维护处理器以及通过 HT 端口接入的 I/O 之间的 Cache 一致性，但硬件不维护通过 PCI 接入到系统中的 I/O 设备的 Cache 一致性。在驱动程序开发时，对通过 PCI 接入的设备进行 DMA（Direct Memory Access）传输时，需要由软件进行 Cache 一致性维护。

## 2.4 系统节点级的物理地址空间分布

龙芯 3 号系列处理器的系统物理地址分布采用全局可访问的层次化寻址设计，以保证系统开发的扩展兼容。整个系统的物理地址宽度为 48 位。按照地址的高 4 位，整个地址空间被均匀分布到 16 个结点上，即每个结点分配 44 位地址空间。

龙芯 3B1500 采用双节点 8 核配置，因此龙芯 3B1500 芯片集成的 DDR 内存控制器、

HT 总线、PCI 总线的对应地址都包含在从 0x0（含）至 0x2000_0000_0000（不含）的 44 位地址空间内，具体各设备地址分布请参见后续相关章节。

表 2-2 节点级的系统全局地址分布

| 节点号 | 地址[47:44]位 | 起始地址 | 结束地址 |
|---|---|---|---|
| 0 | 0 | 0x0000_0000_0000 | 0x1000_0000_0000 |
| 1 | 1 | 0x1000_0000_0000 | 0x2000_0000_0000 |
| 2 | 2 | 0x2000_0000_0000 | 0x3000_0000_0000 |
| 3 | 3 | 0x3000_0000_0000 | 0x4000_0000_0000 |
| 4 | 4 | 0x4000_0000_0000 | 0x5000_0000_0000 |
| 5 | 5 | 0x5000_0000_0000 | 0x6000_0000_0000 |
| 6 | 6 | 0x6000_0000_0000 | 0x7000_0000_0000 |
| 7 | 7 | 0x7000_0000_0000 | 0x8000_0000_0000 |
| 8 | 8 | 0x8000_0000_0000 | 0x9000_0000_0000 |
| 9 | 9 | 0x9000_0000_0000 | 0xa000_0000_0000 |
| 10 | 0xa | 0xa000_0000_0000 | 0xb000_0000_0000 |
| 11 | 0xb | 0xb000_0000_0000 | 0xc000_0000_0000 |
| 12 | 0xc | 0xc000_0000_0000 | 0xd000_0000_0000 |
| 13 | 0xd | 0xd000_0000_0000 | 0xe000_0000_0000 |
| 14 | 0xe | 0xe000_0000_0000 | 0xf000_0000_0000 |
| 15 | 0xf | 0xf000_0000_0000 | 0x1_0000_0000_0000 |

在每个节点的内部，44 位地址空间又进一步均匀分布给结点内连接的可能最多 8 个设备。其中低 43 位地址由 4 个 SCache 模块所拥有，高 43 位地址则进一步按地址的[43:42]位分布给连接在 4 个方向端口的设备上。根据芯片和系统结构配置的不同，如果某端口上没有连接从设备，则对应的地址空间是保留地址空间，不允许访问。

表 2-3 节点内的地址分布

| 设备 | 地址[43:41]位 | 节点内起始地址 | 节点结束地址 |
|---|---|---|---|
| SCache | 0,1,2,3 | 0x000_0000_0000 | 0x800_0000_0000 |
| 东 | 4 | 0x800_0000_0000 | 0xa00_0000_0000 |
| 南 | 5 | 0xa00_0000_0000 | 0xc00_0000_0000 |
| 西 | 6 | 0xc00_0000_0000 | 0xe00_0000_0000 |
| 北 | 7 | 0xe00_0000_0000 | 0x1000_0000_0000 |

例如节点 0 的东端口设备的基地址为 0x0800_0000_0000,节点 1 的东端口设备的基地址为 0x1800_0000_0000，依次类推。

不同于方向端口的映射关系，龙芯 3B1500 可以根据实际应用的访问行为，来决定 SCache 的交叉寻址方式。节点内的 4 个 SCache 模块一共对应 43 位地址空间，而每个 S 模块所对应的地址空间根据地址位的某两位选择位确定，并可以通过软件进行动态配置修改。系统中设置了名为 SCID_SEL 的配置寄存器来确定地址选择位，如下表所示。在缺省情况下采用[6:5]地位散列的方式进行分布，即地址[6:5]两位决定对应的 SCache 编号。节点 0 上的寄存器地址 0x3FF00400，节点 1 上的寄存器地址为 0x1000_3FF04400。两个节点上的散列方式可以分别配置。如果使用双芯片互连系统，节点 2 的寄存器地址为 0x2000_3FF08400，节点 3 的寄存器地址为 0x30003FF0C400。

简单的说，不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

表 2-4 共享缓存散列配置

| SCID_SEL | 地址位选择 | SCID_SEL | 地址位选择 |
|---|---|---|---|
| 4'h0 | 6: 5 | 4'h8 | 23:22 |
| 4'h1 | 9: 8 | 4'h9 | 25:24 |
| 4'h2 | 11:10 | 4'ha | 27:26 |
| 4'h3 | 13:12 | 4'hb | 29:28 |
| 4'h4 | 15:14 | 4'hc | 31:30 |
| 4'h5 | 17:16 | 4'hd | 33:32 |
| 4'h6 | 19:18 | 4'he | 35:34 |
| 4'h7 | 21:20 | 4'hf | 37:36 |

## 2.5 地址路由分布与配置

龙芯 3B1500 的路由主要通过系统的两级交叉开关实现。一级交叉开关可以对每个 Master 端口接收到的请求进行路由配置，每个 Master 端口都拥有 8 个地址窗口，可以完成 8 个地址窗口的目标路由选择。每个地址窗口由 BASE、MASK 和 MMAP 三个 64 位寄存器组成，BASE 以 K 字节对齐；MASK 采用类似网络掩码高位为 1 的格式；MMAP 的低三位表示对应目标 Slave 端口的编号，MMAP[4]表示允许取指，MMAP [5]表示允许块读，MMAP [7]表示窗口使能。

窗口命中公式：　　(IN_ADDR & MASK) == BASE

由于龙芯 3 号缺省采用固定路由，在上电启动时，配置窗口都处于关闭状态，使用时需要系统软件对其进行使能配置。

在一级 XBAR 处，标号与所述模块的对应关系如表 2-5 所示。

表 2-5 一级 XBAR 标号与所述模块的对应关系

| 标号 | 缺省值 |
|---|---|
| 0 | 共享缓存块 0 |
| 1 | 共享缓存块 1 |
| 2 | 共享缓存块 2 |
| 3 | 共享缓存块 3 |
| 4 | 相邻节点 |
| 5、6 | 保留 |
| 7 | HT 模块<br>（对于节点 0，为 HT0；对于节点 1，对 HT1） |

节点 0 的地址窗口转换寄存器如下表所示。节点 1 的地址窗口基址是在节点 0 的地址基础上增加 0x1000_0000_4000，即对应于 Core4_Win0_Base 的地址是 0x1000_3FF0_6000。

不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

表 2-6 节点 0 一级交叉开关地址窗口寄存器

| 地址 | 寄存器 | 地址 | 寄存器 |
|---|---|---|---|
| 0x3ff0_2000 | CORE0_WIN0_BASE | 0x3ff0_2100 | CORE1_WIN0_BASE |
| 0x3ff0_2008 | CORE0_WIN1_BASE | 0x3ff0_2108 | CORE1_WIN1_BASE |
| 0x3ff0_2010 | CORE0_WIN2_BASE | 0x3ff0_2110 | CORE1_WIN2_BASE |
| 0x3ff0_2018 | CORE0_WIN3_BASE | 0x3ff0_2118 | CORE1_WIN3_BASE |
| 0x3ff0_2020 | CORE0_WIN4_BASE | 0x3ff0_2120 | CORE1_WIN4_BASE |
| 0x3ff0_2028 | CORE0_WIN5_BASE | 0x3ff0_2128 | CORE1_WIN5_BASE |
| 0x3ff0_2030 | CORE0_WIN6_BASE | 0x3ff0_2130 | CORE1_WIN6_BASE |
| 0x3ff0_2038 | CORE0_WIN7_BASE | 0x3ff0_2138 | CORE1_WIN7_BASE |
| 0x3ff0_2040 | CORE0_WIN0_MASK | 0x3ff0_2140 | CORE1_WIN0_MASK |
| 0x3ff0_2048 | CORE0_WIN1_MASK | 0x3ff0_2148 | CORE1_WIN1_MASK |
| 0x3ff0_2050 | CORE0_WIN2_MASK | 0x3ff0_2150 | CORE1_WIN2_MASK |
| 0x3ff0_2058 | CORE0_WIN3_MASK | 0x3ff0_2158 | CORE1_WIN3_MASK |
| 0x3ff0_2060 | CORE0_WIN4_MASK | 0x3ff0_2160 | CORE1_WIN4_MASK |
| 0x3ff0_2068 | CORE0_WIN5_MASK | 0x3ff0_2168 | CORE1_WIN5_MASK |
| 0x3ff0_2070 | CORE0_WIN6_MASK | 0x3ff0_2170 | CORE1_WIN6_MASK |

| | | | |
|---|---|---|---|
| 0x3ff0_2078 | CORE0_WIN7_MASK | 0x3ff0_2178 | CORE1_WIN7_MASK |
| 0x3ff0_2080 | CORE0_WIN0_MMAP | 0x3ff0_2180 | CORE1_WIN0_MMAP |
| 0x3ff0_2088 | CORE0_WIN1_MMAP | 0x3ff0_2188 | CORE1_WIN1_MMAP |
| 0x3ff0_2090 | CORE0_WIN2_MMAP | 0x3ff0_2190 | CORE1_WIN2_MMAP |
| 0x3ff0_2098 | CORE0_WIN3_MMAP | 0x3ff0_2198 | CORE1_WIN3_MMAP |
| 0x3ff0_20a0 | CORE0_WIN4_MMAP | 0x3ff0_21a0 | CORE1_WIN4_MMAP |
| 0x3ff0_20a8 | CORE0_WIN5_MMAP | 0x3ff0_21a8 | CORE1_WIN5_MMAP |
| 0x3ff0_20b0 | CORE0_WIN6_MMAP | 0x3ff0_21b0 | CORE1_WIN6_MMAP |
| 0x3ff0_20b8 | CORE0_WIN7_MMAP | 0x3ff0_21b8 | CORE1_WIN7_MMAP |
| | | | |
| 0x3ff0_2200 | CORE2_WIN0_BASE | 0x3ff0_2300 | CORE3_WIN0_BASE |
| 0x3ff0_2208 | CORE2_WIN1_BASE | 0x3ff0_2308 | CORE3_WIN1_BASE |
| 0x3ff0_2210 | CORE2_WIN2_BASE | 0x3ff0_2310 | CORE3_WIN2_BASE |
| 0x3ff0_2218 | CORE2_WIN3_BASE | 0x3ff0_2318 | CORE3_WIN3_BASE |
| 0x3ff0_2220 | CORE2_WIN4_BASE | 0x3ff0_2320 | CORE3_WIN4_BASE |
| 0x3ff0_2228 | CORE2_WIN5_BASE | 0x3ff0_2328 | CORE3_WIN5_BASE |
| 0x3ff0_2230 | CORE2_WIN6_BASE | 0x3ff0_2330 | CORE3_WIN6_BASE |
| 0x3ff0_2238 | CORE2_WIN7_BASE | 0x3ff0_2338 | CORE3_WIN7_BASE |
| 0x3ff0_2240 | CORE2_WIN0_MASK | 0x3ff0_2340 | CORE3_WIN0_MASK |
| 0x3ff0_2248 | CORE2_WIN1_MASK | 0x3ff0_2348 | CORE3_WIN1_MASK |
| 0x3ff0_2250 | CORE2_WIN2_MASK | 0x3ff0_2350 | CORE3_WIN2_MASK |
| 0x3ff0_2258 | CORE2_WIN3_MASK | 0x3ff0_2358 | CORE3_WIN3_MASK |
| 0x3ff0_2260 | CORE2_WIN4_MASK | 0x3ff0_2360 | CORE3_WIN4_MASK |
| 0x3ff0_2268 | CORE2_WIN5_MASK | 0x3ff0_2368 | CORE3_WIN5_MASK |
| 0x3ff0_2270 | CORE2_WIN6_MASK | 0x3ff0_2370 | CORE3_WIN6_MASK |
| 0x3ff0_2278 | CORE2_WIN7_MASK | 0x3ff0_2378 | CORE3_WIN7_MASK |
| 0x3ff0_2280 | CORE2_WIN0_MMAP | 0x3ff0_2380 | CORE3_WIN0_MMAP |
| 0x3ff0_2288 | CORE2_WIN1_MMAP | 0x3ff0_2388 | CORE3_WIN1_MMAP |
| 0x3ff0_2290 | CORE2_WIN2_MMAP | 0x3ff0_2390 | CORE3_WIN2_MMAP |
| 0x3ff0_2298 | CORE2_WIN3_MMAP | 0x3ff0_2398 | CORE3_WIN3_MMAP |
| 0x3ff0_22a0 | CORE2_WIN4_MMAP | 0x3ff0_23a0 | CORE3_WIN4_MMAP |
| 0x3ff0_22a8 | CORE2_WIN5_MMAP | 0x3ff0_23a8 | CORE3_WIN5_MMAP |
| 0x3ff0_22b0 | CORE2_WIN6_MMAP | 0x3ff0_23b0 | CORE3_WIN6_MMAP |
| 0x3ff0_22b8 | CORE2_WIN7_MMAP | 0x3ff0_23b8 | CORE3_WIN7_MMAP |

| | | | |
|---|---|---|---|
| | | | |
| 0x3ff0_2400 | EAST_WIN0_BASE | 0x3ff0_2500 | SOUTH_WIN0_BASE |
| 0x3ff0_2408 | EAST_WIN1_BASE | 0x3ff0_2508 | SOUTH_WIN1_BASE |
| 0x3ff0_2410 | EAST_WIN2_BASE | 0x3ff0_2510 | SOUTH_WIN2_BASE |
| 0x3ff0_2418 | EAST_WIN3_BASE | 0x3ff0_2518 | SOUTH_WIN3_BASE |
| 0x3ff0_2420 | EAST_WIN4_BASE | 0x3ff0_2520 | SOUTH_WIN4_BASE |
| 0x3ff0_2428 | EAST_WIN5_BASE | 0x3ff0_2528 | SOUTH_WIN5_BASE |
| 0x3ff0_2430 | EAST_WIN6_BASE | 0x3ff0_2530 | SOUTH_WIN6_BASE |
| 0x3ff0_2438 | EAST_WIN7_BASE | 0x3ff0_2538 | SOUTH_WIN7_BASE |
| 0x3ff0_2440 | EAST_WIN0_MASK | 0x3ff0_2540 | SOUTH_WIN0_MASK |
| 0x3ff0_2448 | EAST_WIN1_MASK | 0x3ff0_2548 | SOUTH_WIN1_MASK |
| 0x3ff0_2450 | EAST_WIN2_MASK | 0x3ff0_2550 | SOUTH_WIN2_MASK |
| 0x3ff0_2458 | EAST_WIN3_MASK | 0x3ff0_2558 | SOUTH_WIN3_MASK |
| 0x3ff0_2460 | EAST_WIN4_MASK | 0x3ff0_2560 | SOUTH_WIN4_MASK |
| 0x3ff0_2468 | EAST_WIN5_MASK | 0x3ff0_2568 | SOUTH_WIN5_MASK |
| 0x3ff0_2470 | EAST_WIN6_MASK | 0x3ff0_2570 | SOUTH_WIN6_MASK |
| 0x3ff0_2478 | EAST_WIN7_MASK | 0x3ff0_2578 | SOUTH_WIN7_MASK |
| 0x3ff0_2480 | EAST_WIN0_MMAP | 0x3ff0_2580 | SOUTH_WIN0_MMAP |
| 0x3ff0_2488 | EAST_WIN1_MMAP | 0x3ff0_2588 | SOUTH_WIN1_MMAP |
| 0x3ff0_2490 | EAST_WIN2_MMAP | 0x3ff0_2590 | SOUTH_WIN2_MMAP |
| 0x3ff0_2498 | EAST_WIN3_MMAP | 0x3ff0_2598 | SOUTH_WIN3_MMAP |
| 0x3ff0_24a0 | EAST_WIN4_MMAP | 0x3ff0_25a0 | SOUTH_WIN4_MMAP |
| 0x3ff0_24a8 | EAST_WIN5_MMAP | 0x3ff0_25a8 | SOUTH_WIN5_MMAP |
| 0x3ff0_24b0 | EAST_WIN6_MMAP | 0x3ff0_25b0 | SOUTH_WIN6_MMAP |
| 0x3ff0_24b8 | EAST_WIN7_MMAP | 0x3ff0_25b8 | SOUTH_WIN7_MMAP |
| | | | |
| 0x3ff0_2600 | WEST_WIN0_BASE | 0x3ff0_2700 | NORTH_WIN0_BASE |
| 0x3ff0_2608 | WEST_WIN1_BASE | 0x3ff0_2708 | NORTH_WIN1_BASE |
| 0x3ff0_2610 | WEST_WIN2_BASE | 0x3ff0_2710 | NORTH_WIN2_BASE |
| 0x3ff0_2618 | WEST_WIN3_BASE | 0x3ff0_2718 | NORTH_WIN3_BASE |
| 0x3ff0_2620 | WEST_WIN4_BASE | 0x3ff0_2720 | NORTH_WIN4_BASE |
| 0x3ff0_2628 | WEST_WIN5_BASE | 0x3ff0_2728 | NORTH_WIN5_BASE |
| 0x3ff0_2630 | WEST_WIN6_BASE | 0x3ff0_2730 | NORTH_WIN6_BASE |
| 0x3ff0_2638 | WEST_WIN7_BASE | 0x3ff0_2738 | NORTH_WIN7_BASE |

| | | | |
|---|---|---|---|
| 0x3ff0_2640 | WEST_WIN0_MASK | 0x3ff0_2740 | NORTH_WIN0_MASK |
| 0x3ff0_2648 | WEST_WIN1_MASK | 0x3ff0_2748 | NORTH_WIN1_MASK |
| 0x3ff0_2650 | WEST_WIN2_MASK | 0x3ff0_2750 | NORTH_WIN2_MASK |
| 0x3ff0_2658 | WEST_WIN3_MASK | 0x3ff0_2758 | NORTH_WIN3_MASK |
| 0x3ff0_2660 | WEST_WIN4_MASK | 0x3ff0_2760 | NORTH_WIN4_MASK |
| 0x3ff0_2668 | WEST_WIN5_MASK | 0x3ff0_2768 | NORTH_WIN5_MASK |
| 0x3ff0_2670 | WEST_WIN6_MASK | 0x3ff0_2770 | NORTH_WIN6_MASK |
| 0x3ff0_2678 | WEST_WIN7_MASK | 0x3ff0_2778 | NORTH_WIN7_MASK |
| 0x3ff0_2680 | WEST_WIN0_MMAP | 0x3ff0_2780 | NORTH_WIN0_MMAP |
| 0x3ff0_2688 | WEST_WIN1_MMAP | 0x3ff0_2788 | NORTH_WIN1_MMAP |
| 0x3ff0_2690 | WEST_WIN2_MMAP | 0x3ff0_2790 | NORTH_WIN2_MMAP |
| 0x3ff0_2698 | WEST_WIN3_MMAP | 0x3ff0_2798 | NORTH_WIN3_MMAP |
| 0x3ff0_26a0 | WEST_WIN4_MMAP | 0x3ff0_27a0 | NORTH_WIN4_MMAP |
| 0x3ff0_26a8 | WEST_WIN5_MMAP | 0x3ff0_27a8 | NORTH_WIN5_MMAP |
| 0x3ff0_26b0 | WEST_WIN6_MMAP | 0x3ff0_27b0 | NORTH_WIN6_MMAP |
| 0x3ff0_26b8 | WEST_WIN7_MMAP | 0x3ff0_27b8 | NORTH_WIN7_MMAP |

在龙芯 3B1500 的二级 XBAR 中有 CPU 地址空间（包括 HT 空间）、DDR2 地址空间、以及 PCI 地址空间共三个 IP 相关的地址空间。地址窗口是供 CPU 和 PCI-DMA 两个具有 Master 功能的 IP 进行路由选择和地址转换而设置的。CPU 和 PCI-DMA 两者都拥有 8 个地址窗口，可以完成目标地址空间的选择以及从源地址空间到目标地址空间的转换。每个地址窗口由 BASE、MASK 和 MMAP 三个 64 位寄存器组成，BASE 以 K 字节对齐，MASK 采用类似网络掩码高位为 1 的格式，MMAP 的低三位是路由选择。

在二级 XBAR 处，标号与所述模块的对应关系如表 2-7 所示对应新地址空间的编号（其中 DDR2 的标号为 0，PCI/Local IO 编号为 2，配置寄存器模块连接在端口 3 上）。

表 2-7 二级 XBAR 处，标号与所述模块的对应关系

| 标号 | 缺省值 |
|---|---|
| 0 | 0 号 DDR2/3 控制器 |
| 1 | 空 |
| 2 | 低速 I/O（PCI，LPC 等） |
| 3 | 配置寄存器 |

如下表。MMAP[4]表示允许取指，MMAP[5]表示允许块读，MMAP[7]表示窗口使能。

表 2-8 MMAP 字段对应的该空间访问属性

| [4] | [5] | [7] |
|---|---|---|
| 允许取指 | 允许块读 | 窗口使能 |

二级 XBAR 的地址配置与一级 XBAR 的地址配置相比增加了地址转换的功能。相比之下，一级 XBAR 的窗口配置不能对 Cache 一致性的请求进行地址转换，否则在 SCache 处的地址会与处理器一级 Cache 处的地址不一致，导致 Cache 一致性的维护错误。

窗口命中公式：　　(IN_ADDR & MASK) == BASE

新地址换算公式：　　OUT_ADDR = (IN_ADDR & ~MASK) | {MMAP[63:10],10'h0}

节点 0 的地址窗口转换寄存器如下表。节点 1 的地址窗口基址是在节点 0 的地址基础上增加 0x1000_0000_4000，即对应于节点 1 的 CPU_Win0_Base 的地址是 0x1000_3FF0_4000。

不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

表 2-9 二级 XBAR 地址窗口转换寄存器

| 地址 | 寄存器 | 描述 | 缺省值 |
|---|---|---|---|
| 3ff0 0000 | CPU_WIN0_BASE | CPU 窗口 0 的基地址 | 0x0 |
| 3ff0 0008 | CPU_WIN1_BASE | CPU 窗口 1 的基地址 | 0x1000_0000 |
| 3ff0 0010 | CPU_WIN2_BASE | CPU 窗口 2 的基地址 | 0x0 |
| 3ff0 0018 | CPU_WIN3_BASE | CPU 窗口 3 的基地址 | 0x0 |
| 3ff0 0020 | CPU_WIN4_BASE | CPU 窗口 4 的基地址 | 0x0 |
| 3ff0 0028 | CPU_WIN5_BASE | CPU 窗口 5 的基地址 | 0x0 |
| 3ff0 0030 | CPU_WIN6_BASE | CPU 窗口 6 的基地址 | 0x0 |
| 3ff0 0038 | CPU_WIN7_BASE | CPU 窗口 7 的基地址 | 0x0 |
| 3ff0 0040 | CPU_WIN0_MASK | CPU 窗口 0 的掩码 | 0xffff_ffff_f000_0000 |
| 3ff0 0048 | CPU_WIN1_MASK | CPU 窗口 1 的掩码 | 0xffff_ffff_f000_0000 |
| 3ff0 0050 | CPU_WIN2_MASK | CPU 窗口 2 的掩码 | 0x0 |
| 3ff0 0058 | CPU_WIN3_MASK | CPU 窗口 3 的掩码 | 0x0 |
| 3ff0 0060 | CPU_WIN4_MASK | CPU 窗口 4 的掩码 | 0x0 |
| 3ff0 0068 | CPU_WIN5_MASK | CPU 窗口 5 的掩码 | 0x0 |
| 3ff0 0070 | CPU_WIN6_MASK | CPU 窗口 6 的掩码 | 0x0 |
| 3ff0 0078 | CPU_WIN7_MASK | CPU 窗口 7 的掩码 | 0x0 |
| 3ff0 0080 | CPU_WIN0_MMAP | CPU 窗口 0 的新基地址 | 0xf0 |
| 3ff0 0088 | CPU_WIN1_MMAP | CPU 窗口 1 的新基地址 | 0x1000_00f2 |
| 3ff0 0090 | CPU_WIN2_MMAP | CPU 窗口 2 的新基地址 | 0 |
| 3ff0 0098 | CPU_WIN3_MMAP | CPU 窗口 3 的新基地址 | 0 |
| 3ff0 00a0 | CPU_WIN4_MMAP | CPU 窗口 4 的新基地址 | 0x0 |
| 3ff0 00a8 | CPU_WIN5_MMAP | CPU 窗口 5 的新基地址 | 0x0 |
| 3ff0 00b0 | CPU_WIN6_MMAP | CPU 窗口 6 的新基地址 | 0 |
| 3ff0 00b8 | CPU_WIN7_MMAP | CPU 窗口 7 的新基地址 | 0 |
| 3ff0 0100 | PCI_WIN0_BASE | PCI 窗口 0 的基地址 | 0x8000_0000 |
| 3ff0 0108 | PCI_WIN1_BASE | PCI 窗口 1 的基地址 | 0x0 |
| 3ff0 0110 | PCI_WIN2_BASE | PCI 窗口 2 的基地址 | 0x0 |
| 3ff0 0118 | PCI_WIN3_BASE | PCI 窗口 3 的基地址 | 0x0 |
| 3ff0 0120 | PCI_WIN4_BASE | PCI 窗口 4 的基地址 | 0x0 |

| | | | |
|---|---|---|---|
| 3ff0 0128 | PCI_WIN5_BASE | PCI 窗口 5 的基地址 | 0x0 |
| 3ff0 0130 | PCI_WIN6_BASE | PCI 窗口 6 的基地址 | 0x0 |
| 3ff0 0138 | PCI_WIN7_BASE | PCI 窗口 7 的基地址 | 0x0 |
| 3ff0 0140 | PCI_WIN0_MASK | PCI 窗口 0 的掩码 | 0xffff_ffff_8000_0000 |
| 3ff0 0148 | PCI_WIN1_MASK | PCI 窗口 1 的掩码 | 0x0 |
| 3ff0 0150 | PCI_WIN2_MASK | PCI 窗口 2 的掩码 | 0x0 |
| 3ff0 0158 | PCI_WIN3_MASK | PCI 窗口 3 的掩码 | 0x0 |
| 3ff0 0160 | PCI_WIN4_MASK | PCI 窗口 4 的掩码 | 0x0 |
| 3ff0 0168 | PCI_WIN5_MASK | PCI 窗口 5 的掩码 | 0x0 |
| 3ff0 0170 | PCI_WIN6_MASK | PCI 窗口 6 的掩码 | 0x0 |
| 3ff0 0178 | PCI_WIN7_MASK | PCI 窗口 7 的掩码 | 0x0 |
| 3ff0 0180 | PCI_WIN0_MMAP | PCI 窗口 0 的新基地址 | 0xf0 |
| 3ff0 0188 | PCI_WIN1_MMAP | PCI 窗口 1 的新基地址 | 0x0 |
| 3ff0 0190 | PCI_WIN2_MMAP | PCI 窗口 2 的新基地址 | 0 |
| 3ff0 0198 | PCI_WIN3_MMAP | PCI 窗口 3 的新基地址 | 0 |
| 3ff0 01a0 | PCI_WIN4_MMAP | PCI 窗口 4 的新基地址 | 0x0 |
| 3ff0 01a8 | PCI_WIN5_MMAP | PCI 窗口 5 的新基地址 | 0x0 |
| 3ff0 01b0 | PCI_WIN6_MMAP | PCI 窗口 6 的新基地址 | 0 |
| 3ff0 01b8 | PCI_WIN7_MMAP | PCI 窗口 7 的新基地址 | 0 |

根据缺省的寄存器配置，芯片启动后，CPU 的 0x00000000 - 0x0fffffff 的地址区间（256M）映射到 DDR2 的 0x00000000 - 0x0fffffff 的地址区间，CPU 的 0x10000000 - 0x1fffffff 区间（256M）映射到 PCI 的 0x10000000 - 0x1fffffff 区间，PCIDMA 的 0x80000000 - 0x8fffffff 的地址区间（256M）映射到 DDR2 的 0x00000000 - 0x0fffffff 的地址区间。软件可以通过修改相应的配置寄存器实现新的地址空间路由和转换。

此外，当出现由于 CPU 猜测执行引起对非法地址的读访问时，8 个地址窗口都不命中，由配置寄存器模块返回全 0 的数据给 CPU，以防止 CPU 死等。

表 2-10 二级 XBAR 缺省地址配置

| 基地址 | 高位 | 所有者 |
|---|---|---|
| 0x0000_0000_0000_0000 | 0x0000_0000_1000_0000 | 0 号 DDR 控制器 |
| 0x0000_0000_1000_0000 | 0x0000_0000_2000_0000 | 低速 I/O（PCI 等） |

## 2.6 芯片配置、采样及 PLL 相关寄存器

龙芯 3B1500 中的芯片配置寄存器(Chip_config)及芯片采样寄存器(Chip_sample)提供了对芯片的配置进行读写的机制。

表 2-11 芯片配置寄存器（物理地址 0x1fe00180）

| 位域 | 字段名 | 访问 | 复位值 | 描述 |
|---|---|---|---|---|
| 3:0 | - | RW | 4’b7 | 保留 |

| | | | | |
|---|---|---|---|---|
| 4 | MC0_disable_ddr2_confspace | RW | 1’b0 | 是否禁用 MC0 DDR 配置空间 |
| 5 | - | RW | 1’b0 | 保留 |
| 6 | - | RW | 1’b0 | 保留 |
| 7 | MC0_ddr2_resetn | RW | 1’b1 | MC0 软件复位（低有效） |
| 8 | MC0_clken | RW | 1’b1 | 是否使能 MC0 |
| 9 | MC1_disable_ddr2_confspace | RW | 1’b0 | 是否禁用 MC1 DDR 配置空间 |
| 10 | - | RW | 1’b0 | 保留 |
| 11 | - | RW | 1’b0 | 保留 |
| 12 | MC1_ddr2_resetn | RW | 1’b1 | MC1 软件复位（低有效） |
| 13 | MC1_clken | RW | 1’b1 | 是否使能 MC1 |
| 26:24 | HT0_freq_scale_ctrl | RW | 3’b111 | HT 控制器 0 分频 |
| 27 | HT0_clken | RW | 1’b1 | 是否使能 HT0 |
| 30:28 | HT1_freq_scale_ctrl | RW | 3’b111 | HT 控制器 1 分频 |
| 31 | HT1_clken | RW | 1’b1 | 是否使能 HT1 |
| 42:40 | Node0_freq_ ctrl | RW | 3’b111 | 节点 0 分频 |
| 43 | - | RW | 1’b1 | |
| 46:44 | Node1_freq_ ctrl | RW | 3’b111 | 节点 1 分频 |
| 47 | - | RW | 1’b1 | |
| 63:56 | Cpu_version | R | 2’h35 | CPU 版本 |
| 95:64 | | | | （空） |
| 127:96 | Pad1v8_ctrl | RW | 6’h780 | 1v8 pad 控制 |
| 其它 | | R | | 保留 |

表 2-12 芯片采样寄存器（物理地址 0x1fe00190）

| 位域 | 字段名 | 访问 | 复位值 | 描述 |
|---|---|---|---|---|
| 31:0 | Compcode_core | R | | |
| 47:32 | Sys_clkseli | R | | 板上倍频设置 |
| 55:48 | Bad_ip_core | R | | core7-core0 是否坏 |
| 57:56 | Bad_ip_ddr | R | | 2 个 DDR 控制器是否坏 |
| 61:60 | Bad_ip_ht | R | | 2 个 HT 控制器是否坏 |
| 83:80 | Compcode_ok | R | | |
| 88 | Thsens0_overflow | R | | 温度传感器 0 上溢（超过 125℃） |
| 89 | Thsens1_overflow | R | | 温度传感器 1 上溢（超过 125℃） |
| 103:96 | Thsens0_out | R | | 温度传感器 0 摄氏温度<br>结点温度=Thens0_out0 -100<br>温度范围 -40 度 – 125 度 |
| 111:104 | Thsens1_out | R | | 温度传感器 1 摄氏温度<br>结点温度=Thens0_out1 -100<br>温度范围 -40 度 – 125 度 |
| 其它 | | R | | 保留 |

以下几组软件倍频设置寄存器用于设置在 CLKSEL 配置为软件控制模式下，各个时钟

的工作频率。其中，MEM CLOCK 配置对应内存控制器及总线时钟频率；CORE CLOCK 对应处理器核时钟频率；NODE CLOCK 对应片上互连网络及三级共享高速缓存频率；HT CLOCK 对应 HT 控制器时钟频率。

每个时钟配置一般有三个参数，DIV_REFC、DIV_LOOPC、DIV_OUT。最终的时钟频率为（参考时钟 / DIV_REFC * DIV_LOOPC）/ DIV_OUT。

而对于 HT CLOCK 来说，前两个参数是由引脚配置决定的，无法更改，只有 DIV_OUT（HTx_DIV_HT_CORE）是可以软件配置的。其工作频率为 CLKSEL[12:10]配置的 PHY 时钟频率 / DIV_OUT。

软件配置 CORE CLOCK 与 NODE CLOCK 时，建议将 PD_L1 设为 1，PD_L2/PD_L3 设为 0；Serial_mode 设置为 0；BYPASS_REFIN_L2 设为 1。这样不使用 L1 PLL，而使得 SYSPLL 工作在 NODE 与 CORE CLOCK 独立模式下，能够得到更好的时钟精度。

软件控制模式下，默认对应的时钟频率为外部参考时钟的频率（对于 CORE CLOCK 与 NODE CLOCK，都为引脚 SYS_CLK 的对应频率；对于 MEM CLOCK，为引脚 MEM_CLK 对应频率），需要在处理器启动过程中对时钟进行软件设置。各个时钟设置的过程应该按照以下方式：

1） 设置寄存器中除了 SEL_PLL_*及 SOFT_SET_PLL 之外的其它寄存器，也即这两个寄存器在设置的过程中写为 0；

2） 其它寄存器值不变，将 SOFT_SET_PLL 设为 1；

3） 等待寄存器中的锁定信号 LOCKED_*为 1；

4） 将 SEL_PLL_*设为 1，此时对应的时钟频率将切换为软件设置的频率

表 2-13 芯片结点和处理器核软件倍频设置寄存器（物理地址 0x1fe001b0）

| 位域 | 字段名 | 访问 | 复位值 | 描述 |
|---|---|---|---|---|
| 0 | SEL_PLL_NODE | RW | 0x0 | Node 时钟非软件 bypass 整个 PLL |
| 1 | SEL_PLL_CORE | RW | 0x0 | Core 时钟非软件 bypass 整个 PLL |
| 2 | SOFT_SET_PLL | RW | 0x0 | 允许软件设置 PLL |
| 3 | BYPASS_L1 | RW | 0x0 | Bypass L1 PLL |
| 4 | BYPASS_L2_NODE | RW | 0x0 | Bypass L2 Node PLL |
| 5 | BYPASS_L2_CORE | RW | 0x0 | Bypass L2 Core PLL |
| 6 | BYPASS_REFIN_L2 | RW | 0x0 | L2 的输入是否 Bypass L1 PLL |
| 7 | LOCKEN_L1 | RW | 0x0 | 允许锁定 L1 PLL |
| 8 | LOCKEN_L2_NODE | RW | 0x0 | 允许锁定 L2 Node PLL |
| 9 | LOCKEN_L2_CORE | RW | 0x0 | 允许锁定 L2 Core PLL |
| 11:10 | LOCKC_L1 | RW | 0x0 | 判定 L1 PLL 是否锁定使用的相位 |

| | | | | |
|---|---|---|---|---|
| | | | | 的精度 |
| 13:12 | LOCKC_L2_NODE | RW | 0x0 | 判定 L2 Node PLL 是否锁定使用的相位的精度 |
| 15:14 | LOCKC_L2_CORE | RW | 0x0 | 判定 L2 Core PLL 是否锁定使用的相位的精度 |
| 16 | LOCKED_L1 | R | 0x0 | L1 PLL 是否锁定 |
| 17 | LOCKED_L2_NODE | R | 0x0 | L2 Node PLL 是否锁定 |
| 18 | LOCKED_L2_CORE | R | 0x0 | L2 Core PLL 是否锁定 |
| 19 | PD_L1 | R/W | 0x0 | 关闭 L1 PLL（LS3B1500F） |
| 20 | PD_L2_NODE | R/W | 0x0 | 关闭 L2 Node PLL（LS3B1500F） |
| 21 | PD_L2_CORE | R/W | 0x0 | 关闭 L2 Core PLL（LS3B1500F） |
| 23:22 | Serial_mode | R/W | 0x0 | PLL 使用模式（LS3B1500F）<br>00 – L2 PLL 锁定不依赖于 L1 PLL<br>11 – L2 PLL 锁定依赖于 L1 PLL |
| 31:26 | L1_DIV_REFC | RW | 0x1 | L1 PLL 输入参数 |
| 41:32 | L1_DIV_LOOPC | RW | 0x1 | L1 PLL 输入参数 |
| 47:42 | L1_DIV_OUT | RW | 0x1 | L1 PLL 输入参数 |
| 53:48 | L2_DIV_REFC_NODE | RW | 0x1 | L2 Node PLL 输入参数 |
| 63:54 | L2_DIV_LOOPC_NODE | RW | 0x1 | L2 Node PLL 输入参数 |
| 69:64 | L2_DIV_OUT_NODE | RW | 0x1 | L2 Node PLL 输入参数 |
| 75:70 | L2_DIV_REFC_CORE | RW | 0x5 | L2 Core PLL 输入参数 |
| 85:76 | L2_DIV_LOOPC_CORE | RW | 0x0 | L2 Core PLL 输入参数 |
| 91:86 | L2_DIV_OUT_CORE | RW | 0x1 | L2 Core PLL 输入参数 |
| 其它 | | RW | | 保留 |

注：PLL ouput = （clk_ref / div_refc * div_loopc）/ div_out。

PLL constrain 指括号内的值：L1：450M~1.58G　　L2: 850M~3.23G

内存时钟约束同 L1，HT 时钟约束同 L2。

表 2-14 芯片内存和 HT 时钟软件倍频设置寄存器（物理地址 0x1fe001c0）

| 位域 | 字段名 | 访问 | 复位值 | 描述 |
|---|---|---|---|---|
| 0 | SEL _MEM_PLL | RW | 0x0 | MEM 时钟非软件 bypass 整个 PLL |
| 1 | SOFT_SET_MEM_PLL | RW | 0x0 | 允许软件设置 MEM PLL |
| 2 | BYPASS_MEM_PLL | RW | 0x0 | Bypass MEM_PLL |
| 3 | LOCKEN_MEM_PLL | RW | 0x0 | 允许锁定 MEM_PLL |
| 5:4 | LOCKC_ MEM_PLL | RW | 0x0 | 判定 MEM PLL 是否锁定使用的相位的精度 |
| 6 | LOCKED_MEM_PLL | R | 0x0 | MEM_PLL 是否锁定 |
| 13:8 | MEM_PLL_DIV_REFC | RW | 0x1 | MEM PLL 输入参数 |
| 23:14 | MEM_PLL_DIV_LOOPC | RW | 0x41 | MEM PLL 输入参数 |
| 29:24 | MEM_PLL_DIV_OUT | RW | 0x0 | MEM PLL 输入参数 |
| 32 | SEL_HT0_PLL | RW | 0x0 | HT0 非软件 bypass PLL |
| 33 | SOFT_SET_HT0_PLL | RW | 0x0 | 允许软件设置 HT0 PLL |

| | | | | |
|---|---|---|---|---|
| 34 | BYPASS_HT0_PLL | RW | 0x0 | Bypass HT0_PLL |
| 35 | LOCKEN_HT0_PLL | RW | 0x0 | 允许锁定 HT0 PLL |
| 37:36 | LOCKC_HT0_PLL | RW | 0x0 | 判定 HT0 PLL 是否锁定是采用的相位精度 |
| 38 | LOCKED_HT0_PLL | R | 0x0 | HT0_PLL 是否锁定 |
| 45:40 | HT0_DIV_HTCORE | RW | 0x1 | HT0 Core PLL 输入参数 |
| 48 | SEL_HT1_PLL | RW | 0x0 | HT1 非软件 bypass PLL |
| 49 | SOFT_SET_HT1_PLL | RW | 0x0 | 允许软件设置 HT1 PLL |
| 50 | BYPASS_HT1_PLL | RW | 0x0 | Bypass HT1_PLL |
| 51 | LOCKEN_HT1_PLL | RW | 0x0 | 允许锁定 HT1 PLL |
| 53:52 | LOCKC_HT1_PLL | RW | 0x0 | 判定 HT1 PLL 是否锁定是采用的相位精度 |
| 54 | LOCKED_HT1_PLL | R | 0x0 | HT1_PLL 是否锁定 |
| 61:56 | HT1_DIV_HTCORE | RW | 0x1 | HT1 Core PLL 输入参数 |
| 其它 | | RW | | 保留 |

表 2-15 芯片处理器核软件分频设置寄存器（物理地址 0x1fe001d0）

| 位域 | 字段名 | 访问 | 复位值 | 描述 |
|---|---|---|---|---|
| 2:0 | core0_freqctrl | RW | 0x7 | 核 0 分频控制值 |
| 3 | core0_en | RW | 0x1 | 核 0 时钟使能 |
| 6:4 | core1_freqctrl | RW | 0x7 | 核 1 分频控制值 |
| 7 | core1_en | RW | 0x1 | 核 1 时钟使能 |
| 10:8 | core2_freqctrl | RW | 0x7 | 核 2 分频控制值 |
| 11 | core2_en | RW | 0x1 | 核 2 时钟使能 |
| 14:12 | core3_freqctrl | RW | 0x7 | 核 3 分频控制值 |
| 15 | core3_en | RW | 0x1 | 核 3 时钟使能 |
| 18:16 | core4_freqctrl | RW | 0x7 | 核 4 分频控制值 |
| 19 | core4_en | RW | 0x1 | 核 4 时钟使能 |
| 22:20 | core5_freqctrl | RW | 0x7 | 核 5 分频控制值 |
| 23 | core5_en | RW | 0x1 | 核 5 时钟使能 |
| 26:24 | core6_freqctrl | RW | 0x7 | 核 6 分频控制值 |
| 27 | core6_en | RW | 0x1 | 核 6 时钟使能 |
| 30:28 | core7_freqctrl | RW | 0x7 | 核 7 分频控制值 |
| 31 | core7_en | RW | 0x1 | 核 7 时钟使能 |
| | | | 注: | 软件分频后的时钟频率值等于原来的（分频控制值+1）/8 |
| 63:32 | Reserved | RW | 32’hFFFFFF | 保留 |
| 64 | Core0_pre_resetn | RW | 0x1 | 核 0 的预复位控制（LS3B1500F） |
| 65 | Core0_soft_resetn | RW | 0x1 | 核 0 的软件复位控制（LS3B1500F） |
| 66 | Core1_pre_resetn | RW | 0x1 | 核 1 的预复位控制（LS3B1500F） |
| 67 | Core1_soft_resetn | RW | 0x1 | 核 1 的软件复位控制（LS3B1500F） |
| 68 | Core2_pre_resetn | RW | 0x1 | 核 2 的预复位控制（LS3B1500F） |

| | | | | |
|---|---|---|---|---|
| 69 | Core2_soft_resetn | RW | 0x1 | 核 2 的软件复位控制（LS3B1500F） |
| 70 | Core3_pre_resetn | RW | 0x1 | 核 3 的预复位控制（LS3B1500F） |
| 71 | Core3_soft_resetn | RW | 0x1 | 核 3 的软件复位控制（LS3B1500F） |
| 72 | Core4_pre_resetn | RW | 0x1 | 核 4 的预复位控制（LS3B1500F） |
| 73 | Core4_soft_resetn | RW | 0x1 | 核 4 的软件复位控制（LS3B1500F） |
| 74 | Core5_pre_resetn | RW | 0x1 | 核 5 的预复位控制（LS3B1500F） |
| 75 | Core5_soft_resetn | RW | 0x1 | 核 5 的软件复位控制（LS3B1500F） |
| 76 | Core6_pre_resetn | RW | 0x1 | 核 6 的预复位控制（LS3B1500F） |
| 77 | Core6_soft_resetn | RW | 0x1 | 核 6 的软件复位控制（LS3B1500F） |
| 78 | Core7_pre_resetn | RW | 0x1 | 核 7 的预复位控制（LS3B1500F） |
| 79 | Core7_soft_resetn | RW | 0x1 | 核 7 的软件复位控制（LS3B1500F） |
| | | | | |

# 3 GS464 处理器核

GS464 是四发射 64 位的 MIPS 兼容的高性能处理器核，面向高性能计算和高端嵌入式等应用。

在龙芯 3B1500 中的 8 个 GS464 核通过以及 SCache 模块通过 AXI 互联网络形成一个分布式共享 SCache 的多核结构。

GS464 的主要特点如下：

- MIPS64 兼容；
- 提供 X86 虚拟机指令；
- 四发射超标量结构，两个定点、两个浮点、一个访存部件；
- 每个浮点部件都支持全流水双精度浮点乘加运算；
- 访存部件支持 128 位存储访问，虚地址和物理地址各为 48 位；
- 支持寄存器重命名、动态调度、转移预测等乱序执行技术；
- 64 项全相联 TLB，独立的 16 项指令 TLB，可变页大小；
- 一级指令 Cache 64KB，4 路组相联；
- 一级数据 Cache 64KB，4 路组相联；
- 独立不共享 Victim Cache 128KB，减少冲突失效；
- 支持 Non-blocking 访问及 Load-Speculation 等访存优化技术；
- 支持 Cache 一致性协议，可用于片内多核处理器；
- 指令 Cache 实现奇偶校验，数据 Cache 实现 ECC 校验；
- 支持标准的 EJTAG 调试标准，方便软硬件调试；
- 标准的 128 位 AXI 接口。

GS464 的结构如下图所示。GS464 更多的详细介绍请参考 GS464 用户手册以及 MIPS64 用户手册。

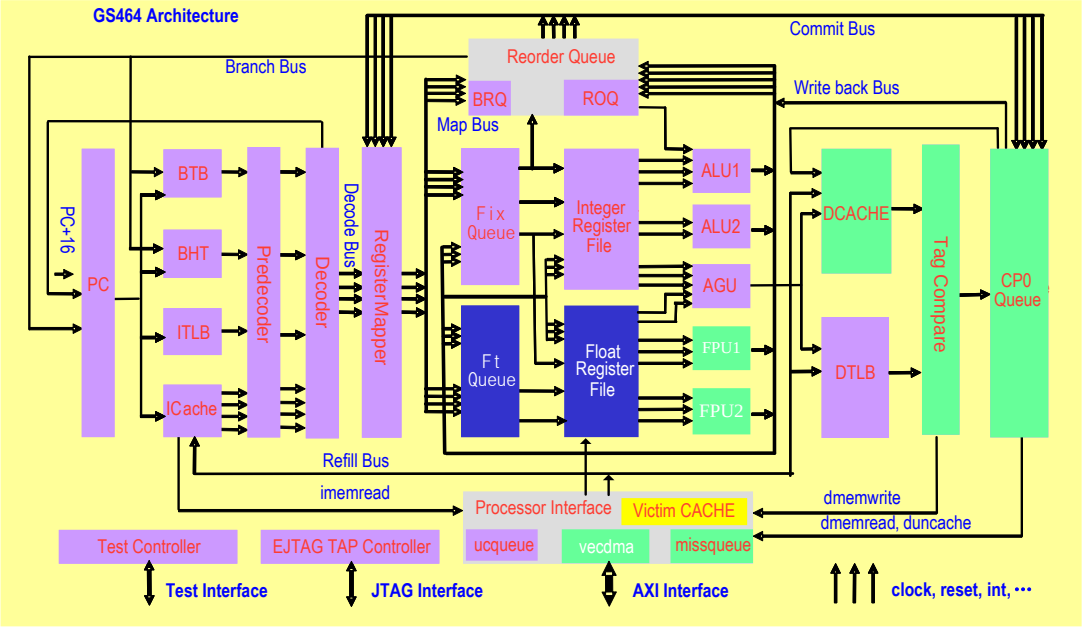



图 3-1 GS464 结构图

# 4 片上共享高速缓存（SCache）

SCache 模块是与 GS464 处理器 IP 配套设计的模块。该模块既可以和 GS464 对接，使 GS464 成为包括 SCache 在内的处理器 IP；也可以通过 AXI 网络连接多个 GS464 以及多个 SCache 模块，形成片内多处理器的 CMP 结构。SCache 模块的主要特征包括：

- 通过目录支持 Cache 一致性协议。
- 采用四路组相联结构。
- 运行时可动态关闭。
- 支持 ECC 校验。
- 支持 DMA 一致性读写和预取读。
- 支持 16 种 SCache 散列方式。
- 支持按窗口锁 SCache。

为降低功耗，SCache 的 TAG、目录和数据可以分开访问，SCache 状态位、w 位与 TAG 一起存储，TAG 存放在 TAG RAM 中，目录存放在 DIR RAM 中，数据存放在 DATA RAM 中。失效请求访问 SCache，同时读出所有路的 TAG、目录和数据，并根据 TAG 来选出数据和目录。替换请求、重填请求和写回请求只操作一路的 TAG、目录和数据。

为提高一些特定计算任务的性能，SCache 增加了锁机制。落在被锁区域中的 SCache 块会被锁住，因而不会被替换出 SCache（除非四路 SCache 中都是被锁住的块）。通过 confbus 可以对 SCache 模块内部的四组锁窗口寄存器进行动态配置，但必须保证四路 SCache 中一定有一路被锁住。因此 SCache 锁住的区域大小没有限制，只要不超过 SCache 大小的 3/4 就可以。此外当 SCache 收到 DMA 写请求时，如果被写的区域在 SCache 中命中且被锁住，那么 DMA 写将直接写入到 SCache 而不是内存。

表 4-1 SCache 锁窗口配置寄存器

| 名称 | 地址 | 位域 | 描述 |
|---|---|---|---|
| Slock0_valid | 0x3ff00200 | [63:63] | 0 号锁窗口有效位 |
| Slock0_addr | 0x3ff00200 | [47:0] | 0 号锁窗口锁地址 |
| Slock0_mask | 0x3ff00240 | [47:0] | 0 号锁窗口掩码 |
| Slock1_valid | 0x3ff00208 | [63:63] | 1 号锁窗口有效位 |
| Slock1_addr | 0x3ff00208 | [47:0] | 1 号锁窗口锁地址 |
| Slock1_mask | 0x3ff00248 | [47:0] | 1 号锁窗口掩码 |
| Slock2_valid | 0x3ff00210 | [63:63] | 2 号锁窗口有效位 |
| Slock2_addr | 0x3ff00210 | [47:0] | 2 号锁窗口锁地址 |
| Slock2_mask | 0x3ff00250 | [47:0] | 2 号锁窗口掩码 |
| Slock3_valid | 0x3ff00218 | [63:63] | 3 号锁窗口有效位 |
| Slock3_addr | 0x3ff00218 | [47:0] | 3 号锁窗口锁地址 |

| | | | |
|---|---|---|---|
| Slock3_mask | 0x3ff00258 | [47:0] | 3 号锁窗口掩码 |

举例来说，当一个地址 addr 使得 slock0_valid && ((addr & slock0_mask) == (slock0_addr & slock0_mask) )为 1 时，这个地址就被锁窗口 0 锁住了。

不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

# 5 矩阵处理加速器

龙芯 3B1500 中内置了两个独立于处理器核的矩阵处理加速器，其基本功能是通过软件的配置，实现对存放在内存中矩阵进行从源矩阵到目标矩阵的行列转置或搬移功能。两个加速器分别集成在龙芯 3B1500 的两个 HyperTransport 控制器内部，通过 1 级交叉开关实现对 SCache 及内存的读写，并且能够维护 Cache 一致性。

完成转置功能时，由于转置前同一 Cache 块的元素顺序在转置后的矩阵中是分散的，如果提前读入多行数据，使得这些数据能够以 Cache 行为单位进行写入目标矩阵的地址空间，则能提高读写效率。因此矩阵处理加速器中设置了一个大小为 32 行的缓存区，实现横向方式写入（从源矩阵读入到缓冲区），纵向读出（由缓冲区写入到目标矩阵）。

矩阵处理的工作过程为先读入 32 行源矩阵数据，再将该 32 行数据写入到目标矩阵，依次下去，直至完成整个矩阵的转置或搬运。矩阵处理加速器还可以根据需要，仅进行预取源矩阵而不写目标矩阵，以此方式来实现对数据进行预取到 SCache 的操作。

与转置不同，进行矩阵搬移工作时可以直接以 Cache 块为单位进行读写操作。因为读写操作可能不是同时进行，需要对读回来的数据进行缓冲处理，因此矩阵搬移模块实现了一块 64*64bit 的缓存区，提供四个读端口和四个写端口。为了提高 Cache 块的读写效率，本模块要求源矩阵的起始地址，每行搬运字节数及行宽均是 32 字节的整数倍，对矩阵的列数则没有要求，并且硬件不考虑矩阵地址不对齐的情况。

转置和搬移操作涉及的源矩阵可能是位于一个大矩阵中的一个小矩阵，因此，其矩阵地址可能不是完全连续，相邻行之间的地址会有间隔，需要实现更多的编程控制接口。通过设置矩阵元素的大小，本模块还支持对源矩阵的行和列同时进行放大处理，但要求放大后的矩阵大小不能超出所在大矩阵。

下面表 5-1 到 5-4 说明了矩阵处理涉及到的编程接口。

表 5-1 矩阵处理编程接口说明

| 地址 | 名称 | 属性 | 说明 |
|---|---|---|---|
| 0x3ff00700 | src_start_addr | RW | 源矩阵起始地址 |
| 0x3ff00708 | dst_start_addr | RW | 目标矩阵起始地址 |
| 0x3ff00710 | row | RW | 源矩阵的一行元素个数 |
| 0x3ff00718 | col | RW | 源矩阵的一列元素个数 |
| 0x3ff00720 | src_stride | RW | 源矩阵所在大矩阵的行跨度（字节） |
| 0x3ff00728 | dst_stride | RW | 目标矩阵所在大矩阵的行跨度（字节） |
| 0x3ff00730 | trans_ctrl | RW | 转置控制寄存器 |
| 0x3ff00738 | trans_status | RO | 转置状态寄存器 |

表 5-2 矩阵处理寄存器地址说明

| 地址 | 名称 |
|---|---|
| 0x3ff00700 | 0 号转置模块的 src_start_addr |
| 0x3ff00708 | 0 号转置模块的 dst_start_addr |
| 0x3ff00710 | 0 号转置模块的 row |
| 0x3ff00718 | 0 号转置模块的 col |
| 0x3ff00720 | 0 号转置模块的 src_stride |
| 0x3ff00728 | 0 号转置模块的 dst_stride |
| 0x3ff00730 | 0 号转置模块的 trans_ctrl |
| 0x3ff00738 | 0 号转置模块的 trans_status |
| 0x10003ff04700 | 1 号转置模块的 src_start_addr |
| 0x10003ff04708 | 1 号转置模块的 dst_start_addr |
| 0x10003ff04710 | 1 号转置模块的 row |
| 0x10003ff04718 | 1 号转置模块的 col |
| 0x10003ff04720 | 1 号转置模块的 src_stride |
| 0x10003ff04728 | 1 号转置模块的 dst_stride |
| 0x10003ff04730 | 1 号转置模块的 trans_ctrl |
| 0x10003ff04738 | 1 号转置模块的 trans_status |

对于节点 2 和节点 3 来说，对应的转置模块基地址分别为 0x20003ff08700 和 0x30003ff0c700。

表 5-3 trans_ctrl 寄存器

| 字段 | 说明 |
|---|---|
| 0 | fetch_en，是否允许读源矩阵。 |
| 1 | store_en，是否允许写目标矩阵。为 0 时，转置过程只预取源矩阵，但不写目标矩阵。 |
| 2 | 源矩阵读取完毕后，是否有效中断。 |
| 3 | 目标矩阵写入完毕后，是否有效中断， |
| 7..4 | Arcmd，读命令内部控制位。当 arcache 为 4’hf 时，必须设为 4’he 或 4’hf，其中 4’he 表示读分配，4’hf 表示读不分配。当 arcache 为其它值时无意义。 |
| 11..8 | Arcache，读命令内部控制位。为 4’hf 时，使用 cache 通路，为 4’h0 时，使用 uncache 通路。其它值无意义。 |
| 15..12 | Awcmd，写命令内部控制位。当 awcache 为 4’hf 时，必须设为 4’hb。当 awcache 为其它值时无意义。 |
| 19..16 | Awcache，写命令内部控制位。为 4’hf 时，使用 cache 通路，为 4’h0 时，使用 uncache 通路。其它值无意义。 |

| | |
|---|---|
| 21..20 | element_width，矩阵的元素大小。00 表示 1 个字节，01 表示 2 个字节，10 表示 4 个字节，11 表示 8 个字节 |
| 22 | trans_yes，为 1 表示进行转置；为 0 表示不转置 |

表 5-4 trans_status 寄存器

| 字段 | 说明 |
|---|---|
| 0 | 源矩阵读取完毕 |
| 1 | 目标矩阵写入完毕 |

# 6 处理器核间中断与通信

龙芯 3B1500 为每个处理器核都实现了 8 个核间中断寄存器（IPI）以支持多核 BIOS 启动和操作系统运行时在处理器核之间进行中断和通信，其说明和地址见表 6-1 到表 6-5。

表 6-1 处理器核间中断相关的寄存器及其功能描述

| 名称 | 读写权限 | 描述 |
|---|---|---|
| IPI_Status | R | 32 位状态寄存器，任何一位有被置 1 且对应位使能情况下，处理器核 INT4 中断线被置位。 |
| IPI_Enable | RW | 32 位使能寄存器，控制对应中断状态位是否有效 |
| IPI_Set | W | 32 位中断置位寄存器，往对应的位写 1，则 STATUS 寄存器对应的位被置 1 |
| IPI_Clear | W | 32 位中断清除寄存器，往对应的位写 1，则 STATUS 寄存器对应的位被清 0 |
| MailBox0 | RW | 缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。 |
| MailBox01 | RW | 缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。 |
| MailBox02 | RW | 缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。 |
| MailBox03 | RW | 缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。 |

龙芯 3B1500 节点 0 的处理器核间中断相关的寄存器及其功能描述如下，节点 1 的处理器核相关寄存器的地址是在节点 0 的地址基础上增加 0x1000_0000_4000，如对应于 Core4_IPI_Status 的地址是 0x1000_3FF0_5000。

不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

表 6-2 0 号处理器核的核间中断与通信寄存器

| 名称 | 地址 | 权限 | 描述 |
|---|---|---|---|
| Core0_IPI_Status | 0x3ff01000 | R | 0 号处理器核的 IPI_Status 寄存器 |
| Core0_IPI_Enalbe | 0x3ff01004 | RW | 0 号处理器核的 IPI_Enalbe 寄存器 |
| Core0_IPI_Set | 0x3ff01008 | W | 0 号处理器核的 IPI_Set 寄存器 |
| Core0 _IPI_Clear | 0x3ff0100c | W | 0 号处理器核的 IPI_Clear 寄存器 |
| Core0_MailBox0 | 0x3ff01020 | RW | 0 号处理器核的 IPI_MailBox0 寄存器 |
| Core0_ MailBox1 | 0x3ff01028 | RW | 0 号处理器核的 IPI_MailBox1 寄存器 |
| Core0_ MailBox2 | 0x3ff01030 | RW | 0 号处理器核的 IPI_MailBox2 寄存器 |
| Core0_ MailBox3 | 0x3ff01038 | RW | 0 号处理器核的 IPI_MailBox3 寄存器 |

表 6-3 1 号处理器核的核间中断与通信寄存器

| 名称 | 地址 | 权限 | 描述 |
|---|---|---|---|
| Core1_IPI_Status | 0x3ff01100 | R | 1 号处理器核的 IPI_Status 寄存器 |
| Core1_IPI_Enalbe | 0x3ff01104 | RW | 1 号处理器核的 IPI_Enalbe 寄存器 |
| Core1_IPI_Set | 0x3ff01108 | W | 1 号处理器核的 IPI_Set 寄存器 |
| Core1 _IPI_Clear | 0x3ff0110c | W | 1 号处理器核的 IPI_Clear 寄存器 |
| Core1_MailBox0 | 0x3ff01120 | R | 1 号处理器核的 IPI_MailBox0 寄存器 |
| Core1_ MailBox1 | 0x3ff01128 | RW | 1 号处理器核的 IPI_MailBox1 寄存器 |
| Core1_ MailBox2 | 0x3ff01130 | W | 1 号处理器核的 IPI_MailBox2 寄存器 |
| Core1_ MailBox3 | 0x3ff01138 | W | 1 号处理器核的 IPI_MailBox3 寄存器 |

表 6-4 2 号处理器核的核间中断与通信寄存器

| 名称 | 地址 | 权限 | 描述 |
|---|---|---|---|
| Core2_IPI_Status | 0x3ff01200 | R | 2 号处理器核的 IPI_Status 寄存器 |
| Core2_IPI_Enalbe | 0x3ff01204 | RW | 2 号处理器核的 IPI_Enalbe 寄存器 |
| Core2_IPI_Set | 0x3ff01208 | W | 2 号处理器核的 IPI_Set 寄存器 |
| Core2 _IPI_Clear | 0x3ff0120c | W | 2 号处理器核的 IPI_Clear 寄存器 |
| Core2_MailBox0 | 0x3ff01220 | R | 2 号处理器核的 IPI_MailBox0 寄存器 |
| Core2_ MailBox1 | 0x3ff01228 | RW | 2 号处理器核的 IPI_MailBox1 寄存器 |
| Core2_ MailBox2 | 0x3ff01230 | W | 2 号处理器核的 IPI_MailBox2 寄存器 |
| Core2_ MailBox3 | 0x3ff01238 | W | 2 号处理器核的 IPI_MailBox3 寄存器 |

表 6-5 3 号处理器核的核间中断与通信寄存器

| 名称 | 地址 | 权限 | 描述 |
|---|---|---|---|
| Core3_IPI_Status | 0x3ff01300 | R | 3 号处理器核的 IPI_Status 寄存器 |
| Core3_IPI_Enalbe | 0x3ff01304 | RW | 3 号处理器核的 IPI_Enalbe 寄存器 |
| Core3_IPI_Set | 0x3ff01308 | W | 3 号处理器核的 IPI_Set 寄存器 |
| Core3 _IPI_Clear | 0x3ff0130c | W | 3 号处理器核的 IPI_Clear 寄存器 |
| Core3_MailBox0 | 0x3ff01320 | R | 3 号处理器核的 IPI_MailBox0 寄存器 |
| Core3_ MailBox1 | 0x3ff01328 | RW | 3 号处理器核的 IPI_MailBox1 寄存器 |
| Core3_ MailBox2 | 0x3ff01330 | W | 3 号处理器核的 IPI_MailBox2 寄存器 |
| Core3_ MailBox3 | 0x3ff01338 | W | 3 号处理器核的 IPI_MailBox3 寄存器 |

上面列出的是龙芯 3B1500 多处理器芯片的节点 0 核间中断相关寄存器列表。在采用两片龙芯 3B1500 互连构成四节点 CC-NUMA 系统时，每个芯片内的节点对应一个系统全局节点编号，节点内处理器核的 IPI 寄存器地址按上表与其所在节点的基地址成固定偏移关系。例如，0 号节点 0 号处理器核的 IPI_Status 地址为 0x3ff01000，其它节点内的 0 号处理器核地址偏移增加 0x1000_0000_4000，比如 1 号节点的 0 号处理器地址则为 0x10003ff05000，依次类推。

不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

# 7 I/O 中断

龙芯 3B1500 芯片最多支持 32 个中断源，连入两个节点各自的中断配置寄存器，如下图 7-1 所示，任意一个 IO 中断源可以被配置为是否使能、触发的方式、以及被路由的目标处理器核中断脚。

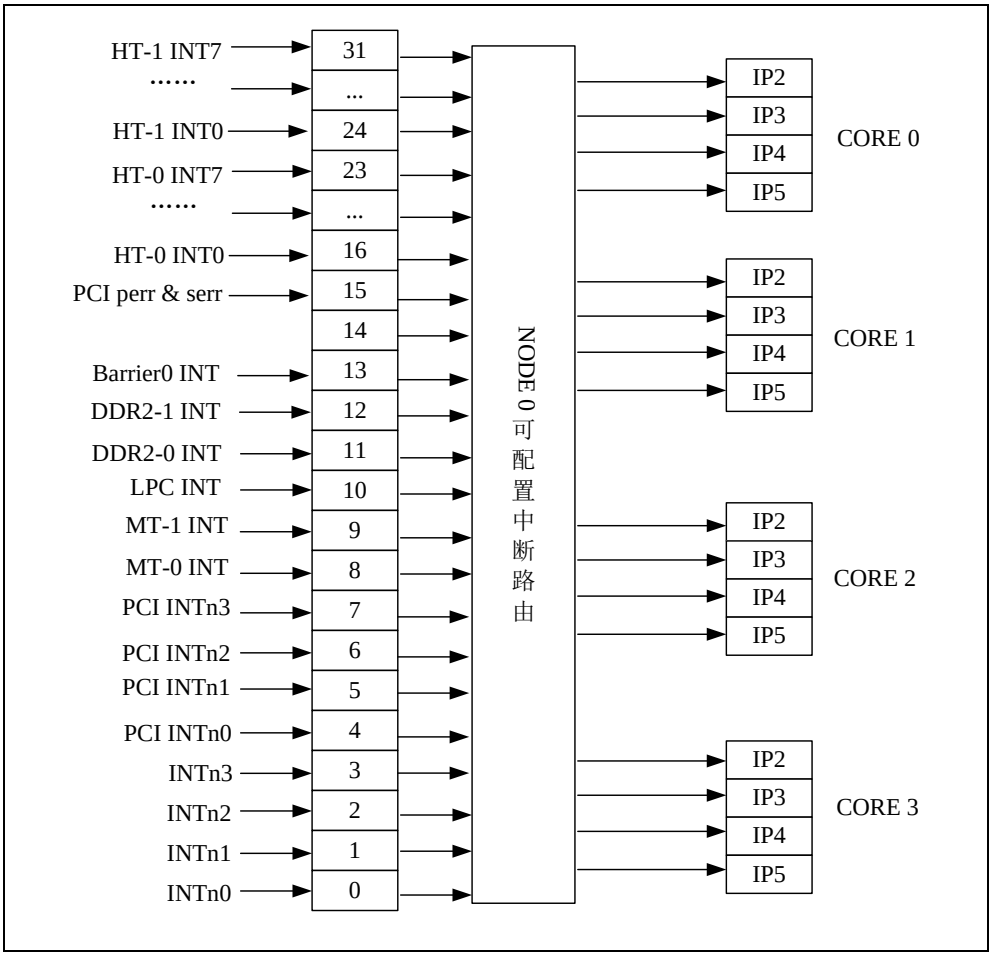



图 7-1 龙芯 3B1500 处理器中断路由示意图

中断相关配置寄存器都是以位的形式对相应的中断线进行控制，中断控制位连接及属性配置见下表 7-1。中断使能（Enable）的配置有三个寄存器：Intenset、Intenclr 和 Inten。Intenset 设置中断使能，

Intenset 寄存器写 1 的位对应的中断使能被设置，写 0 没有任何作用。Intenclr 清除中断使能，写 0 没有任何作用，Intenclr 寄存器写 1 的位对应的中断使能被清除，写 0 没有任何作用。Inten 寄存器读取当前各中断使能的情况，为 1 时对应的中断源产生的中断可以被路由，为 0 时对应的中断源产生的中断被屏蔽。

脉冲形式的中断信号（如 PCI_SERR）由 Intedge 配置寄存器来选择，写 1 表示脉冲触发，写 0 表示电平触发。中断处理程序可以通过 Intenclr 的相应位来清除脉冲记录。

所有的片内中断（包括 HT、DDR、MT、LPC）都为电平触发。

表 7-1 中断控制寄存器

| 位域 | 访问属性/缺省值 | | | | |
|---|---|---|---|---|---|
| | Intedge | Inten | Intenset | Intenclr | 中断源 |
| 3 : 0 | RW / 0 | R / 0 | W / 0 | W / 0 | Sys_int0-3 |
| 7 : 4 | RO / 0 | R / 0 | RW / 0 | RW / 0 | PCI_INTn |
| 8 | RO / 0 | R / 0 | RW / 0 | RW / 0 | Matrix_int0 |
| 9 | RO / 0 | R / 0 | RW / 0 | RW / 0 | Matrix_int1 |
| 10 | RO / 0 | R / 0 | RW / 0 | RW / 0 | Lpc |
| 12 : 11 | RW / 0 | R / 0 | RW / 0 | RW / 0 | Mc0-1 |
| 13 | RW / 0 | R / 0 | RW / 0 | RW / 0 | Barrier |
| 14 | RW / 0 | R / 0 | RW / 0 | RW / 0 | 保留 |
| 15 | RW / 0 | R / 0 | RW / 0 | RW / 0 | Pci_perr |
| 23 : 16 | RW / 0 | R / 0 | RW / 0 | RW / 0 | HT0 int0-7 |
| 31 : 24 | RW / 0 | R / 0 | RW / 0 | RW / 0 | HT1 int0-7 |

表 7-2 节点 0 IO 控制寄存器地址

| 名称 | 地址偏移 | 描述 |
|---|---|---|
| Intisr | 0x3ff01420 | 32 位中断状态寄存器 |
| Inten | 0x3ff01424 | 32 位中断使能状态寄存器 |
| Intenset | 0x3ff01428 | 32 位使能设置寄存器 |
| Intenclr | 0x3ff0142c | 32 位使能清除寄存器 |
| Intedge | 0x3ff01434 | 32 位触发方式寄存器 |
| CORE0_INTISR | 0x3ff01440 | 路由给 CORE0 的 32 位中断状态 |
| CORE1_INTISR | 0x3ff01448 | 路由给 CORE1 的 32 位中断状态 |
| CORE2_INTISR | 0x3ff01450 | 路由给 CORE2 的 32 位中断状态 |
| CORE3_INTISR | 0x3ff01458 | 路由给 CORE3 的 32 位中断状态 |

节点 1 的处理器核相关寄存器的地址是在节点 0 的地址基础上增加 0x1000_0000_4000，如节点 1 的 Intisr 的地址是 0x1000_3FF0_5420。

不同结点的配置空间基址分别为 0x3FF00000 + 0x1000_00004000 * NODE。这个原则对所有的 0x3FF00000 空间配置寄存器都适用。

在龙芯 3B1500 中集成了两个节点，每个节点包括 4 个处理器核，上述的 32 位中断源可以通过软件配置选择期望中断的目标处理器核。进一步，中断源可以选择路由到处理器核中断 IP2 到 IP5 中的任意一个，即对应 CP0_Status 的 IP2 到 IP5。每个节点内的 32 个 I/O 中断源中每一个都对应一个 8 位的路由控制器，其格式和地址如下表 7-3 和 7-4 所示。路由寄存器采用向量的方式进行路由选择，如 0x48 标示路由到 3 号处理器的 IP4 上。

表 7-3 中断路由寄存器的说明

| 位域 | 说 明 |
|---|---|
| 3:0 | 路由的处理器核向量号（第 0 位代表 0 号处理器核，第 1 位代表 1 号...） |
| 7:4 | 路由的处理器核中断引脚向量号（第 0 位代表 IP2，第 1 位代表 IP3...） |

表 7-4 中断路由寄存器地址

| 名称 | 地址偏移 | 描述 | 名称 | 地址偏移 | 描述 |
|---|---|---|---|---|---|
| Entry0 | 0x3ff01400 | Sys_int0 | Entry16 | 0x3ff01410 | HT0-int0 |
| Entry1 | 0x3ff01401 | Sys_int1 | Entry17 | 0x3ff01411 | HT0-int1 |
| Entry2 | 0x3ff01402 | Sys_int2 | Entry18 | 0x3ff01412 | HT0-int2 |
| Entry3 | 0x3ff01403 | Sys_int3 | Entry19 | 0x3ff01413 | HT0-int3 |
| Entry4 | 0x3ff01404 | Pci_int0 | Entry20 | 0x3ff01414 | HT0-int4 |
| Entry5 | 0x3ff01405 | Pci_int1 | Entry21 | 0x3ff01415 | HT0-int5 |
| Entry6 | 0x3ff01406 | Pci_int2 | Entry22 | 0x3ff01416 | HT0-int6 |
| Entry7 | 0x3ff01407 | Pci_int3 | Entry23 | 0x3ff01417 | HT0-int7 |
| Entry8 | 0x3ff01408 | Matrix int0 | Entry24 | 0x3ff01418 | HT1-int0 |
| Entry9 | 0x3ff01409 | Matrix int1 | Entry25 | 0x3ff01419 | HT1-int1 |
| Entry10 | 0x3ff0140a | Lpc int | Entry26 | 0x3ff0141a | HT1-int2 |
| Entry11 | 0x3ff0140b | Mc0 | Entry27 | 0x3ff0141b | HT1-int3 |
| Entry12 | 0x3ff0140c | Mc1 | Entry28 | 0x3ff0141c | HT1-int4 |
| Entry13 | 0x3ff0140d | Barrier | Entry29 | 0x3ff0141d | HT1-int5 |
| Entry14 | 0x3ff0140e | 保留 | Entry30 | 0x3ff0141e | HT1-int6 |
| Entry15 | 0x3ff0140f | Pci_perr/serr | Entry31 | 0x3ff0141f | HT1-int7 |

# 8 全局时钟

龙芯 3B1500 芯片内部提供了两个 64 位全局时钟供软件使用。

两个全局时钟的物理地址分别为 0x3FF0_0408 和 0x1000_3FF0_4408。访问时必须使用 UNCACHE 地址。

全局时钟的运行频率与节点时钟（NODE CLOCK）相同，每个时钟周期自增 1。该时钟为只读 64 位，对该时钟的写不产生任何效果。

对于双路系统，NODE2 与 NODE3 的两个全局时钟物理地址分别为 0x2000_3FF0_8408 和 0x3000_3FF0_C408。

# 9 DDR2/3 SDRAM 控制器配置

龙芯 3B1500 处理器内部集成的内存控制器的设计遵守 DDR2/3 SDRAM 的行业标准（JESD79）。

## 9.1 DDR2/3 SDRAM 参数配置格式

参数列表及说明如下表：

表 9-1 DDR2/3 参数列表

| | 63:56 | 55:48 | 47:40 | 39:32 | 31:24 | 23:16 | 15:8 | 7:0 |
|---|---|---|---|---|---|---|---|---|
| 0x000 | Dll_value_0(RD) | | Dll_value_ck(RD) | | Dll_init_done(RD) | | Version(RD) | |
| 0x008 | Dll_value_4(RD) | | Dll_value_3(RD) | | Dll_value_2(RD) | | Dll_value_1(RD) | |
| 0x010 | Dll_value_8(RD) | | Dll_value_7(RD) | | Dll_value_6(RD) | | Dll_value_5(RD) | |
| 0x018 | Dll_ck_3 | Dll_ck_2 | Dll_ck_1 | Dll_ck_0 | Dll_increment | Dll_start_point | Dll_bypass | Init_start |
| 0x020 | Dq_oe_end_0 | Dq_oe_begin_0 | Dq_stop_edge_0 | Dq_start_edge_0 | Rddata_delay_0 | Rddqs_lt_half_0 | Wrdqs_lt_half_0 | Wrdq_lt_half_0 |
| 0x028 | Rd_oe_end_0 | Rd_oe_begin_0 | Rd_stop_edge_0 | Rd_start_edge_0 | Dqs_oe_end_0 | Dqs_oe_begin_0 | Dqs_stop_edge_0 | Dqs_start_edge_0 |
| 0x030 | | | | Wrdq_clkdelay_0 | Odt_oe_end_0 | Odt_oe_begin_0 | Odt_stop_edge_0 | Odt_start_edge_0 |
| 0x038 | | | | Dll_rddqs_n_0 | Dll_rddqs_p_0 | Dll_wrdqs_0 | Dll_wrdata_0 | Dll_gate_0 |
| 0x040 | Dq_oe_end_1 | Dq_oe_begin_1 | Dq_stop_edge_1 | Dq_start_edge_1 | Rddata_delay_1 | Rddqs_lt_half_1 | Wrdqs_lt_half_1 | Wrdq_lt_half_1 |
| 0x048 | Rd_oe_end_1 | Rd_oe_begin_1 | Rd_stop_edge_1 | Rd_start_edge_1 | Dqs_oe_end_1 | Dqs_oe_begin_1 | Dqs_stop_edge_1 | Dqs_start_edge_1 |
| 0x050 | | | | Wrdq_clkdelay_1 | Odt_oe_end_1 | Odt_oe_begin_1 | Odt_stop_edge_1 | Odt_start_edge_1 |
| 0x058 | | | | Dll_rddqs_n_1 | Dll_rddqs_p_1 | Dll_wrdqs_1 | Dll_wrdata_1 | Dll_gate_1 |
| 0x060 | Dq_oe_end_2 | Dq_oe_begin_2 | Dq_stop_edge_2 | Dq_start_edge_2 | Rddata_delay_2 | Rddqs_lt_half_2 | Wrdqs_lt_half_2 | Wrdq_lt_half_2 |
| 0x068 | Rd_oe_end_2 | Rd_oe_begin_2 | Rd_stop_edge_2 | Rd_start_edge_2 | Dqs_oe_end_2 | Dqs_oe_begin_2 | Dqs_stop_edge_2 | Dqs_start_edge_2 |
| 0x070 | | | | Wrdq_clkdelay_2 | Odt_oe_end_2 | Odt_oe_begin_2 | Odt_stop_edge_2 | Odt_start_edge_2 |
| 0x078 | | | | Dll_rddqs_n_2 | Dll_rddqs_p_2 | Dll_wrdqs_2 | Dll_wrdata_2 | Dll_gate_2 |
| 0x080 | Dq_oe_end_3 | Dq_oe_begin_3 | Dq_stop_edge_3 | Dq_start_edge_3 | Rddata_delay_3 | Rddqs_lt_half_3 | Wrdqs_lt_half_3 | Wrdq_lt_half_3 |
| 0x088 | Rd_oe_end_3 | Rd_oe_begin_3 | Rd_stop_edge_3 | Rd_start_edge_3 | Dqs_oe_end_3 | Dqs_oe_begin_3 | Dqs_stop_edge_3 | Dqs_start_edge_3 |
| 0x090 | | | | Wrdq_clkdelay_3 | Odt_oe_end_3 | Odt_oe_begin_3 | Odt_stop_edge_3 | Odt_start_edge_3 |
| 0x098 | | | | Dll_rddqs_n_3 | Dll_rddqs_p_3 | Dll_wrdqs_3 | Dll_wrdata_3 | Dll_gate_3 |
| 0x0A0 | Dq_oe_end_4 | Dq_oe_begin_4 | Dq_stop_edge_4 | Dq_start_edge_4 | Rddata_delay_4 | Rddqs_lt_half_4 | Wrdqs_lt_half_4 | Wrdq_lt_half_4 |
| 0x0A8 | Rd_oe_end_4 | Rd_oe_begin_4 | Rd_stop_edge_4 | Rd_start_edge_4 | Dqs_oe_end_4 | Dqs_oe_begin_4 | Dqs_stop_edge_4 | Dqs_start_edge_4 |
| 0x0B0 | | | | Wrdq_clkdelay_4 | Odt_oe_end_4 | Odt_oe_begin_4 | Odt_stop_edge_4 | Odt_start_edge_4 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x0B8 | | | | Dll_rddqs_n_4 | Dll_rddqs_p_4 | Dll_wrdqs_4 | Dll_wrdata_4 | Dll_gate_4 |
| 0x0C0 | Dq_oe_end_5 | Dq_oe_begin_5 | Dq_stop_edge_5 | Dq_start_edge_5 | Rddata_delay_5 | Rddqs_lt_half_5 | Wrdqs_lt_half_5 | Wrdq_lt_half_5 |
| 0x0C8 | Rd_oe_end_5 | Rd_oe_begin_5 | Rd_stop_edge_5 | Rd_start_edge_5 | Dqs_oe_end_5 | Dqs_oe_begin_5 | Dqs_stop_edge_5 | Dqs_start_edge_5 |
| 0x0D0 | | | | Wrdq_clkdelay_5 | Odt_oe_end_5 | Odt_oe_begin_5 | Odt_stop_edge_5 | Odt_start_edge_5 |
| 0x0D8 | | | | Dll_rddqs_n_5 | Dll_rddqs_p_5 | Dll_wrdqs_5 | Dll_wrdata_5 | Dll_gate_5 |
| 0x0E0 | Dq_oe_end_6 | Dq_oe_begin_6 | Dq_stop_edge_6 | Dq_start_edge_6 | Rddata_delay_6 | Rddqs_lt_half_6 | Wrdqs_lt_half_6 | Wrdq_lt_half_6 |
| 0x0E8 | Rd_oe_end_6 | Rd_oe_begin_6 | Rd_stop_edge_6 | Rd_start_edge_6 | Dqs_oe_end_6 | Dqs_oe_begin_6 | Dqs_stop_edge_6 | Dqs_start_edge_6 |
| 0x0F0 | | | | Wrdq_clkdelay_6 | Odt_oe_end_6 | Odt_oe_begin_6 | Odt_stop_edge_6 | Odt_start_edge_6 |
| 0x0F8 | | | | Dll_rddqs_n_6 | Dll_rddqs_p_6 | Dll_wrdqs_6 | Dll_wrdata_6 | Dll_gate_6 |
| 0x100 | Dq_oe_end_7 | Dq_oe_begin_7 | Dq_stop_edge_7 | Dq_start_edge_7 | Rddata_delay_7 | Rddqs_lt_half_7 | Wrdqs_lt_half_7 | Wrdq_lt_half_7 |
| 0x108 | Rd_oe_end_7 | Rd_oe_begin_7 | Rd_stop_edge_7 | Rd_start_edge_7 | Dqs_oe_end_7 | Dqs_oe_begin_7 | Dqs_stop_edge_7 | Dqs_start_edge_7 |
| 0x110 | | | | Wrdq_clkdelay_7 | Odt_oe_end_7 | Odt_oe_begin_7 | Odt_stop_edge_7 | Odt_start_edge_7 |
| 0x118 | | | | Dll_rddqs_n_7 | Dll_rddqs_p_7 | Dll_wrdqs_7 | Dll_wrdata_7 | Dll_gate_7 |
| 0x120 | Dq_oe_end_8 | Dq_oe_begin_8 | Dq_stop_edge_8 | Dq_start_edge_8 | Rddata_delay_8 | Rddqs_lt_half_8 | Wrdqs_lt_half_8 | Wrdq_lt_half_8 |
| 0x128 | Rd_oe_end_8 | Rd_oe_begin_8 | Rd_stop_edge_8 | Rd_start_edge_8 | Dqs_oe_end_8 | Dqs_oe_begin_8 | Dqs_stop_edge_8 | Dqs_start_edge_8 |
| 0x130 | | | | Wrdq_clkdelay_8 | Odt_oe_end_8 | Odt_oe_begin_8 | Odt_stop_edge_8 | Odt_start_edge_8 |
| 0x138 | | | | Dll_rddqs_n_8 | Dll_rddqs_p_8 | Dll_wrdqs_8 | Dll_wrdata_8 | Dll_gate_8 |
| 0x140 | Pad_ocd_clk | Pad_ocd_ctl | Pad_ocd_dqs | Pad_ocd_dq | Pad_enzi | | Pad_en_ctl | Pad_en_clk |
| 0x148 | Pad_adj_code_dqs | Pad_code_dqs | Pad_adj_code_dq | Pad_code_dq | | Pad_vref_internal | Pad_odt_se | Pad_modezi1v8 |
| 0x150 | | | Pad_adj_code_clk | Pad_code_lk | Pad_adj_code_cmd | Pad_code_cmd | Pad_adj_code_addr | Pad_code_addr |
| 0x158 | | | Pad_comp_okn | Pad_comp_code_o | Pad_comp_code_i | Pad_comp_mode | Pad_comp_tm | Pad_comp_pd |
| 0x160 | Rdfifo_empty(RD) | | Overflow(RD) | | Dram_init(RD) | Rdfifo_valid | Cmd_timming | Ddr3_mode |
| 0x168 | | Addr_mirror | Cmd_delay | Burst_length | Bank | Cs_zq | Cs_mrs | Cs_enable |
| 0x170 | Odt_wr_cs_map | | Odt_wr_length | Odt_wr_delay | Odt_rd_cs_map | | Odt_rd_length | Odt_rd_delay |
| 0x178 | | | | | | | | |
| 0x180 | Lvl_resp_0(RD) | Lvl_done(RD) | Lvl_ready(RD) | | Lvl_cs | tLVL_DELAY | Lvl_req(WR) | Lvl_mode |
| 0x188 | Lvl_resp_8(RD) | Lvl_resp_7(RD) | Lvl_resp_6(RD) | Lvl_resp_5(RD) | Lvl_resp_4(RD) | Lvl_resp_3(RD) | Lvl_resp_2(RD) | Lvl_resp_1(RD) |
| 0x190 | Cmd_a | | Cmd_ba | Cmd_cmd | Cmd_cs | Status_cmd(RD) | Cmd_req(WR) | Command_mode |
| 0x198 | | | Status_sref(RD) | Srefresh_req | Pre_all_done(RD) | Pre_all_req(RD) | Mrs_done(RD) | Mrs_req(WR) |
| 0x1A0 | Mr_3_cs_0 | | Mr_2_cs_0 | | Mr_1_cs_0 | | Mr_0_cs_0 | |
| 0x1A8 | Mr_3_cs_1 | | Mr_2_cs_1 | | Mr_1_cs_1 | | Mr_0_cs_1 | |
| 0x1B0 | Mr_3_cs_2 | | Mr_2_cs_2 | | Mr_1_cs_2 | | Mr_0_cs_2 | |
| 0x1B8 | Mr_3_cs_3 | | Mr_2_cs_3 | | Mr_1_cs_3 | | Mr_0_cs_3 | |
| 0x1C0 | tRESET | tCKE | tXPR | tMOD | tZQCL | tZQ_CMD | tWLDQSEN | tRDDATA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x1C8 | tFAW | tRRD | tRCD | tRP | tREF | tRFC | tZQCS | tZQperiod |
| 0x1D0 | tODTL | tXSRD | tPHY_RDLAT | tPHY_WRLAT | tRAS_max | | | tRAS_min |
| 0x1D8 | tXPDLL | tXP | tWR | tRTP | tRL | tWL | tCCD | tWTR |
| 0x1E0 | tW2R_diffCS | tW2W_diffCS | tR2P_sameBA | tW2P_sameBA | tR2R_sameBA | tR2W_sameBA | tW2R_sameBA | tW2W_sameBA |
| 0x1E8 | tR2R_diffCS | tR2W_diffCS | tR2P_sameCS | tW2P_sameCS | tR2R_sameCS | tR2W_sameCS | tW2R_sameCS | tW2W_sameCS |
| 0x1F0 | Power_up | Age_step | tCPDED | Cs_map | Bs_config | Nc | Pr_r2w | Placement_en |
| 0x1F8 | Hw_pd_3 | Hw_pd_2 | Hw_pd_1 | Hw_pd_0 | Credit_16 | Credit_32 | Credit_64 | Selection_en |
| 0x200 | Cmdq_age_16 | | Cmdq_age_32 | | Cmdq_age_64 | | tCKESR | tRDPDEN |
| 0x208 | Wfifo_age | | Ffifo_age | | Power_stat3 | Power_stat2 | Power_stat1 | Power_stat0 |
| 0x210 | Active_age | | Cs_place_0 | Addr_win_0 | Cs_diff_0 | Row_diff_0 | Ba_diff_0 | Col_diff_0 |
| 0x218 | Fastpd_age | | Cs_place_1 | Addr_win_1 | Cs_diff_1 | Row_diff_1 | Ba_diff_1 | Col_diff_1 |
| 0x220 | Slowpd_age | | Cs_place_2 | Addr_win_2 | Cs_diff_2 | Row_diff_2 | Ba_diff_2 | Col_diff_2 |
| 0x228 | Selfref_age | | Cs_place_3 | Addr_win_3 | Cs_diff_3 | Row_diff_3 | Ba_diff_3 | Col_diff_3 |
| 0x230 | Win_mask_0 | | | | Win_base_0 | | | |
| 0x238 | Win_mask_1 | | | | Win_base_1 | | | |
| 0x240 | Win_mask_2 | | | | Win_base_2 | | | |
| 0x248 | Win_mask_3 | | | | Win_base_3 | | | |
| 0x250 | | Cmd_monitor | Axi_monitor | | Ecc_code(RD) | Ecc_enable | Int_vector | Int_enable |
| 0x258 | | | | | | | | |
| 0x260 | Ecc_addr(RD) | | | | | | | |
| 0x268 | Ecc_data(RD) | | | | | | | |
| 0x270 | Lpbk_ecc_mask(RD) | Prbs_init | | | Lpbk_error(RD) | Prbs_23 | Lpbk_start | Lpbk_en |
| 0x278 | Lpbk_ecc(RD) | | Lpbk_data_mask(RD) | | Lpbk_correct(RD) | | Lpbk_counter(RD) | |
| 0x280 | Lpbk_data_r(RD) | | | | | | | |
| 0x288 | Lpbk_data_f(RD) | | | | | | | |
| 0x290 | Axi0_bandwidth_w | | | | Axi0_bandwidth_r | | | |
| 0x298 | Axi0_latency_w | | | | Axi0_latency_r | | | |
| 0x2A0 | Axi1_bandwidth_w | | | | Axi1_bandwidth_r | | | |
| 0x2A8 | Axi1_latency_w | | | | Axi1_latency_r | | | |
| 0x2B0 | Axi2_bandwidth_w | | | | Axi2_bandwidth_r | | | |
| 0x2B8 | Axi2_latency_w | | | | Axi2_latency_r | | | |
| 0x2C0 | Axi3_bandwidth_w | | | | Axi3_bandwidth_r | | | |
| 0x2C8 | Axi3_latency_w | | | | Axi3_latency_r | | | |
| 0x2D0 | Axi4_bandwidth_w | | | | Axi4_bandwidth_r | | | |

| | | |
|---|---|---|
| 0x2D8 | Axi4_latency_w | Axi4_latency_r |
| 0x2E0 | Cmdq0_bandwidth_w | Cmdq0_bandwidth_r |
| 0x2E8 | Cmdq0_latency_w | Cmdq0_latency_r |
| 0x2F0 | Cmdq1_bandwidth_w | Cmdq1_bandwidth_r |
| 0x2F8 | Cmdq1_latency_w | Cmdq1_latency_r |
| 0x300 | Cmdq2_bandwidth_w | Cmdq2_bandwidth_r |
| 0x308 | Cmdq2_latency_w | Cmdq2_latency_r |
| 0x310 | Cmdq3_bandwidth_w | Cmdq3_bandwidth_r |
| 0x318 | Cmdq3_latency_w | Cmdq3_latency_r |

详细说明列表：

表 9-2 DDR2 SDRAM 配置参数寄存器格式

| 0x000 | 位域 | 读写 | 缺省值 | 说明 |
|---|---|---|---|---|
| Dll_value_0 | 56:48 | 只读 | 0x0 | 第 0 组数据 DLL 锁定值 |
| Dll_value_ck | 40:32 | 只读 | 0x0 | 时钟组 DLL 锁定值 |
| Dll_init_done | 25:16 | 只读 | 0x0 | 控制器内部 DLL 锁定信号<br>[25:17]：对应 9 组数据的 DLL 锁定信号<br>[16:16]：对应时钟组 DLL 锁定信号 |
| Version | 15:0 | 只读 | 0x1 | 控制器版本号 |
| 0x008 | | | | |
| Dll_value_4 | 56:48 | 只读 | 0x0 | 第 4 组数据 DLL 锁定值 |
| Dll_value_3 | 40:32 | 只读 | 0x0 | 第 3 组数据 DLL 锁定值 |
| Dll_value_2 | 24:16 | 只读 | 0x0 | 第 2 组数据 DLL 锁定值 |
| Dll_value_1 | 8:0 | 只读 | 0x0 | 第 1 组数据 DLL 锁定值 |
| 0x010 | | | | |
| Dll_value_8 | 56:48 | 只读 | 0x0 | 第 8 组数据 DLL 锁定值 |
| Dll_value_7 | 40:32 | 只读 | 0x0 | 第 7 组数据 DLL 锁定值 |
| Dll_value_6 | 24:16 | 只读 | 0x0 | 第 6 组数据 DLL 锁定值 |
| Dll_value_5 | 8:0 | 只读 | 0x0 | 第 5 组数据 DLL 锁定值 |
| 0x018 | | | | |
| Dll_ck_3 | 63:56 | 读写 | 0x0 | 时钟 3 延迟值<br>[63:63]：bypass 控制<br>[62:56]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |

| | | | | |
|---|---|---|---|---|
| Dll_ck_2 | 55:48 | 读写 | 0x0 | 时钟 2 延迟值<br>[55:55]：bypass 控制<br>[54:48]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_ck_1 | 47:40 | 读写 | 0x0 | 时钟 1 延迟值<br>[47:47]：bypass 控制<br>[46:40]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_ck_0 | 39:32 | 读写 | 0x0 | 时钟 0 延迟值<br>[39:39]：bypass 控制<br>[38:32]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_increment | 31:24 | 读写 | 0x4 | 每次 DLL 下溢时，起始延迟单元增加个数 |
| Dll_start_point | 23:16 | 读写 | 0x10 | DLL 初始化的起始延迟单元个数 |
| Dll_bypass | 8:8 | 读写 | 0x0 | DLL 初始化 bypass 控制。<br>该位只有当 Dll_init_done 一直无法锁定时需要设置，以使内存控制器初始化可以继续进行。<br>正确设置该位的方法是在 Init_start 有效一段时间后再设为 1。而且设置之前相应的 DLL 延迟的最高位（bypass 控制）也应该设置 |
| Init_start | 0:0 | 读写 | 0x0 | 控制器初始化开始。<br>只有当其它的所有相关参数设置好了之后才可以将该位置位，使控制器进行初始化，并向内存发起初始化。只有这个操作完成后内存空间才可以被访问，否则内存空间不可被外部访问。 |
| 0x020 | | | | |
| Dq_oe_end_0 | 59:56 | 读写 | 0x2 | 第 0 组数据输出有效时期的结束时间，不可小于 Dq_oe_begin_0 |
| Dq_oe_begin_0 | 51:48 | 读写 | 0x2 | 第 0 组数据输出有效时期的开始时间，不可大于 Dq_oe_end_0 |
| Dq_stop_edge_0 | 41:40 | 读写 | 0x0 | 第 0 组数据输出有效时期的结束相位，其与 Dq_oe_end_0 组合得到的时钟边沿不可早于 Dq_start_edge_0 与 Dq_oe_begin_0 组合得到的时钟边沿 |

| | | | | |
|---|---|---|---|---|
| | | | | 0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| Dq_start_edge_0 | 33:32 | 读写 | 0x0 | 第 0 组数据输出有效时期的开始相位，其与 Dq_oe_begin_0 组合得到的时钟边沿不可晚于 Dq_stop_edge_0 与 Dq_oe_end_0 组合得到的时钟边沿<br>0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| Rddata_delay_0 | 24:24 | 读写 | 0x1 | 读返回数据在 FIFO 中延迟一周期输出 |
| Rddqs_lt_half_0 | 16:16 | 读写 | 0x0 | 当读返回 DQS 信号(延时后)相比内部时钟的延迟小于半周期时需要设为 1 |
| Wrdqs_lt_half_0 | 8:8 | 读写 | 0x0 | 当 Dll_wrdqs_0 的设置小于 0x40 时需要设为 1 |
| Wrdq_lt_half_0 | 0:0 | 读写 | 0x0 | 当 Dll_wrdata_0 的设置小于 0x40 时需要设为 1 |
| 0x028 | | | | |
| Rd_oe_end_0 | 59:56 | 读写 | 0x1 | 第 0 组数据读采样有效时期的结束时间，不可小于 Rd_oe_begin_0 |
| Rd_oe_begin_0 | 51:48 | 读写 | 0x1 | 第 0 组数据读采样有效时期的开始时间，不可大于 Rd_oe_end_0 |
| Rd_stop_edge_0 | 41:40 | 读写 | 0x0 | 第 0 组数据读采样有效时期的结束相位，其与 Rd_oe_end_0 组合得到的时钟边沿不可早于 Rd_start_edge_0 与 Rd_oe_begin_0 组合得到的时钟边沿<br>0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| Rd_start_edge_0 | 33:32 | 读写 | 0x0 | 第 0 组数据读采样有效时期的开始相位，其与 Rd_oe_begin_0 组合得到的时钟边沿不可晚于 Rd_stop_edge_0 与 Rd_oe_end_0 组合得到的时钟边沿<br>0 – 比为 1 时提前 1/4 周期 |

| | | | | |
|---|---|---|---|---|
| | | | | 1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| Dqs_oe_end_0 | 27:24 | 读写 | 0x2 | 第 0 组数据写 DQS 有效时期的结束时间，不可小于 Dqs_oe_begin_0 |
| Dqs_oe_begin_0 | 19:16 | 读写 | 0x1 | 第 0 组数据写 DQS 有效时期的开始时间，不可大于 Dqs_oe_end_0 |
| Dqs_stop_edge_0 | 9:8 | 读写 | 0x1 | 第 0 组数据写 DQS 有效时期的结束相位，其与 Dqs_oe_end_0 组合得到的时钟边沿不可早于 Dqs_start_edge_0 与 Dqs_oe_begin_0 组合得到的时钟边沿<br>0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| Dqs_start_edge_0 | 1:0 | 读写 | 0x1 | 第 0 组数据写 DQS 有效时期的开始相位，其与 Dqs_oe_begin_0 组合得到的时钟边沿不可晚于 Dqs_stop_edge_0 与 Dqs_oe_end_0 组合得到的时钟边沿<br>0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| 0x030 | | | | |
| Wrdq_clkdelay_0 | 32:32 | 读写 | 0x0 | 第 0 组数据写 DQ 延迟控制信号<br>在 Wrdq_lt_half_0 = 0 的时候将本组数据延迟增加一拍 |
| Odt_oe_end_0 | 27:24 | 读写 | 0x2 | 第 0 组数据读 ODT（控制器内部）有效时期的结束时间，不可小于 Odt_oe_begin_0 |
| Odt_oe_begin_0 | 19:16 | 读写 | 0x1 | 第 0 组数据读 ODT（控制器内部）有效时期的开始时间，不可大于 Odt_oe_end_0 |
| Odt_stop_edge_0 | 9:8 | 读写 | 0x0 | 第 0 组数据读 ODT（控制器内部）有效时期的结束相位，其与 Odt_oe_end_0 组合得到的时钟边沿不可早于 Odt_start_edge_0 与 Odt_oe_begin_0 组合得到的 |

| | | | | |
|---|---|---|---|---|
| | | | | 时钟边沿<br>0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| Odt_start_edge_0 | 1:0 | 读写 | 0x0 | 第 0 组数据读 ODT（控制器内部）有效时期的开始相位，其与 Odt_oe_begin_0 组合得到的时钟边沿不可晚于 Odt_stop_edge_0 与 Odt_oe_end_0 组合得到的时钟边沿<br>0 – 比为 1 时提前 1/4 周期<br>1 – 对应于 wrdqs_0（第 0 组写 DQS）的上升沿<br>2 – 比为 1 时推后 1/4 周期<br>3 – 比为 1 时推后 1/2 周期 |
| 0x038 | | | | |
| Dll_rddqs_n_0 | 39:32 | 读写 | 0x20 | 读 DQSn 采样延迟值<br>[39:39]：bypass 控制<br>[38:32]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_rddqs_p_0 | 31:24 | 读写 | 0x20 | 读 DQSp 采样延迟值<br>[31:31]：bypass 控制<br>[30:24]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_wrdqs_0 | 23:16 | 读写 | 0x7F | 写 DQS 延迟值<br>[23:23]：bypass 控制<br>[22:16]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_wrdata_0 | 15:8 | 读写 | 0x60 | 写数据延迟值（应该比 DQS 提前 1/4 周期）<br>[15:15]：bypass 控制<br>[14:8]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |
| Dll_gate_0 | 7:0 | 读写 | 0x0 | 读 DQS 采样有效时期控制延迟值<br>[7:7]：bypass 控制<br>[6:0]：当 bypass = 0 时，表示 n/128 个时钟周期<br>当 bypass = 1 时，表示 n 个延迟单元 |

| | | | | |
|---|---|---|---|---|
| 0x040 | | | | |
| | | | | |
| 0x138 | | | | |
| | | | | |
| 0x140 | | | | |
| Pad_ocd_clk | 58:56 | 读写 | 0x0 | 时钟引脚输出阻抗控制<br>000 – 40 欧姆<br>001 – 30 欧姆<br>010 – 24 欧姆<br>011 – 20 欧姆<br>100 – 15 欧姆 |
| Pad_ocd_ctl | 50:48 | 读写 | 0x0 | 控制引脚输出阻抗控制<br>000 – 40 欧姆<br>001 – 30 欧姆<br>010 – 24 欧姆<br>011 – 20 欧姆<br>100 – 15 欧姆 |
| Pad_ocd_dqs | 40:40 | 读写 | 0x0 | DQS 引脚输出阻抗控制<br>0 – 34 欧姆<br>1 – 40 欧姆 |
| Pad_ocd_dq | 32:32 | 读写 | 0x0 | DQ 引脚输出阻抗控制<br>0 – 34 欧姆<br>1 – 40 欧姆 |
| Pad_enzi | 24:16 | 读写 | 0x0 | 分别对应 9 个数据组的引脚输入使能<br>1 – 使能<br>0 – 高阻 |
| Pad_en_ctl | 8:8 | 读写 | 0x0 | 控制引脚输出使能<br>1 – 使能<br>0 – 高阻 |
| Pad_en_clk | 7:0 | 读写 | 0x0 | 时钟引脚输出使能<br>1 – 使能<br>0 – 高阻 |
| 0x148 | | | | |
| Pad_adj_code_dqs | 63:56 | 读写 | 0x0 | 设置当 Pad_code_dqs[0]有效时 DQS 信号附加 CODE |

| | | | | |
|---|---|---|---|---|
| | | | | [7:4] N_CODE：1 使能，0 关闭<br>[3:0] P_CODE：0 关闭，1 使能 |
| Pad_code_dqs | 50:48 | 读写 | 0x0 | DQS 信号附加 CODE 使能设置<br>Bit 2：0 有效，表示附加码作用于输出及 ODT<br>Bit 1：0 有效，表示附加码作用于 SLEWRATE<br>Bit 0：附加 CODE 符号位，0 为正，1 为负 |
| Pad_adj_code_dq | 47:40 | 读写 | 0x0 | 设置当 Pad_code_dq[0]有效时 DQ 信号附加 CODE<br>[7:4] N_CODE：1 使能，0 关闭<br>[3:0] P_CODE：0 关闭，1 使能 |
| Pad_code_dq | 34:32 | 读写 | 0x0 | DQ 信号附加 CODE 使能设置<br>Bit 2：0 有效，表示附加码作用于输出及 ODT<br>Bit 1：0 有效，表示附加码作用于 SLEWRATE<br>Bit 0：附加 CODE 符号位，0 为正，1 为负 |
| Pad_vref_internal | 16:16 | 读写 | 0x0 | 使能内部 VREF 分压电路<br>1 - 同时使用内部 VREF 分压与外部引脚输出电压<br>0 - 只使用外部引脚输出电压 |
| Pad_odt_se | 8:8 | 读写 | 0x0 | 引脚匹配电阻值控制<br>0 – 60 欧姆<br>1 – 120 欧姆 |
| Pad_modezi1v8 | 0:0 | 读写 | 0x0 | PAD MODE ZI 1v8<br>1 – 使用 PAD 的 ZITEST 输入<br>0 – 使用 PAD 的 ZI 输入 |
| 0x150 | | | | |
| Pad_adj_code_clk | 47:40 | 读写 | 0x0 | 设置当 Pad_code_clk[0]有效时 CLK 信号附加 CODE<br>[7:4] N_CODE：1 使能，0 关闭<br>[3:0] P_CODE：0 关闭，1 使能 |
| Pad_code_clk | 34:32 | 读写 | 0x0 | CLK 信号附加 CODE 使能设置<br>Bit 2：0 有效，表示附加码作用于输出及 ODT<br>Bit 1：0 有效，表示附加码作用于 SLEWRATE<br>Bit 0：附加 CODE 符号位，0 为正，1 为负 |
| Pad_adj_code_cmd | 31:24 | 读写 | 0x0 | 设置当 Pad_code_cmd[0]有效时 CMD 信号附加 CODE<br>[7:4] N_CODE：1 使能，0 关闭<br>[3:0] P_CODE：0 关闭，1 使能 |

| | | | | |
|---|---|---|---|---|
| Pad_code_cmd | 18:16 | 读写 | 0x0 | CMD 信号附加 CODE 使能设置<br>Bit 2：0 有效，表示附加码作用于输出及 ODT<br>Bit 1：0 有效，表示附加码作用于 SLEWRATE<br>Bit 0：附加 CODE 符号位，0 为正，1 为负 |
| Pad_adj_code_addr | 15:8 | 读写 | 0x0 | 设置当 Pad_code_addr[0]有效时 ADDR 信号附加 CODE<br>[7:4] N_CODE：1 使能，0 关闭<br>[3:0] P_CODE：0 关闭，1 使能 |
| Pad_code_addr | 2:0 | 读写 | 0x0 | ADDR 信号附加 CODE 使能设置<br>Bit 2：0 有效，表示附加码作用于输出及 ODT<br>Bit 1：0 有效，表示附加码作用于 SLEWRATE<br>Bit 0：附加 CODE 符号位，0 为正，1 为负 |
| 0x158 | | | | |
| Pad_comp_okn | 40:40 | 只读 | 0x0 | 引脚补偿单元自动调节完成标志 |
| Pad_comp_code_o | 39:32 | 只读 | 0x0 | 引脚补偿单元自动调节调整值 |
| Pad_comp_code_i | 31:24 | 读写 | 0xF0 | 引脚补偿单元手动设置值<br>[7:4] N_CODE：1 使能，0 关闭<br>[3:0] P_CODE：0 关闭，1 使能 |
| Pad_comp_mode | 16:16 | 读写 | 0x0 | 引脚补偿单元设置<br>1 – 手动设置 CODE<br>0 – 自动调节 CODE |
| Pad_comp_tm | 8:8 | 读写 | 0x0 | 外部引脚测试模块使能<br>1 – 使用引脚 COMP_NOUT/COMP_POUT 连接电阻<br>0 – 使用引脚 COMP_REXT 连接电阻 |
| Pad_comp_pd | 0:0 | 读写 | 0x1 | 引脚补偿单元 Power Down<br>1 – Power Down<br>0 – 正常工作 |
| 0x160 | | | | |
| Rdfifo_empty | 56:48 | 只读 | 0x0 | PHY 中收集每个 SLICE 的读 FIFO 错误读出标志，当对应的 FIFO 为空时发生出队列操作时有效。可以用于判断 Rdfifo_valid 无效时，tPHY_RDLAT 的值是否设置过小<br>每一位分别对应于 SLICE8 … SLICE0 |
| Overflow | 40:32 | 只读 | 0x0 | PHY 中每个 SLICE 中的读 FIFO 溢出标志，每一位分 |

| | | | | |
|---|---|---|---|---|
| | | | | 别对应于 SLICE8 … SLICE0 |
| Dram_init | 27:24 | 只读 | 0x0 | DRAM 初始化完成标志，在 Init_start 设置之后才会生效，每一位分别对应于一个片选 |
| Rdfifo_valid | 16:16 | 读写 | 0x1 | 表示使用 PHY 内部逻辑控制读数据同步时间<br>该位无效时，这个同步时间由 tPHY_RDLAT 决定 |
| Cmd_timming | 9:8 | 读写 | 0x0 | 控制线 2T/3T 功能使能<br>0 – 1T<br>1 – 2T<br>2 – 3T<br>与其他几个参数需要满足关系式：<br>tRDDATA - Cmd_delay - Cmd_timing = CASLAT – 3<br>tPHY_WRLAT – Cmd_delay – Cmd_timing = WRLAT - 4 |
| Ddr3_mode | 0:0 | 读写 | 0x1 | 使用 DDR3 模式时将该位设为 1 |
| 0x168 | | | | |
| Addr_mirror | 51:48 | 读写 | 0x0 | 表示该 CS 对应的地址需要进行地址镜像 |
| Cmd_delay | 41:40 | 读写 | 0x0 | 表示命令总线需要的附加延迟<br>有效值为 0/1/2<br>与其他几个参数需要满足关系式：<br>tRDDATA - Cmd_delay - Cmd_timing = CASLAT – 3<br>tPHY_WRLAT – Cmd_delay – Cmd_timing = WRLAT - 4 |
| Burst_length | 35:32 | 读写 | 0x7 | 表示 DRAM 总线上的突发请求长度，该参数设置应与 MRS 参数一致。突发长度为 8 时设为 4’h7，突发长度为 4 时设为 4’h3 |
| Bank | 27:24 | 读写 | 0x7 | 表示每个片选上的 Bank 数量<br>Bank 数为 2 时：<br>需设置 Ba_diff = 2 , Addr_win[3:2] = 2’b01<br>Bank 数为 4 时：<br>需设置 Ba_diff = 1 , Addr_win[3:2] = 2’b10<br>Bank 数为 8 时：<br>需设置 Ba_diff = 0 , Addr_win[3:2] = 2’b11 |
| Cs_zq | 19:16 | 读写 | 0x1 | 使能对应片选信号的 ZQ 请求 |
| Cs_mrs | 11:8 | 读写 | 0x1 | 使能对应片选信号的 MRS 请求 |

| | | | | |
|---|---|---|---|---|
| Cs_enable | 3:0 | 读写 | 0x1 | 使能对应片选信号 |
| 0x170 | | | | |
| Odt_wr_cs_map | 63:48 | 读写 | 0x8421 | 对应 CS 发送写命令时，使能的 ODT 信号<br>Bit [15:12]：CS3 发读时对应 ODTx 是否有效，x=3..0<br>Bit [11: 8]：CS2 发读时对应 ODTx 是否有效，x=3..0<br>Bit [ 7: 4]：CS1 发读时对应 ODTx 是否有效，x=3..0<br>Bit [ 3: 0]：CS0 发读时对应 ODTx 是否有效，x=3..0 |
| Odt_wr_length | 43:40 | 读写 | 0x5 | 发送写命令时，ODT 信号有效周期数 |
| Odt_wr_delay | 35:32 | 读写 | 0x0 | 发送写命令时，ODT 信号与写命令的起始间隔 |
| Odt_rd_cs_map | 31:16 | 读写 | 0x4812 | 对应 CS 发送读命令时，使能的 ODT 信号<br>Bit [15:12]：CS3 发读时对应 ODTx 是否有效，x=3..0<br>Bit [11: 8]：CS2 发读时对应 ODTx 是否有效，x=3..0<br>Bit [ 7: 4]：CS1 发读时对应 ODTx 是否有效，x=3..0<br>Bit [ 3: 0]：CS0 发读时对应 ODTx 是否有效，x=3..0 |
| Odt_rd_length | 11:8 | 读写 | 0x5 | 发送读命令时，ODT 信号有效周期数 |
| Odt_rd_delay | 3:0 | 读写 | 0x1 | 发送读命令时，ODT 信号与读命令的起始间隔 |
| 0x178 | | | | |
| | | | | |
| 0x180 | | | | |
| Lvl_resp_0 | 63:56 | 只读 | 0x0 | Leveling 操作时，第 0 数据组的反馈信号 |
| Lvl_done | 48:48 | 只读 | 0x0 | Leveling 操作时，表示 Lvl_resp_*有效信号 |
| Lvl_ready | 40:40 | 只读 | 0x0 | Leveling 操作时，表示当前控制器已经进入 Leveling 操作模式。（用户程序正确设置 Lvl_mode 的值后，应该对这个寄存器进行采样，如果值为 1 表示可以对控制器发起 Leveling 请求，也就是说，此时才可以将设 Lvl_req 为 1） |
| Lvl_cs | 27:24 | 读写 | 0x1 | Leveling 操作时，当前控制的片选信号 |
| tLVL_DELAY | 23:16 | 读写 | 0x10 | Leveling 操作时，有效采样延迟周期<br>单位为时钟周期 |
| Lvl_req | 8:8 | 只写 | 0x0 | Leveling 操作时，向外发起 Leveling 操作请求 |
| Lvl_mode | 1:0 | 读写 | 0x0 | Leveling 模式使能<br>00 – 正常功能模式<br>01 – Write Leveling 模式<br>10 – Gate Leveling 模式 |

| | | | | |
|---|---|---|---|---|
| 0x188 | | | | |
| Lvl_resp_8 | 63:56 | 只读 | 0x0 | Leveling 操作时，第 8 数据组的反馈信号<br>当 Lvl_mode == 1 时，为数据线上的反馈<br>当 Lvl_mode == 2 时<br>Bit[7:5]：内部有效读 DQS 时钟上升沿计数器<br>Bit[4:2]：内部有效读 DQS 时钟下升沿计数器<br>Bit[1:0]：内部 DQS 采样信号采样 DQS 的反馈<br>对于一个正确配置的 Gate Leveling 操作来说<br>Bit[7:5]与 Bit[4:2]应该在每一次 Leveling_req 增加 Burst_length/2 个计数，否则需要调整 Dll_gate_x 的值 |
| Lvl_resp_7 | 55:48 | 只读 | 0x0 | Leveling 操作时，第 7 数据组的反馈信号 |
| Lvl_resp_6 | 47:40 | 只读 | 0x0 | Leveling 操作时，第 6 数据组的反馈信号 |
| Lvl_resp_5 | 39:32 | 只读 | 0x0 | Leveling 操作时，第 5 数据组的反馈信号 |
| Lvl_resp_4 | 31:24 | 只读 | 0x0 | Leveling 操作时，第 4 数据组的反馈信号 |
| Lvl_resp_3 | 23:16 | 只读 | 0x0 | Leveling 操作时，第 3 数据组的反馈信号 |
| Lvl_resp_2 | 15:8 | 只读 | 0x0 | Leveling 操作时，第 2 数据组的反馈信号 |
| Lvl_resp_1 | 7:0 | 只读 | 0x0 | Leveling 操作时，第 1 数据组的反馈信号 |
| 0x190 | | | | |
| Cmd_a | 63:48 | 读写 | 0x0 | 命令发送模式下，对 DRAM 发出的地址线信号 （最高位 bit[15]保留，cmd_a[15]==0) |
| Cmd_ba | 42:40 | 读写 | 0x0 | 命令发送模式下，对 DRAM 发出的 ba 线信号 |
| Cmd_cmd | 34:32 | 读写 | 0x0 | 命令发送模式下，对 DRAM 发出的控制信号<br>bit2 – RASn<br>bit1 – CASn<br>bit0 – WEn |
| Cmd_cs | 27:24 | 读写 | 0x0 | 命令发送模式下，对 DRAM 发出的片选信号 |
| Status_cmd | 16:16 | 只读 | 0x0 | 表示控制器进入命令发送模式，在 command_mode 设置之后才会生效 |
| Cmd_req | 8:8 | 只写 | 0x0 | 命令发送模式下，对 DRAM 发出一次控制命令 |
| Command_mode | 0:0 | 读写 | 0x0 | 使控制器进入命令发送模式 |
| 0x198 | | | | |
| Status_sref | 43:40 | 只读 | 0x0 | 已经进入自刷新模式，每位分别对应一个片选 |
| Srefresh_req | 35:32 | 读写 | 0x0 | 自刷新控制信号，设 1 进入自刷新，设 0 退出自刷新 |
| Pre_all_done | 27:24 | 只读 | 0x0 | Precharge All 操作完成 |

| | | | | |
|---|---|---|---|---|
| Pre_all_req | 19:16 | 只写 | 0x0 | 请求发出 Precharge All 命令，每位分别对应一个片选 |
| Mrs_done | 8:8 | 只读 | 0x0 | 命令模式下，表示 MRS 命令发送完毕 |
| Mrs_req | 0:0 | 只写 | 0x0 | 命令模式下，向 DRAM 发出一次 MRS 命令，发送的命令序列为<br>MRS2、MRS3、MRS1、MRS0 |
| 0x1A0 | | | | |
| Mr_3_cs_0 | 63:48 | 读写 | 0x0000 | 向 DRAM CS 0 发送 MRS 3 命令时对应的值 |
| Mr_2_cs_0 | 47:32 | 读写 | 0x0018 | 向 DRAM CS 0 发送 MRS 2 命令时对应的值 |
| Mr_1_cs_0 | 31:16 | 读写 | 0x0004 | 向 DRAM CS 0 发送 MRS 1 命令时对应的值 |
| Mr_0_cs_0 | 15:0 | 读写 | 0x0d60 | 向 DRAM CS 0 发送 MRS 0 命令时对应的值 |
| 0x1A8 | | | | |
| Mr_3_cs_1 | 63:48 | 读写 | 0x0000 | 向 DRAM CS 1 发送 MRS 3 命令时对应的值 |
| Mr_2_cs_1 | 47:32 | 读写 | 0x0018 | 向 DRAM CS 1 发送 MRS 2 命令时对应的值 |
| Mr_1_cs_1 | 31:16 | 读写 | 0x0004 | 向 DRAM CS 1 发送 MRS 1 命令时对应的值 |
| Mr_0_cs_1 | 15:0 | 读写 | 0x0d60 | 向 DRAM CS 1 发送 MRS 0 命令时对应的值 |
| 0x1B0 | | | | |
| Mr_3_cs_2 | 63:48 | 读写 | 0x0000 | 向 DRAM CS 2 发送 MRS 3 命令时对应的值 |
| Mr_2_cs_2 | 47:32 | 读写 | 0x0018 | 向 DRAM CS 2 发送 MRS 2 命令时对应的值 |
| Mr_1_cs_2 | 31:16 | 读写 | 0x0004 | 向 DRAM CS 2 发送 MRS 1 命令时对应的值 |
| Mr_0_cs_2 | 15:0 | 读写 | 0x0d60 | 向 DRAM CS 2 发送 MRS 0 命令时对应的值 |
| 0x1B8 | | | | |
| Mr_3_cs_3 | 63:48 | 读写 | 0x0000 | 向 DRAM CS 3 发送 MRS 3 命令时对应的值 |
| Mr_2_cs_3 | 47:32 | 读写 | 0x0018 | 向 DRAM CS 3 发送 MRS 2 命令时对应的值 |
| Mr_1_cs_3 | 31:16 | 读写 | 0x0004 | 向 DRAM CS 3 发送 MRS 1 命令时对应的值 |
| Mr_0_cs_3 | 15:0 | 读写 | 0x0D60 | 向 DRAM CS 3 发送 MRS 0 命令时对应的值 |
| 0x1C0 | | | | |
| tRESET | 63:56 | 读写 | 0x28 | DRAM 初始化前的复位时间<br>单位为 4096 个时钟周期 |
| tCKE | 55:48 | 读写 | 0x70 | DRAM 初始化从复位释放到 CKE 有效时间<br>单位为 4096 个时钟周期 |
| tXPR | 47:40 | 读写 | 0x80 | DRAM 初始化从 CKE 有效到 MRS 命令的时间<br>单位为时钟周期 |
| tMOD | 39:32 | 读写 | 0x0C | 发送 MRS 命令后至下一条命令的时间间隔<br>单位为时钟周期 |

| | | | | |
|---|---|---|---|---|
| tZQCL | 31:24 | 读写 | 0x03 | 发送 ZQCL 命令后至下一条命令的时间间隔<br>单位为 256 个时钟周期 |
| tZQ_CMD | 23:16 | 读写 | 0x03 | 不同片选之间发送 ZQ 命令的时间间隔<br>单位为时钟周期 |
| tWLDQSEN | 15:8 | 读写 | 0x20 | Write Leveling 中，从 MRS 到 DQS 为低的时间间隔<br>单位为时钟周期 |
| tRDDATA | 7:0 | 读写 | 0x07 | 从发送读命令到发送读数据有效命令的时间间隔。<br>单位为时钟周期<br>与其他几个参数需要满足关系式：<br>tRDDATA - Cmd_delay - Cmd_timing = CASLAT – 3<br>该参数最小设置值为 2 |
| 0x1C8 | | | | |
| tFAW | 61:56 | 读写 | 0x30 | 连续打开 4 个 Bank 的最小允许时间<br>单位为时钟周期 |
| tRRD | 50:48 | 读写 | 0x06 | 打开两个行之间的最小间隔时间<br>单位为时钟周期 |
| tRCD | 43:40 | 读写 | 0x09 | 打开行到对应行的读写操作之间的最小间隔时间<br>单位为时钟周期 |
| tRP | 39:32 | 读写 | 0x09 | Precharge 操作需要时间<br>单位为时钟周期 |
| tREF | 31:24 | 读写 | 0x03 | 同一片选刷新操作之间的时间间隔<br>单位为 256 个时钟周期 |
| tRFC | 23:16 | 读写 | 0x85 | 刷新操作需要时间<br>单位为时钟周期 |
| tZQCS | 15:8 | 读写 | 0x40 | ZQCS 操作需要时间<br>单位为时钟周期 |
| tZQperiod | 7:0 | 读写 | 0x04 | 同一片选 ZQCS 操作之间的时间间隔<br>单位为 tREF（刷新操作之间的时间间隔） |
| 0x1D0 | | | | |
| tODTL | 59:56 | 读写 | 0x0A | Write Leveling 中从 ODT 无效到 MRS 命令时间间隔<br>单位为时钟周期 |
| tXSRD | 55:48 | 读写 | 0x02 | 从自刷新模式恢复到第一条访问的最小时间间隔<br>单位为 256 个时钟周期 |
| tPHY_RDLAT | 43:40 | 读写 | 0x9 | 读内存操作的 PHY 内部读数据同步时间 |

| | | | | |
|---|---|---|---|---|
| | | | | 0x5 时一般可以正常工作，极端情况下可以将这个值增加或减小。减小可能会影响到读数据的正确性，增加会增加读操作的延迟<br>当 Rdfifo_valid 有效时，这个配置不起作用<br>单位为时钟周期 |
| tPHY_WRLAT | 36:32 | 读写 | 0x04 | 从发送写命令到发送写数据的时间间隔。<br>单位为时钟周期<br>与其他几个参数需要满足关系式：<br>tPHY_WRLAT – Cmd_delay – Cmd_timing = WRLAT – 4<br>该参数最小设置值为 2 |
| tRAS_max | 25:8 | 读写 | 0x20000 | 行打开的最长有效时间<br>单位为时钟周期 |
| tRAS_min | 5:0 | 读写 | 0x1C | 行打开的最短有效时间<br>单位为时钟周期 |
| 0x1D8 | | | | |
| tXPDLL | 63:56 | 读写 | 0x14 | 从离开 Power down （DLL 关闭）状态到下一个命令的间隔时间<br>单位为始终周期 |
| tXP | 55:48 | 读写 | 0x05 | 从离开 Power down （DLL 打开）状态到下一个命令的间隔时间 |
| tWR | 44:40 | 读写 | 0x0C | 写恢复时间<br>单位为时钟周期<br>该参数设置值应大于等于 MRS 中设置值 |
| tRTP | 34:32 | 读写 | 0x06 | 读到 Precharge 操作的间隔时间<br>单位为时钟周期 |
| tRL | 27:24 | 读写 | 0x0a | 读操作延迟，相当于 CASLAT<br>单位为时钟周期<br>该参数设置值应大于等于 MRS 中设置值 |
| tWL | 19:16 | 读写 | 0x08 | 写操作延迟，相当于 WRLAT<br>单位为时钟周期<br>该参数设置值应大于等于 MRS 中设置值 |
| tCCD | 11:8 | 读写 | 0x04 | 两个读写操作之间的最小间隔时间<br>单位为时钟周期 |

| | | | | |
|---|---|---|---|---|
| tWTR | 3:0 | 读写 | 0x06 | 写操作到读操作之间的最小间隔时间<br>单位为时钟周期 |
| 0x1E0 | | | | |
| tW2R_diffCS_dly | 61:56 | 读写 | 0x03 | 不同 CS 上的写操作到读操作之间间隔时间减 1<br>单位为时钟周期，最小值等于 tCCD+tWL-tRL. |
| tW2W_diffCS_adj_dly | 53:48 | 读写 | 0x0 | 不同 CS 上的写操作到写操作之间的附加间隔时间<br>单位为时钟周期，最小值等于 tCCD-1 |
| tR2P_sameBA_adj_dly | 45:40 | 读写 | 0x0 | 相同 Bank 上的读操作到 Precharge 之间附加间隔时间<br>单位为时钟周期 |
| tW2P_sameBA_adj_dly | 37:32 | 读写 | 0x0 | 相同 Bank 上的写操作到 Precharge 之间附加间隔时间<br>单位为时钟周期 |
| tR2R_sameBA_adj_dly | 29:24 | 读写 | 0x0 | 相同 Bank 上的读操作到读操作之间的附加间隔时间<br>单位为时钟周期 |
| tR2W_sameBA_adj_dly | 21:16 | 读写 | 0x0 | 相同 Bank 上的读操作到写操作之间的附加间隔时间<br>单位为时钟周期 |
| tW2R_sameBA_adj_dly | 13:8 | 读写 | 0x0 | 相同 Bank 上的写操作到读操作之间的附加间隔时间<br>单位为时钟周期 |
| tW2W_sameBA_adj_dly | 5:0 | 读写 | 0x0 | 相同 Bank 上的写操作到写操作之间的附加间隔时间<br>单位为时钟周期 |
| 0x1E8 | | | | |
| tR2R_diffCS_adj_dly | 61:56 | 读写 | 0x0 | 不同 CS 上的读操作到读操作之间的附加间隔时间<br>单位为时钟周期，最小值等于 tCCD-1 |
| tR2W_diffCS_dly | 53:48 | 读写 | 0x05 | 不同 CS 上的读操作到写操作之间间隔时间减 1<br>单位为时钟周期，最小值等于<br>tCCD+tRL(+1)-tWL. |
| tR2P_sameCS_dly | 45:40 | 读写 | 0x0 | 相同 CS 上的读操作到 Precharge 之间的间隔时间减去 1 的值<br>单位为时钟周期 |
| tW2P_sameCS_dly | 37:32 | 读写 | 0x0 | 相同 CS 上的写操作到 Precharge 之间的间隔时间减去 1 的值<br>单位为时钟周期 |
| tR2R_sameCS_adj_dly | 29:24 | 读写 | 0x0 | 相同 CS 上的读操作到读操作之间的附加间隔时间<br>单位为时钟周期 |
| tR2W_sameCS_adj | 21:16 | 读写 | 0x0 | 相同 CS 上的读操作到写操作之间的附加间隔时间 |

| | | | | |
|---|---|---|---|---|
| _dly | | | | 单位为时钟周期 |
| tW2R_sameCS_adj _dly | 13:8 | 读写 | 0x0 | 相同 CS 上的写操作到读操作之间的附加间隔时间<br>单位为时钟周期 |
| tW2W_sameCS_adj_dly | 5:0 | 读写 | 0x0 | 相同 CS 上的写操作到写操作之间的附加间隔时间<br>单位为时钟周期 |
| 0x1F0 | | | | |
| Power_up | 59:56 | 读写 | 0x0 | 分别对应四个 CS。设为 1 时，可以使对应的 CS 离开或者不进入 Power down 状态。 |
| Age_step | 55:48 | 读写 | 0x08 | Power down 计数器步长。 |
| tCPDED | 47:40 | 读写 | 0x01 | CKE 为 0 后，命令和地址总线失效时间<br>单位为时钟周期 |
| Cs_map | 39:32 | 读写 | 0x0 | CS 地址映射控制，每两位分别对应地址译码后的 CS 与真实 CS 之间的映射关系 |
| Bs_config | 31:24 | 读写 | 0xFF | 命令调度 CS 状态使能<br>Bit7：CS3 对应状态机状态使能<br>1-使能；0-禁用（下同）<br>Bit6：CS3 对应状态机状态重置<br>1-解复位；0-重置（下同）<br>Bit5：CS2 对应状态机状态使能<br>Bit4：CS2 对应状态机状态重置<br>Bit3：CS1 对应状态机状态使能<br>Bit2：CS1 对应状态机状态重置<br>Bit1：CS0 对应状态机状态使能<br>Bit0：CS0 对应状态机状态重置 |
| Nc | 18:16 | 读写 | 0x0 | 多通道模式使能<br>000 – 普通 64 位模式<br>001 – 多通道模式<br>011 – 普通 16 位模式<br>101 – 普通 32 位模式<br>其它 – 保留 |
| Pr_r2w | 11:8 | 读写 | 0x1 | 读操作优先级是否高于写操作 |
| Placement_en | 0:0 | 读写 | 0x1 | 使能读写命令重排逻辑 |
| 0x1F8 | | | | |
| Hw_pd_3 | 59:56 | 读写 | 0x0 | 从低到高分别对应 Active Standby，Fast Power Down， |

| | | | | |
|---|---|---|---|---|
| | | | | Slow Power Down 和 Self Refresh。设为 1 表示允许 CS3 进入对应的低功耗状态。 |
| Hw_pd_2 | 51:48 | 读写 | 0x0 | 设为 1 表示允许 CS2 进入对应的低功耗状态。 |
| Hw_pd_1 | 43:40 | 读写 | 0x0 | 设为 1 表示允许 CS1 进入对应的低功耗状态。 |
| Hw_pd_0 | 35:32 | 读写 | 0x0 | 设为 1 表示允许 CS0 进入对应的低功耗状态。 |
| Credit_16 | 29:24 | 读写 | 0x4 | 16 位通道优先级设置 |
| Credit_32 | 21:16 | 读写 | 0x8 | 32 位通道优先级设置 |
| Credit_64 | 13:8 | 读写 | 0x10 | 64 位通道优先级设置 |
| Selection_en | 0:0 | 读写 | 0x1 | 不同通道优先级调度使能 |
| 0x200 | | | | |
| Cmdq_age_16 | 59:48 | 读写 | 0xC00 | 16 位通道调度初始年龄。0xFFF 为超时。 |
| Cmdq_age_32 | 43:32 | 读写 | 0xC00 | 32 位通道调度初始年龄。0xFFF 为超时。 |
| Cmdq_age_64 | 27:16 | 读写 | 0xC00 | 64 位通道调度初始年龄。0xFFF 为超时。 |
| tCKESR | 15:8 | 读写 | 0x07 | 进入自刷新时，CKE 为低的最短时间<br>单位为时钟周期 |
| tRDPDEN | 7:0 | 读写 | 0x0C | 从发出 RD/RDA 命令到进入低功耗状态的时间间隔<br>单位为时钟周期 |
| 0x208 | | | | |
| Wfifo_age | 59:48 | 读写 | 0xC00 | 写队列中命令初始年龄。0xFFF 为超时。 |
| Rfifo_age | 43:32 | 读写 | 0xC00 | 读队列中命令初始年龄。0xFFF 为超时。 |
| Power_stat3 | 27:24 | 只读 | 0x0 | 从低到高分别对应 Active Standby，Fast Power Down，Slow Power Down 和 Self Refresh。<br>设为 1 表示 CS3 处于对应的低功耗状态。 |
| Power_stat2 | 19:16 | 只读 | 0x0 | 设为 1 表示 CS2 处于对应的低功耗状态。 |
| Power_stat1 | 11:8 | 只读 | 0x0 | 设为 1 表示 CS1 处于对应的低功耗状态。 |
| Power_stat0 | 3:0 | 只读 | 0x0 | 设为 1 表示 CS0 处于对应的低功耗状态。 |
| 0x210 | | | | |
| Active_age | 63:48 | 读写 | 0x0008 | Active Standby 低功耗状态计数器 |
| Cs_place_0 | 40:40 | 读写 | 0x0 | 普通模式或窗口 0 译码时 CS 在地址中的位置<br>0 – 译码方式为{CS、ROW、BA、COL}<br>1 – 译码方式为{ROW、CS、BA、COL} |
| Addr_win_0 | 35:32 | 读写 | 0xF | 普通模式或窗口 0 地址命中及配置<br>Bit [3:2]：窗口使用 DRAM 的 Bank 数<br>11 – 8 bank；10 – 4 bank；01 – 2 bank；00 – 保留 |

| | | | | |
|---|---|---|---|---|
| | | | | Bit [1:0]：窗口使用 DRAM 位数<br>11 – 64 位；10 – 32 位；01 – 16 位；00 – 保留 |
| Cs_diff_0 | 27:24 | 读写 | 0x0 | 普通模式或窗口 0 实际使用的 CS 译码前地址与 2 之差<br>当 Pm_nc 有效时，<br>对于 64 位窗口，应该为 2；<br>对于 32 位窗口，应该为 1；<br>对于 16 位窗口，应该为 0 |
| Row_diff_0 | 19:16 | 读写 | 0x2 | 普通模式或窗口 0 实际使用的行地址线个数与 16 之差<br>这个值等于 16 – 实际使用的行地址线个数 |
| Ba_diff_0 | 9:8 | 读写 | 0x0 | 普通模式或窗口 0 实际使用的 BA 线个数与 3 之差<br>这个值等于 3 – 实际使用的 BA 线个数 |
| Col_diff_0 | 3:0 | 读写 | 0x6 | 普通模式或窗口 0 实际使用的列地址线个数与 16 之差<br>这个值等于 16 – 实际使用的列地址线个数 |
| 0x218 | | | | |
| Fastpd_age | 63:48 | 读写 | 0x0008 | Fast Powerdown 低功耗状态计数器 |
| Cs_place_1 | 40:40 | 读写 | 0x0 | 普通模式或窗口 1 译码时 CS 在地址中的位置<br>0 – 译码方式为{CS、ROW、BA、COL}<br>1 – 译码方式为{ROW、CS、BA、COL} |
| Addr_win_1 | 35:32 | 读写 | 0xF | 普通模式或窗口 1 地址命中及配置<br>Bit [3:2]：窗口使用 DRAM 的 Bank 数<br>11 – 8 bank；10 – 4 bank；01 – 2 bank；00 – 保留<br>Bit [1:0]：窗口使用 DRAM 位数<br>11 – 64 位；10 – 32 位；01 – 16 位；00 – 保留 |
| Cs_diff_1 | 27:24 | 读写 | 0x0 | 普通模式或窗口 1 实际使用 CS 译码前地址与 2 之差<br>当 Pm_nc 有效时，<br>对于 64 位窗口，应该为 2；<br>对于 32 位窗口，应该为 1；<br>对于 16 位窗口，应该为 0 |
| Row_diff_1 | 19:16 | 读写 | 0x2 | 普通模式或窗口 1 实际使用行地址线个数与 16 之差<br>这个值等于 16 – 实际使用的行地址线个数 |
| Ba_diff_1 | 9:8 | 读写 | 0x0 | 普通模式或窗口 1 实际使用的 BA 线个数与 3 之差 |

| | | | | |
|---|---|---|---|---|
| | | | | 这个值等于 3 – 实际使用的 BA 线个数 |
| Col_diff_1 | 3:0 | 读写 | 0x6 | 普通模式或窗口 1 实际使用列地址线个数与 16 之差<br>这个值等于 16 – 实际使用的列地址线个数 |
| 0x220 | | | | |
| Slowpd_age | 63:48 | 读写 | 0x0008 | Slow Powerdown 低功耗状态计数器 |
| Cs_place_2 | 40:40 | 读写 | 0x0 | 普通模式或窗口 2 译码时 CS 在地址中的位置<br>0 – 译码方式为{CS、ROW、BA、COL}<br>1 – 译码方式为{ROW、CS、BA、COL} |
| Addr_win_2 | 35:32 | 读写 | 0xF | 普通模式或窗口 2 地址命中及配置<br>Bit [3:2]：窗口使用 DRAM 的 Bank 数<br>11 – 8 bank；10 – 4 bank；01 – 2 bank；00 – 保留<br>Bit [1:0]：窗口使用 DRAM 位数<br>11 – 64 位；10 – 32 位；01 – 16 位；00 – 保留 |
| Cs_diff_2 | 27:24 | 读写 | 0x0 | 普通模式或窗口 2 实际使用 CS 译码前地址与 2 之差<br>当 Pm_nc 有效时，<br>对于 64 位窗口，应该为 2；<br>对于 32 位窗口，应该为 1；<br>对于 16 位窗口，应该为 0 |
| Row_diff_2 | 19:16 | 读写 | 0x2 | 普通模式或窗口 2 实际使用行地址线个数与 16 之差<br>这个值等于 16 – 实际使用的行地址线个数 |
| Ba_diff_2 | 9:8 | 读写 | 0x0 | 普通模式或窗口 2 实际使用的 BA 线个数与 3 之差<br>这个值等于 3 – 实际使用的 BA 线个数 |
| Col_diff_2 | 3:0 | 读写 | 0x6 | 普通模式或窗口 2 实际使用列地址线个数与 16 之差<br>这个值等于 16 – 实际使用的列地址线个数 |
| 0x228 | | | | |
| Selfref_age | 63:48 | 读写 | 0x0008 | Selfrefresh 低功耗状态计数器 |
| Cs_place_3 | 40:40 | 读写 | 0x0 | 普通模式或窗口 3 译码时 CS 在地址中的位置<br>0 – 译码方式为{CS、ROW、BA、COL}<br>1 – 译码方式为{ROW、CS、BA、COL} |
| Addr_win_3 | 35:32 | 读写 | 0xF | 普通模式或窗口 3 地址命中及配置<br>Bit [3:2]：窗口使用 DRAM 的 Bank 数<br>11 – 8 bank；10 – 4 bank；01 – 2 bank；00 – 保留<br>Bit [1:0]：窗口使用 DRAM 位数<br>11 – 64 位；10 – 32 位；01 – 16 位；00 – 保留 |

| | | | | |
|---|---|---|---|---|
| Cs_diff_3 | 27:24 | 读写 | 0x0 | 普通模式或窗口 3 实际使用 CS 译码前地址与 2 之差<br>当 Pm_nc 有效时，<br>对于 64 位窗口，应该为 2；<br>对于 32 位窗口，应该为 1；<br>对于 16 位窗口，应该为 0 |
| Row_diff_3 | 19:16 | 读写 | 0x2 | 普通模式或窗口 3 实际使用行地址线个数与 16 之差<br>这个值等于 16 – 实际使用的行地址线个数 |
| Ba_diff_3 | 9:8 | 读写 | 0x0 | 普通模式或窗口 3 实际使用的 BA 线个数与 3 之差<br>这个值等于 3 – 实际使用的 BA 线个数 |
| Col_diff_3 | 3:0 | 读写 | 0x6 | 普通模式或窗口 3 实际使用列地址线个数与 16 之差<br>这个值等于 16 – 实际使用的列地址线个数 |
| 0x230 | | | | |
| Win_mask_0 | 59:32 | 读写 | 0xFFFFF00 | 0 号窗口 MASK，对应地址[47:20] |
| Win_base_0 | 27:0 | 读写 | 0x0000000 | 0 号窗口 BASE，对应地址[47:20] |
| 0x238 | | | | |
| Win_mask_1 | 59:32 | 读写 | 0xFFFFF00 | 1 号窗口 MASK，对应地址[47:20] |
| Win_base_1 | 27:0 | 读写 | 0x0000100 | 1 号窗口 BASE，对应地址[47:20] |
| 0x240 | | | | |
| Win_mask_2 | 59:32 | 读写 | 0xFFFFF00 | 2 号窗口 MASK，对应地址[47:20] |
| Win_base_2 | 27:0 | 读写 | 0x0000200 | 2 号窗口 BASE，对应地址[47:20] |
| 0x248 | | | | |
| Win_mask_3 | 59:32 | 读写 | 0xFFFFF00 | 3 号窗口 MASK，对应地址[47:20] |
| Win_base_3 | 27:0 | 读写 | 0x0000300 | 3 号窗口 BASE，对应地址[47:20] |
| 0x250 | | | | |
| Cmd_monitor | 55:48 | 读写 | 0x0 | Bit 7：使能命令队列 3 监控功能<br>Bit 6：复位命令队列 3 性能计数值 |

| | | | | |
|---|---|---|---|---|
| | | | | Bit 5：使能命令队列 2 监控功能<br>Bit 4：复位命令队列 2 性能计数值<br>Bit 3：使能命令队列 1 监控功能<br>Bit 2：复位命令队列 1 性能计数值<br>Bit 1：使能命令队列 0 监控功能<br>Bit 0：复位命令队列 0 性能计数值 |
| Axi_monitor | 41:32 | 读写 | 0x0 | 使能 AXI 命令队列性能监控，每两位控制一个 AXI 监控模块，控制方法与 Cmd_monitor 相同 |
| Ecc_code | 31:24 | 只读 | 0x0 | 第一次发生 ECC 错误时从内存读出的校验码<br>记录的出错信息的时机由 Int_vector[0] 或 Int_vector[1]由 0 变 1 时触发，使用 Ecc_enable[3]进行配置 |
| Ecc_enable | 18:16 | 读写 | 0x0 | ECC 功能使能<br>Bit-3：设置保存 ECC 出错信号时机<br>0 - 出现 ECC 错时触发；1 – 出现两位错时触发<br>Bit-2：使能写时 ECC 校验错的内部总线报错（异常）<br>Bit-1：使能读时 ECC 校验错的内部总线报错（异常）<br>Bit-0：使能 ECC 功能（只在 64 位模式下有效） |
| Int_vector | 9:8 | 读写 | 0x0 | 中断向量寄存器<br>Bit-1：ECC 两位校验错<br>Bit-0：ECC 校验错（包括一位错与两位错）<br>对这个寄存器的读操作将得到当前的ECC出错情况，对这个寄存器的“写 1”操作将清除对应的位 |
| Int_enable | 1:0 | 读写 | 0x0 | 中断使能寄存器<br>Bit-1：ECC 两位较验错中断使能<br>Bit-0：ECC 较验错中断使能（包括一位错与两位错） |
| 0x258 | | | | |
| | | | | |
| 0x260 | | | | |
| Ecc_addr | 63:0 | 只读 | 0x0 | 第一次发生 ECC 错误时向内存读的出错地址<br>记录的出错信息的时机由 Int_vector[0] 或 Int_vector[1]由 0 变 1 时触发，使用 Ecc_enable[3]进行配置 |
| 0x268 | | | | |

| | | | | |
|---|---|---|---|---|
| Ecc_data | 63:0 | 只读 | 0x0 | 第一次发生 ECC 错误时从内存读出的数据<br>记录的出错信息的时机由 Int_vector[0] 或 Int_vector[1]由 0 变 1 时触发，使用 Ecc_enable[3]进行配置 |
| 0x270 | | | | |
| Lpbk_ecc_mask | 58:57 | 只读 | 0x0 | 自循环测试第一次出错时的 ECC MASK 值<br>Bit 1：对应于 ECC MASK 的上升沿数据<br>Bit 0：对应于 ECC MASK 的下降沿数据 |
| Prbs_init | 54:32 | 只写 | 0x10 | 自循环测试时使用的 PRBS 初始值 |
| Lpbk_error | 25:25 | 只读 | 0x0 | 自循环测试出错 |
| Prbs_23 | 16:16 | 读写 | 0x0 | 自循环测试时使用的编码方式<br>1 – PRBS 23<br>0 – PRBS 7 |
| Lpbk_start | 8:8 | 读写 | 0x0 | 自循环测试开始 |
| Lpbk_en | 0:0 | 读写 | 0x0 | 自循环测试模式使能 |
| 0x278 | | | | |
| Lpbk_ecc | 63:49 | 只读 | 0x0 | 自循环测试第一次出错时的 ECC 值<br>Bit [63:57]：对应于 ECC 的上升沿数据的低 15 位<br>Bit [56:49]：对应于 ECC 的下降沿数据 |
| Lpbk_data_mask | 48:33 | 只读 | 0x0 | 自循环测试第一次出错时的 DQM 值<br>Bit [48:41]：对应于 DQM 的上升沿数据<br>Bit [40:33]：对应于 DQM 的下降沿数据 |
| Lpbk_correct | 32:17 | 只读 | 0x0 | 自循环测试第一次出错时的 PRBS 编码<br>Bit [32:25]：对应于上升沿数据<br>Bit [24:17]：对应于下降沿数据 |
| Lpbk_counter | 16:1 | 只读 | 0x0 | 自循环测试第一次出错时的计数周期 |
| 0x280 | | | | |
| Lpbk_data_r | 63:0 | 只读 | 0x0 | 自循环测试第一次出错时的 DQ 上升沿数据 |
| 0x288 | | | | |
| Lpbk_data_f | 63:0 | 只读 | 0x0 | 自循环测试第一次出错时的 DQ 下降沿数据 |
| 0x290 | | | | |
| Axi0_bw_w | 63:32 | 只读 | 0x0 | AXI0 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Axi0_bw_r | 31:0 | 只读 | 0x0 | AXI0 读带宽性能计数值 |

| | | | | |
|---|---|---|---|---|
| | | | | 表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x298 | | | | |
| Axi0_latency_w | 63:32 | 只读 | 0x0 | AXI0 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Axi0_latency_r | 31:0 | 只读 | 0x0 | AXI0 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x2A0 | | | | |
| Axi1_bw_w | 63:32 | 只读 | 0x0 | AXI1 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Axi1_bw_r | 31:0 | 只读 | 0x0 | AXI1 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x2A8 | | | | |
| Axi1_latency_w | 63:32 | 只读 | 0x0 | AXI1 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Axi1_latency_r | 31:0 | 只读 | 0x0 | AXI1 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x2B0 | | | | |
| Axi2_bw_w | 63:32 | 只读 | 0x0 | AXI2 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Axi2_bw_r | 31:0 | 只读 | 0x0 | AXI2 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x2B8 | | | | |
| Axi2_latency_w | 63:32 | 只读 | 0x0 | AXI2 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Axi2_latency_r | 31:0 | 只读 | 0x0 | AXI2 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x2C0 | | | | |
| Axi3_bw_w | 63:32 | 只读 | 0x0 | AXI3 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Axi3_bw_r | 31:0 | 只读 | 0x0 | AXI3 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x2C8 | | | | |
| Axi3_latency_w | 63:32 | 只读 | 0x0 | AXI3 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |

| | | | | |
|---|---|---|---|---|
| Axi3_latency_r | 31:0 | 只读 | 0x0 | AXI3 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x2D0 | | | | |
| Axi4_bw_w | 63:32 | 只读 | 0x0 | AXI4 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Axi4_bw_r | 31:0 | 只读 | 0x0 | AXI4 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x2D8 | | | | |
| Axi4_latency_w | 63:32 | 只读 | 0x0 | AXI4 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Axi4_latency_r | 31:0 | 只读 | 0x0 | AXI4 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x2E0 | | | | |
| Cmdq0_bw_w | 63:32 | 只读 | 0x0 | 命令队列 0 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Cmdq0_bw_r | 31:0 | 只读 | 0x0 | 命令队列 0 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x2E8 | | | | |
| Cmdq0_latency_w | 63:32 | 只读 | 0x0 | 命令队列 0 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Cmdq0_latency_r | 31:0 | 只读 | 0x0 | 命令队列 0 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x2f0 | | | | |
| Cmdq1_bw_w | 63:32 | 只读 | 0x0 | 命令队列 1 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Cmdq1_bw_r | 31:0 | 只读 | 0x0 | 命令队列 1 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x2f8 | | | | |
| Cmdq1_latency_w | 63:32 | 只读 | 0x0 | 命令队列 1 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Cmdq1_latency_r | 31:0 | 只读 | 0x0 | 命令队列 1 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x300 | | | | |
| Cmdq2_bw_w | 63:32 | 只读 | 0x0 | 命令队列 2 写带宽性能计数值 |

| | | | | |
|---|---|---|---|---|
| | | | | 表示 64K 个时钟周期里总线数据有效的周期数 |
| Cmdq2_bw_r | 31:0 | 只读 | 0x0 | 命令队列 2 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x308 | | | | |
| Cmdq2_latency_w | 63:32 | 只读 | 0x0 | 命令队列 2 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Cmdq2_latency_r | 31:0 | 只读 | 0x0 | 命令队列 2 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| 0x310 | | | | |
| Cmdq3_bw_w | 63:32 | 只读 | 0x0 | 命令队列 3 写带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| Cmdq3_bw_r | 31:0 | 只读 | 0x0 | 命令队列 3 读带宽性能计数值<br>表示 64K 个时钟周期里总线数据有效的周期数 |
| 0x318 | | | | |
| Cmdq3_latency_w | 63:32 | 只读 | 0x0 | 命令队列 3 写延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |
| Cmdq3_latency_r | 31:0 | 只读 | 0x0 | 命令队列 3 读延迟性能计数值<br>这个值表示 64K 个访问的总延迟周期之和 |

## 9.2 DDR2/3 SDRAM 软件编程指南

### 9.2.1 初始化操作

初始化操作由软件向寄存器 Init_start（0x018）写入 1 时开始，在设置 Init_start 信号之前，必须将其它所有寄存器设置为正确的值。

软硬件协同的 DRAM 初始化过程如下：

(1) 软件向所有的寄存器写入正确的配置值，但是 Init_start（0x018）在这一过程中必须保持为 0；

(2) 软件将 Init_start（0x018）设置为 1，这将导致硬件初始化的开始；

(3) PHY 内部开始初始化操作，DLL 将尝试进行锁定操作。如果锁定成功，则可以从 Dll_init_done（0x000）读出对应状态，并可以从 Dll_value_ck（0x000）读写当前锁定延迟线个数；如果锁定不成功，则初始化不会继续进行（此时可以通过设置 Dll_bypass（0x018）使得初始化继续执行）；

(4) DLL 锁定（或者 bypass 设置）之后，控制器将根据对应 DRAM 的初始化要求向

DRAM 发出相应的初始化序列，例如对应的 MRS 命令，ZQCL 命令等等；

(5) 软件可以通过采样 Dram_init（0x160）寄存器来判断内存初始化操作是否完成。

### 9.2.2 Leveling

Leveling 操作是在 DDR3 中，用于智能配置内存控制器读写操作中各种信号间相位关系的操作。通常它包括了 Write Leveling、Read Leveling 和 Gate Leveling。在本控制器中，只实现了 Write Leveling 与 Gate Leveling，Read Leveling 没有实现，软件需要通过判断读写的正确性来实现 Read Leveling 所完成的功能。除了在 Leveling 过程中操作的 DQS 相位、GATE 相位之外，还可以根据这些最后确认的相位来计算出写 DQ 相位、读 DQ 相位的配置方法。

**Write Leveling**

Write Leveling 用于配置写 DQS 与时钟之间的相位关系，软件编程需要参照如下步骤。

(1) 完成控制器初始化，参见上一小节内容；

(2) 将 Dll_wrdqs_x（x = 0…8）设置为 0；

(3) 设置 Lvl_mode（0x180）为 2'b01；

(4) 采样 Lvl_ready（0x180）寄存器，如果为 1，表示可以开始 Write Leveling 请求；

(5) 设置 Lvl_req（0x180）为 1；

(6) 采样 Lvl_done（0x180）寄存器，如果为 1，表示一次 Write Leveling 请求完成；

(7) 采样 Lvl_resp_x（0x180、0x188）寄存器，如果为 0 则跳至步骤(8)。否则(采样 Lvl_resp_x 结果为 1)，将对应的 Dll_wrdqs_x[6:0]增加 1，并重复执行 5-7 直至采样 Lvl_resp_x 结果为 0；

(8) 采样 Lvl_resp_x（0x180、0x188）寄存器，如果为 0，则将对应的 Dll_wrdqs_x[6:0]增加 1，并重复执行步骤 5，6，8 直至采样 Lvl_resp_x 结果为 1；

(9) 此时 Dll_wrdqs_x 的值就应该是正确的设置值。

至此 Write Leveling 操作结束。

(10) 接着根据 Dll_wrdqs_x 的值是否小于 0x40 来设置 Wrdqs_lt_half_x；

(11) 根据 Dll_wrdqs_x 的值是否小于 0x20 来设置 Dll_wrdata_x。如果 Dll_wrdqs_x > 0x20，Dll_wrdata_x = Dll_wrdqs_x – 0x20，否则 Dll_wrdata_x = Dll_wrdqs_x + 0x60；

(12) 根据 Dll_wrdata_x 的值是否小于 0x40 来设置 Wrdata_lt_half_x；

(13) 接下来需要根据 DIMM 类型进行不同的操作

对于 UDIMM，Slice0-7 的 Dll_wrdata 的值依次增大(当跨越 0x7F 边界时也视为依次增大)。如果所有 Wrdq_lt_half_x 都为 1，即所有 Dll_wrdata_x 都小于 0x40，则将 tPHY_WRDATA 与 tRDDATA 的值减 1；如果有的 Wrdq_lt_half_x 为 1，有的 Wrdq_lt_half_x 为 0，即 Dll_wrdata_x 的值有跨越 0x40 边界的情况，则将 0x40 边界右边(不一定是大于 0x40，因为有可能有的 Dll_wrdata_x 的值跨越 0x7F 边界)的 Slice 对应的 Wrdq_clkdelay_x 设为 1，然后

将 tPHY_WRDATA 与 tRDDATA 的值减 1；如果所有 Wrdq_lt_half_x 都为 0，即所有 Dll_wrdata_x 都大于等于 0x40，不做任何处理。

对于 RDIMM，tPHY_WRDATA 与 tRDDATA 的默认配置值需要在 UDIMM 的基础上增加 1。然后分别对于 Slice8,3,2,1,0 和 Slice4,5,6,7 作同 UDIMM 的处理。

(14)将 Lvl_mode（0x180）设置为 2'b00，退出 Write Leveling 模式；

**Gate Leveling**

Gate Leveling 用于配置控制器内使能采样读 DQS 窗口的时机，软件编程参照如下步骤。

(1) 完成控制器初始化，参见上一小节内容；

(2) 完成 Write Leveling，参见上一小节内容；

(3) 将 Dll_gate_x（x = 0…8）设置为 0；

(4) 设置 Lvl_mode（0x180）为 2'b10；

(5) 采样 Lvl_ready（0x180）寄存器，如果为 1，表示可以开始 Gate Leveling 请求；

(6) 设置 Lvl_req（0x180）为 1；

(7) 采样 Lvl_done（0x180）寄存器，如果为 1，表示一次 Gate Leveling 请求完成；

(8) 采样 Lvl_resp_x[0]（0x180、0x188）寄存器。如果第一次采样发现 Lvl_resp_x[0]为 1，则将对应的 Dll_gate_x[6:0]增加 1，并重复执行 6-8，直至采样结果为 0，否则进行下一步。在这个过程中如果 Dll_gate_x[6:0]的值增加到 0x7F 还没采样到 Lvl_resp_x[0]为 0，则将对应的 Rd_oe_begin_x 和 Rd_oe_end_x 增加 1；

(9) 采样 Lvl_resp_x[0]（0x180、0x188）寄存器，如果采样结果为 0，则将对应的 Dll_gate_x[6:0]增加 1，并重复执行 6、7、9，直至采样结果 1，则表示 Gate Leveling 操作已经成功；

至此 Gate Leveling 操作结束，此时 Dll_gate_x[6:0]与 Dll_wrdata_x[6:0]的和实际上就是读 DQS 相对于 PHY 内部时钟的相位关系。下面根据 Leveling 的结果对各个参数进行调整。

(10)如果 Dll_gate_x[6:0]与 Dll_wrdata_x[6:0]的和小于 0x20 或者大于 0x60，那么 Dll_rddqs_lt_halt 设置为 1。因为 rddqs 的相位关系实际上等于在输入的读 DQS 基础上再延迟 1/4。

(11)此时如果 Dll_gate_x 的值大于 0x20，则将 Dll_gate_x 的值减去 0x20；否则将其设为 0 即可。

(12)调整完毕后，再分别进行两次 Lvl_req 操作，观察 Lvl_resp_x[7:5]与 Lvl_resp_x[4:2]的值变化，如果各增加为 Burst_length/2，则继续进行第 13 步操作；如果不为 Burst_length/2，可能需要对 Rd_oe_begin_x 进行加一或减一操作，如果大于 Burst_length/2，很可能需要对 Dll_gate_x 的值进行一些微调

(13)将 Lvl_mode（0x180）设置为 2'b00，退出 Gate Leveling 模式；

### 9.2.3 单独发起 MRS 命令

内存控制器向内存发出的 MRS 命令次序分别为：

MR2_CS0、MR2_CS1、MR2_CS2、MR2_CS3、

MR3_CS0、MR3_CS1、MR3_CS2、MR3_CS3、

MR1_CS0、MR1_CS1、MR1_CS2、MR1_CS3、

MR0_CS0、MR1_CS1、MR1_CS2、MR1_CS3。

其中，对应 CS 的 MRS 命令是否有效，是由 Cs_mrs 决定，只有 Cs_mrs 上对应每个片选的位有效，才会真正向 DRAM 发出这个 MRS 命令。对应的每个 MR 的值由寄存器 Mr*_cs* 决定。这些值同时也用于初始化内存时的 MRS 命令。

具体操作如下：

(1) 将寄存器 Cs_mrs（0x168）、Mr*_cs*（0x190 – 0x1B8）设置为正确的值；

(2) 设置 Command_mode（0x190）为 1，使控制器进入命令发送模式；

(3) 采样 Status_cmd（0x190），如果为 1，则表示控制器已进入命令发送模式，可以进行下一步操作，如果为 0，则需要继续等待；

(4) 写 Mrs_req（0x198）为 1，向 DRAM 发送 MRS 命令；

(5) 采样 Mrs_done（0x198），如果为 1，则表示 MRS 命令已经发送完毕，可以退出，如果为 0，则需要继续等待；

(6) 设置 Command_mode（0x190）为 0，使控制器退出命令发送模式。

### 9.2.4 任意操作控制总线

内存控制器可以通过命令发送模式向 DRAM 发出任意的命令组合，软件可以设置 Cmd_cs、Cmd_cmd、Cmd_ba、Cmd_a（0x190），在命令发送模式下向 DRAM 发出。

具体操作如下：

(1) 将寄存器 Cmd_cs、Cmd_cmd、Cmd_ba、Cmd_a（0x190）设置为正确的值；

(2) 设置 Command_mode（0x190）为 1，使控制器进入命令发送模式；

(3) 采样 Status_cmd（0x190），如果为 1，则表示控制器已进入命令发送模式，可以进行下一步操作，如果为 0，则需要继续等待；

(4) 写 Cmd_req（0x190）为 1，向 DRAM 发送命令；

(5) 设置 Command_mode（0x190）为 0，使控制器退出命令发送模式。

### 9.2.5 自循环测试模式控制

自循环测试模式可以分别在测试模式下或者正常功能模式下使用，为此，本内存控制器分别实现了两套独立的控制接口，一套用于在测试模式下由测试端口直接控制，另一套用于

在正常功能模式下由寄存器配置模块进行配置使能测试。

这两套接口的复用使用端口 test_phy 进行控制，当 test_phy 有效时，使用控制器的 test_* 端口进行控制，此时的自测试完全由硬件控制；当 test_phy 无效时，使用软件编程的 pm_* 的参数进行控制。使用测试端口的具体信号含义可以参考寄存器参数中的同名部分。

这两套接口从控制的参数来说基本一致，仅仅是接入点不同，在此介绍软件编程时的控制方法。具体操作如下：

(1) 将内存控制器所有的参数全部正确设置。需要注意的是，Dqs_oe_begin_*、Dqs_oe_end_* 应该设为 0，Rd_oe_begin_*、Rd_oe_end_* 应该设为 1 或 2，否则可能会导致测试出错；

(2) 将寄存器 Lpbk_en（0x270）设为 1；

(3) 将寄存器 Init_start（0x018）设为 1；

(4) 采样寄存器 Dll_init_done（0x000），如果这个值为 1，表示 DLL 已经锁定，可以进行下一步操作；如果这个值为 0，则需要继续等待；（当使用测试端口进行控制的时候，因为看不到这个寄存器的输出，所以不需要采样这个寄存器，而只需要在此处等待一定的时间，以确保 DLL 锁定完成，再进行下一步操作）；

(5) 将寄存器 Lpbk_start（0x270）设为 1；此时自循环测试正式开始。

到此为止自循环测试已经开始，软件需要经常检测是否有错误发生，具体操作如下：

(6) 采样寄存器 Lpbk_error（0x270），如果这个值为 1，表示有错误发生，此时可以通过 Lpbk_*等观测用寄存器（0x270、0x278、0x280、0x288）来观测第一个出错时的错误数据和正确数据；如果这个值为 0，表示还没有出现过数据错误。

### 9.2.6 细粒度多通道模式控制

**硬件连接**

需要使用细粒度多通道模式，硬件连接时每个 CS/ODT 连接 16 位 DQ，也即 CS0/ODT0 对应 DQ[15:0]，CS1/ODT1 对应 DQ[31:16]，CS2/ODT2 对应 DQ[47:32]，CS3/ODT3 对应 DQ[63:48]。地址线与其它控制线连接所有内存颗粒。

**软件编程**

进行软件初始化时，与普通模式时不同的是，需要打开所有的 Cs_enable、Cs_mrs、Lvl_cs，以便对所有的颗粒进行初始化及 Leveling 操作。

需要将 Rdfifo_valid 设为 0，并正确设置 tPHY_RDLAT 的值。

接下来按照需要设置四个 Addr_win/Cs_diff/Row_diff/Ba_diff/Col_diff 的值。

四组不同的窗口每组表示总地址空间的 1/4。需要进行分别配置。

### 9.2.7 ECC 功能使用控制

ECC 功能只有在 64 位模式下可以使用。

Ecc_enable 包括以下 4 个控制位：

Ecc_enable[0]控制是否使能 ECC 功能，只有设置了这个有效位，才会使能 ECC 功能。

Ecc_enable[1]控制是否通过处理器内部的读响应通路进行报错，以使得出现 ECC 两位错的读访问能立即导致处理器核的异常发生。

Ecc_enable[2]控制是否通过处理器内部的写响应通路进行报错，以使得出现 ECC 两位错的写访问（读后写）能立即导致处理器核的异常发生。

Ecc_enable[3]控制寄存器内记录出错信息的触发时机。这些出错信息在没有软件进行处理的情况下不会连续触发，只会记录第一次出错时的信息。这些信息包括 Ecc_code，Ecc_addr，Ecc_data。当 Ecc_enable[3]为 0 的情况下，只要出现了 ECC 错误（包括 1 位错与 2 位错），这个记录就会被触发，当 Ecc_enable[3]为 1 的情况下，只有出现了 ECC 两位错，这个记录才会被触发。而这个“第一次”指的是中断向量寄存器的对应位被置位。也就是说，记录的是导致中断发生的那一次访问。

除此之外，ECC 出错还可以通过中断方式通知处理器核。这个中断通过 Int_enable 进行控制。中断包括两个向量，Int_vector[0]表示出现 ECC 错误（包括 1 位错与 2 位错），Int_vecotr[1]表示出现 ECC 两位错。Int_vector 的清除通过向对应位写 1 实现。

### 9.2.8 32/16 位窄通道使用控制

除了 64 位模式，通过配置还可以将内存控制器置于 32 位或者 16 位工作模式，具体的配置如下：

对于 32 位模式：

(1) 将寄存器 Nc（0x1F0）设为 0x5；

(2) 将寄存器 Addr_win_*（0x210，0x218，0x220，0x228）设为 0xe；

(3) 将寄存器 Cs_diff_*（0x210，0x218，0x220，0x228）设为 0x1。

对于 16 位模式：

(1) 将寄存器 Nc（0x1F0）设为 0x3；

(2) 将寄存器 Addr_win_*（0x210，0x218，0x220，0x228）设为 0xd；

(3) 将寄存器 Cs_diff_*（0x210，0x218，0x220，0x228）设为 0x0。

# 10 HyperTransport 控制器

龙芯 3B1500 中，HyperTransport 总线用于实现外部设备连接以及多芯片互联。用于外设连接时，可由用户程序自由选择是否支持 IO Cache 一致性（通过地址窗口 Uncache 进行设置，详见 10.5.13 节）：当配置为支持 Cache 一致性模式时，IO 设备对内 DMA 的访问对于 Cache 层次透明，即由硬件自动维护其一致性,而无需软件通过程序 Cache 指令进行维护；当 HyperTransport 总线用于多芯片互联时，HT0 控制器(初始地址为 0x0C00_0000_0000 – 0x0DFF_FFFF_FFFF)可通过硬件自动维护各个 CPU 之间的 Cache 一致性，而 HT1 控制器(初始地址为 0x0E00_0000_0000 – 0x0FFF_FFFF_FFFF)不支持片间 Cache 一致性维护，详见 9.6 节。

HyperTransport 控制器最高支持双向 16 位宽度以及 800MHz 运行频率。在系统自动初始化建立连接后，用户程序可以通过修改协议中相应的配置寄存器，实现对宽度和运行频率的更改，并重新进行初始化，具体方法见 10.1 节。

龙芯 3B1500 HyperTransport 控制器的主要特征如下：

- 支持 200/400/800MHz 运行频率
- 支持 8/16 位宽度
- 每个 HT 控制器（HT0/HT1）可以配置为两个 8 位 HT 控制器
- 总线控制信号（包括 PowerOK，Rstn，LDT_Stopn）方向可配置
- 外设 DMA 空间 Cache/Uncache 可配置
- HT0 控制器用于多片互联时可配置为 Cache 一致性模式

## 10.1 HyperTransport 硬件设置及初始化

HyperTransport 总线由传输信号总线和控制信号引脚等组成，下表给出了 HyperTransport 总线相关的引脚及其功能描述。

表 10-1 HyperTransport 总线相关引脚信号

| 引脚 | 名称 | 描述 |
|---|---|---|
| HT0_8x2 | 总线宽度配置 | 1：将 16 位 HyperTransport 总线配置为两个独立的 8 位总线，分别由两个独立的控制器控制，地址空间的区分为<br>HT0_Lo：address[40] = 0；<br>HT0_Hi：address[40] = 1;<br>0:将 16 位 HyperTransport 总线作为一个 16 位总线使用，由 HT0_Lo 控制，地址空间为 HT0_Lo 的地址，即 address[40] =0；HT0_Hi 所有信号无效。 |
| HT0_Lo_mode | 主设备模式 | 1：将 HT0_Lo 设为主设备模式，这个模式下，总线控制信号等由 HT0_Lo 驱动，这些控制信号包括 HT0_Lo_Powerok， |

| | | |
|---|---|---|
| | | HT0_Lo_Rstn，HT0_Lo_Ldt_Stopn。这个模式下，这些控制信号也可以为双向驱动。同时这个引脚决定（取反）寄存器“Act as Slave”的初始值，这个寄存器为 0 时，HyperTransport 总线上的包中的 Bridge 位为 1，否则为 0。另外，这个寄存器为 0 时，如果 HyperTransport 总线上的请求地址没有在控制器的接收窗口命中时，将作为 P2P 请求重新发回总线，如果这个寄存器为 1 时，没有命中，则作为错误请求做出响应。<br>0：将 HT0_Lo 设为从设备模式，这个模式下，总线控制信号等由对方设备驱动，这些控制信号包括 HT0_Lo_Powerok，HT0_Lo_Rstn，HT0_Lo_Ldt_Stopn。这个模式下，这些控制信号由对方设备驱动，如果没有被正确驱动，则 HT 总线不能正确工作。 |
| HT0_Lo_Powerok | 总线 Powerok | HyperTransport 总线 Powerok 信号，<br>HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制；<br>HT0_Lo_Mode 为 0 时，由对方设备控制。 |
| HT0_Lo_Rstn | 总线 Rstn | HyperTransport 总线 Rstn 信号，<br>HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制；<br>HT0_Lo_Mode 为 0 时，由对方设备控制。 |
| HT0_Lo_Ldt_Stopn | 总线 Ldt_Stopn | HyperTransport 总线 Ldt_Stopn 信号，<br>HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制；<br>HT0_Lo_Mode 为 0 时，由对方设备控制。 |
| HT0_Lo_Ldt_Reqn | 总线 Ldt_Reqn | HyperTransport 总线 Ldt_Reqn 信号， |
| HT0_Hi_mode | 主设备模式 | 1：将 HT0_Hi 设为主设备模式，这个模式下，总线控制信号等由 HT0_Hi 驱动，这些控制信号包括 HT0_Hi_Powerok，HT0_Hi_Rstn，HT0_Hi_Ldt_Stopn。这个模式下，这些控制信号也可以为双向驱动。同时这个引脚决定（取反）寄存器“Act as Slave”的初始值，这个寄存器为 0 时，HyperTransport 总线上的包中的 Bridge 位为 1，否则为 0。另外，这个寄存器为 0 时，如果 HyperTransport 总线上的请求地址没有在控制器的接收窗口命中时，将作为 P2P 请求重新发回总线，如果这个寄存器为 1 时，没有命中，则作为错误请求做出响应。<br>0：将 HT0_Hi 设为从设备模式，这个模式下，总线控制信号等由对方设备驱动，这些控制信号包括 HT0_Hi_Powerok，HT0_Hi_Rstn，HT0_Hi_Ldt_Stopn。这个模式下，这些控制信号由对方设备驱动，如果没有被正确驱动，则 HT 总线不能正确工作。 |
| HT0_Hi_Powerok | 总线 Powerok | HyperTransport 总线 Powerok 信号，<br>HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制；<br>HT0_Lo_Mode 为 0 时，由对方设备控制。<br>HT0_8x2 为 1 时，控制高 8 位总线；<br>HT0_8x2 为 0 时，无效。 |
| HT0_Hi_Rstn | 总线 Rstn | HyperTransport 总线 Rstn 信号，<br>HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制；<br>HT0_Lo_Mode 为 0 时，由对方设备控制。<br>HT0_8x2 为 1 时，控制高 8 位总线； |

| | | |
|---|---|---|
| | | HT0_8x2 为 0 时，无效。 |
| HT0_Hi_Ldt_Stopn | 总线 Ldt_Stopn | HyperTransport 总线 Ldt_Stopn 信号，<br>HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制；<br>HT0_Lo_Mode 为 0 时，由对方设备控制。<br>HT0_8x2 为 1 时，控制高 8 位总线；<br>HT0_8x2 为 0 时，无效。 |
| HT0_Hi_Ldt_Reqn | 总线 Ldt_Reqn | HyperTransport 总线 Ldt_Reqn 信号，<br>HT0_8x2 为 1 时，控制高 8 位总线；<br>HT0_8x2 为 0 时，无效。 |
| HT0_Rx_CLKp[1:0]<br>HT0_Rx_CLKn[1:0]<br>HT0_Tx_CLKp[1:0]<br>HT0_Tx_CLKp[1:0] | CLK[1:0] | HyperTransport 总线 CLK 信号<br>HT0_8x2 为 1 时，CLK[1]由 HT0_Hi 控制<br>CLK[0]由 HT0_Lo 控制<br>HT0_8x2 为 0 时，CLK[1:0]由 HT0_Lo 控制 |
| HT0_Rx_CTLp[1:0]<br>HT0_Rx_CTLn[1:0]<br>HT0_Tx_CTLp[1:0]<br>HT0_Tx_CTLn[1:0] | CTL[1:0] | HyperTransport 总线 CTL 信号<br>HT0_8x2 为 1 时，CTL[1]由 HT0_Hi 控制<br>CTL[0]由 HT0_Lo 控制<br>HT0_8x2 为 0 时，CTL[1]无效<br>CTL[0]由 HT0_Lo 控制 |
| HT0_Rx_CADp[15:0]<br>HT0_Rx_CADn[15:0]<br>HT0_Tx_CADp[15:0]<br>HT0_Tx_CADn[15:0] | CAD[15:0] | HyperTransport 总线 CAD 信号<br>HT0_8x2 为 1 时，CAD[15:8]由 HT0_Hi 控制<br>CAD[ 7:0]由 HT0_Lo 控制<br>HT0_8x2 为 0 时，CAD[15:0]由 HT0_Lo 控制 |

HyperTransport 的初始化在每次复位完成后自动开始，冷启动后 HyperTransport 总线将自动工作在最低频率 (200MHz)与最小宽度(8bit)，并尝试进行总线初始化握手。初始化是否已处于完成状态可以由寄存器“Init Complete”（见 10.5.2 节）读出。初始化完成后，总线的宽度可以由寄存器“Link Width Out”与“Link Width In” （见 10.5.2 节）读出。初始化完成后，用户可重写寄存器“Link Width Out”、“Link Width In”以及“Link Freq”，同时还需要配置对方设备的相应寄存器，配置完成后需要热复位总线或者通过“HT_Ldt_Stopn”信号进行重新初始化操作，以便使寄存器重写后的值生效。重新初始化完成后 HyperTransport 总线将工作在新的频率和宽度。需要注意的是，HyperTransport 两端的设备的配置需要一一对应，否则将使得 HyperTransport 接口不能正常工作。

## 10.2 HyperTransport 协议支持

龙芯 3B1500 的 HyperTransport 总线支持 1.03 版协议中的大部分命令，并且在支持多芯片互联的扩展一致性协议中加入了一些扩展指令。在以上两种模式下，HyperTransport 接收端可接收的命令如下表所示。需要注意的是，不支持 HyperTransport 总线的原子操作命令。

表 10-2 HyperTransport 接收端可接收的命令

| 编码 | 通道 | 命令 | 标准模式 | 扩展（一致性） |
|---|---|---|---|---|
| 000000 | - | NOP | 空包或流控 | |
| 000001 | NPC | FLUSH | 无操作 | |

| | | | | |
|---|---|---|---|---|
| x01xxx | NPC or PC | Write | bit 5：0 - Nonposted<br>1 - Posted<br>bit 2：0 – Byte<br>1 – Doubleword<br>bit 1：Don’t Care<br>bit 0：Don’t Care | bit 5：必为 1，POSTED<br>bit 2：0 – Byte<br>1 – Doubleword<br>bit 1：Don’t Care<br>bit 0：必为 1 |
| 01xxxx | NPC | Read | bit 3：Don’t Care<br>bit 2：0 – Byte<br>1 – Doubleword<br>bit 1：Don’t Care<br>bit 0：Don’t Care | bit 3：Don’t Care<br>bit 2：0 – Byte<br>1 – Doubleword<br>bit 1：Don’t Care<br>bit 0：必为 1 |
| 110000 | R | RdResponse | 读操作返回 | |
| 110011 | R | TgtDone | 写操作返回 | |
| 110100 | PC | WrCoherent | ---- | 写命令扩展 |
| 110101 | PC | WrAddr | ---- | 写地址扩展 |
| 111000 | R | RespCoherent | ---- | 读响应扩展 |
| 111001 | NPC | RdCoherent | ---- | 读命令扩展 |
| 111010 | PC | Broadcast | 无操作 | |
| 111011 | NPC | RdAddr | ---- | 读地址扩展 |
| 111100 | PC | FENCE | 保证序关系 | |
| 111111 | - | Sync/Error | Sync/Error | |

对于发送端，在两种模式下会向外发送的命令如下表所示。

表 10-3 两种模式下会向外发送的命令

| 编码 | 通道 | 命令 | 标准模式 | 扩展（一致性） |
|---|---|---|---|---|
| 000000 | - | NOP | 空包或流控 | |
| x01x0x | NPC or PC | Write | bit 5：0 - Nonposted<br>1 - Posted<br>bit 2：0 – Byte<br>1 – Doubleword<br>bit 0：必为 0 | bit 5：必为 1，POSTED<br>bit 2：0 – Byte<br>1 – Doubleword<br>bit 0：必为 1 |
| 010x0x | NPC | Read | bit 2：0 – Byte<br>1 – Doubleword<br>bit 0：Don’t Care | bit 2：0 – Byte<br>1 – Doubleword<br>bit 0：必为 1 |
| 110000 | R | RdResponse | 读操作返回 | |
| 110011 | R | TgtDone | 写操作返回 | |
| 110100 | PC | WrCoherent | ---- | 写命令扩展 |
| 110101 | PC | WrAddr | ---- | 写地址扩展 |
| 111000 | R | RespCoherent | ---- | 读响应扩展 |
| 111001 | NPC | RdCoherent | ---- | 读命令扩展 |
| 111011 | NPC | RdAddr | ---- | 读地址扩展 |
| 111111 | - | Sync/Error | 只会转发 | |

## 10.3 HyperTransport 中断支持

HyperTransport 控制器提供了 256 个中断向量，可以支持 Fix，Arbiter 等类型的中断，但是，没有对硬件自动 EOI 提供支持。对于以上两种支持类型的中断，控制器在接收之后会自动写入中断寄存器中，并根据中断屏蔽寄存器的设置对系统中断控制器进行中断通知。

具体的中断控制请见 10.5 节中的中断控制寄存器组。

另外，控制器对 PIC 中断做了专门的支持，以加速该类型的中断处理。

一个典型的 PIC 中断由下述步骤完成：①PIC 控制器向系统发送 PIC 中断请求；②系统向 PIC 控制器发送中断向量查询；③PIC 控制器向系统发送中断向量号；④系统清除 PIC 控制器上的对应中断。只有上述 4 步都完成后，PIC 控制器才会对系统发出下一个中断。对于龙芯 3B1500 HyperTransport 控制器，将自动进行前 3 步的处理，并将 PIC 中断向量写入 256 个中断向量中的对应位置。而软件系统在处理了该中断之后，需要进行第 4 步处理，即向 PIC 控制器发出清中断。之后开始下一个中断的处理过程。

# 10.4 HyperTransport 地址窗口

## 10.4.1 HyperTransport 空间

龙芯 3B1500 处理器中，默认的 4 个 HyperTransport 接口的地址窗口分布如下：

表 10-4 默认的 4 个 HyperTransport 接口的地址窗口分布

| 基地址 | 结束地址 | 大小 | 定义 |
|---|---|---|---|
| 0x0E00_0000_0000 | 0x0EFF_FFFF_FFFF | 1 Tbytes | HT0_LO 窗口 |
| 0x0F00_0000_0000 | 0x0FFF_FFFF_FFFF | 1 Tbytes | HT0_HI 窗口 |
| 0x1E00_0000_0000 | 0x1EFF_FFFF_FFFF | 1 Tbytes | HT1_LO 窗口 |
| 0x1F00_0000_0000 | 0x1FFF_FFFF_FFFF | 1 Tbytes | HT1_HI 窗口 |

在默认情况下（未对系统地址窗口另行配置），软件根据上述地址空间对各个 HyperTransport 接口进行访问，此外，软件还可以通过对交叉开关上的地址窗口进行配置实现用其它的地址空间对其进行访问（详见 2.5 节）。每个 HyperTransport 接口的内部 40 位地址空间其地址窗口分布如下表所示。

表 10-5 龙芯 3 号处理器 HyperTransport 接口内部的地址窗口分布

| 基地址 | 结束地址 | 大小 | 定义 |
|---|---|---|---|
| 0x00_0000_0000 | 0xFC_FFFF_FFFF | 1012 Gbytes | MEM 空间 |
| 0xFD_0000_0000 | 0xFD_F7FF_FFFF | 3968 Mbytes | 保留 |
| 0xFD_F800_0000 | 0xFD_F8FF_FFFF | 16 Mbytes | 中断 |
| 0xFD_F900_0000 | 0xFD_F90F_FFFF | 1 Mbyte | PIC 中断响应 |
| 0xFD_F910_0000 | 0xFD_F91F_FFFF | 1 Mbyte | 系统信息 |
| 0xFD_F920_0000 | 0xFD_FAFF_FFFF | 30 Mbytes | 保留 |
| 0xFD_FB00_0000 | 0xFD_FBFF_FFFF | 16 Mbytes | HT 控制器配置空间 |
| 0xFD_FC00_0000 | 0xFD_FDFF_FFFF | 32 Mbytes | I/O 空间 |
| 0xFD_FE00_0000 | 0xFD_FFFF_FFFF | 32 Mbytes | HT 总线配置空间 |
| 0xFE_0000_0000 | 0xFF_FFFF_FFFF | 8 Gbytes | 保留 |

### 10.4.2 HyperTransport 控制器内部窗口配置

龙芯 3B1500 处理器的 HyperTransport 接口中提供了多种丰富的地址窗口供用户使用，这些地址窗口的作用和功能描述如下表所示。

表 10-6 龙芯 3B1500 处理器 HyperTransport 接口中提供的地址窗口

| 地址窗口 | 窗口数 | 接受总线 | 作用 | 备注 |
|---|---|---|---|---|
| 接收窗口（窗口配置见 10.5.7 节） | 3 | HyperTransport | 判断是否接收 HyperTransport 总线上发出的访问。 | 处于主桥模式时（即配置寄存器中 act_as_slave 为 0），只有落在这些地址窗口中的访问会被内部总线所响应，其它访问将会被认为是 P2P 访问重新发回到 HyperTransport 总线上；处于设备模式时（即配置寄存器中 act_as_slave 为 1），只有落在这些地址窗口中的访问会被内部总线所接收并处理，其它访问将会按照协议给出错误返回。 |
| Post 窗口（窗口配置见 10.5.11 节） | 2 | 内部总线 | 判断是否将内部总线对 HyperTransport 总线的写访问作为 Post Write | 落在这些地址空间中的对外写访问将作为 Post Write。<br>Post Write：HyperTransport 协议中，这种写访问不需要等待写完成响应，即在控制器向总线发出这个写访问之后就将对处理器进行写访问完成响应。 |
| 可预取窗口（窗口配置见 10.5.12 节） | 2 | 内部总线 | 判断是否接收内部的 Cache 访问，取指访问。 | 当处理器核乱序执行时，会对总线发出一些猜测读访问或是取指访问，这种访问对于某些 IO 空间是错误的。在默认情况下，这种访问 HT 控制器将直接返回而不对 HyperTransport 总线进行访问。通过这些窗口可以使能对 HyperTransport 总线的这类访问。 |
| Uncache 窗口（窗口配置见 10.5.13 节） | 2 | HyperTransport | 判断是否将 HyperTransport 总线上的访问作为对内部的 Uncache 访问 | 龙芯 3B1500 处理器内部的 IO DMA 访问，在情况下将作为 Cache 方式访问经由 SCache 判断是命中，从而维护其 IO 一致性信息。而通过这些窗口的配置，可以使在这些窗口命中的访问以 Uncache 的方式直接访问内存，而不通过硬件维护其 IO 一致性信息。 |

## 10.5 配置寄存器

配置寄存器模块主要用于控制从 AXI SLAVE 端或是 HT RECEIVER 端到达的配置寄存器访问请求，进行外部中断处理，并保存了大量软件可见的用于控制系统各种工作方式的配置寄存器。

首先，用于控制 HT 控制器各种行为的配置寄存器的访问与存储都在本模块中，本模块

的访问偏移地址在 AXI 端为 0xFD_FB00_0000 到 0xFD_FBFF_FFFF。本模块中所有软件可见寄存器如下表所示：

表 10-7 本模块中所有软件可见寄存器

| 偏移地址 | 名称 | 描述 |
|---|---|---|
| 0x30 | | |
| 0x34 | | |
| 0x38 | | |
| 0x3c | Bridge Control | Bus Reset Control |
| 0x40 | Capability Registers | Command，Capabilities Pointer，Capability ID |
| 0x44 | | Link Config，Link Control |
| 0x48 | | Revision ID，Link Freq，Link Error，Link Freq Cap |
| 0x4c | | Feature Capability |
| 0x50 | 自定义寄存器 | MISC |
| 0x54 | 接收诊断寄存器 | 用于诊断接收端采样的信号 |
| 0x58 | 中断路由方式选择寄存器 | 对应于 3 种中断路由方式 |
| 0x5c | 接收缓存寄存器 | |
| 0x60 | 接收地址窗口配置寄存器 | HT 总线接收地址窗口 0 使能（外部访问） |
| 0x64 | | HT 总线接收地址窗口 0 基址（外部访问） |
| 0x68 | | HT 总线接收地址窗口 1 使能（外部访问） |
| 0x6c | | HT 总线接收地址窗口 1 基址（外部访问） |
| 0x70 | | HT 总线接收地址窗口 2 使能（外部访问） |
| 0x74 | | HT 总线接收地址窗口 2 基址（外部访问） |
| 0x78 | | |
| 0x7c | | |
| 0x80 | 中断向量寄存器 | HT 总线中断向量寄存器[31:0] |
| 0x84 | | HT 总线中断向量寄存器[63:32] |
| 0x88 | | HT 总线中断向量寄存器[95:64] |
| 0x8c | | HT 总线中断向量寄存器[127:96] |
| 0x90 | | HT 总线中断向量寄存器[159:128] |
| 0x94 | | HT 总线中断向量寄存器[191:160] |
| 0x98 | | HT 总线中断向量寄存器[223:192] |
| 0x9C | | HT 总线中断向量寄存器[255:224] |
| 0xA0 | 中断使能寄存器 | HT 总线中断使能寄存器[31:0] |
| 0xA4 | | HT 总线中断使能寄存器[63:32] |
| 0xA8 | | HT 总线中断使能寄存器[95:64] |
| 0xAC | | HT 总线中断使能寄存器[127:96] |
| 0xB0 | | HT 总线中断使能寄存器[159:128] |
| 0xB4 | | HT 总线中断使能寄存器[191:160] |
| 0xB8 | | HT 总线中断使能寄存器[223:192] |
| 0xBC | | HT 总线中断使能寄存器[255:224] |
| 0xC0 | Interrupt Discovery & Configuration | Interrupt Capability |
| 0xC4 | | DataPort |
| 0xC8 | | IntrInfo[31:0] |
| 0xCC | | IntrInfo[63:32] |
| 0xD0 | POST 地址窗口配置寄存器 | HT 总线 POST 地址窗口 0 使能（内部访问） |
| 0xD4 | | HT 总线 POST 地址窗口 0 基址（内部访问） |
| 0xD8 | | HT 总线 POST 地址窗口 1 使能（内部访问） |
| 0xDC | | HT 总线 POST 地址窗口 1 基址（内部访问） |
| 0xE0 | 可预取地址窗口 | HT 总线可预取地址窗口 0 使能（内部访问） |

| | | |
|---|---|---|
| 0xE4 | 配置寄存器 | HT 总线可预取地址窗口 0 基址（内部访问） |
| 0xE8 | | HT 总线可预取地址窗口 1 使能（内部访问） |
| 0xEC | | Ht 总线可预取地址窗口 1 基址（内部访问） |
| 0xF0 | Uncache 地址窗口配置寄存器 | HT 总线 Uncache 地址窗口 0 使能（内部访问） |
| 0xF4 | | HT 总线 Uncache 地址窗口 0 基址（内部访问） |
| 0xF8 | | HT 总线 Uncache 地址窗口 1 使能（内部访问） |
| 0xFC | | HT 总线 Uncache 地址窗口 1 基址（内部访问） |
| 0x100 | 发送端缓存大小寄存器 | 发送端命令缓存大小寄存器 |
| 0x104 | | 发送端数据缓存大小寄存器 |
| 0x108 | 发送端缓存调试寄存器 | 用于人工设置发送端缓存的大小（调试用） |
| 0x10C | 预加重和输出阻抗寄存器 | 用于设置 HT 接口预加重和输出阻抗 |

每个寄存存器的具体含义如下节如示：

## 10.5.1 Bridge Control

偏移量：　0x3C

复位值：　0x00000000

名称：　Bus Reset Control

表 10-8 Bus Reset Control 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:23 | Reserved | 4 | 0x0 | | 保留 |
| 22 | Reset | 12 | 0x0 | R/W | 总线复位控制：<br>0->1：HT_RSTn 置 0，总线复位<br>1->0：HT_RSTn 置 1，总线解复位 |
| 21:0 | Reserved | 5 | 0x0 | | 保留 |

## 10.5.2 Capability Registers

偏移量：　0x40

复位值：　0x20010008

名称：　Command，Capabilities Pointer，Capability ID

表 10-9 Command，Capabilities Pointer，Capability ID 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:29 | HOST/Sec | 3 | 0x1 | R | Command 格式为 HOST/Sec |
| 28:27 | Reserved | 2 | 0x0 | R | 保留 |

| | | | | | |
|---|---|---|---|---|---|
| 26 | Act as Slave | 1 | 0x0<br>/0x1 | R/W | HOST/SLAVE 模式<br>初始值由引脚 HOSTMODE 决定<br>HOSTMODE 上拉：0<br>HOSTMODE 下拉：1 |
| 25 | Reserved | 1 | 0x0 | | 保留 |
| 24 | Host Hide | 1 | 0x0 | R/W | 是否禁止来自 HT 总线的寄存器访问 |
| 23 | Reserved | 1 | 0x0 | | 保留 |
| 22:18 | Unit ID | 5 | 0x0 | R/W | HOST 模式时：可用于记录使用 ID 个数<br>SLAVE 模式时：记录自身 Unit ID |
| 17 | Double Ended | 1 | 0x0 | R | 不采用双 HOST 模式 |
| 16 | Warm Reset | 1 | 0x1 | R | Bridge Control 中 reset 采用热复位方式 |
| 15:8 | Capabilities Pointer | 8 | 0xa0 | R | 下一个 Cap 寄存器偏移地址 |
| 7:0 | Capability ID | 8 | 0x08 | R | HyperTransport capability ID |

偏移量： 0x44

复位值： 0x00112000

名称： Link Config，Link Control

表 10-10 Link Config，Link Control 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_phase_select<br>_disable | 1 | 0x0 | | 相位选择使能<br>0：使能相位选择功能<br>1：禁用相位选择功能 |
| 30:28 | Link Width Out | 3 | 0x0 | R/W | 发送端宽度<br>冷复位后的值为当前连接的最大宽度，写入此寄存器的值将会在下次热复位或是 HT Disconnect 之后生效<br>000：8 位方式<br>001：16 位方式 |
| 27 | Reserved | 1 | 0x0 | | 保留 |
| 26:24 | Link Width In | 3 | 0x0 | R/W | 接收端宽度<br>冷复位后的值为当前连接的最大宽度，写入此寄存器的值将会在下次热复位或是 HT Disconnect 之后生效 |
| 23 | Dw Fc out | 1 | 0x0 | R | 发送端不支持双字流控 |
| 22:20 | Max Link Width out | 3 | 0x1 | R | HT 总线发送端最大宽度：16bits |

| | | | | | |
|---|---|---|---|---|---|
| 19 | Dw Fc In | 1 | 0x0 | R | 接收端不支持双字流控 |
| 18:16 | Max Link Width In | 3 | 0x1 | R | HT 总线接收端最大宽度：16bits |
| 15:14 | Reserved | 2 | 0x0 | | 保留 |
| 13 | LDTSTOP# Tristate Enable | 1 | 0x1 | R/W | 当 HT 总线进入 HT Disconnect 状态时，是否关闭 HT PHY<br>1：关闭<br>0：不关闭 |
| 12:10 | Reserved | 3 | 0x0 | | 保留 |
| 9 | CRC Error (hi) | 1 | 0x0 | R/W | 高 8 位发生 CRC 错 |
| 8 | CRC Error (lo) | 1 | 0x0 | R/W | 低 8 位发生 CRC 错 |
| 7 | Trans off | 1 | 0x0 | R/W | HT PHY 关闭控制<br>处于 16 位总线工作方式时<br>1：关闭 高/低 8 位 HT PHY<br>0：使能 低 8 位 HT PHY，<br>高 8 位 HT PHY 由 bit 0 控制 |
| 6 | End of Chain | 0 | 0x0 | R | HT 总线末端 |
| 5 | Init Complete | 1 | 0x0 | R | HT 总线初始化是否完成 |
| 4 | Link Fail | 1 | 0x0 | R | 指示连接失败 |
| 3:2 | Reserved | 2 | 0x0 | | 保留 |
| 1 | CRC Flood Enable | 1 | 0x0 | R/W | 发生 CRC 错误时，是否 flood HT 总线 |
| 0 | Trans off (hi) | 1 | 0x0 | R/W | 使用 16 位 HT 总线运行 8 位协议时，<br>高 8 位 PHY 关闭控制<br>1：关闭 高 8 位 HT PHY<br>0：使能 高 8 位 HT PHY |

偏移量： 0x48

复位值： 0x80250023

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

表 10-11 Revision ID，Link Freq，Link Error，Link Freq Cap 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | Link Freq Cap | 16 | 0x0025 | R | 支持的 HT 总线频率（根据 CLKSEL[15:10]的设置，可支持的频率各不相同，详见表 2-1 的设置说明），包括<br>200Mhz，400Mhz，800Mhz，1000MHz，1200MHz，1400MHz，1600MHz |

| | | | | | |
|---|---|---|---|---|---|
| 15:14 | Reserved | 2 | 0x0 | | 保留 |
| 13 | Over Flow Error | 1 | 0x0 | R | HT 总线包溢出 |
| 12 | Protocol Error | 1 | 0x0 | R/W | 协议错误，<br>指 HT 总线上收到不可识别的命令 |
| 11:8 | Link Freq | 4 | 0x0 | R/W | HT 总线工作频率<br>写入此寄存器的值后将在下次热复位或是 HT Disconnect 之后生效<br>0000：200MHz<br>0010：400MHz<br>0101：800MHz<br>0110：1000MHz<br>0111：1200MHz<br>1000：1400MHz<br>1001：1600MHz |
| 7:0 | Revision ID | 8 | 0x23 | R/W | 版本号：1.03 |

偏移量： 0x4C

复位值： 0x00000002

名称： Feature Capability

表 10-12 Feature Capability 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:9 | Reserved | 25 | 0x0 | | 保留 |
| 8 | Extended Register | 1 | 0x0 | R | 没有 |
| 7:4 | Reserved | 3 | 0x0 | | 保留 |
| 3 | Extended CTL Time | 1 | 0x0 | R | 不需要 |
| 2 | CRC Test Mode | 1 | 0x0 | R | 不支持 |
| 1 | LDTSTOP# | 1 | 0x1 | R | 支持 LDTSTOP# |
| 0 | Isochronous Mode | 1 | 0x0 | R | 不支持 |

## 10.5.3 自定义寄存器

偏移量： 0x50

复位值： 0x00904321

名称： MISC

表 10-13 MISC 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | Reserved | 1 | 0x0 | | 保留 |
| 30 | Ldt Stop Gen | 1 | 0x0 | R/W | 使总线进入 LDT DISCONNECT 模式<br>正确的方法是：0 -> 1 |
| 29 | Ldt Req Gen | 1 | 0x0 | R/W | 从 LDT DISCONNECT 中唤醒 HT 总线，设置 LDT_REQ_n<br>正确的方法是先置 0 再置 1：0 -> 1<br>除此之外，直接向总线发出读写请求也可以自动唤醒总线 |
| 28:24 | Interrupt Index | 5 | 0x0 | R/W | 将除了标准中断之外的其它中断重定向到哪个中断向量中（包括 SMI，NMI，INIT，INTA，INTB，INTC，INTD）<br>总共 256 个中断向量，本寄存器表示的是中断向量的高 5 位，内部中断向量如下：<br>000：SMI<br>001：NMI<br>010：INIT<br>011：Reservered<br>100：INTA<br>101：INTB<br>110：INTC<br>111：INTD |
| 23 | Dword Write | 1 | 0x1 | R/W | 对于 32/64/128/256 位的写访问，是否采用 Dword Write 命令格式<br>1：使用 Dword Write<br>0：使用 Byte Write（带 MASK） |
| 22 | Coherent Mode | 1 | 0x0 | R | 是否是处理器一致性模式<br>由引脚 ICCC_EN 决定 |
| 21 | Not Care Seqid | 1 | 0x0 | R/W | 是否不关心 HT 序关系 |

| | | | | | |
|---|---|---|---|---|---|
| 20 | Not Axi2Seqid | 1 | 0x1 | R | 是否把 Axi 总线上的命令转换成不同的 SeqID，如果不转换，则所有的读写命令都会采用 Fixed Seqid 中的固定 ID 号<br>1：不转换<br>0：转换 |
| 19:16 | Fixed Seqid | 4 | 0x0 | R/W | 当 Not Axi2Seqid 有效时，配置 HT 总线发出的 Seqid |
| 15:12 | Priority Nop | 4 | 0x4 | R/W | HT 总线 Nop 流控包优先级 |
| 11:8 | Priority NPC | 4 | 0x3 | R/W | Non Post 通道读写优先级 |
| 7:4 | Priority RC | 4 | 0x2 | R/W | Response 通道读写先级 |
| 3:0 | Priority PC | 4 | 0x1 | R/W | Post 通道读写优先级<br>0x0：最高优先级<br>0xF：最低优先级<br>对于各个通道的优先级均采用根据时间变化提高的优先级策略，该组存器用于配置各个通道的初始优先级 |

## 10.5.4 接收诊断寄存器

偏移量：　　0x54

复位值：　　0x00000000

名称：　　接收诊断寄存器

表 10-14 接收诊断寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 0 | Sample_en | 1 | 0x0 | R/W | 使能采样输入的 cad 和 ctl<br>0x0：禁止<br>0x1：使能 |
| 15:8 | rx_ctl_catch | 24 | 0x0 | R/W | 保存采样得到的输入 ctl<br>（0、2、4、6）对应 CTL0 采样的四个相位<br>（1、3、5、7）对应 CTL1 采样的四个相位 |
| 31:16 | rx_cad_phase_0 | 24 | 0x0 | R/W | 保存采样得到的输入 CAD[15:0]的值 |

## 10.5.5 中断路由方式选择寄存器

偏移量：　　0x58

复位值： 0x00000000

名称： 中断路由方式选择寄存器

表 10-15 中断路由方式选择寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 9:8 | ht_int_stripe | 2 | 0x0 | R/W | 对应于 3 种中断路由方式，具体描述见 10.5.8 中断向量寄存器<br>0x0：ht_int_stripe_1<br>0x1：ht_int_stripe_2<br>0x2：ht_int_stripe_4 |

## 10.5.6 接收缓冲区初始寄存器

偏移量： 0x5c

复位值： 0x07778888

名称： 接收缓冲区初始化配置寄存器

表 10-16 接收缓冲区初始寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 27:24 | rx_buffer_r_data | 4 | 0x0 | R/W | 接收缓冲区的读数据 buffer 初始化信息 |
| 23:20 | rx_buffer_npc_data | 4 | 0x0 | R/W | 接收缓冲区的 npc 数据 buffer 初始化信息 |
| 19:16 | rx_buffer_pc_data | 4 | 0x0 | R/W | 接收缓冲区的 pc 数据 buffer 初始化信息 |
| 15:12 | rx_buffer_b_cmd | 4 | 0x0 | R/W | 接收缓冲区的 bresponse 命令 buffer 初始化信息 |
| 11:8 | rx_buffer_r_cmd | 4 | 0x0 | R/W | 接收缓冲区的读命令 buffer 初始化信息 |
| 7:4 | rx_buffer_npc_cmd | 4 | 0x0 | R/W | 接收缓冲区的 npc 命令 buffer 初始化信息 |
| 3:0 | rx_buffer_pc_cmd | 4 | 0x0 | R/W | 接收缓冲区的 pc 命令 buffer 初始化信息 |

## 10.5.7 接收地址窗口配置寄存器

HT 控制器中的地址窗口命中公式如下：

hit = ( BASE & MASK ) == ( ADDR & MASK )

addr_out = TRANS_EN ? TRANS | ADDR & ~MASK : ADDR

需要说明的是，配置地址窗口寄存器时，MASK 高位应全为 1，低位应全为 0。MASK 中 0 的实际位数表示的就是地址窗口的大小。

接收地址窗口的地址为 HT 总线上接收的地址。落在本窗口内的 HT 地址将被发往 CPU 内，其它地址的命令将作为 P2P 命令被转发回 HT 总线。

偏移量： 0x60

复位值： 0x00000000

名称： HT 总线接收地址窗口 0 使能（外部访问）

表 10-17 HT 总线接收地址窗口 0 使能（外部访问）寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_rx_image0_en | 1 | 0x0 | R/W | HT 总线接收地址窗口 0，使能信号 |
| 30 | ht_rx_image0_ trans_en | 1 | 0x0 | R/W | HT 总线接收地址窗口 0，映射使能信号 |
| 29:0 | ht_rx_image0_ trans[53:24] | 30 | 0x0 | R/W | HT 总线接收地址窗口 0，映射后地址的[53:24] |

偏移量： 0x64

复位值： 0x00000000

名称： HT 总线接收地址窗口 0 基址（外部访问）

表 10-18 HT 总线接收地址窗口 0 基址（外部访问）寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_rx_image0_ base[39:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 0，地址基址的[39:24] |
| 15:0 | ht_rx_image0_ mask[39:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 0，地址屏蔽的[39:24] |

偏移量： 0x68

复位值： 0x00000000

名称： HT 总线接收地址窗口 1 使能（外部访问）

表 10-19 HT 总线接收地址窗口 1 使能（外部访问）寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_rx_image1_en | 1 | 0x0 | R/W | HT 总线接收地址窗口 1，使能信号 |
| 30 | ht_rx_image1_ trans_en | 1 | 0x0 | R/W | HT 总线接收地址窗口 1，映射使能信号 |
| 29:0 | ht_rx_image1_ trans[53:24] | 30 | 0x0 | R/W | HT 总线接收地址窗口 1，映射后地址的[53:24] |

偏移量： 0x6c

复位值： 0x00000000

名称： HT 总线接收地址窗口 1 基址（外部访问）

表 10-20 HT 总线接收地址窗口 1 基址（外部访问）寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_rx_image1_base[39:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 1，地址基址的[39:24] |
| 15:0 | ht_rx_image1_mask[39:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 1，地址屏蔽的[39:24] |

偏移量： 0x70

复位值： 0x00000000

名称： HT 总线接收地址窗口 2 使能（外部访问）

表 10-21 HT 总线接收地址窗口 2 使能（外部访问）寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_rx_image2_en | 1 | 0x0 | R/W | HT 总线接收地址窗口 2，使能信号 |
| 30 | ht_rx_image2_trans_en | 1 | 0x0 | R/W | HT 总线接收地址窗口 2，映射使能信号 |
| 29:0 | ht_rx_image2_trans[53:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 2，转译后地址的[53:24] |

偏移量： 0x74

复位值： 0x00000000

名称： HT 总线接收地址窗口 2 基址（外部访问）

表 10-22 HT 总线接收地址窗口 2 基址（外部访问）寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_rx_image2_base[39:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 2，地址基址的[39:24] |
| 15:0 | ht_rx_image2_mask[39:24] | 16 | 0x0 | R/W | HT 总线接收地址窗口 2，地址屏蔽的[39:24] |

## 10.5.8 中断向量寄存器

中断向量寄存器共 256 个，其中除去 HT 总线上的 Fix、Arbiter 以及 PIC 中断直接映射到此 256 个中断向量之中，其它的中断，如 SMI，NMI，INIT，INTA，INTB，INTC，INTD 可以通过寄存器 0x50 的[28:24]映射到任意一个 8 位中断向量上去，映射的顺序为{INTD，INTC，INTB，INTA，1’b0，INIT，NMI，SMI}。此时中断向量对应值为{Interrupt Index, 内部向量[2:0]}。

LS3A1000E 及以上版本，256 个中断向量根据中断路由方式选择寄存器配置的不同映射

到不同的中断线上，具体的映射方式为：

ht_int_stripe_1:

[0,1,2,3……63]对应中断线 0 /HT HI 对应中断线 4

[64,65,66,67……127]对应中断线 1 /HT HI 对应中断线 5

[128,129,130,131……191]对应中断线 2 /HT HI 对应中断线 6

[192,193,194,195……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_2:

[0,2,4,6……126]对应中断线 0 /HT HI 对应中断线 4

[1,3,5,7……127]对应中断线 1 /HT HI 对应中断线 5

[128,130,132,134……254]对应中断线 2 /HT HI 对应中断线 6

[129,131,133,135……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_4:

[0,4,8,12……252]对应中断线 0 /HT HI 对应中断线 4

[1,5,9,13……253]对应中断线 1 /HT HI 对应中断线 5

[2,6,10,14……254]对应中断线 2 /HT HI 对应中断线 6

[3,7,11,15……255]对应中断线 3 /HT HI 对应中断线 7

以下中断向量的描述对应于 ht_int_stripe_1，另外两种方式可由以上说明得到。

对于 LS3A1000D 及以下版本，只能使用 ht_int_stripe_1 的方式。

偏移量：　　0x80

复位值：　　0x00000000

名称：　　HT 总线中断向量寄存器[31:0]

表 10-23 HT 总线中断向量寄存器定义（1）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [31:0] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[31:0]，对应中断线 0 /HT HI 对应中断线 4 |

偏移量：　　0x84

复位值：　　0x00000000

名称：　　HT 总线中断向量寄存器[63:32]

表 10-24 HT 总线中断向量寄存器定义（2）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [63:32] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[63:32]，对应中断线 0 /HT HI 对应中断线 4 |

偏移量： 0x88

复位值： 0x00000000

名称： HT 总线中断向量寄存器[95:64]

表 10-25 HT 总线中断向量寄存器定义（3）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [95:64] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[95:64]，对应中断线 1 /HT HI 对应中断线 5 |

偏移量： 0x8c

复位值： 0x00000000

名称： HT 总线中断向量寄存器[127:96]

表 10-26 HT 总线中断向量寄存器定义（4）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [127:96] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[127:96]，对应中断线 1 /HT HI 对应中断线 5 |

偏移量： 0x90

复位值： 0x00000000

名称： HT 总线中断向量寄存器[159:128]

表 10-27 HT 总线中断向量寄存器定义（5）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [159:128] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[159:128]，对应中断线 2 /HT HI 对应中断线 6 |

偏移量： 0x94

复位值： 0x00000000

名称： HT 总线中断向量寄存器[191:160]

表 10-28 HT 总线中断向量寄存器定义（6）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [191:160] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[191:160]，对应中断线 2 /HT HI 对应中断线 6 |

偏移量： 0x98

复位值： 0x00000000

名称： HT 总线中断向量寄存器[223:192]

表 10-29 HT 总线中断向量寄存器定义（7）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [223:192] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[223:192]，对应中断线 3 /HT HI 对应中断线 7 |

偏移量：　　0x9c

复位值：　　0x00000000

名称：　　HT 总线中断向量寄存器[255:224]

表 10-30 HT 总线中断向量寄存器定义（8）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_case [255:224] | 32 | 0x0 | R/W | HT 总线中断向量寄存器[255:224]，对应中断线 3 /HT HI 对应中断线 7 |

## 10.5.9 中断使能寄存器

中断使能寄存器共 256 个，与中断向量寄存器一一对应。 置 1 为对应中断打开，置 0 则为中断屏蔽。

256 个中断向量根据中断路由方式选择寄存器配置的不同映射到不同的中断线上，具体的映射方式为：

ht_int_stripe_1:

[0,1,2,3……63]对应中断线 0 /HT HI 对应中断线 4

[64,65,66,67……127]对应中断线 1 /HT HI 对应中断线 5

[128,129,130,131……191]对应中断线 2 /HT HI 对应中断线 6

[192,193,194,195……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_2:

[0,2,4,6……126]对应中断线 0 /HT HI 对应中断线 4

[1,3,5,7……127]对应中断线 1 /HT HI 对应中断线 5

[128,130,132,134……254]对应中断线 2 /HT HI 对应中断线 6

[129,131,133,135……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_4:

[0,4,8,12……252]对应中断线 0 /HT HI 对应中断线 4

[1,5,9,13……253]对应中断线 1 /HT HI 对应中断线 5

[2,6,10,14……254]对应中断线 2 /HT HI 对应中断线 6

[3,7,11,15……255]对应中断线 3 /HT HI 对应中断线 7

以下中断向量的描述对应于 ht_int_stripe_1，另外两种方式可由以上说明得到。

偏移量：　　0xa0

复位值： 0x00000000

名称： HT 总线中断使能寄存器[31:0]

表 10-31 HT 总线中断使能寄存器定义（1）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [31:0] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[31:0]，<br>对应中断线 0 /HT HI 对应中断线 4 |

偏移量： 0xa4

复位值： 0x00000000

名称： HT 总线中断使能寄存器[63:32]

表 10-32 HT 总线中断使能寄存器定义（2）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [63:32] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[63:32]，<br>对应中断线 0 /HT HI 对应中断线 4 |

偏移量： 0xa8

复位值： 0x00000000

名称： HT 总线中断使能寄存器[95:64]

表 10-33 HT 总线中断使能寄存器定义（3）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [95:64] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[95:64]，<br>对应中断线 1 /HT HI 对应中断线 5 |

偏移量： 0xac

复位值： 0x00000000

名称： HT 总线中断使能寄存器[127:96]

表 10-34 HT 总线中断使能寄存器定义（4）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [127:96] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[127:96]，<br>对应中断线 1 /HT HI 对应中断线 5 |

偏移量： 0xb0

复位值： 0x00000000

名称： HT 总线中断使能寄存器[159:128]

表 10-35 HT 总线中断使能寄存器定义（5）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [159:128] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[159:128]，对应中断线 2 /HT HI 对应中断线 6 |

偏移量： 0xb4

复位值： 0x00000000

名称： HT 总线中断使能寄存器[191:160]

表 10-36 HT 总线中断使能寄存器定义（6）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [191:160] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[191:160]，对应中断线 2 /HT HI 对应中断线 6 |

偏移量： 0xb8

复位值： 0x00000000

名称： HT 总线中断使能寄存器[223:192]

表 10-37 HT 总线中断使能寄存器定义（7）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [223:192] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[223:192]，对应中断线 3 /HT HI 对应中断线 7 |

偏移量： 0xbc

复位值： 0x00000000

名称： HT 总线中断使能寄存器[255:224]

表 10-38 HT 总线中断使能寄存器定义（8）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Interrupt_mask [255:224] | 32 | 0x0 | R/W | HT 总线中断使能寄存器[255:224]，对应中断线 3 /HT HI 对应中断线 7 |

## 10.5.10 Interrupt Discovery & Configuration

偏移量： 0xc0

复位值： 0x80000008

名称： Interrupt Capability

表 10-39 Interrupt Capability 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:24 | Capabilities Pointer | 8 | 0x80 | R | Interrupt discovery and configuration block |
| 23:16 | Index | 8 | 0x0 | R/W | 读寄存器偏移地址 |

| | | | | | |
|---|---|---|---|---|---|
| 15:8 | Capabilities Pointer | 8 | 0x0 | R | Capabilities Pointer |
| 7:0 | Capability ID | 8 | 0x08 | R | Hypertransport Capablity ID |

偏移量： 0xc4

复位值： 0x00000000

名称： Dataport

表 10-40 Dataport 寄存器定义

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | Dataport | 32 | 0x0 | R/W | 当上一寄存器 Index 为 0x10 时，本寄存器读写结果为 0xa8 寄存器，否则为 0xac |

偏移量： 0xc8

复位值： 0xF8000000

名称： IntrInfo[31:0]

表 10-41 IntrInfo 寄存器定义（1）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:24 | IntrInfo[31:24] | 32 | 0xF8 | R | 保留 |
| 23:2 | IntrInfo[23:2] | 22 | 0x0 | R/W | IntrInfo[23:2]，当发出 PIC 中断时，IntrInfo 的值用来表示中断向量 |
| 1:0 | Reserved | 2 | 0x0 | R | 保留 |

偏移量： 0xcc

复位值： 0x00000000

名称： IntrInfo[63:32]

表 10-42 IntrInfo 寄存器定义（2）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:0 | IntrInfo[63:32] | 32 | 0x0 | R | 保留 |

## 10.5.11 POST 地址窗口配置寄存器

地址窗口命中公式详见 10.5.7 节。

本窗口的地址是 AXI 总线上接收到的地址。落在本窗口的所有写访问将立即在 AXI B 通道返回，并以 POST WRITE 的命令格式发给 HT 总线。而不在本窗口的写请求则以 NONPOST WRITE 的方式发送到 HT 总线，并等待 HT 总线响应后再返回 AXI 总线。

偏移量： 0xd0

复位值： 0x00000000

名称： HT 总线 POST 地址窗口 0 使能（内部访问）

表 10-43 HT 总线 POST 地址窗口 0 使能（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_post0_en | 1 | 0x0 | R/W | HT 总线 POST 地址窗口 0，使能信号 |
| 30:23 | Reserved | 15 | 0x0 | | 保留 |
| 15:0 | ht_post0_trans [39:24] | 16 | 0x0 | R/W | HT 总线 POST 地址窗口 0，转译后地址的[39:24] |

偏移量： 0xd4

复位值： 0x00000000

名称： HT 总线 POST 地址窗口 0 基址（内部访问）

表 10-44 HT 总线 POST 地址窗口 0 基址（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_post0_base [39:24] | 16 | 0x0 | R/W | HT 总线 POST 地址窗口 0，地址基址的[39:24] |
| 15:0 | ht_post0_mask [39:24] | 16 | 0x0 | R/W | HT 总线 POST 地址窗口 0，地址屏蔽的[39:24] |

偏移量： 0xd8

复位值： 0x00000000

名称： HT 总线 POST 地址窗口 1 使能（内部访问）

表 10-45 HT 总线 POST 地址窗口 1 使能（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_post1_en | 1 | 0x0 | R/W | HT 总线 POST 地址窗口 1，使能信号 |
| 30:16 | Reserved | 15 | 0x0 | | 保留 |
| 15:0 | ht_post1_trans [39:24] | 16 | 0x0 | R/W | HT 总线 POST 地址窗口 1，转译后地址的[39:24] |

偏移量： 0xdc

复位值： 0x00000000

名称： HT 总线 POST 地址窗口 1 基址（内部访问）

表 10-46 HT 总线 POST 地址窗口 1 基址（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_post1_base [39:24] | 16 | 0x0 | R/W | HT 总线 POST 地址窗口 1，地址基址的[39:24] |

| | | | | | |
|---|---|---|---|---|---|
| 15:0 | ht_post1_mask [39:24] | 16 | 0x0 | R/W | HT 总线 POST 地址窗口 1，地址屏蔽的[39:24] |

## 10.5.12 可预取地址窗口配置寄存器

地址窗口命中公式详见 10.5.7 节。

本窗口的地址是 AXI 总线上接收到的地址。落在本窗口的取指指令以及 CACHE 访问才会被发往 HT 总线，其它的取指或 CACHE 访问将不会被发往 HT 总线，而是立即返回，如果是读命令，则会返回相应个数的无效读数据。

偏移量： 0xe0

复位值： 0x00000000

名称： HT 总线可预取地址窗口 0 使能（内部访问）

表 10-47 HT 总线可预取地址窗口 0 使能（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_prefetch0_en | 1 | 0x0 | R/W | HT 总线可预取地址窗口 0，使能信号 |
| 30:23 | Reserved | 15 | 0x0 | | 保留 |
| 15:0 | ht_prefetch0_trans [39:24] | 16 | 0x0 | R/W | HT 总线可预取地址窗口 0,转译后地址的[39:24] |

偏移量： 0xe4

复位值： 0x00000000

名称： HT 总线可预取地址窗口 0 基址（内部访问）

表 10-48 HT 总线可预取地址窗口 0 基址（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_prefetch0_ base[39:24] | 16 | 0x0 | R/W | HT 总线可预取地址窗口 0，地址基址的[39:24] 位地址 |
| 15:0 | ht_prefetch0_ mask[39:24] | 16 | 0x0 | R/W | HT 总线可预取地址窗口 0，地址屏蔽的[39:24] |

偏移量： 0xe8

复位值： 0x00000000

名称： HT 总线可预取地址窗口 1 使能（内部访问）

表 10-49 HT 总线可预取地址窗口 1 使能（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 31 | ht_prefetch1_en | 1 | 0x0 | R/W | HT 总线可预取地址窗口 1，使能信号 |
| 30:23 | Reserved | 15 | 0x0 | | 保留 |
| 15:0 | ht_prefetch1_ trans[39:24] | 16 | 0x0 | R/W | HT 总线可预取地址窗口 1,转译后地址的[39:24] |

偏移量： 0xec

复位值： 0x00000000

名称： HT 总线可预取地址窗口 1 基址（内部访问）

表 10-50 HT 总线可预取地址窗口 1 基址（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_prefetch1_ base[39:24] | 16 | 0x0 | R/W | HT 总线可预取地址窗口 1，地址基址的[39:24] |
| 15:0 | ht_prefetch1_ mask[39:24] | 16 | 0x0 | R/W | HT 总线可预取地址窗口 1，地址屏蔽的[39:24] |

## 10.5.13 UNCACHE 地址窗口配置寄存器

地址窗口命中公式详见 10.5.7 节。

本窗口的地址是 HT 总线上接收到的地址。落在本窗口地址的读写命令，将不会被送往 SCACHE，也不会使一级 CACHE 发生失效，而是会被直接送至内存或是其它的地址空间，也即该地址窗口中的读写命令将不会维持 IO 的 CACHE 一致性。该窗口主要针对一些不会在 CACHE 中命中所以可以提高存问效率的操作，如显存的访问等。

偏移量： 0xf0

复位值： 0x00000000

名称： HT 总线 Uncache 地址窗口 0 使能（内部访问）

表 10-51 HT 总线 Uncache 地址窗口 0 使能（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_uncache0_en | 1 | 0x0 | R/W | HT 总线 uncache 地址窗口 0，使能信号 |
| 30 | ht_uncache0_ trans_en | 1 | 0x0 | R/W | HT 总线 uncache 地址窗口 1，映射使能信号 |
| 29:0 | ht_uncache0_ trans[53:24] | 16 | 0x0 | R/W | HT 总线 uncache 地址窗口 0，转译后地址的[53:24] |

偏移量： 0xf4

复位值： 0x00000000

名称： HT 总线 Uncache 地址窗口 0 基址（内部访问）

表 10-52 HT 总线 Uncache 地址窗口 0 基址（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_uncache0_base[39:24] | 16 | 0x0 | R/W | HT 总线 uncache 地址窗口 0，地址基址的[39:24] |
| 15:0 | ht_uncache0_mask[39:24] | 16 | 0x0 | R/W | HT 总线 uncache 地址窗口 0，地址屏蔽的[39:24] |

偏移量： 0xf8

复位值： 0x00000000

名称： HT 总线 Uncache 地址窗口 1 使能（内部访问）

表 10-53 HT 总线 Uncache 地址窗口 1 使能（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | ht_uncache1_en | 1 | 0x0 | R/W | HT 总线 uncache 地址窗口 1，使能信号 |
| 30 | ht_uncache1_trans_en | 1 | 0x0 | R/W | HT 总线 uncache 地址窗口 1，映射使能信号 |
| 29:0 | ht_uncache1_trans[53:24] | 16 | 0x0 | R/W | HT 总线 uncache 地址窗口 1，转译后地址的[53:24] |

偏移量： 0xfc

复位值： 0x00000000

名称： HT 总线 Uncache 地址窗口 1 基址（内部访问）

表 10-54 HT 总线 Uncache 地址窗口 1 基址（内部访问）

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:16 | ht_uncache1_base[39:24] | 16 | 0x0 | R/W | HT 总线 uncache 地址窗口 1，地址基址的[39:24] |
| 15:0 | ht_uncache1_mask[39:24] | 16 | 0x0 | R/W | HT 总线 uncache 地址窗口 1，地址屏蔽的[39:24] |

## 10.5.14 命令发送缓存大小寄存器

命令发送缓存大小寄存器用于观测发送端可用各个命令通道的缓存个数。

偏移量：　　0x100

复位值：　　0x00000000

名称：　　命令发送缓存大小寄存器

表 10-55 命令发送缓存大小寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:24 | B_CMD_txbuffer | 8 | 0x0 | R | 发送端 B 通道命令缓存个数 |
| 2316 | R_CMD_txbuffer | 8 | 0x0 | R | 发送端 R 通道命令缓存个数 |
| 15:8 | NPC_CMD_txbuffer | 8 | 0x0 | R | 发送端 NPC 通道命令缓存个数 |
| 7:0 | PC_CMD_txbuffer | 8 | 0x0 | R | 发送端 PC 通道命令缓存个数 |

## 10.5.15 数据发送缓存大小寄存器

数据发送缓存大小寄存器用于观测发送端可用各个数据通道的缓存个数。

偏移量：　　0x104

复位值：　　0x00000000

名称：　　数据发送缓存大小寄存器

表 10-56 数据发送缓存大小寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31:24 | Reserved | 8 | 0x0 | R | 保留 |
| 2316 | R_DATA_txbuffer | 8 | 0x0 | R | 发送端 R 通道数据缓存个数 |
| 15:8 | NPC_DATA_txbuffer | 8 | 0x0 | R | 发送端 NPC 通道数据缓存个数 |
| 7:0 | PC_DATA_txbuffer | 8 | 0x0 | R | 发送端 PC 通道数据缓存个数 |

## 10.5.16 发送缓存调试寄存器

发送缓存调试寄存器用于人工设置 HT 控制器发送端缓冲区的个数，通过增或减的方式对不同的发送缓存个数进行调整。

偏移量：　　0x108

复位值：　　0x00000000

名称：　　发送缓存调试寄存器

表 10-57 发送缓存调试寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 31 | B_interleave | 1 | 0x0 | R/W | 一致性写响应通道交错使能 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | （片间互连模式时必须设置） |
| 30 | NOP_interleave | 1 | 0x0 | R/W | 流控包交错使能（片间互连模式时必须设置） |
| 29 | Tx_neg | 1 | 0x0 | R/W | 发送端缓存调试符号<br>0：增加相应个数<br>1：减少（相应寄存器个数+1）个 |
| 28 | Tx_buff_adj_en | 1 | 0x0 | R/W | 发送端缓存调试使能寄存器<br>0->1：使本寄存器的值产生一次增减效果 |
| 27:24 | R_DATA_txadj | 4 | 0x0 | R/W | 发送端 R 通道数据缓存增减个数<br>当 tx_neg 为 0 时，增加 R_DATA_txadj 个；<br>当 tx_neg 为 1 时，减少 R_DATA_txadj+1 个 |
| 23:20 | NPC_DATA_txadj | 4 | 0x0 | R/W | 发送端 NPC 通道数据缓存增减个数<br>当 tx_neg 为 0 时，增加 NPC_DATA_txadj 个；<br>当 tx_neg 为 1 时，减少 NPC_DATA_txadj+1 个 |
| 19:16 | PC_DATA_txadj | 4 | 0x0 | R/W | 发送端 PC 通道数据缓存增减个数<br>当 tx_neg 为 0 时，增加 PC_DATA_txadj 个；<br>当 tx_neg 为 1 时，减少 PC_DATA_txadj+1 个 |
| 15:12 | B_CMD_txadj | 4 | 0x0 | R/W | 发送端 B 通道命令缓存增减个数<br>当 tx_neg 为 0 时，增加 B_CMD_txadj 个；<br>当 tx_neg 为 1 时，减少 B_CMD_txadj+1 个 |
| 11:8 | R_CMD_txadj | 4 | 0x0 | R/W | 发送端 R 通道命令缓存增减个数<br>当 tx_neg 为 0 时，增加 R_CMD_txadj 个；<br>当 tx_neg 为 1 时，减少 R_CMD_txadj+1 个 |
| 7:4 | NPC_CMD_txadj | 4 | 0x0 | R/W | 发送端 NPC 通道命令/数据缓存增减个数<br>当 tx_neg 为 0 时，增加 NPC_CMD_txadj 个；<br>当 tx_neg 为 1 时，减少 NPC_CMD_txadj+1 个 |
| 3:0 | PC_CMD_txadj | 4 | 0x0 | R/W | 发送端 PC 通道命令缓存增减个数<br>当 tx_neg 为 0 时，增加 PC_CMD_txadj 个；<br>当 tx_neg 为 1 时，减少 PC_CMD_txadj+1 个 |

## 10.5.17 预加重和输出阻抗寄存器

调整 TX 端预加重和输出阻抗

偏移量： 0x10c

复位值： 0x00000030

名称：　　预加重和输出阻抗寄存器

表 10-58 预加重和输出阻抗寄存器

| 位域 | 位域名称 | 位宽 | 复位值 | 访问 | 描述 |
|---|---|---|---|---|---|
| 7 | csr_phy_tx_comp_ok | 1 | 0x0 | R | 1：输出阻抗自动校准完成 |
| 6 | csr_phy_tx_code_en | 1 | 0x1 | R/W | 0：输出阻抗自动校准<br>1：输出阻抗人工校准 |
| 5:4 | csr_phy_tx_ncode | 2 | 0x3 | R/W | 下拉阻抗<br>csr_phy_tx_code_en=0：not care<br>csr_phy_tx_code_en=1：人工设置 |
| 3:2 | csr_phy_tx_pcode | 2 | 0x0 | R/W | 上拉阻抗<br>csr_phy_tx_code_en=0：not care<br>csr_phy_tx_code_en=1：人工设置 |
| 1:0 | csr_phy_tx_de_tap | 2 | 0x0 | R/W | 预加重<br>00：　normal<br>01 or 10: 3dB<br>11：　6dB |

## 10.5.18 HyperTransport 总线配置空间的访问方法

HyperTransport 接口软件层的协议与 PCI 协议基本一致，由于配置空间的访问直接与底层协议相关，具体访问细节略有不同。在表 9-5 中已列出，HT 总线配置空间的地址范围是 0xFD_FE00_0000 ～0xFD_FFFF_FFFF。对于 HT 协议中的配置访问，在龙芯 3B1500 中采用如下格式实现。

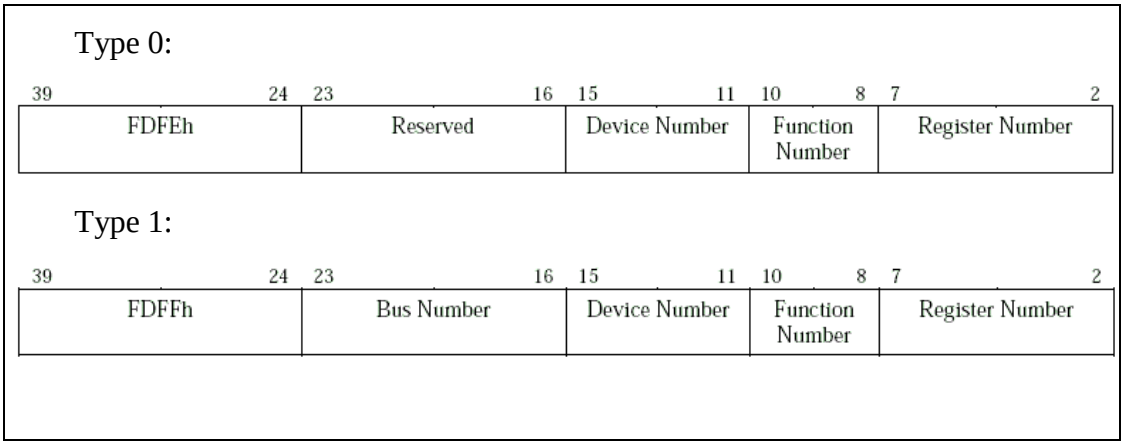



图 10-1 龙芯 3B1500 中 HT 协议的配置访问

## 10.6 HyperTransport 多处理器支持

龙芯 3B1500 处理器使用 HyperTransport 接口进行多处理器互联，并且硬件可以自动维护 2 个芯片之间的一致性请求。下面介绍两片龙芯 3B1500 处理器的互联方法。

### 两片龙芯 3 号互联结构

根据固定路由算法的特性，我们在构建两片处理器互联时，有两种不同的方法。首先是采用 8 位 HT 总线互联。在这种互联方式下，两个处理器之间只能采用 8 位 HT 互联。如下所示：

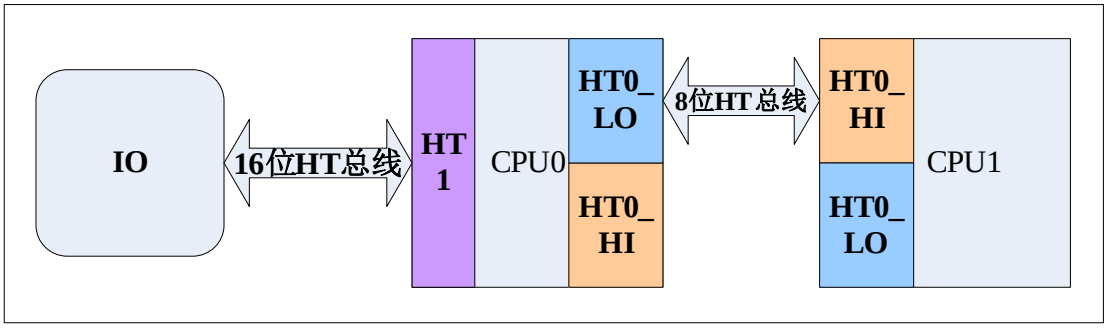



图 10-2 两片龙芯 3B1500 8 位互联结构

此外，龙芯 3B1500 的 HT 总线最高可以采用 16 位宽度，因此还可以采用最大化带宽的 16 位互联方式。将龙芯 3B1500 中的 HT0 控制器设置为 16 位模式后，所有发到 HT0 控制器的命令都会被发往 HT0_LO，而不会像第一种互联方式一样按照路由表分别发至 HT0_HI 或是 HT0_LO。所以只需正确配置 CPU0 与 CPU1 的 16 位模式，并正确连接高低位总线，即可使用 16 位的 HT 总线互联方式。同时，该互联结构也支持使用 8 位的 HT 总线协议进行相互访问。该互联结构如下图所示：

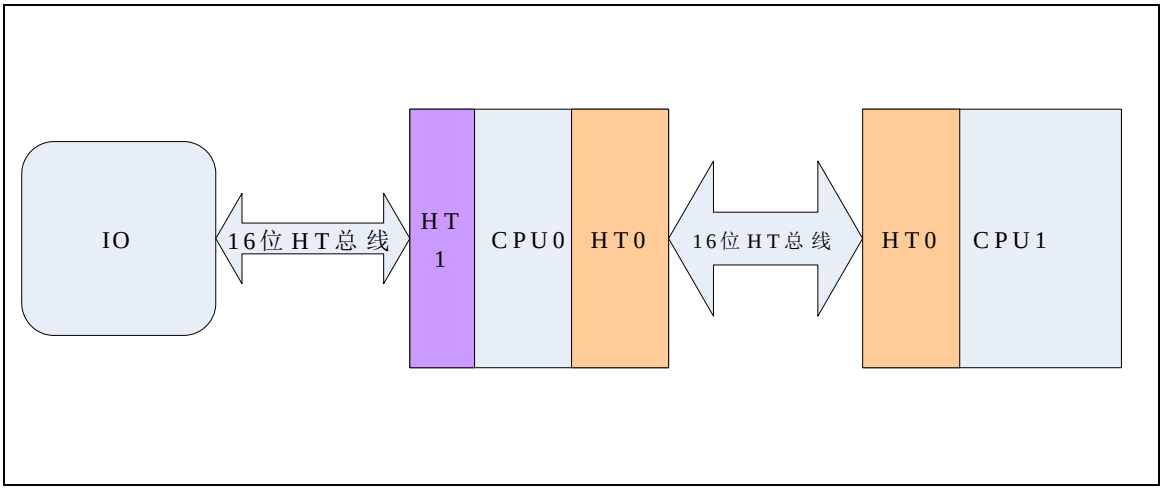



图 10-3 两片龙芯 3B1500 16 位互联结构

# 11 低速 IO 控制器配置

龙芯 3B1500 I/O 控制器包括 PCI/PCI-X 控制器、LPC 控制器、UART 控制器、SPI 控制器、GPIO 以及配置寄存器。这些 I/O 控制器共享一个交叉开关端口，CPU 的请求经过地址译码后发送到相应的设备。

## 11.1 PCI/PCI-X 控制器

龙芯 3B1500 的 PCI/PCI-X 控制器既可以作为主桥控制整个系统，也可以作为普通 PCI/PCI-X 设备工作在总线上。龙芯 3B1500 的 PCI/PCI-X 控制器内置 PCI 仲裁器。

PCI/PCI-X 控制器的配置头位于 0x1FE00000 开始的 256 字节，如表 11-1 所示。

表 11-1 PCIX 控制器配置头

| 字节 3 | 字节 2 | 字节 1 | 字节 0 | 地址 |
|---|---|---|---|---|
| Device ID | | Vendor ID | | 00 |
| Status | | Command | | 04 |
| Class Code | | | Revision ID | 08 |
| BIST | Header Type | Latency Timer | CacheLine Size | 0C |
| Base Address Register 0 | | | | 10 |
| Base Address Register 1 | | | | 14 |
| Base Address Register 2 | | | | 18 |
| Base Address Register 3 | | | | 1C |
| Base Address Register 4 | | | | 20 |
| Base Address Register 5 | | | | 24 |
| | | | | 28 |
| Subsystem ID | | Subsystem Vendor ID | | 2C |
| | | | | 30 |
| | | | Capabilities Pointer | 34 |
| | | | | 38 |
| Maximum Latency | Minimum Grant | Interrupt Pin | Interrupt Line | 3C |
| Implementation Specific Register(ISR40) | | | | 40 |
| Implementation Specific Register(ISR44) | | | | 44 |
| Implementation Specific Register(ISR48) | | | | 48 |
| Implementation Specific Register(ISR4C) | | | | 4C |
| Implementation Specific Register(ISR50) | | | | 50 |
| Implementation Specific Register(ISR54) | | | | 54 |
| Implementation Specific Register(ISR58) | | | | 58 |
| | | | | ... |
| PCIX Command Register | | | | E0 |
| PCIX Status Register | | | | E4 |

龙芯 3B1500 的 PCIX 控制器支持三个 64 位窗口，由{BAR1, BAR0}、{BAR3, BAR2}、{BAR5, BAR4}三对寄存器配置窗口 0、1、2 的基址。窗口的大小、使能以及其它细节由另外三个对应寄存器 PCI_Hit0_Sel, PCI_Hit1_Sel, PCI_Hit2_Sel 控制，具体位域请参见表 11-2。

表 11-2 PCI 控制寄存器

| 位域 | 字段名 | 访问 | 复位值 | 说明 |
|---|---|---|---|---|
| REG_40 | | | | |
| 31 | tar_read_io | 读写<br>(写 1 清) | 0 | target 端收到对 IO 或者是不可预取区域的访问 |
| 30 | tar_read_discard | 读写<br>(写 1 清) | 0 | target 端的 delay 请求被丢弃 |
| 29 | tar_resp_delay | 读写 | 0 | target 访问何时给出 delay/split<br>0: 超时后<br>1: 马上 |
| 28 | tar_delay_retry | 读写 | 0 | target 访问重试策略<br>0: 根据内部逻辑（见 29 位）<br>1: 马上重试 |
| 27 | tar_read_abort_en | 读写 | 0 | 若 target 对内部的读请求超时，是否允许 target-abort 回应 |
| 26:25 | Reserved | - | 0 | |
| 24 | tar_write_abort_en | 读写 | 0 | 若 target 对内部的写请求超时，是否允许 target-abort 回应 |
| 23 | tar_master_abort | 读写 | 0 | 是否允许 master-abort |
| 22:20 | tar_subseq_timeout | 读写 | 000 | target 后续延迟超时<br>000: 8 周期<br>其它: 不支持 |
| 19:16 | tar_init_timeout | 读写 | 0000 | target 初始延迟超时<br>PCI 模式下<br>0: 16 周期<br>1-7: 禁用计数器<br>8-15: 8-15 周期<br>PCIX 模式下超时计数固定为 8 周期，此处配置影响最大的 delay 访问数<br>0: 8 delay 访问<br>8: 1 delay 访问<br>9: 2 delay 访问<br>10: 3 delay 访问<br>11: 4 delay 访问<br>12: 5 delay 访问<br>13: 6 delay 访问<br>14: 7 delay 访问<br>15: 8 delay 访问 |
| 15:4 | tar_pref_boundary | 读写 | 000h | 可预取边界配置（以 16 字节为单位）<br>FFF: 64KB 到 16byte |

| | | | | |
|---|---|---|---|---|
| | | | | FFE: 64KB 到 32byte<br>FF8: 64KB 到 128byte |
| 3 | tar_pref_bound_en | 读写 | 0 | 使用 tar_pref_boundary 的配置<br>0: 预取到设备边界<br>1: 使用 tar_pref_boundary |
| 2 | Reserved | - | 0 | |
| 1 | tar_splitw_ctrl | 读写 | 0 | target split 写控制<br>0: 阻挡除 Posted Memory Write 以外的访问<br>1: 阻挡所有访问，直至 split 完成 |
| 0 | mas_lat_timeout | 读写 | 0 | 禁用 mater 访问超时<br>0: 允许 master 访问超时<br>1: 不允许 |
| REG_44 | | | | |
| 31:0 | Reserved | - | - | |
| REG_48 | | | | |
| 31:0 | tar_pending_seq | 读写 | 0 | target 未处理完的请求号位向量<br>对应位写 1 可清 |
| REG_4C | | | | |
| 31:30 | Reserved | - | - | |
| 29 | mas_write_defer | 读写 | 0 | 允许后续的读越过前面未完成的写<br>(只对 PCI 有效) |
| 28 | mas_read_defer | 读写 | 0 | 允许后续的读写越过前面未完成的读<br>(只对 PCI 有效) |
| 27 | mas_io_defer_cnt | 读写 | 0 | 在外的最大 IO 请求数<br>0: 由控制<br>1: 1 |
| 26:24 | mas_read_defer_cnt | 读写 | 010 | master 支持在外读的最大数(只对 PCI 有效）<br>0: 8<br>1-7: 1-7<br>注：一个双地址周期访问占两项 |
| 23:16 | err_seq_id | 只读 | 00h | target/master 错误号 |
| 15 | err_type | 只读 | 0 | target/master 出错的命令类型<br>0: |
| 14 | err_module | 只读 | 0 | 出错的模块<br>0: target<br>1: master |
| 13 | system_error | 读写 | 0 | target/master 系统错（写 1 清） |
| 12 | data_parity_error | 读写 | 0 | target/master 数据奇偶错（写 1 清） |
| 11 | ctrl_parity_error | 读写 | 0 | target/master 地址奇偶错（写 1 清） |
| 10:0 | Reserved | - | - | |
| REG_50 | | | | |
| 31:0 | mas_pending_seq | 读写 | 0 | master 未处理完的请求号位向量 |

| | | | | |
|---|---|---|---|---|
| | | | | 对应位写 1 可清 |
| REG_54 | | | | |
| 31:0 | mas_split_err | 读写 | 0 | split 返回出错的请求号位向量 |
| REG_58 | | | | |
| 31:30 | Reserved | - | - | |
| 29:28 | tar_split_priority | 读写 | 0 | target split 返回优先级<br>0 最高, 3 最低 |
| 27:26 | mas_req_priority | 读写 | 0 | master 对外的优先级<br>0 最高, 3 最低 |
| 25 | Priority_en | 读写 | 0 | 仲裁算法（在 master 的访问和 target 的 split 返回间做仲裁）<br>0: 固定优先级<br>1: 轮转 |
| 24:18 | 保留 | - | - | |
| 17 | mas_retry_aborted | 读写 | 0 | master 重试取消（写 1 清） |
| 16 | mas_trdy_timeout | 读写 | 0 | master TRDY 超时计数 |
| 15:8 | mas_retry_value | 读写 | 00h | master 重试次数<br>0: 无限重试<br>1-255: 1-255 次 |
| 7:0 | mas_trdy_count | 读写 | 00h | master TRDY 超时计数器<br>0: 禁用<br>1-255: 1-255 拍 |

在发起配置空间读写前，应用程序应先配置好 PCIMap_Cfg 寄存器，告诉控制器欲发起的配置操作的类型和高 16 位地址线上的值。然后对 0x1fe80000 开始的 2K 空间进行读写即可访问对应设备的配置头。设备号根据 PCIMap_Cfg[15:0]从低到高优先编码得到。

配置操作地址生成见图 11-1。

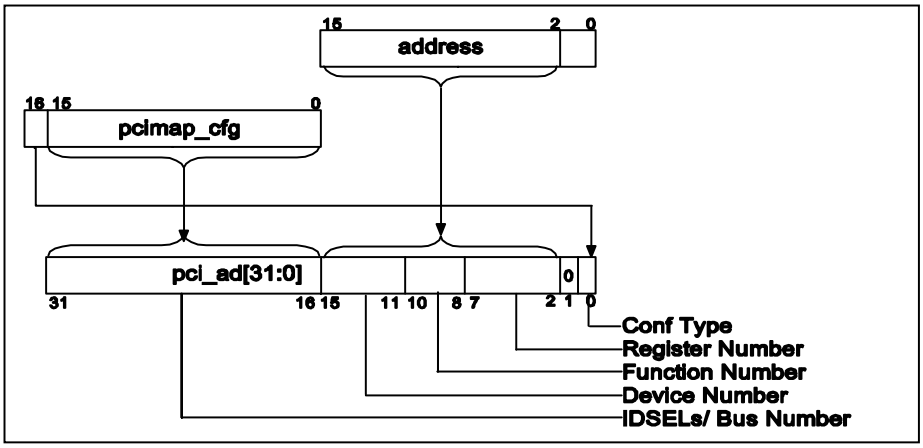



图 11-1 配置读写总线地址生成

PCI/PCIX 仲裁器实现了两级轮转仲裁、总线停靠和损坏主设备的隔离。其配置和状态见 PXArb_Config 和 PXArb_Status 寄存器。PCI/PCIX 总线请求与应答线分配见表 11-3。

表 11-3 PCI/PCIX 总线请求与应答线分配

| 请求与应答线 | 描述 |
|---|---|
| 0 | 内部集成 PCI/PCIX 控制器 |
| 7:1 | 外部请求 6~0 |

基于轮转的仲裁算法提供两个级别，第二级整体作为第一级中的一员一起调度。当多个设备同时申请总线时每轮转完一次第一级设备，第二级中优先级最高的设备可以得到总线。

仲裁器被设计成任何时候只要条件允许就可以切换，对于一些不符合协议的 PCI 设备，这样做可能会使之不正常。使用强制优先级可以让这些设备通过持续请求来占有总线。

总线停靠是指当没有设备请求使用总线时是否选择其中一个给出允许信号。对于已经得到允许的设备而言，直接发起总线操作能够提高效率。龙芯 2F 的 PCI 仲裁器提供两种停靠模式：最后一个主设备和默认主设备。如果在特殊场合下不能够停靠，可以将仲裁器设置为停靠到默认 0 号主设备（内部控制器），且依靠延迟为 0。

## 11.2 LPC 控制器

LPC 控制器具有以下特性：

- 符合 LPC1.1 规范
- 支持 LPC 访问超时计数器
- 支持 Memory Read、Memory write 访问类型
- 支持 Firmware Memory Read、Firmware Memory Write 访问类型（单字节）
- 支持 I/O read、I/O write 访问类型
- 支持 Memory 访问类型地址转换
- 支持 Serizlized IRQ 规范，提供 17 个中断源

LPC 控制器的地址空间分布见表 11-4：

表 11-4 LPC 控制器地址空间分布

| 地址名称 | 地址范围 | 大小 |
|---|---|---|
| LPC Boot | 0X1FC0_0000-0X1FD0_0000 | 1MByte |
| LPC Memory | 0X1C00_0000-0X1E00_0000 | 32MByte |
| LPC I/O | 0X1FF0_0000-0X1FF1_0000 | 64KByte |
| LPC Register | 0X1FE0_0200-0X1FE0_0300 | 256Byte |

LPC Boot 地址空间是系统启动时处理器最先访问的地址空间。这个地址空间支持 LPC Memory 或 Firmware Memory 访问类型。系统启动时发出哪种访问类型由 LPC_ROM_INTEL 引脚控制。LPC_ROM_INTEL 引脚上拉时发出 LPC Firmware Memory 访问，LPC_ROM_INTEL 引脚下拉时发出 LPC Memory 访问类型。

LPC Memory 地址空间是系统用 Memory/Firmware Memory 访问的地址空间。LPC 控制器发出哪种类型的 Memory 访问，由 LPC 控制器的配置寄存器 LPC_MEM_IS_FWH 决定。处理器发往这个地址空间的地址可以进行地址转换。转换后的地址由 LPC 控制器的配置寄存器 LPC_MEM_TRANS 设置。

处理器发往 LPC I/O 地址空间的访问按照 LPC I/O 访问类型发往 LPC 总线。地址为地址空间低 16 位。

LPC 控制器配置寄存器共有 3 个 32 位寄存器，基地址为 0x1FE0_0200。配置寄存器的含义见表 11-5：

表 11-5 LPC 配置寄存器含义

| 位域 | 字段名 | 访问 | 复位值 | 说明 |
|---|---|---|---|---|
| REG0 | | | | |
| REG0[31:31] | SIRQ_EN | 读写 | 0 | SIRQ 使能控制 |
| REG0[23:23] | LPC_MEM_TRANS_EN | 读写 | 0 | LPC Memory 空间地址转换控制使能控制 |
| REG0[22:16] | LPC_MEM_TRANS | 读写 | 0 | LPC Memory 空间地址转换控制，对应于 LPC 总线地址的[31:25] |
| REG0[15:0] | LPC_SYNC_TIMEOUT | 读写 | 0 | LPC 访问超时计数器 |
| REG1 | | | | |
| REG1[31:31] | LPC_MEM_IS_FWH | 读写 | 0 | LPC Memory 空间 Firmware Memory 访问类型设置 |
| REG1[17:0] | LPC_INT_EN | 读写 | 0 | LPC SIRQ 中断使能 |
| REG2 | | | | |
| REG2[17:0] | LPC_INT_SRC | 读写 | 0 | LPC SIRQ 中断源指示 |
| REG3 | | | | |
| REG3[17:0] | LPC_INT_CLEAR | 写 | 0 | LPC SIRQ 中断清除 |

## 11.3 UART 控制器

UART 控制器具有以下特性

- 全双工异步数据接收/发送
- 可编程的数据格式
- 16 位可编程时钟计数器

- 支持接收超时检测
- 带仲裁的多中断系统
- 仅工作在 FIFO 方式
- 在寄存器与功能上兼容 NS16550A

本模块有两个并行工作 UART 接口，功能寄存器完全一样，只是访问基址不同。

UART0 寄存器物理地址基址为 0x1FE001E0。

UART1 寄存器物理地址基址为 0x1FE001E8。

## 11.3.1 数据寄存器（DAT）

中文名： 数据传输寄存器

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0x00

表 11-6 数据传输寄存器定义

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:0 | Tx FIFO | 8 | W | 数据传输寄存器 |

## 11.3.2 中断使能寄存器（IER）

中文名： 中断使能寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x00

表 11-7 中断使能寄存器定义

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:4 | Reserved | 4 | RW | 保留 |
| 3 | IME | 1 | RW | Modem 状态中断使能 ‘0’ – 关闭 ‘1’ – 打开 |
| 2 | ILE | 1 | RW | 接收器线路状态中断使能 ‘0’ – 关闭 ‘1’ – 打开 |
| 1 | ITxE | 1 | RW | 传输保存寄存器为空中断使能 ‘0’ – 关闭 ‘1’ – 打开 |
| 0 | IRxE | 1 | RW | 接收有效数据中断使能 ‘0’ – 关闭 ‘1’ – 打开 |

## 11.3.3 中断标识寄存器（IIR）

中文名： 中断源寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0xc1

表 11-8 中断源寄存器定义

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:4 | Reserved | 4 | R | 保留 |
| 3:1 | II | 3 | R | 中断源标志位，详见下表 |
| 0 | INTp | 1 | R | 中断标志位 |

表 11-9 中断控制功能表

| Bit 3 | Bit 2 | Bit 1 | 优先级 | 中断类型 | 中断源 | 中断复位控制 |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1st | 接收线路状态 | 奇偶、溢出或帧错误，或打断中断 | 读 LSR |
| 0 | 1 | 0 | 2nd | **接收到有效数据** | FIFO 的字符个数达到 trigger 的水平 | FIFO 的字符个数低于 trigger 的值 |
| 1 | 1 | 0 | 2nd | 接收超时 | 在 FIFO 至少有一个字符，但在 4 个字符时间内没有任何操作，包括读和写操作 | 读接收 FIFO |
| 0 | 0 | 1 | 3rd | 传输保存寄存器为空 | 传输保存寄存器为空 | 写数据到 THR 或者多 IIR |
| 0 | 0 | 0 | 4th | Modem 状态 | CTS, DSR, RI or DCD. | 读 MSR |

## 11.3.4 FIFO 控制寄存器（FCR）

中文名： FIFO 控制寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0xc0

表 11-10 FIFO 控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:6 | TL | 2 | W | 接收 FIFO 提出中断申请的 trigger 值<br>‘00’ – 1 字节 ‘01’ – 4 字节<br>‘10’ – 8 字节 ‘11’ – 14 字节 |
| 5:3 | Reserved | 3 | W | 保留 |
| 2 | Txset | 1 | W | ‘1’ 清除发送 FIFO 的内容，复位其逻辑 |
| 1 | Rxset | 1 | W | ‘1’ 清除接收 FIFO 的内容，复位其逻辑 |

| | | | | |
|---|---|---|---|---|
| 0 | Reserved | 1 | W | 保留 |

## 11.3.5 线路控制寄存器（LCR）

中文名： 线路控制寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x03

表 11-11 线路控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7 | dlab | 1 | RW | 分频锁存器访问位<br>‘1’ – 读写分频锁存器<br>‘0’ – 读写正常寄存器 |
| 6 | bcb | 1 | RW | 打断控制位<br>‘1’ – 此时串口的输出被置为 0(打断状态).<br>‘0’ – 正常操作 |
| 5 | spb | 1 | RW | 指定奇偶校验位<br>‘0’ – 不用指定奇偶校验位<br>‘1’ – 如果 LCR[4]位是 1 则传输和检查奇偶校验位为 0。如果 LCR[4]位是 0 则传输和检查奇偶校验位为 1。 |
| 4 | eps | 1 | RW | 奇偶校验位选择<br>‘0’ – 在每个字符中有奇数个 1（包括数据和奇偶校验位）<br>‘1’ –在每个字符中有偶数个 1 |
| 3 | pe | 1 | RW | 奇偶校验位使能<br>‘0’ – 没有奇偶校验位<br>‘1’ – 在输出时生成奇偶校验位，输入则判断奇偶校验位 |
| 2 | sb | 1 | RW | 定义生成停止位的位数<br>‘0’ – 1 个停止位<br>‘1’ – 在 5 位字符长度时是 1.5 个停止位，其他长度是 2 个停止位 |

| | | | | |
|---|---|---|---|---|
| 1:0 | bec | 2 | RW | 设定每个字符的位数<br>‘00’ – 5 位 ‘01’ – 6 位<br>‘10’ – 7 位 ‘11’ – 8 位 |

## 11.3.6 MODEM 控制寄存器（MCR）

中文名： Modem 控制寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x00

表 11-12 Modem 控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:5 | Reserved | 3 | W | 保留 |
| 4 | Loop | 1 | W | 回环模式控制位<br>‘0’ – 正常操作<br>‘1’ –回环模式。在在回环模式中，TXD 输出一直为 1，输出移位寄存器直接连到输入移位寄存器中。其他连接如下。<br>DTR DSR<br>RTS CTS<br>Out1 RI<br>Out2 DCD |
| 3 | OUT2 | 1 | W | 在回环模式中连到 DCD 输入 |
| 2 | OUT1 | 1 | W | 在回环模式中连到 RI 输入 |
| 1 | RTSC | 1 | W | RTS 信号控制位 |
| 0 | DTRC | 1 | W | DTR 信号控制位 |

## 11.3.7 线路状态寄存器（LSR）

中文名： 线路状态寄存器

寄存器位宽： [7：0]

偏移量： 0x05

复位值： 0x00

表 11-13 线路状态寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7 | ERROR | 1 | R | 错误表示位<br>‘1’ – 至少有奇偶校验位错误，帧错误或打断中断 |

| | | | | |
|---|---|---|---|---|
| | | | | 的一个。<br>‘0’ – 没有错误 |
| 6 | TE | 1 | R | 传输为空标志位<br>‘1’ – 传输 FIFO 和传输移位寄存器都为空。给传输 FIFO 写数据时清零<br>‘0’ – 有数据 |
| 5 | TFE | 1 | R | 传输 FIFO 为空标志位<br>‘1’ – 当前传输 FIFO 为空，给传输 FIFO 写数据时清零<br>‘0’ – 有数据 |
| 4 | BI | 1 | R | 打断中断标志位<br>‘1’ –接收到 起始位＋数据＋奇偶位＋停止位都是 0，即有打断中断<br>‘0’ – 没有打断 |
| 3 | FE | 1 | R | 帧错误标志位<br>‘1’ – 接收的数据没有停止位<br>‘0’ – 没有错误 |
| 2 | PE | 1 | R | 奇偶校验位错误标志位<br>‘1’ – 当前接收数据有奇偶错误<br>‘0’ – 没有奇偶错误 |
| 1 | OE | 1 | R | 数据溢出标志位<br>‘1’ – 有数据溢出<br>‘0’ – 无溢出 |
| 0 | DR | 1 | R | 接收数据有效标志位<br>‘0’ – 在 FIFO 中无数据<br>‘1’ – 在 FIFO 中有数据 |

对这个寄存器进行读操作时，LSR[4:1]和 LSR[7]被清零，LSR[6:5]在给传输 FIFO 写数据时清零，LSR[0]则对接收 FIFO 进行判断。

## 11.3.8 MODEM 状态寄存器 （MSR）

中文名： Modem 状态寄存器

寄存器位宽： [7：0]

偏移量： 0x06

复位值： 0x00

表 11-14 Modem 状态寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 7 | CDCD | 1 | R | DCD 输入值的反，或者在回环模式中连到 Out2 |
| 6 | CRI | 1 | R | RI 输入值的反，或者在回环模式中连到 OUT1 |
| 5 | CDSR | 1 | R | DSR 输入值的反，或者在回环模式中连到 DTR |
| 4 | CCTS | 1 | R | CTS 输入值的反，或者在回环模式中连到 RTS |
| 3 | DDCD | 1 | R | DDCD 指示位 |
| 2 | TERI | 1 | R | RI 边沿检测。RI 状态从低到高变化 |
| 1 | DDSR | 1 | R | DDSR 指示位 |
| 0 | DCTS | 1 | R | DCTS 指示位 |

### 11.3.9 分频锁存器

中文名：　　分频锁存器 1

寄存器位宽：　[7：0]

偏移量：　　0x00

复位值：　　0x00

表 11-15 分频锁存器 1

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:0 | LSB | 8 | RW | 存放分频锁存器的低 8 位 |

中文名：　　分频锁存器 2

寄存器位宽：　[7：0]

偏移量：　　0x01

复位值：　　0x00

表 11-16 分频锁存器 2

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:0 | MSB | 8 | RW | 存放分频锁存器的高 8 位 |

## 11.4 SPI Flash 控制器

串行外围设备接口 SPI 总线技术是 Motorola 公司推出的多种微处理器、微控制器以及外围设备之间的一种全双工、同步、串行数据接口标准。

SPI 控制器具有以下特性：

- 全双工同步串口数据传输
- 支持到 4 个的变长字节传输
- 主模式支持
- 极性和相位可编程的串行时钟
- 支持系统启动

- 支持标准读、连续地址读、快速读、双路 I/O 等 SPI Flash 读模式

龙芯 3A 中 SPI 控制器模块寄存器物理地址基址为 0x1FE001F0。

龙芯 3B1500 中 SPI 控制器模块寄存器物理地址基址为 0x1FE00220。

## 11.4.1 控制寄存器（SPCR）

中文名： 控制寄存器

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0x10

表 11-17 控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7 | Spie | 1 | RW | 中断输出使能信号 高电平有效 |
| 6 | spe | 1 | RW | 系统工作使能信号高电平有效 |
| 5 | Reserved | 1 | RW | 保留 |
| 4 | mstr | 1 | RW | master 模式选择位，此位一直保持 1 |
| 3 | cpol | 1 | RW | 时钟极性位 |
| 2 | cpha | 1 | RW | 时钟相位，为 1 位则相位相反，为 0 则相同 |
| 1:0 | spr | 2 | RW | sclk_o 分频设定，需要与 sper 的 spre 一起使用 |

## 11.4.2 状态寄存器（SPSR）

中文名： 状态寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x05

表 11-18 状态寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7 | spif | 1 | RW | 中断标志，1 表示有中断申请，写 1 则清零 |
| 6 | wcol | 1 | RW | 写寄存器溢出标志位，1 表示已经溢出,写 1 则清零 |
| 5:4 | Reserved | 2 | RW | 保留 |
| 3 | wffull | 1 | RW | 写寄存器满标，1 表示已经满 |
| 2 | wfempty | 1 | RW | 写寄存器空标志，1 表示空 |
| 1 | rffull | 1 | RW | 读寄存器满标志，1 表示已经满 |
| 0 | rfempty | 1 | RW | 读寄存器空标志，1 表示空 |

## 11.4.3 数据寄存器（TxFIFO）

中文名： 数据传输寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x00

表 11-19 数据传输寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:0 | Tx FIFO | 8 | W | 数据传输寄存器 |

## 11.4.4 外部寄存器（SPER）

中文名： 外部寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

表 11-20 外部寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:6 | icnt | 2 | RW | 在传输完多少个字节后送出中断请求信号<br>00 -1 字节 01 - 2 字节<br>10 -3 字节 11 - 3 字节 |
| 5:3 | Reserved | 4 | RW | 保留 |
| 2 | mode | 1 | RW | spi 接口模式控制<br>0: 采样与发送时机同时<br>1: 采样与发送时机错开半周期 |
| 1:0 | spre | 2 | RW | 与 Spr 一起设定分频的比率 |

表 11-21 分频系数

| spre | 00 | 00 | 00 | 00 | 01 | 01 | 01 | 01 | 10 | 10 | 10 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| spr | 00 | 01 | 10 | 11 | 00 | 01 | 10 | 11 | 00 | 01 | 10 | 11 |
| 分频系数 | 2 | 4 | 16 | 32 | 8 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |

## 11.4.5 参数控制寄存器（SFC_PARAM）

中文名： SPI Flash 参数控制寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x21

表 11-22 SPI Flash 参数控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:4 | clk_div | 4 | RW | 时钟分频数选择（分频系数与{spre,spr}组合相同） |
| 3 | dual_io | 1 | RW | 使用双 I/O 模式，优先级高于快速读模式 |
| 2 | fast_read | 1 | RW | 使用快速读模式 |
| 1 | burst_en | 1 | RW | spi flash 支持连续地址读模式 |
| 0 | memory_en | 1 | RW | spi flash 读使能，无效时 csn[0]可由软件控制。 |

## 11.4.6 片选控制寄存器（SFC_SOFTCS）

中文名： SPI Flash 片选控制寄存器

寄存器位宽： [7：0]

偏移量： 0x05

复位值： 0x00

表 11-23 SPI Flash 片选控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:4 | csn | 4 | RW | csn 引脚输出值 |
| 3:0 | csen | 4 | RW | 为 1 时对应位的 cs 线由 7:4 位控制 |

## 11.4.7 时序控制寄存器（SFC_TIMING）

中文名： SPI Flash 时序控制寄存器

寄存器位宽： [7：0]

偏移量： 0x06

复位值： 0x03

表 11-24 SPI Flash 时序控制寄存器

| 位域 | 位域名称 | 位宽 | 访问 | 描述 |
|---|---|---|---|---|
| 7:2 | Reserved | 6 | RW | 保留 |
| 1:0 | tCSH | 2 | RW | SPI Flash 的片选信号最短无效时间，以分频后时钟周期 T 计算<br>00: 1T<br>01: 2T<br>10: 4T |

| | | | | |
|---|---|---|---|---|
| | | | | 11: 8T |

## 11.4.8 SPI Flash 控制器结构

串行外围设备接口 SPI 总线技术是 Motorola 公司推出的多种微处理器、微控制器以及外围设备之间的一种全双工、同步、串行数据接口标准。

SPI Flash 控制器是在一个简单 SPI 控制器的基础上增加专门的硬件逻辑，使得控制器对外(软件)除了有若干 IO 寄存器外还有一段只读 memory 空间。这段 memory 空间映射到 SPI Flash 中，复位后不需要软件干预就可以直接访问。从而支持处理器从 SPI Flash 启动。

本模块结构如下图所示，由 AXI 接口、简单的 SPI 主控制器、SPI Flash 读引擎和总线选择模块组成。根据访问的地址和类型，AXI 上的合法请求转发到 SPI 主控制器或者 SPI Flash 读引擎中(非法请求被丢弃)。

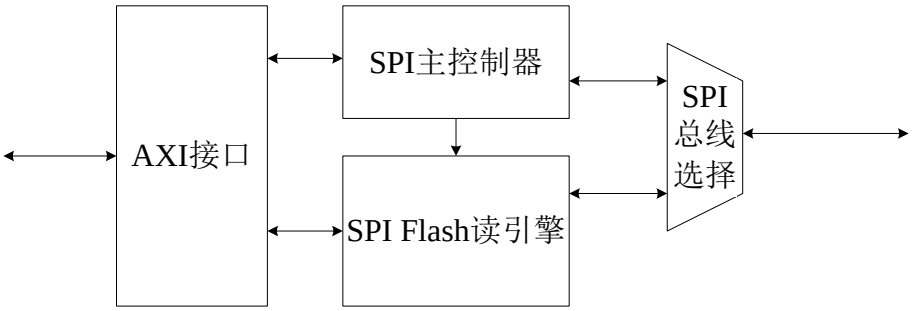



图 11-2 SPI Flash 控制器结构

SPI 主控制器只接收单字节的读写访问，实现了 SPI 主设备的功能，并为 SPI Flash 读引擎提供参数配置。SPI 主控制器的结构如图 11-3 所示。系统寄存器包括控制寄存器，状态寄存器、外部寄存器和 SPI Flash 相关控制寄存器。分频器生成 SPI 总线工作的时钟信号，数据读、写缓冲器（FIFO）允许 SPI 同时进行串行发送和接收数据。

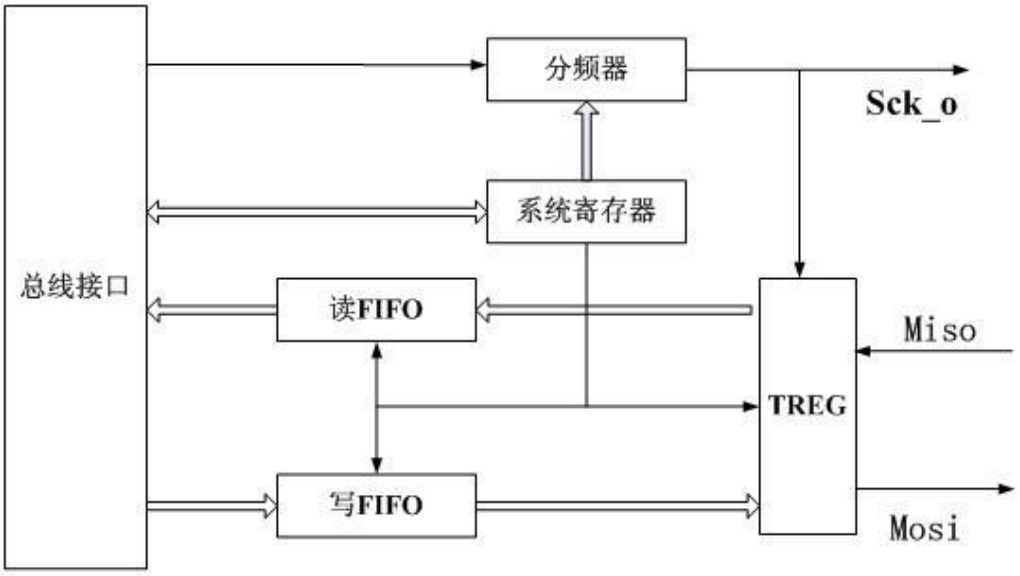



图 11-3 SPI 主控制器结构

SPI Flash 读引擎结构如图 11-3 所示。它将读请求打包成 SPI Flash 的读命令，读出数据后返回。为了加快接口速度，SPI Flash 生产厂商在最早的 SPI Flash 协议基础上作了若干扩展。根据受支持的情况和实现开销，本引擎实现了三种增强模式，分别为

- 连续地址读：在传送完一个字节数据后 SPI Flash 自动准备好下一个地址的数据，只要片选不拉高就可以继续传输。
- 快速读模式：可工作在高频率的读模式，在命令地址后还附了一个字节的空数据供数据读出。
- 双 I/O 模式：在发完命令编码后，SDI 和 SDO 两根线不再遵循 SPI 协议，而是同时同向进行数据传输。

双 I/O 模式与快速读模式互斥，二者只能选其一。连续地址读则与其它两种模式可共存。

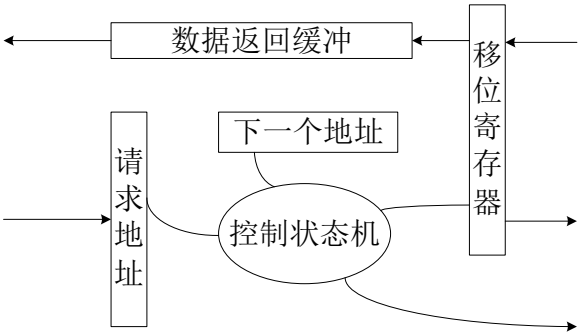



图 11-4 SPI Flash 读引擎结构

## 11.4.9 SPI 主控制器外部接口时序图

如图 11-5 所示，SPI 主控制器发送数据时，数据提前半拍放在 MOSI 引线上，接着从设

备端用时钟边沿锁存数据。根据时钟极性（CPOL）和时钟相位（CPHA）的设定，有 4 种可能的时序关系(sper.mode=1)。

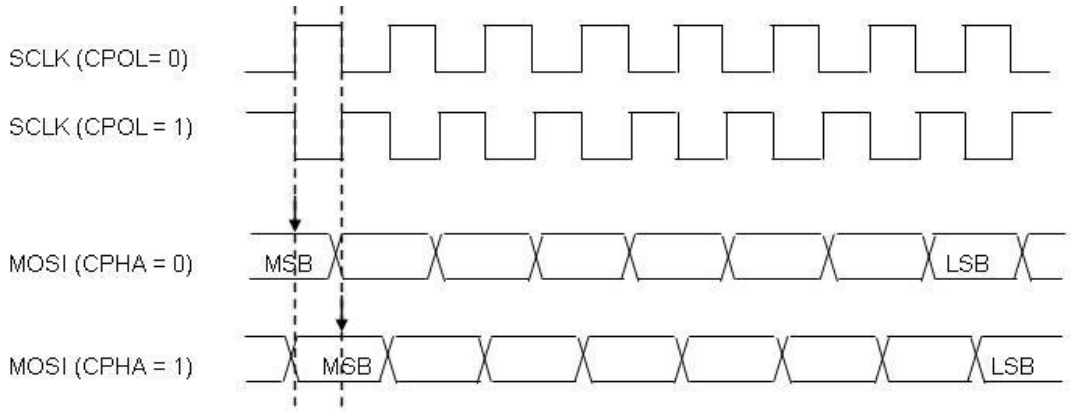



图 11-5 SPI 主控制器时序图

## 11.4.10 SPI Flash 访问时序图

- 标准读模式

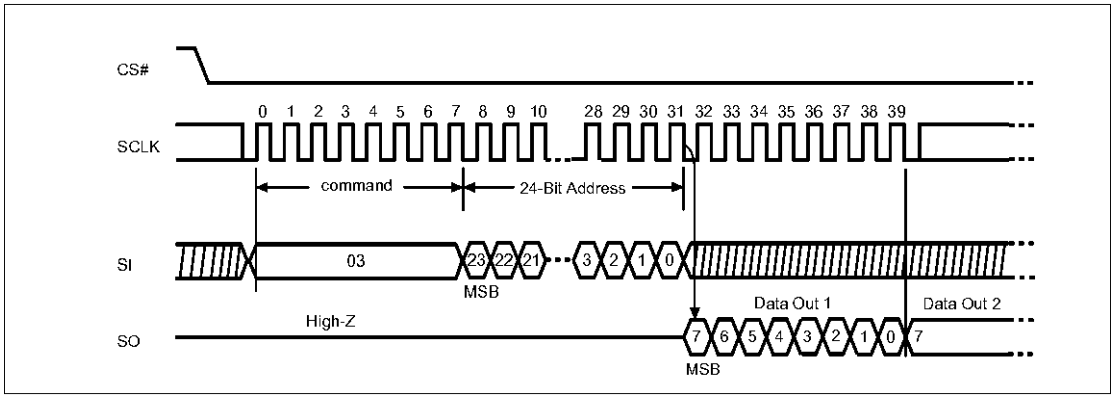



图 11-6 SPI 标准读时序

- 快速读模式

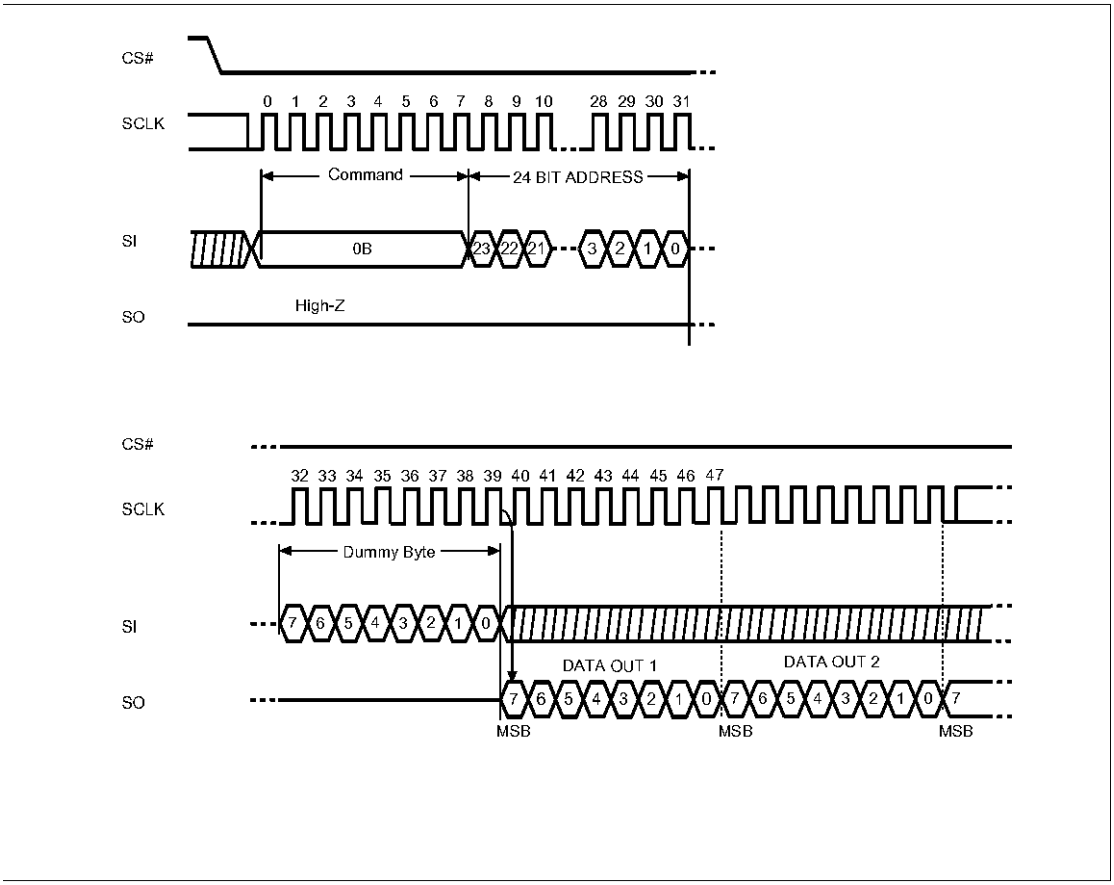



图 11-7 SPI 快速读时序

- 双 I/O 模式

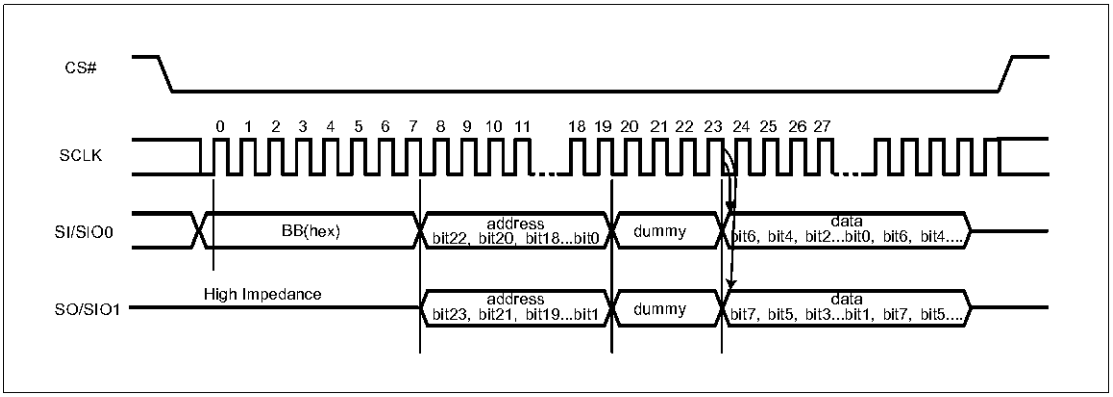



图 11-8 SPI 双 I/O 读时序

在所有模式下，若没有使能连续地址读，则 CS 将在传输完一个字节数据后拉高。

## 11.4.11 SPI Flash 控制器使用指南

本模块寄存器物理地址基址为 0x1FE00220。

**SPI 主控制器的读写操作**

### 1. 模块初始化

- 设置片选控制寄存器（SFC_SOFTCS），并使能控制寄存器
- 停止 SPI 控制器工作，对控制寄存器 spcr 的 spe 位写 0
- 重置状态寄存器 spsr，对寄存器写入 8'b1100_0000
- 设置外部寄存器 sper，包括中断申请条件 sper[7:6]和分频系数 sper[1:0]，具体参考寄存器说明
- 配置 SPI 时序，包括 spcr 的 cpol、cpha 和 sper 的 mode 位。mode 为 1 时是标准 SPI 实现，为 0 时为兼容模式。
- 配置中断使能，spcr 的 spie 位
- 启动 SPI 控制器，对控制寄存器 spcr 的 spe 位写 1

### 2. 模块的发送/传输操作

- 往数据传输寄存器写入数据
- 传输完成后从数据传输寄存器读出数据。由于发送和接收同时进行，即使 SPI 从设备没有发送有效数据也必须进行读出操作。

### 3. 中断处理

- 接收到中断申请
- 读状态寄存器 spsr 的值，若 spsr[2]为 1 则表示数据发送完成，若 spsr[0]为 1 则表示已经接收数据
- 读或写数据传输寄存器
- 往状态寄存器 spsr 的 spif 位写 1，清除控制器的中断申请

**硬件 SPI Flash 读**

### 1. 初始化

- 将 SFC_PARAM 的 memory_en 位写 1。当 SPI 被选为启动设备时此位复位为 1。
- 设置读参数(时钟分频、连续地址读、快速读、双 I/O、tCSH 等)。这些参数复位值均为最保守的值。

### 2. 更改参数

如果所使用的 SPI Flash 支持更高的频率或者提供增强功能，修改相应参数可以大大加快 Flash 的访问速度。参数的修改不需要关闭 SPI Flash 读使能(memory_en)。具体参考寄存器说明。

**混合访问 SPI Flash 和 SPI 主控制器**

### 1. 使用多个 SPI 外设

SPI Flash 控制器提供 4 个软件可控制的片选 cs[3:0]，其中 cs[0]专用于 SPI Flash。在龙芯 3B1500 中，SPI Flash 控制器的片选与 GPIO[3:0]对应。软件可通过 SFC_SOFTCS 控制寄存器设置 cs[3:1]，来访问所选择的 SPI 设备，当 SFC_SOFTCS 的 csen 有效时，对应的 GPIO 位由 SPI Flash 控制器控制。如果软件选中了其它 SPI 设备，此时又出现 SPI Flash 读访问(比如取指)，cs[3:1]会自动无效，Flash 访问结束后恢复原值。

**2. 对 SPI Flash 进行读以外的访问**

将 SPI Flash 读使能关闭后，软件就可直接控制 cs[0]，并通过 SPI 主控制器访问 SPI 总线。这意味着在进行此操作时，不能从 SPI Flash 中取指。除了读以外，SPI Flash 还实现了很多命令(如擦除、写入)，具体参见相关文档。

## 11.5 IO 控制器配置

配置寄存器主要用于配置 PCI/PIC-X 控制器的地址窗口、仲裁器以及 GPIO 控制器。表 10-25 列出了这些寄存器，表 10-26 给出寄存器的详细说明。这部分寄存器基地址为 0x1FE00100。

表 11-25 IO 控制寄存器

| 地 址 | 寄存器 | 说 明 |
|---|---|---|
| 00 | PonCfg | 上电配置 |
| 04 | GenCfg | 常规配置 |
| 08 | 保留 | |
| 0C | 保留 | |
| 10 | PCIMap | PCI 映射 |
| 14 | PCIX_Bridge_Cfg | PCI/X 桥相关配置 |
| 18 | PCIMap_Cfg | PCI 配置读写设备地址 |
| 1C | GPIO_Data | GPIO 数据 |
| 20 | GPIO_EN | GPIO 方向 |
| 24 | 保留 | |
| 28 | 保留 | |
| 2C | 保留 | |
| 30 | 保留 | |
| 34 | 保留 | |
| 38 | 保留 | |
| 3C | 保留 | |
| 40 | Mem_Win_Base_L | 可预取窗口基址低 32 位 |
| 44 | Mem_Win_Base_H | 可预取窗口基址高 32 位 |
| 48 | Mem_Win_Mask_L | 可预取窗口掩码低 32 位 |

| | | |
|---|---|---|
| 4C | Mem_Win_Mask_H | 可预取窗口掩码高 32 位 |
| 50 | PCI_Hit0_Sel_L | PCI 窗口 0 控制低 32 位 |
| 54 | PCI_Hit0_Sel_H | PCI 窗口 0 控制高 32 位 |
| 58 | PCI_Hit1_Sel_L | PCI 窗口 1 控制低 32 位 |
| 5C | PCI_Hit1_Sel_H | PCI 窗口 1 控制高 32 位 |
| 60 | PCI_Hit2_Sel_L | PCI 窗口 2 控制低 32 位 |
| 64 | PCI_Hit2_Sel_H | PCI 窗口 2 控制高 32 位 |
| 68 | PXArb_Config | PCIX 仲裁器配置 |
| 6C | PXArb_Status | PCIX 仲裁器状态 |
| 70 | | |
| 74 | | |
| 78 | | |
| 7C | | |
| 80~f0 | Chip configure related | 芯片配置相关寄存器 |

表 11-26 寄存器详细描述

| 位域 | 字段名 | 访问 | 复位值 | 说明 |
|---|---|---|---|---|
| CR00: PonCfg | | | | |
| 15:0 | pcix_bus_dev | 只读 | | PCIX Agent 模式下 CPU 取指所使用的总线、设备号 |
| 23:16 | pon_pci_configi | 只读 | pci_configi | PCI_Configi 引脚值 |
| 31:24 | 保留 | 只读 | | |
| CR04: 保留 | | | | |
| 31:0 | 保留 | 只读 | 0 | |
| CR08: 保留 | | | | |
| 31:0 | 保留 | 只读 | 0 | |
| CR10: PCIMap | | | | |
| 5:0 | trans_lo0 | 读写 | 0 | PCI_Mem_Lo0 窗口映射地址高 6 位 |
| 11:6 | trans_lo1 | 读写 | 0 | PCI_Mem_Lo1 窗口映射地址高 6 位 |
| 17:12 | trans_lo2 | 读写 | 0 | PCI_Mem_Lo2 窗口映射地址高 6 位 |
| 31:18 | 保留 | 只读 | 0 | |
| CR14: PCIX_Bridge_Cfg | | | | |
| 5:0 | pcix_rgate | 读写 | 6'h18 | PCIX 模式下向 DDR2 发读取数门限 |
| 6 | pcix_ro_en | 读写 | 0 | PCIX 桥是否允许写越过读 |
| 31:18 | 保留 | 只读 | 0 | |

| | | | | |
|---|---|---|---|---|
| CR18: PCIMap_Cfg | | | | |
| 15:0 | dev_addr | 读写 | 0 | PCI 配置读写时 AD 线高 16 位 |
| 16 | conf_type | 读写 | 0 | 配置读写的类型 |
| 31:17 | 保留 | 只读 | 0 | |
| CR1C: GPIO_Data | | | | |
| 15:0 | gpio_out | 读写 | 0 | GPIO 输出数据 |
| 31:16 | gpio_in | 读写 | 0 | GPIO 输入数据 |
| CR20: GPIO_EN | | | | |
| 15:0 | gpio_en | 读写 | FFFF | 高为输入，低输出 |
| 31:16 | 保留 | 只读 | 0 | |
| CR3C: 保留 | | | | |
| 31:0 | 保留 | 只读 | 0 | 保留 |
| CR24,2C,30,34,38:保留 | | | | |
| 见表 11-3 | | | | |
| CR50,54/58,5C/60,64: PCI_Hit*_Sel_* | | | | |
| 0 | 保留 | 只读 | 0 | |
| 2:1 | pci_img_size | 读写 | 2'b11 | 00: 32 位；10: 64 位；其它: 无效 |
| 3 | pref_en | 读写 | 0 | 预取使能 |
| 11:4 | 保留 | 只读 | 0 | |
| 62:12 | bar_mask | 读写 | 0 | 窗口大小掩码（高位 1，低位 0） |
| 63 | burst_cap | 读写 | 1 | 是否允许突发传送 |
| CR68:PXArb_Config | | | | |
| 0 | device_en | 读写 | 1 | 外部设备允许 |
| 1 | disable_broken | 读写 | 0 | 禁用损坏的主设备 |
| 2 | default_mas_en | 读写 | 1 | 总线停靠到默认主设备<br>0：停靠到最后一个主设备<br>1：停靠到默认主设备 |
| 5:3 | default_master | 读写 | 0 | 总线停靠默认主设备号 |
| 7:6 | park_delay | 读写 | 2'b11 | 从没有设备请求总线开始到触发停靠默认设备行为的延迟<br>00: 0 周期<br>01: 8 周期<br>10: 32 周期<br>11: 128 周期 |
| 15:8 | level | 读写 | 8'h01 | 处于第一级的设备 |
| 23:16 | rude_dev | 读写 | 0 | 强制优先级设备 |

| | | | | |
|---|---|---|---|---|
| | | | | 为 1 的位对应的 PCI 设备在得到总线后可以通过持续请求来占住总线 |
| 31:13 | 保留 | 只读 | 0 | |
| CR6C: PXArb_Status | | | | |
| 7:0 | broken_master | 只读 | 0 | 损坏的主设备（改变禁用策略时清零） |
| 10:8 | Last_master | 只读 | 0 | 最后使用总线的主设备 |
| 31:11 | 保留 | 只读 | 0 | |
| CR80: Chip config（见 2.6 节） | | | | |
| CR90: Chip sample（见 2.6 节） | | | | |
| CRa0: 保留 | | | | |
| CRb0:（见 2.6 节） | | | | |
| CRc0:（见 2.6 节） | | | | |
| CRd0:（见 2.6 节） | | | | |