龙芯中科
LOONGSON TECHNOLOGY

LOONGSON

龙芯 3A3000/3B3000 处理器
用户手册

多核处理器架构、寄存器描述与系统软件编程指南

V1.5

龙芯中科技术股份有限公司

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。

龙芯中科技术股份有限公司

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 3A3000/3B3000 处理器用户手册》介绍龙芯 3A3000/3B3000 多核处理器架构与寄存器描述，对芯片系统架构、主要模块的功能与配置、寄存器列表及位域进行详细说明。

修订历史

文档更新记录	文档名:	龙芯 3A3000/3B3000 处理器用户手册
	版本号	V1.5
	创建人:	芯片研发部
	创建日期 :	2021-07-28

更新历史

序号	更新日期	版本号	更新内容
1	2016-06-14	V1.0	初始版本
2	2016-09-14	V1.1	更新部分 PLL 配置寄存器说明，更新软硬件改动说明
3	2016-11-25	V1.2	增加 BBGEN 部分的寄存器说明
4	2017-04-13	V1.3	增加 SPI 地址空间说明
5	2020-07-16	V1.4	修正 10.5.28 节的频率说明 修正部分文字错误，补充部分保留位说明
5	2021-07-28	V1.5	修订龙芯系列处理器介绍

手册信息反馈: service@loongson.cn

也可通过问题反馈网站 http://bugs.loongnix.org/ 向我司提交芯片产品使用过程中的问题，并获取技术支持。

目 录

图 目 录

表 目 录

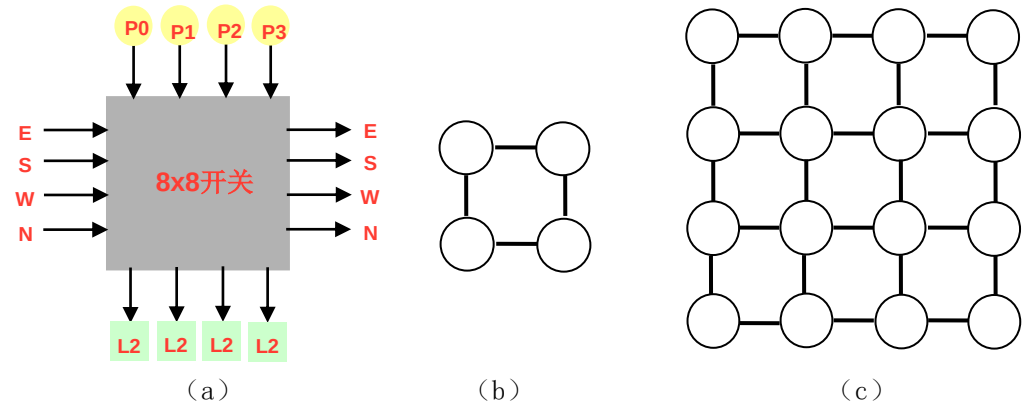
1 概述

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 号系列处理器采用 32 位处理器核，集成各种外围接口，形成面向特定应用的单片解决方案，主要应用于物联终端、仪器设备、数据采集等领域。龙芯 2 号系列处理器采用 32 位或 64 位处理器核，集成各种外围接口，形成面向网络设备、行业终端、智能制造等的高性能低功耗 SoC 芯片。龙芯 3 号系列处理器片内集成多个 64 位处理器核以及必要的存储和 IO 接口，面向高端嵌入式计算机、桌面、服务器等应用。

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 2 级 Cache，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共享 Cache，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、共享 Cache、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）

连接处理器和共享 Cache。第二级交叉开关（称为 X2 Switch，简称 X2）连接共享 Cache 和内存控制器。

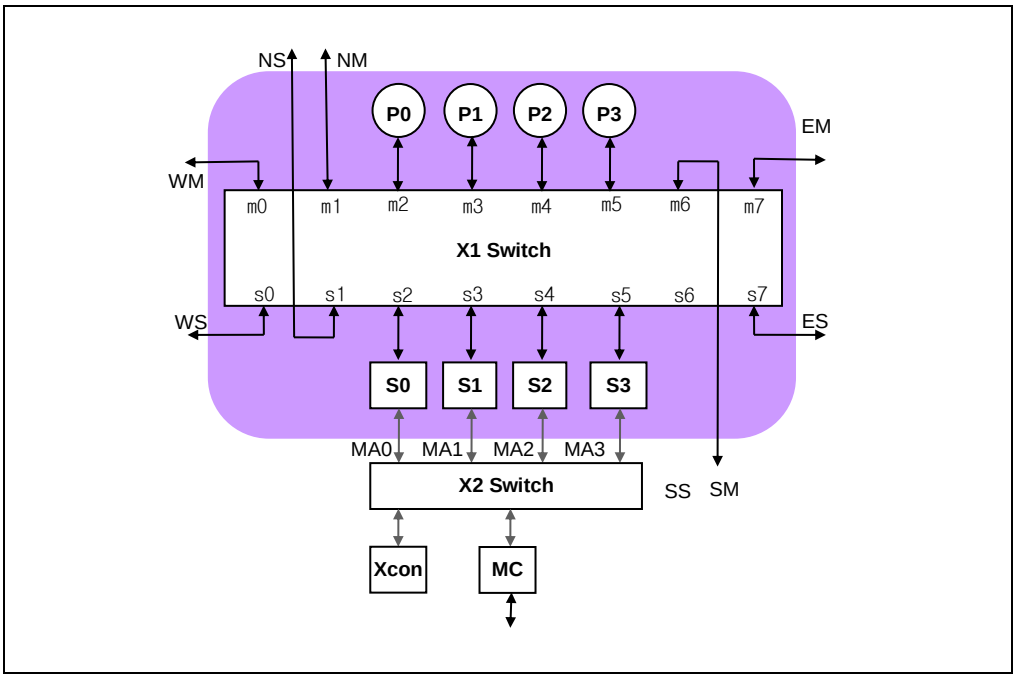


图 1-2 龙芯 3 号节点结构

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 interleave 共享 Cache 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 端口连接四个共享 Cache，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。

1.2 龙芯 3A3000/3B3000 简介

龙芯 3A3000/3B3000 是龙芯 3A2000/3B2000 四核处理器的工艺升级版本，封装引脚与龙芯 3A1000 相比，PLL_AVDD 由 2.5V 改为 1.8V，与龙芯 3A2000/3B2000 相比，多使用了 PLL_AVDD。龙芯 3A3000/3B3000 是一个配置为单节点 4 核的处理器，采用 28nm 工艺制造，工作主频为 1.2GHz-1.5GHz，主要技术特征如下：

- 片内集成 4 个 64 位的四发射超标量 GS464e 高性能处理器核；
- 片内集成 8MB 的分体共享三级 Cache(由 4 个体模块组成，每个体模块容量为 2MB）；

- 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 72 位带 ECC，667MHz 的 DDR2/3 控制器；
- 3B3000 集成 2 个 16 位 2.4GHz 的 HyperTransport 接口（以下简称 HT）；
- 3A3000 片内 HT1 为 16 位 2.4GHz 的 HT 接口，HT0 不可用；
- 每个 16 位的 HT 端口拆分成两个 8 路的 HT 端口使用。
- 片内集成 32 位 33MHz PCI；
- 片内集成 1 个 LPC、2 个 UART、1 个 SPI、16 路 GPIO 接口。

相比龙芯 3A2000/3B2000，其主要改进如下：

- 处理器核微结构升级；
- 内存控制器结构、频率升级；
- HT 控制器结构、频率升级；
- 全芯片的性能优化提升。

龙芯 3A3000/3B3000 芯片整体架构基于两级互连实现，结构如下图 1-3 所示。

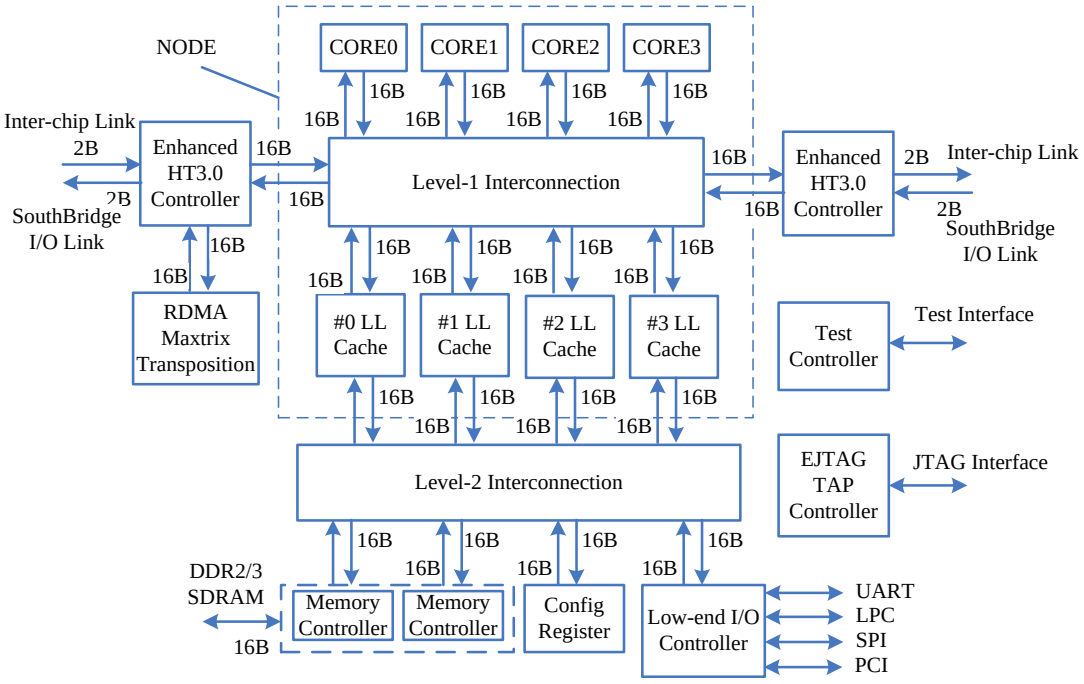


图 1-3 龙芯 3A3000/3B3000 芯片结构

第一级互连采用 6x6 的交叉开关，用于连接四个 GS464e 核（作为主设备）、四个共享 Cache 模块（作为从设备）、以及两个 IO 端口（每个端口使用一个 Master 和一个 Slave）。一级互连开关连接的每个 IO 端口连接一个 16 位的 HT 控制器，每个 16 位的 HT 端口还可以作为两个 8 位的 HT 端口使用。HT 控制器通过一个 DMA 控制器和一级互联开关相连，DMA 控制器负责 IO 的 DMA 控制并负责片间一致性的维护。龙芯 3 号的 DMA 控制器还可以通过配置

实现预取和矩阵转置或搬移。

第二级互连采用5x4的交叉开关，连接4个共享Cache模块(作为主设备)，两个DDR2/3内存控制器、低速高速I/O（包括PCI、LPC、SPI等）以及芯片内部的配置寄存器模块。

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。

2 系统配置与控制

2.1 芯片工作模式

根据组成系统的结构，龙芯 3A3000/3B3000 主要包括三种工作模式：

- 单芯片模式。系统只包含 1 片龙芯 3A3000/3B3000，是一个对称多处理器系统（SMP）；
- 多芯片互连模式。系统中包含 2 片或 4 片龙芯 3A3000/3B3000，通过龙芯 3A3000/3B3000 的 HT 端口进行互连，是一个非均匀访存多处理器系统（CC-NUMA）；
- 大规模互连模式。通过专用扩展桥片进行多芯片大规模扩展互连，构成大规模非均匀访存多处理器系统（CC-NUMA）。

2.2 控制引脚说明

主要控制引脚包括 DO_TEST、ICCC_EN、NODE_ID[1:0]、CLKSEL[15:0]、PCI_CONFIG。表中未专门声明的说明部分，表示引脚电平上拉时的作用。

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
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
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</td></tr>
<tr><td>CLKSEL[15:0]</td><td></td><td>HT 时钟控制
<table>
<tr><th>信号</th><th>作用</th></tr>
<tr><td>CLKSEL[15]</td><td>1’b1 表示 HT 控制器频率仅采用硬件设置
1’b0 表示 HT 控制器频率可采用软件设置</td></tr>
<tr><td>CLKSEL[14]</td><td>1’b1 表示 HT PLL 采用普通时钟输入
1’b0 表示 HT PLL 采用差分时钟输入</td></tr>
<tr><td>CLKSEL[13:12]</td><td>2’b00 表示 PHY 时钟为 1.6GHZ
2’b01 表示 PHY 时钟为 3.2GHZ
2’b10 表示 PHY 时钟为 1.2GHz</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>2’b11 表示 PHY 时钟为 2.4GHz</td></tr>
<tr><td>CLKSEL[11:10]</td><td>2’b00 表示 HT 控制器时钟为 PHY 时钟 8 分频
2’b01 表示 HT 控制器时钟为 PHY 时钟 4 分频
2’b10 表示 HT 控制器时钟为 PHY 时钟 2 分频
2’b11 表示 HT 控制器时钟为 SYSCLOCK</td></tr>
</table>
注：CLKSEL[13:10] == 4’b1111 时，HT 控制器时钟直接使用外部输入 100MHz 参考时钟
MEM 时钟控制
<table>
<tr><th>信号</th><th>作用</th></tr>
<tr><td>CLKSEL[9:5]</td><td>5’b11111 表示 MEM 时钟直接采用 memclk
5’b01111 表示 MEM 时钟采用软件设置，设置方法见 2.7 节说明
其它情况下 MEM 时钟为
memclk*(clksel[8:5]+30)/(clksel[9]+3)
注：
memclk*(clksel[8:5]+30)必须为 1.2GHz~3.2GHz
memclk 为输入参考时钟，必须为 20~40MHz</td></tr>
</table>
CORE 时钟控制
<table>
<tr><th>信号</th><th>作用</th></tr>
<tr><td>CLKSEL[4:0]</td><td>5’b11111 表示 CORE 时钟直接采用 sysclk
5’b011xx 表示 CORE 时钟采用软件设置，设置方法见 2.7 节说明。
5’b01111 为正常工作模式，其它情况为调试模式
5’b0110x 表示处理器接口为异步模式
5’b011x0 表示延迟调试控制模式
其它情况下 CORE 时钟为
sysclk*(clksel[3:0]+30)/(clksel[4]+1)
注：
sysclk*(clksel[3:0]+30) 必须为 1.2GHz~3.2GHz
sysclk 为输入参考时钟，必须为 20~40MHz</td></tr>
</table>
</td></tr>
<tr><td>PCI_CONFIG[7:0]</td><td></td><td>IO 配置控制
7 HT 总线冷启动强制设为 1.0 模式
6:4 需设置为 000
3 PCI 主设备模式</td></tr>
</table>

		2 需设置为 0 1 使用外部 PCI 仲裁 0 使用 SPI 启动功能

2.3 Cache 一致性

龙芯 3A3000/3B3000 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 3A3000/3B3000 处理器可以直接采用 4 芯片直连构建 CC-NUMA 系统，每个芯片的处理器号由引脚 NODEID 决定，每个芯片的地址空间分布如下：

表 2-2 节点级的系统全局地址分布

芯片节点号（NODEID）	地址[47:44]位	起始地址	结束地址
0	0	0x0000_0000_0000	0x0FFF_FFFF_FFFF
1	1	0x1000_0000_0000	0x1FFF_FFFF_FFFF
2	2	0x2000_0000_0000	0x2FFF_FFFF_FFFF
3	3	0x3000_0000_0000	0x3FFF_FFFF_FFFF

龙芯 3A3000/3B3000 采用单节点 4 核配置，因此龙芯 3A3000/3B3000 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 0x1000_0000_0000（不含）的 44 位地在每个节点的内部，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 个共享 Cache 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。

龙芯 3A3000/3B3000 芯片内部一级交叉开关的地址空间对应的各个从设备端如下：

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
共享 Cache	0,1,2,3	0x000_0000_0000	0x7FF_FFFF_FFFF
HT0 控制器	6	0xC00_0000_0000	0xDFF_FFFF_FFFF
HT1 控制器	7	0xE00_0000_0000	0xFFF_FFFF_FFFF

不同于方向端口的映射关系，龙芯 3A3000/3B3000 可以根据实际应用的访问行为，来决定共享 Cache 的交叉寻址方式。节点内的 4 个共享 Cache 模块一共对应 43 位地址空间，而每个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[7:6]地位散列的方式进行分布，即地址[7:6]两位决定对应的共享 Cache 编号。该寄存器地址 0x3FF00400。

表 2-4 节点内的地址分布

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

2.5 节点内的物理地址空间分布

龙芯 3A3000/3B3000 处理器每个节点的内部 44 位物理地址的默认分布如下表所示：

表 2-5 节点内 44 位物理地址分布

起始地址	结束地址	名称	说明
0x0000_0000_0000	0x0000_0FFF_FFFF	内存	需要使用二级交叉开关进行映射
0x0000_1000_0000	0x0000_1FFF_FFFF	低速 IO	需要使用二级交叉开关进行映射

2.6 地址路由分布与配置

龙芯 3A3000/3B3000 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 [6]表示允许对Scache的交错访问使能，MMAP [7]表示窗口使能。

表 2-6 MMAP 字段对应的该空间访问属性

[7]	[6]	[5]	[4]
窗口使能	允许对 SCACHE 进行交错访问，当 Slave 号为 0 时有效，按照上一节 SCID_SEL 的配置对命中窗口地址的请求进行路由	允许块读	允许取指

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。

地址窗口转换寄存器如下表所示。

表 2-7 一级交叉开关地址窗口寄存器表

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_2600	HT0_WIN0_BASE	0x3ff0_2700	HT1_WIN0_BASE
0x3ff0_2608	HT0_WIN1_BASE	0x3ff0_2708	HT1_WIN1_BASE
0x3ff0_2610	HT0_WIN2_BASE	0x3ff0_2710	HT1_WIN2_BASE
0x3ff0_2618	HT0_WIN3_BASE	0x3ff0_2718	HT1_WIN3_BASE
0x3ff0_2620	HT0_WIN4_BASE	0x3ff0_2720	HT1_WIN4_BASE
0x3ff0_2628	HT0_WIN5_BASE	0x3ff0_2728	HT1_WIN5_BASE
0x3ff0_2630	HT0_WIN6_BASE	0x3ff0_2730	HT1_WIN6_BASE
0x3ff0_2638	HT0_WIN7_BASE	0x3ff0_2738	HT1_WIN7_BASE
0x3ff0_2640	HT0_WIN0_MASK	0x3ff0_2740	HT1_WIN0_MASK
0x3ff0_2648	HT0_WIN1_MASK	0x3ff0_2748	HT1_WIN1_MASK
0x3ff0_2650	HT0_WIN2_MASK	0x3ff0_2750	HT1_WIN2_MASK
0x3ff0_2658	HT0_WIN3_MASK	0x3ff0_2758	HT1_WIN3_MASK
0x3ff0_2660	HT0_WIN4_MASK	0x3ff0_2760	HT1_WIN4_MASK
0x3ff0_2668	HT0_WIN5_MASK	0x3ff0_2768	HT1_WIN5_MASK
0x3ff0_2670	HT0_WIN6_MASK	0x3ff0_2770	HT1_WIN6_MASK
0x3ff0_2678	HT0_WIN7_MASK	0x3ff0_2778	HT1_WIN7_MASK
0x3ff0_2680	HT0_WIN0_MMAP	0x3ff0_2780	HT1_WIN0_MMAP
0x3ff0_2688	HT0_WIN1_MMAP	0x3ff0_2788	HT1_WIN1_MMAP
0x3ff0_2690	HT0_WIN2_MMAP	0x3ff0_2790	HT1_WIN2_MMAP
0x3ff0_2698	HT0_WIN3_MMAP	0x3ff0_2798	HT1_WIN3_MMAP
0x3ff0_26a0	HT0_WIN4_MMAP	0x3ff0_27a0	HT1_WIN4_MMAP
0x3ff0_26a8	HT0_WIN5_MMAP	0x3ff0_27a8	HT1_WIN5_MMAP
0x3ff0_26b0	HT0_WIN6_MMAP	0x3ff0_27b0	HT1_WIN6_MMAP
0x3ff0_26b8	HT0_WIN7_MMAP	0x3ff0_27b8	HT1_WIN7_MMAP

在龙芯 3 号的二级 XBAR 中有配置寄存器地址空间、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 两个具有主设备功能的 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 中包含转换后地址、路由选择及使能控制等位，如下表所示：

表 2-8 MMAP 寄存器说明

[63：48]	[47：10]	[7：4]	[3：0]
交错选择位	转换后地址	窗口使能	从设备号

其中，从设备号对应的设备如下表所示：

表 2-9 二级 XBAR 处，从设备号与所述模块的对应关系

从设备号	缺省值
0	0 号 DDR2/3 控制器
1	1 号 DDR2/3 控制器
2	低速 I/O（PCI，LPC 等）
3	配置寄存器

窗口使能位的含义如下表所示：

表 2-10 MMAP 字段对应的该空间访问属性

[7]	[6]	[5]	[4]
窗口使能	允许对 DDR 进行交错访问，当从设备号为 0 时有效，按照“交错选择位”的配置对命中窗口地址的请求进行路由。要求交错使能位大于 10	允许块读	允许取指

需要注意的是，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}

地址窗口转换寄存器如下表：

表 2-11 二级 XBAR 地址窗口转换寄存器表

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 2-12 二级 XBAR 缺省地址配置

基地址	高位	所有者
0x0000_0000_0000_0000	0x0000_0000_0FFF_FFFF	0 号 DDR 控制器
0x0000_0000_1000_0000	0x0000_0000_1FFF_FFFF	低速 I/O（PCI 等）

2.7 芯片配置及采样寄存器

龙芯 3A3000/3B3000 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。

表中未专门声明的说明部分，表示位域值为 1 时的作用。

表 2-13 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	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	使能 MC0
9	MC1_disable_ddr2_confspace	RW	1'b0	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	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	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	使能 HT1
42:40	Node0_freq_ ctrl	RW	3'b111	节点 0 分频
43	-	RW	1'b1	保留
46:44	Node1_freq_ ctrl	RW	3'b111	节点 1 分频
47	-	RW	1'b1	保留
63:56	Cpu_version	R	2'h39	CPU 版本
95:64				保留
127:96	Pad1v8_ctrl	RW	6'h780	1v8 pad 控制
其它		R		保留

表 2-14 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		保留
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		保留
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℃）
111:96	Thsens0_out	R		温度传感器 0 摄氏温度 结 点 温 度 =Thens0_out

				*731/0x4000 - 273 温度范围 -40 度 – 125 度
127:112	Thsens1_out	R		温度传感器 1 摄氏温度 结 点 温 度 =Thens1_out *731/0x4000 - 273 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（参考 2.2 节的 CLKSEL 设置方法）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、片上网络及高速共享缓存的时钟频率；HT CLOCK 对应 HT 控制器时钟频率。

每个时钟配置一般有两个参数， DIV_LOOPC、DIV_OUT。最终的时钟频率为(参考时钟* DIV_LOOPC) / DIV_OUT。

对于 HT CLOCK 的配置方法比较特殊，请参考 10.5.28 节的具体配置方法。

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，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。

3A3000/3B3000 中，可以采用两种不同的 PLL 来提供处理器核时钟，在下表中的控制分别为 L1 和 L2。软件可以通过控制寄存器中的 serial_mode 来选择采用哪一个 PLL 的输出作为主时钟。

表 2-15 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_NODE	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
15:4	-	RW	0x0	保留
16	LOCKED_L1	R	0x0	L1 PLL 是否锁定
17	LOCKED_L2	R	0x0	L2 PLL 是否锁定
18:17	-	R	0x0	保留

19	PD_L1	RW	0x0	关闭 L1 PLL
20	PD_L2	RW	0x0	关闭 L2 PLL
21				保留
22	Serial_mode	RW	0x0	0：选用 L1 PLL 作为主时钟 1：选用 L2 PLL 作为主时钟
23	Serial_mode3	RW	0x0	0：选用 NODE 时钟作为核时钟 1：选用 CORE 时钟作为核时钟
25:24	-	RW		保留
31:26	L1_DIV_REFC	RW	0x1	L1 PLL 输入参数
40:32	L1_DIV_LOOPC	RW	0x1	L1 PLL 输入参数
41				保留
47:42	L1_DIV_OUT	RW	0x1	L1 PLL 输入参数
50:48	L2_DIV_REFC	RW	0x1	L2 PLL 输入参数
53:51				保留
63:54	L2_DIV_LOOPC	RW	0x1	L2 PLL 输入参数
69:64	L2_DIV_OUT	RW	0x1	L2 PLL 输入参数 必须且仅有一位为 1
95:70				保留
96	BBGEN_enable	RW	0x0	偏压使能
97	BBMUX_first	RW	0x0	设置为先切换电压模式 （必须设置为 0）
99:98	BBMUX_SEL_0	RW	0x0	BBMUX_SEL_0 设置值
101:100	BBGEN_feedback	RW	0x0	禁用 BBGEN 反馈信号
103:102				保留
107:104	BBGEN_vbbp_val	WO	0x0	Vbbp 的设置值
111:108	BBGEN_vbbn_val	WO	0x0	Vbbn 的设置值
121:112				保留
123:122	BBMUX_SEL_1	RW	0x0	BBMUX_SEL_1 设置值
125:124	BBMUX_SEL_2	RW	0x0	BBMUX_SEL_2 设置值
127:126	BBMUX_SEL_3	RW	0x0	BBMUX_SEL_3 设置值
其它	-	RW		保留

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

L1 PLL 的 VCO 频率（上述式中括号内部分）必须在范围 1.2GHz - 3.2GHz 之内。该要求对 MEM PLL 和 HT PLL 同样适用。L2 PLL 的 VCO 频率必须在 范围 3.2GHz - 6.4GHz 之内。

表 2-16 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

5:3				保留
6	LOCKED_MEM_PLL	R	0x0	MEM_PLL 是否锁定
7	PD_MEM_PLL	RW	0x0	关闭 MEM PLL
13:8	MEM_PLL_DIV_REFC	RW	0x1	MEM PLL 输入参数 当选用 NODE 时钟（NODE_CLOCK_SEL 为 1）时，作为分频输入
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
30	NODE_CLOCK_SEL	RW	0x0	0：使用 MEM_PLL 作为 MEM 时钟 1：使用 NODE_CLOCK 作为分频输入
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 2-17 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
			注:	软件分频后的时钟频率值等于原来的（分频控制值+1）/8

3 GS464e 处理器核

GS464e 是四发射 64 位的高性能处理器核。该处理器核既可以作为单核面向高端嵌入式应用和桌面应用，也可以作为基本的处理器核构成片内多核系统面向服务器和高性能机应用。在龙芯 3A3000/3B3000 中的多个 GS464 核通过以及共享 Cache 模块通过 AXI 互联网络形成一个分布式共享片上末级 Cache 的多核结构。GS464 的主要特点如下：

- MIPS64 兼容，支持龙芯扩展指令集；
- 四发射超标量结构，两个定点、两个浮点、两个访存部件；
- 每个浮点部件都支持全流水 64 位/双 32 位浮点乘加运算；
- 访存部件支持 128 位存储访问，虚地址为 64 位，物理地址为 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 项全相联外加 8 路组相连 1024 项，共计 1088 项 TLB，64 项指令 TLB，可变页大小；
- 一级指令 Cache 和数据 Cache 大小各为 64KB，4 路组相联；
- Victim Cache 作为私有二级 Cache，大小为 256KB，16 路组相连；
- 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。

GS464e 的结构如下图所示。GS464e 更多的详细介绍请参考 GS464e 用户手册以及 MIPS64 用户手册。

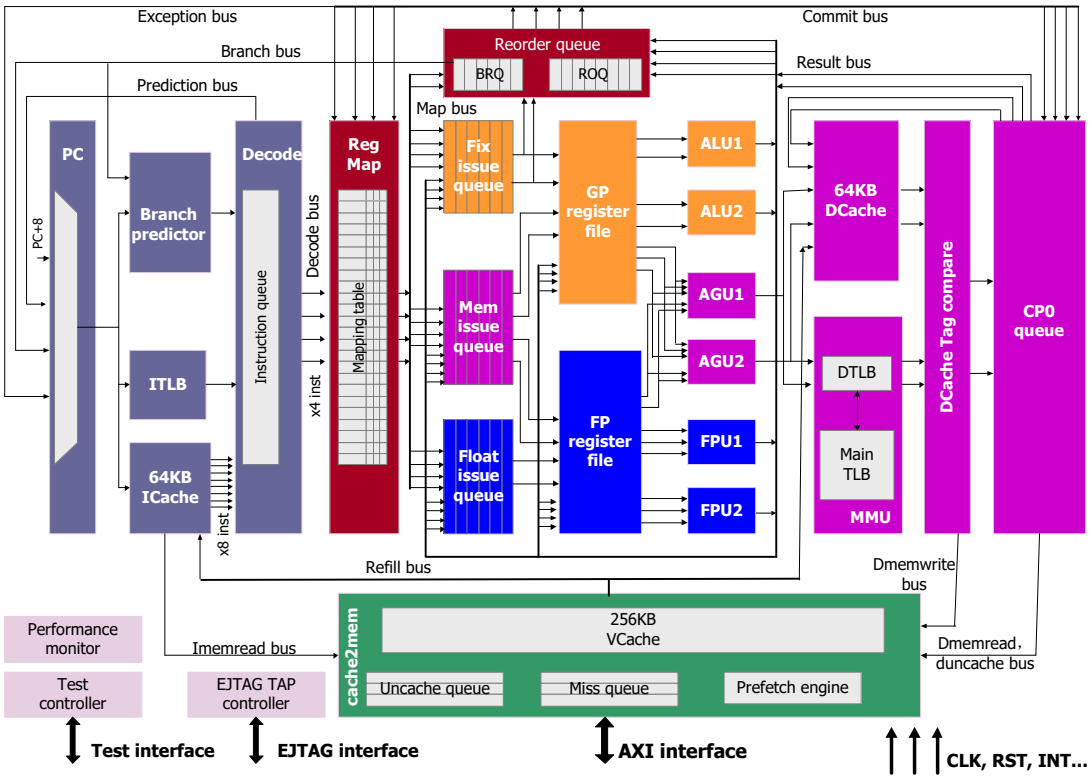


图 3-1 GS464e 结构图

4 共享 Cache（SCache）

SCache 模块是龙芯 3A3000/3B3000 处理器内部所有处理器核所共享的三级 Cache。SCache 模块的主要特征包括：

- 采用 128 位 AXI 接口。
- 16 项 Cache 访问队列。
- 关键字优先。
- 接收读失效请求到返回数据最快 12 拍。
- 通过目录支持 Cache 一致性协议。
- 可用于片上多核结构，也可直接和单处理器 IP 对接。
- 采用 16 路组相联结构。
- 支持 ECC 校验。
- 支持 DMA 一致性读写和预取读。
- 支持 16 种共享 Cache 散列方式。
- 支持按窗口锁共享 Cache。
- 保证读数据返回原子性。

共享 Cache 模块包括共享 Cache 管理模块 scachemanage 及共享 Cache 访问模块 scacheaccess。Scachemanage 模块负责处理器来自处理器和 DMA 的访问请求，而共享 Cache 的 TAG、目录和数据等信息存放在 scacheaccess 模块中。为降低功耗，共享 Cache 的 TAG、目录和数据可以分开访问，共享 Cache 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。失效请求访问共享 Cache，同时读出所有路的 TAG、目录，并根据 TAG 来选出目录，并根据命中情况读取数据。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，共享 Cache 增加了锁机制。落在被锁区域中的共享 Cache 块会被锁住，因而不会被替换出共享 Cache。通过芯片配置寄存器空间可以对共享 Cache 模块内部的四组锁窗口寄存器进行动态配置，锁 Cache 时连续区域大小需要小于 6MB。

表 4-1 共享 Cache 锁窗口寄存器配置

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。

5 矩阵处理加速器

龙芯 3A3000/3B3000 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 3A3000/3B3000 的两个 HyperTransport 控制器内部，通过一级交叉开关实现对 SCache 及内存的读写。

由于转置前同一 Cache 行的元素顺序在转置后的矩阵中是分散的，为了提高读写效率，需要读入多行数据，使得这些数据可以在转置后的矩阵中以 Cache 行为单位进行写入，因此在模块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移。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。

转置或搬移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。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
0x3ff00600	src_start_addr	RW	源矩阵起始地址
0x3ff00608	dst_start_addr	RW	目标矩阵起始地址
0x3ff00610	row	RW	源矩阵的一行元素个数
0x3ff00618	col	RW	源矩阵的一列元素个数
0x3ff00620	length	RW	源矩阵所在大矩阵的行跨度（字节）
0x3ff00628	width	RW	目标矩阵所在大矩阵的行跨度（字节）
0x3ff00630	trans_ctrl	RW	转置控制寄存器
0x3ff00638	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
0x3ff00600	0 号转置模块的 src_start_addr
0x3ff00608	0 号转置模块的 dst_start_addr
0x3ff00610	0 号转置模块的 row
0x3ff00618	0 号转置模块的 col
0x3ff00620	0 号转置模块的 length
0x3ff00628	0 号转置模块的 width
0x3ff00630	0 号转置模块的 trans_ctrl
0x3ff00638	0 号转置模块的 trans_status
0x3ff00700	1 号转置模块的 src_start_addr
0x3ff00708	1 号转置模块的 dst_start_addr
0x3ff00710	1 号转置模块的 src_row_stride
0x3ff00718	1 号转置模块的 src_last_row_addr
0x3ff00720	1 号转置模块的 length
0x3ff00728	1 号转置模块的 width
0x3ff00730	1 号转置模块的 trans_ctrl
0x3ff00738	1 号转置模块的 trans_status

表 5-3 trans_ctrl 寄存器说明

字段	说明
0	使能位
1	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’hc。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	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 寄存器说明

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 3A3000/3B3000 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，控制对应中断位是否有效
IPI_Set	W	32 位置位寄存器，往对应的位写 1，则对应的 STATUS 寄存器位被置 1
IPI_Clear	W	32 位清除寄存器，往对应的位写 1，则对应的 STATUS 寄存器位被清 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。

在龙芯 3A3000/3B3000 与处理器核间中断相关的寄存器及其功能描述如下：

表 6-2 0 号处理器核核间中断与通信寄存器列表

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列表

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列表

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列表

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

上面列出的是单个龙芯 3A3000/3B3000 芯片所组成的单节点多处理器系统的的核间中断相关寄存器列表。在采用多片龙芯3A3000/3B3000互连构成多节点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，而 1 号节点的 0 号处理器地址则为 0x10003ff01000，依次类推。

7 I/O 中断

龙芯 3A3000/3B3000 芯片最多支持 32 个中断源，以统一方式进行管理，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。

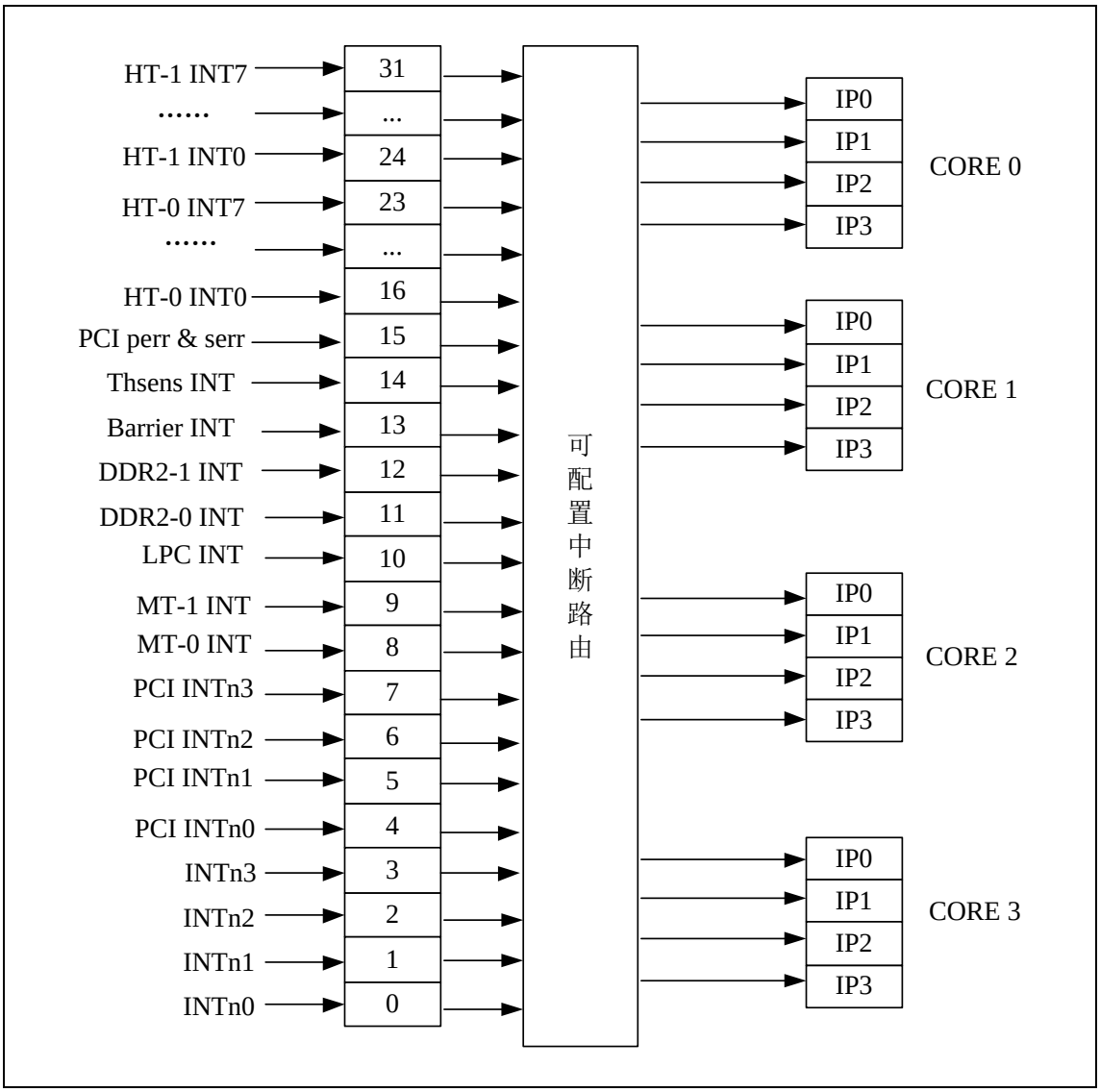


图 7-1 龙芯 3A3000/3B3000 处理器中断路由示意图

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 的位对应的中断被使能。Intenclr 清除中断使能，Intenclr 寄存器写 1 的位对应的中断被清除。Inten 寄存器读取当前各中断使能的情况。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。

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 / 1	R / 0	RW / 0	RW / 0	Matrix_int1
10	RO / 1	R / 0	RW / 0	RW / 0	Lpc
12 : 11	RW / 0	保留	保留	保留	Mc0-1
13	RW / 0	R / 0	RW / 0	RW / 0	Barrier
14	RW / 0	R / 0	RW / 0	RW / 0	Thsens int
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 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 位设置使能寄存器
Intenclr	0x3ff0142c	32 位清除使能寄存器
Intedge	0x3ff01438	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

在龙芯 3A3000/3B3000 中集成了 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 INT0 到 INT3 中的任意一个，即对应 CP0_Status 的 IP2 到 IP5。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 INT2 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
7:4	路由的处理器核中断引脚向量号

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	Thsens int	Entry30	0x3ff0141e	HT1-int6
Entry15	0x3ff0140f	Pci_perr/serr	Entry31	0x3ff0141f	HT1-int7

8 温度传感器

8.1 实时温度采样

龙芯 3A3000/3B3000 内部集成两个温度传感器，可以通过 0x1FE00198 开始的采样寄存器进行观测，同时，可以使用灵活的高低温中断报警或者自动调频功能进行控制。温度传感器在采样寄存器的对应位如下（基地址为 0x1FE00198）：

表 8-1 温度采样寄存器说明

位域	字段名	访问	复位值	描述
24	Thsens0_overflow	R		温度传感器 0 上溢（超过 125℃）
25	Thsens1_overflow	R		温度传感器 1 上溢（超过 125℃）
47:32	Thsens0_out	R		温度传感器 0 摄氏温度 结 点 温 度 =Thens0_out *731/0x4000 - 273 温度范围 -40 度 – 125 度
65:48	Thsens1_out	R		温度传感器 1 摄氏温度 结 点 温 度 =Thens1_out -*731/0x4000 - 273 温度范围 -40 度 – 125 度

通过对控制寄存器的设置，可以实现超过预设温度中断、低于预设温度中断及高温自动降频功能。

8.2 高低温中断触发

对于高低温中断报警功能，分别有 4 组控制寄存器对其阈值进行设置。每组寄存器包含以下三个控制位：

GATE：设置高温或低温的阈值。当输入温度高于高温阈值或低于低温阈值时，将会产生中断；

EN：中断使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 3A3000/3B3000 内部集成两个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入。可以使用 0 或者 1。

高温中断控制寄存器中包含 4 组用于控制高温中断触发的设置位；低温中断控制寄存器中包含 4 组用于控制低温中断触发的设置位。另外还有一组寄存器用于显示中断状态，分别对应于高温中断和低温中断，对该寄存器进行任意写操作将清除中断状态。

这几个寄存器的具体描述如下：

表 8-2 高低温中断寄存器说明

寄存器	地址	控制	说明
高温中断控制寄存器 Thsens_int_ctrl_Hi	0x3ff01460	RW	[7:0]：Hi_gate0：高温阈值 0，超过这个温度将产生中断 [8:8]：Hi_en0：高温中断使能 0 [11:10]：Hi_Sel0：选择高温中断 0 的温度传感器输入源 [23:16]：Hi_gate1：高温阈值 1，超过这个温度将产生中断 [24:24]：Hi_en1：高温中断使能 1 [27:26]：Hi_Sel1：选择高温中断 1 的温度传感器输入源 [39:32]：Hi_gate2：高温阈值 2，超过这个温度将产生中断 [40:40]：Hi_en2：高温中断使能 2 [43:42]：Hi_Sel2：选择高温中断 2 的温度传感器输入源 [55:48]：Hi_gate3：高温阈值 3，超过这个温度将产生中断 [56:56]：Hi_en3：高温中断使能 3 [59:58]：Hi_Sel3：选择高温中断 3 的温度传感器输入源
低温中断控制寄存器 Thsens_int_ctrl_Lo	0x3ff01468	RW	[7:0]：Lo_gate0：低温阈值 0，低于这个温度将产生中断 [8:8]：Lo_en0：低温中断使能 0 [11:10]：Lo_Sel0：选择低温中断 0 的温度传感器输入源 [23:16]：Lo_gate1：低温阈值 1，低于这个温度将产生中断 [24:24]：Lo_en1：低温中断使能 1 [27:26]：Lo_Sel1：选择低温中断 1 的温度传感器输入源 [39:32]：Lo_gate2：低温阈值 2，低于这个温度将产生中断 [40:40]：Lo_en2：低温中断使能 2 [43:42]：Lo_Sel2：选择低温中断 2 的温度传感器输入源 [55:48]：Lo_gate3：低温阈值 3，低于这个温度将产生中断 [56:56]：Lo_en3：低温中断使能 3 [59:58]：Lo_Sel3：选择低温中断 3 的温度传感器输入源
中断状态寄存器 Thsens_int_status/clr	0x3ff01470	RW	中断状态寄存器，写任意值清除中断 [0]：高温中断触发 [1]：低温中断触发

8.3 高温自动降频设置

为了在高温环境中保证芯片的运行，可以设置令高温自动降频，使得芯片在超过预设范围时主动进行时钟分频，达到降低芯片翻转率的效果。

对于高温降频功能，有 4 组控制寄存器对其行为进行设置。每组寄存器包含以下四个控制位：

GATE：设置高温或低温的阈值。当输入温度高于高温阈值或低于低温阈值时，将触发分频操作；

EN：中断使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 3A3000/3B3000 内部集成两个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入。可以使用 0 或者 1。

FREQ：分频数。当触发分频操作时，将频率调整为当前时钟频率的 FREQ/8 倍。

表 8-3 高温降频控制寄存器说明

寄存器	地址	控制	说明
高温降频控制寄存器 Thsens_freq_scale	0x3ff01480	RW	四组设置优先级由高到低 [7:0]：Scale_gate0：高温阈值 0，超过这个温度将降频 [8:8]：Scale_en0：高温降频使能 0 [11:10]：Scale_Sel0：选择高温降频 0 的温度传感器输入源 [14:12]：Scale_freq0：降频时的分频值 [23:16]：Scale_gate1：高温阈值 1，超过这个温度将降频 [24:24]：Scale_en1：高温降频使能 1 [27:26]：Scale_Sel1：选择高温降频 1 的温度传感器输入源 [30:28]：Scale_freq1：降频时的分频值 [39:32]：Scale_gate2：高温阈值 2，超过这个温度将降频 [40:40]：Scale_en2：高温降频使能 2 [43:42]：Scale_Sel2：选择高温降频 2 的温度传感器输入源 [46:44]：Scale_freq2：降频时的分频值 [55:48]：Scale_gate3：高温阈值 3，超过这个温度将降频 [56:56]：Scale_en3：高温降频使能 3 [59:58]：Scale_Sel3：选择高温降频 3 的温度传感器输入源 [62:60]：Scale_freq3：降频时的分频值

9 DDR2/3 SDRAM 控制器配置

龙芯 3 号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（JESD79-2 和 JESD79-3）。在龙芯 3 号处理器中，所实现的所有内存读/写操作都遵守 JESD79-2B 及 JESD79-3 的规定。

9.1 DDR2/3 SDRAM控制器功能概述

龙芯 3 号处理器支持最大 4 个 CS（由 4 个 DDR2 SDRAM 片选信号实现，即两个双面内存条），一共含有 19 位的地址总线（即：16 位的行列地址总线和 3 位的逻辑 Bank 总线）。

龙芯 3 号处理器在具体选择使用不同内存芯片类型时，可以调整 DDR2/3 控制器参数设置进行支持。其中，支持的最大片选（CS_n）数为 4，行地址（RAS_n）数为 16，列地址（CAS_n）数为 15，逻辑体选择（BANK_n）数为 3。

CPU 发送的内存请求物理地址可以根据控制器内部不同的配置进行多种不同的地址映射。

龙芯 3 号处理器所集成的内存控制电路只接受来自处理器或者外部设备的内存读/写请求，在所有的内存读/写操作中，内存控制电路处于从设备状态（Slave State）。

龙芯 3 号处理器中内存控制器具有如下特征：

- 接口上命令、读写数据全流水操作
- 内存命令合并、排序提高整体带宽
- 配置寄存器读写端口，可以修改内存设备的基本参数
- 内建动态延迟补偿电路（DCC），用于数据的可靠发送和接收
- ECC 功能可以对数据通路上的 1 位和 2 位错误进行检测，并能对 1 位错误进行自动纠错
- 支持 133-667MHZ 工作频率

9.2 DDR2/3 SDRAM读操作协议

DDR2/3 SDRAM 读操作的协议如图 11-2 所示。在图中命令（Command，简称 CMD）由 RAS_n，CAS_n 和 WE_n，共三个信号组成。对于读操作，RAS_n=1，CAS_n＝0，WE_n＝1。

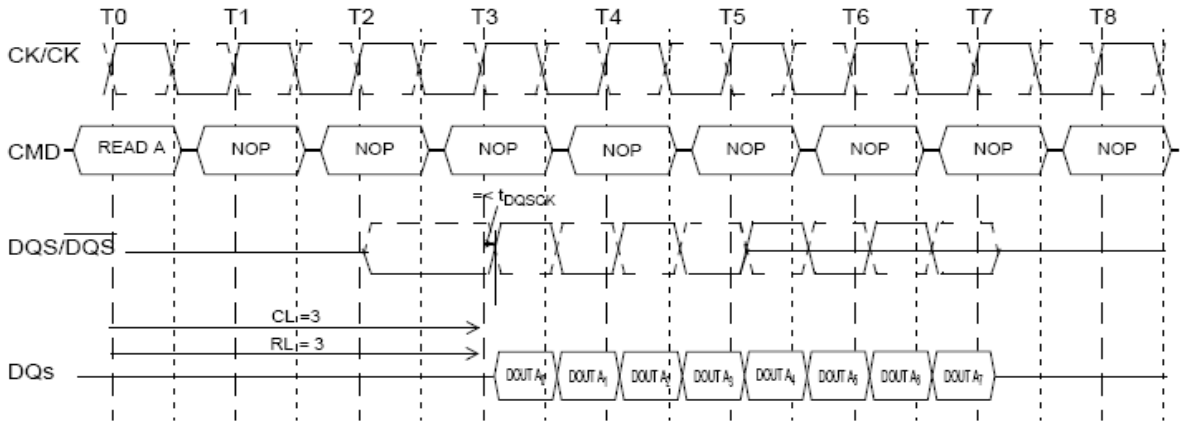


图 9-1 DDR2 SDRAM 读操作协议

上图中，Cas Latency (CL) = 3，Read Latency (RL) = 3，Burst Length = 8。

9.3 DDR2/3 SDRAM 写操作协议

DDR2/3 SDRAM 写操作的协议如图 11-3 所示。在图中命令 CMD 是由 RAS_n，CAS_n 和 WE_n，共三个信号组成的。对于写操作，RAS_n=1，CAS_n＝0，WE_n＝0。另外，与读操作不同，写操作需要 DQM 来标识写操作的掩码，即需要写入的字节数。DQM 与图中 DQs 信号同步。

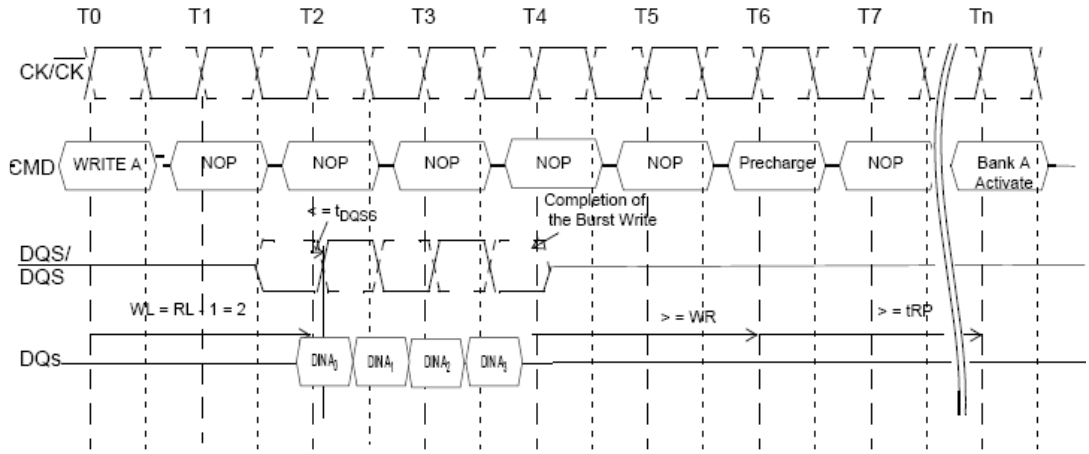


图 9-2 DDR2 SDRAM 写操作协议

上图中，Cas Latency (CL) = 3，Write Latency (WL) = Read Latency (RL) - 1 = 2，Burst Length = 4。

9.4 DDR2/3 SDRAM参数配置格式

内存控制器软件可见的参数列表及说明如下表：

	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
)x000	dll_close_disable dll_sync_disable	Dll_adj_cnt	Dll_value_ck(RD)		Dll_init_done(RD)		Version(RD)	
)x008								
)x010								
)x018	Dll_ck_3	Dll_ck_2	Dll_ck_1	Dll_ck_0	Dll_increment	Dll_start_point	Dll_bypass	Init_start
)x020	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
)x028	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
)x030	Enzi_end_0	Enzi_begin_0	Wrclk_sel_0	Wrdq_clkdelay_0	Odt_oe_end_0	Odt_oe_begin_0	Odt_stop_edge_0	Odt_start_edge_0
)x038	Enzi_stop_0	Enzi_start_0	Dll_oe_shorten_0	Dll_rddqs_n_0	Dll_rddqs_p_0	Dll_wrdqs_0	Dll_wrdata_0	Dll_gate_0
)x040	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
)x048	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
)x050	Enzi_end_1	Enzi_begin_1	Wrclk_sel_1	Wrdq_clkdelay_1	Odt_oe_end_1	Odt_oe_begin_1	Odt_stop_edge_1	Odt_start_edge_1
)x058	Enzi_stop_1	Enzi_start_1	Dll_oe_shorten_1	Dll_rddqs_n_1	Dll_rddqs_p_1	Dll_wrdqs_1	Dll_wrdata_1	Dll_gate_1
)x060	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
)x068	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
)x070	Enzi_end_2	Enzi_begin_2	Wrclk_sel_2	Wrdq_clkdelay_2	Odt_oe_end_2	Odt_oe_begin_2	Odt_stop_edge_2	Odt_start_edge_2
)x078	Enzi_stop_2	Enzi_start_2	Dll_oe_shorten_2	Dll_rddqs_n_2	Dll_rddqs_p_2	Dll_wrdqs_2	Dll_wrdata_2	Dll_gate_2
)x080	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
)x088	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
)x090	Enzi_end_3	Enzi_begin_3	Wrclk_sel_3	Wrdq_clkdelay_3	Odt_oe_end_3	Odt_oe_begin_3	Odt_stop_edge_3	Odt_start_edge_3
)x098	Enzi_stop_3	Enzi_start_3	Dll_oe_shorten_3	Dll_rddqs_n_3	Dll_rddqs_p_3	Dll_wrdqs_3	Dll_wrdata_3	Dll_gate_3
)x0A0	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
)x0A8	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
)x0B0	Enzi_end_4	Enzi_begin_4	Wrclk_sel_4	Wrdq_clkdelay_4	Odt_oe_end_4	Odt_oe_begin_4	Odt_stop_edge_4	Odt_start_edge_4
)x0B8	Enzi_stop_4	Enzi_start_4	Dll_oe_shorten_4	Dll_rddqs_n_4	Dll_rddqs_p_4	Dll_wrdqs_4	Dll_wrdata_4	Dll_gate_4
)x0C0	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
)x0C8	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
)x0D0	Enzi_end_5	Enzi_begin_5	Wrclk_sel_5	Wrdq_clkdelay_5	Odt_oe_end_5	Odt_oe_begin_5	Odt_stop_edge_5	Odt_start_edge_5
)x0D8	Enzi_stop_5	Enzi_start_5	Dll_oe_shorten_5	Dll_rddqs_n_5	Dll_rddqs_p_5	Dll_wrdqs_5	Dll_wrdata_5	Dll_gate_5
)x0E0	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
)x0E8	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
)x0F0	Enzi_end_6	Enzi_begin_6	Wrclk_sel_6	Wrdq_clkdelay_6	Odt_oe_end_6	Odt_oe_begin_6	Odt_stop_edge_6	Odt_start_edge_6
)x0F8	Enzi_stop_6	Enzi_start_6	Dll_oe_shorten_6	Dll_rddqs_n_6	Dll_rddqs_p_6	Dll_wrdqs_6	Dll_wrdata_6	Dll_gate_6
)x100	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
)x108	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
)x110	Enzi_end_7	Enzi_begin_7	Wrclk_sel_7	Wrdq_clkdelay_7	Odt_oe_end_7	Odt_oe_begin_7	Odt_stop_edge_7	Odt_start_edge_7
)x118	Enzi_stop_7	Enzi_start_7	Dll_oe_shorten_7	Dll_rddqs_n_7	Dll_rddqs_p_7	Dll_wrdqs_7	Dll_wrdata_7	Dll_gate_7
)x120	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
)x128	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
)x130	Enzi_end_8	Enzi_begin_8	Wrclk_sel_8	Wrdq_clkdelay_8	Odt_oe_end_8	Odt_oe_begin_8	Odt_stop_edge_8	Odt_start_edge_8
)x138	Enzi_stop_8	Enzi_start_8	Dll_oe_shorten_8	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_reset_po	Pad_adj_code_clk	Pad_code_lk	Pad_adj_code_cmd	Pad_code_cmd	Pad_adj_code_addr	Pad_code_addr
0x158		Pad_comp_code_o	Pad_comp_okn		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	Ba_xor_row_offset	Addr_mirror	Cmd_delay	Burst_length	Bank/Cs_resync	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		Rfifo_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			
0x320	tRESYNC_length	tRESYNC_shift	tRESYNC_max	tRESYNC_min	Pre_predict		tXS	tREF_low
0x328								tRESYNC_delay
0x330	Stat_en	Rdbuffer_max	Retry	Wr_pkg_num	Rwq_rb	Stb_en	Addr_new	tRDQidle
0x338				Rd_fifo_depth	Retry_cnt			
0x340	tREFretention					Ref_num	tREF_IDLE	Ref_sch_en
0x348								
0x350	Lpbk_data_en							
0x358						Lpbk_ecc_mask_en	Lpbk_ecc_en	Lpbk_data_mask_en
0x360			Int_ecc_cnt_fatal	Int_ecc_cnt_error	Ecc_cnt_cs_3	Ecc_cnt_cs_2	Ecc_cnt_cs_1	Ecc_cnt_cs_0
0x368								
0x370	Prior_age3		Prior_age2		Prior_age1		Prior_age_0	
0x378								Row_hit_place
0x380	Zq_cnt_1				Zq_cnt_0			
0x388	Zq_cnt_3				Zq_cnt_2			

9.5 软件编程指南

9.5.1 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。

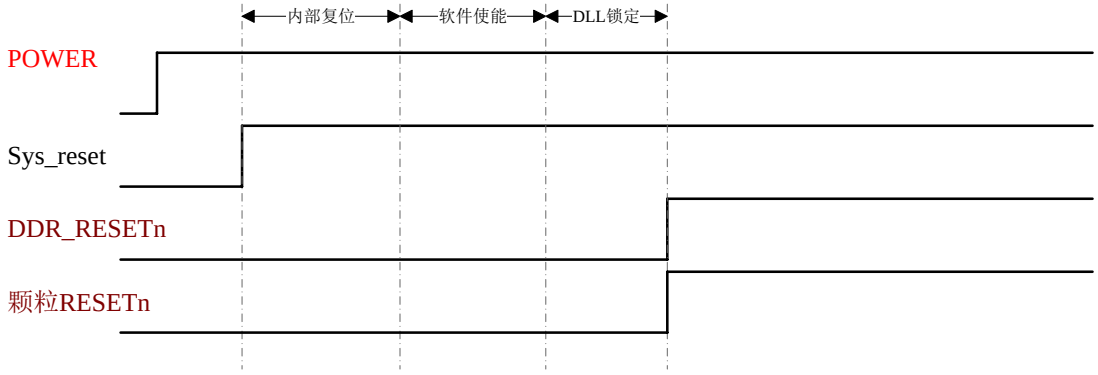
9.5.2 复位引脚的控制

为了在 STR 等状态下更加简单地控制复位引脚，可以通过 reset_ctrl（0x150）寄存器进行特别的复位引脚（DDR_RESETn）控制，主要的控制模式有两种：

(1) 一般模式，reset_ctrl[1:0] == 2’b00。这种模式下，复位信号引脚的行为与一般的控制模式相兼容。主板上直接将 DDR_RESETn 与内存槽上的对应引脚相连。引脚的行为是：

- 未上电时：引脚状态为低；
- 上电时：引脚状态为低；
- 控制器开始初始化时，引脚状态为高；
- 正常工作时，引脚状态为高。

时序如下图所示：

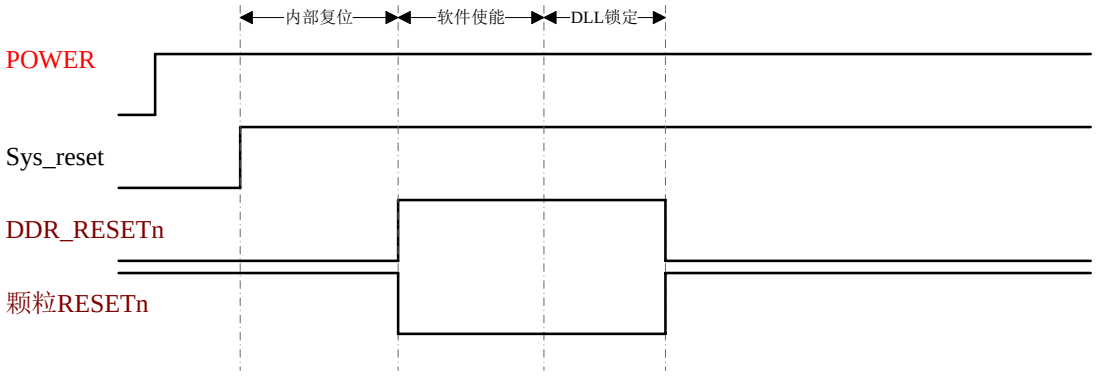


(2) 反向模式，reset_ctrl[1:0] == 2’b10。这种模式下，复位信号引脚在进行内存实际控制的时候，有效电平与一般的控制模式相反。所以主板上需要将

DDR_RESETn 通过反向器与内存槽上的对应引脚相连。引脚的行为是：

- 未上电时：引脚状态为低；
- 上电时：引脚状态为低；
- 控制器开始配置时：引脚状态为高；
- 控制器开始初始化时：引脚状态为低；
- 正常工作时：引脚状态为低。

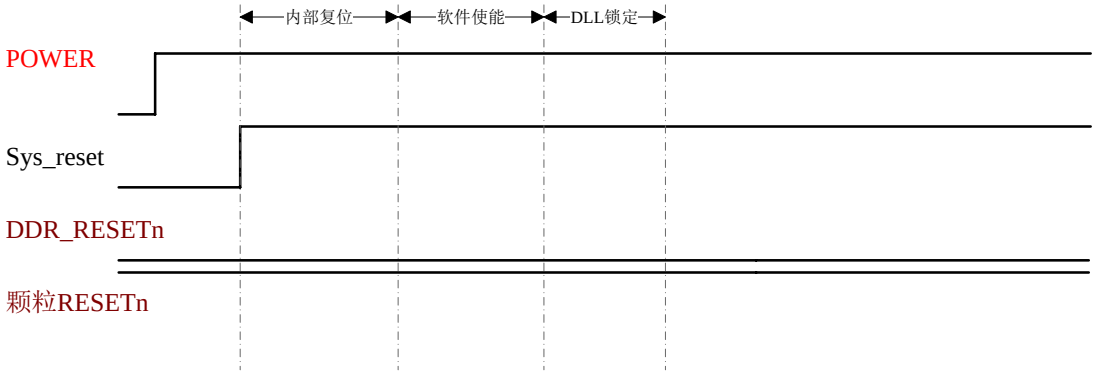
时序如下图所示：



(3) 复位禁止模式，pm_reset_ctrl[1:0] == 2’b01。这种模式下，复位信号引脚在整个内存工作期间，保持低电平。所以主板上需要将 DDR_RESETn 通过反向器与内存槽上的对应引脚相连。引脚的行为是：

- 始终为低；

时序如下图所示：



由后两种复位模式相配合，就可以直接在使用内存控制器的复位信号的情况下实现 STR 控制。当整个系统从关闭状态下启动时，使用（2）中的方法来使用内存条正常复位并开始工作。当系统从 STR 中恢复的时候，使用（3）中的方法来重新配置内存条，使得在不破坏内存条原有状态的条件上使其重新开始正常工作。

9.5.3 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。

9.5.3.1 Write Leveling

(1) 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。

(2) 完成控制器初始化，参见上一小节内容；

(3) 将 Dll_wrdqs_x（x = 0…8）设置为 0；

(4) 设置 Lvl_mode（0x180）为 2’b01；

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

(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 5-7；如果为 1，则表示 Write Leveling 操作已经成功；

(9) 此时 Dll_wrdqs_x 的值就应该是正确的设置值。

(10) 至此 Write Leveling 操作结束。如果这个过程中，第一次采样就发现 Lvl_resp_x 为 1，则这个结果是有问题的，应该检查其它的寄存器是否有错误的设置，这些寄存器可能包括 Wrdqs_lt_half、Dqs_start_edge、Dqs_stop_edge、Dqs_oe_begin、Dqs_oe_end。

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

(12) 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；

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

(14) 判断是否存在以下情况：不同的 Dll_wrdata_x 值在 0x40 附近，且有跨越 0x40 边界的情况出现（指有的 Dll_wrdata_x 略小于 0x40，有的 Dll_wrdata_x 略大于

0x40）。如果出现这种情况，设置对应 Wrdata_lt_half_x == 0 数据组的 Write_clk_delay_x 为 1。再将 tPHY_WRDATA 与 tRDDATA 的值减 1；

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

9.5.3.2 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；

(9) 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 6-9；如果为 1，则表示 Gate Leveling 操作已经成功；

(10)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。

(11)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。

(12)此时如果 Dll_gate_x 的值大于 0x40，则将 Dll_gate_x 的值减去 0x40；否则将其设为 0 即可。

(13)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 4，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

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

9.5.4 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。

9.5.5 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（0x168），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。

9.5.6 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；

(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。

9.5.7 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。

10 HyperTransport 控制器

龙芯 3A3000/3B3000 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)可通过引脚配置来支持片间 Cache 一致性传输，而 HT1 控制器(初始地址为 0x0E00_0000_0000 - 0x0FFF_FFFF_FFFF)可通过软件配置来支持片间 Cache 一致性维护，详见 10.7 节。

HyperTransport 控制器最高支持双向 16 位宽度以及 2.4GHz 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 3A3000/3B3000 HyperTransport 控制器的主要特征如下：

- 支持 HT1.0/HT3.0 协议
- 支持 200/400/800/1600/2000/2400MHz 运行频率
- HT1.0 支持 8 位宽度
- HT3.0 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
- 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

		HT0_Lo：address[40] = 0； HT0_Hi：address[40] = 1; 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。 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， HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制； 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， HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制； 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， HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制； 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。 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， HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。 HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Hi_Rstn	总线 Rstn	HyperTransport 总线 Rstn 信号， HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。 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， HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。 HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Hi_Ldt_Reqn	总线 Ldt_Reqn	HyperTransport 总线 Ldt_Reqn 信号， HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Rx_CLKp[1:0] HT0_Rx_CLKn[1:0] HT0_Tx_CLKp[1:0] HT0_Tx_CLKp[1:0]	CLK[1:0]	HyperTransport 总线 CLK 信号 HT0_8x2 为 1 时，CLK[1]由 HT0_Hi 控制 CLK[0]由 HT0_Lo 控制 HT0_8x2 为 0 时，CLK[1:0]由 HT0_Lo 控制
HT0_Rx_CTLp[1:0] HT0_Rx_CTLn[1:0] HT0_Tx_CTLp[1:0] HT0_Tx_CTLn[1:0]	CTL[1:0]	HyperTransport 总线 CTL 信号 HT0_8x2 为 1 时，CTL[1]由 HT0_Hi 控制 CTL[0]由 HT0_Lo 控制 HT0_8x2 为 0 时，CTL[1]无效 CTL[0]由 HT0_Lo 控制
HT0_Rx_CADp[15:0] HT0_Rx_CADn[15:0] HT0_Tx_CADp[15:0] HT0_Tx_CADn[15:0]	CAD[15:0]	HyperTransport 总线 CAD 信号 HT0_8x2 为 1 时，CAD[15:8]由 HT0_Hi 控制 CAD[7:0]由 HT0_Lo 控制 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 3A3000/3B3000 的 HyperTransport 总线支持 1.03/3.0 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 1 - Posted bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：Don't Care	bit 5：必为 1，POSTED bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：必为 1
01xxxx	NPC	Read	bit 3：Don't Care bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：Don't Care	bit 3：Don't Care bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：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 1 - Posted bit 2：0 – Byte 1 – Doubleword bit 0：必为 0	bit 5：必为 1，POSTED bit 2：0 – Byte 1 – Doubleword bit 0：必为 1
010x0x	NPC	Read	bit 2：0 – Byte 1 – Doubleword bit 0：Don't Care	bit 2：0 – Byte 1 – Doubleword 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 10.5.8 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 3A3000/3B3000 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 3A3000/3B3000 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
0x0C00_0000_0000	0x0CFF_FFFF_FFFF	1 Tbytes	HT0_LO 窗口
0x0D00_0000_0000	0x0DFF_FFFF_FFFF	1 Tbytes	HT0_HI 窗口
0x0E00_0000_0000	0x0EFF_FFFF_FFFF	1 Tbytes	HT1_LO 窗口
0x0F00_0000_0000	0x0FFF_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 2.6 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 3A3000/3B3000 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 3A3000/3B3000 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。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 3A3000/3B3000 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 HT 控制器端为 0xFD_FB00_0000 到 0xFD_FBFF_FFFF。HT 控制器中所有软件可见寄存器如下表所示：

表 10-7 软件可见寄存器列表

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）
0x148		HT 总线接收地址窗口 3 使能（外部访问）
0x14c		HT 总线接收地址窗口 3 基址（外部访问）
0x150		HT 总线接收地址窗口 4 使能（外部访问）
0x154		HT 总线接收地址窗口 4 基址（外部访问）
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	可预取地址窗口配置寄存器	HT 总线可预取地址窗口 0 使能（内部访问）
0xE4		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（外部访问）
0xF4		HT 总线 Uncache 地址窗口 0 基址（外部访问）
0xF8		HT 总线 Uncache 地址窗口 1 使能（外部访问）
0xFC		HT 总线 Uncache 地址窗口 1 基址（外部访问）
0x168		HT 总线 Uncache 地址窗口 2 使能（外部访问）
0x16C		HT 总线 Uncache 地址窗口 2 基址（外部访问）
0x170		HT 总线 Uncache 地址窗口 3 使能（外部访问）
0x174		HT 总线 Uncache 地址窗口 3 基址（外部访问）
0x158	P2P 地址窗口配置寄存器	HT 总线 P2P 地址窗口 0 使能（外部访问）
0x15C		HT 总线 P2P 地址窗口 0 基址（外部访问）
0x160		HT 总线 P2P 地址窗口 1 使能（外部访问）
0x164		HT 总线 P2P 地址窗口 1 基址（外部访问）
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	PHY 阻抗匹配配置寄存器	用于配置 PHY 发送端和接收端的阻抗匹配配置
0x110	Revision ID 寄存器	用于配置控制器版本
0x118	Error Retry 控制寄存器	Retry Count Rollover,Short Retry Attempts
0x11C	Retry Count 寄存器	用于 HyerTransport 3.0 模式下错误重传计数
0x130	Link Train 寄存器	HyperTransport 3.0 链路初始化及链路训练控制
0x134	Training 0 超时短计数寄存器	用于 Training 0 短计时超时阈值配置
0x138	Training 0 超时长计数寄存器	用于 Training 0 长计数超时阈值配置
0x13C	Training 1 计数寄存器	用于 Training 1 计数阈值配置

0x140	Training 2 计数寄存器	用于 Training 2 计数阈值配置
0x144	Training 3 计数寄存器	用于 Training 3 计数阈值配置
0x178	软件频率配置寄存器	实现控制器在工作过程的频率切换
0x17C	PHY 配置寄存器	用于配置 PHY 相关的物理参数
0x180	链路初始化调试寄存器	用于忽略 PHY CDR lock 信号，并自定义等待时间
0x184	LDT 调试寄存器	用于配置 LDT 信号无效到链路开始初始化的时间

每个寄存存器的具体含义如下节如示。如无特别说明，寄存器位域描述为对应位为 1 的情况。

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： 0->1：HT_RSTn 置 0，总线复位 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 /0x1	R/W	HOST/SLAVE 模式 初始值由引脚 HOSTMODE 决定 HOSTMODE 上拉：0 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 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 _disable	1	0x0		相位选择使能 0：使能相位选择功能 1：禁用相位选择功能
30:28	Link Width Out	3	0x0	R/W	发送端宽度 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 000：8 位方式 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 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 000：8 位方式 001：16 位方式 其它：错误
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 1：关闭 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 处于 16 位总线工作方式时 1：关闭 高/低 8 位 HT PHY 0：使能 低 8 位 HT PHY， 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， 高 8 位 PHY 关闭控制 1：关闭 高 8 位 HT PHY 0：使能 高 8 位 HT PHY

偏移量： 0x48
复位值： 0x80250023（与硬件配置相关）
名称： Revision ID，Link Freq，Link Error，Link Freq Cap
频率设置需要与 HT1.0/3.0 相对应。HT1.0 时，对应的频率应设置为 200MHz - 800MHz，HT3.0 时，对应频率应设置为 1.0GHz - 3.2GHz。

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，根据外部 PLL 的设置产生不同的值（当使用软件配置 PLL（0x178）时，该位无意义） {3.2G,2.6G,2.4G,2.2G,2.0G,1.8G,1.6G,1.4G,1.2G,1.0G,800M,600M,500M,400M,300M,200M}
15:14	Reserved	2	0x0		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， 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 写入此寄存器的值后将在下次热复位或是 HT Disconnect 之后生效，设置的值与 Link Freq Cap 的位相对应（硬件配置时）
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

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 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 正确的方法是先置 0 再置 1：0 -> 1 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） 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： 000：SMI 001：NMI 010：INIT 011：Reservered 100：INTA 101：INTB 110：INTC 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 1：使用 Dword Write 0：使用 Byte Write（带 MASK）
22	Coherent Mode	1	0x0	R	处理器一致性模式 由引脚 ICCC_EN 决定
21	Not Care Seqid	1	0x0	R/W	设置为不关心 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 1：不转换 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 0x0：最高优先级 0xF：最低优先级 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 0x0：禁止 0x1：使能
15:8	rx_ctl_catch	24	0x0	R/W	保存采样得到的输入 ctl （0、2、4、6）对应 CTL0 采样的四个相位 （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 0 中断向量寄存器 0x0：ht_int_stripe_1 0x1：ht_int_stripe_2 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 P2P 窗口内的 HT 地址将作为 P2P 命令转发回 HT 总线，落在正常接收窗口内且不在 P2P 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]

位域	位域名称	位宽	复位值	访问	描述
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]

偏移量： 0x148
复位值： 0x00000000
名称： HT 总线接收地址窗口 3 使能（外部访问）

表 10-23 HT 总线接收地址窗口 3 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_image3_en	1	0x0	R/W	HT 总线接收地址窗口 3，使能信号
30	ht_rx_image3_ trans_en	1	0x0	R/W	HT 总线接收地址窗口 3，映射使能信号
29:0	ht_rx_image3_ trans[53:24]	16	0x0	R/W	HT 总线接收地址窗口 3，转译后地址的[53:24]

偏移量： 0x14C
复位值： 0x00000000
名称： HT 总线接收地址窗口 3 基址（外部访问）

表 10-24 HT 总线接收地址窗口 3 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_image3_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 3，地址基址的[39:24]
15:0	ht_rx_image3_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 3，地址屏蔽的[39:24]

偏移量：　　0x150
复位值：　　0x00000000
名称：　　HT 总线接收地址窗口 4 使能（外部访问）

表 10-25 HT 总线接收地址窗口 4 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_image4_en	1	0x0	R/W	HT 总线接收地址窗口 4，使能信号
30	ht_rx_image4_trans_en	1	0x0	R/W	HT 总线接收地址窗口 4，映射使能信号
29:0	ht_rx_image4_trans[53:24]	16	0x0	R/W	HT 总线接收地址窗口 4，转译后地址的[53:24]

偏移量：　　0x154
复位值：　　0x00000000
名称：　　HT 总线接收地址窗口 4 基址（外部访问）

表 10-26 HT 总线接收地址窗口 4 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_image4_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 4，地址基址的[39:24]
15:0	ht_rx_image4_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 4，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 10-27 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 10-28 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

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 10-29 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 10-30 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 10-31 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 10-31 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 10-32 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 10-33 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 10-34 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]， 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 10-35 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]， 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 10-36 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]， 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 10-37 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

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]，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 10-38 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 10-39 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 10-40 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 10-41 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 10-42 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 10-43 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 10-44 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 10-45 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 10-46 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	ht_depart0_en	1	0x0	R/W	HT 访问拆包使能(对应于 CPU 核的对外 uncache ACC 操作窗口)
29:23	Reserved	14	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 10-47 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 10-48 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

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	ht_depart1_en	1	0x0	R/W	HT 访问拆包使能(对应于 CPU 核的对外 uncache ACC 操作窗口)
29:16	Reserved	14	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 10-49 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 10-50 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 10-51 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 10-52 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 10-53 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 10-54 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 10-55 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 10-56 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 10-57 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

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]

偏移量： 0x168
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 2 使能（内部访问）

表 10-58 HT 总线 Uncache 地址窗口 2 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 2，使能信号
30	ht_uncache1_ trans_en	1	0x0	R/W	HT 总线 uncache 地址窗口 2，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 2，转译后地址的[53:24]

偏移量： 0x16c
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 2 基址（内部访问）

表 10-59 HT 总线 Uncache 地址窗口 2 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 2，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 2，地址屏蔽的[39:24]

偏移量： 0x170
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 3 使能（内部访问）

表 10-60 HT 总线 Uncache 地址窗口 3 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 3，使能信号
30	ht_uncache1_ trans_en	1	0x0	R/W	HT 总线 uncache 地址窗口 3，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 3，转译后地址的[53:24]

偏移量：　　0x174
复位值：　　0x00000000
名称：　　HT 总线 Uncache 地址窗口 3 基址（内部访问）

表 10-61 HT 总线 Uncache 地址窗口 3 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 3，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 3，地址屏蔽的[39:24]

10.5.14 P2P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，直接作为 P2P 命令转发回总线，相对于正常接收窗口和 Uncache 窗口，该窗口具有最高优先级。

偏移量：　　0x158
复位值：　　0x00000000
名称：　　HT 总线 P2P 地址窗口 0 使能（外部访问）

表 10-62 HT 总线 P2P 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_image2_en	1	0x0	R/W	HT 总线 P2P 地址窗口 0，使能信号
30	ht_rx_image2_trans_en	1	0x0	R/W	HT 总线 P2P 地址窗口 0，映射使能信号
29:0	ht_rx_image2_trans[53:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 0，转译后地址的[53:24]

偏移量：　　0x15c
复位值：　　0x00000000
名称：　　HT 总线 P2P 地址窗口 0 基址（外部访问）

表 10-63 HT 总线 P2P 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_image2_base[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址基址的[39:24]

位域	位域名称	位宽	复位值	访问	描述
15:0	ht_rx_image2_ mask[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址屏蔽的[39:24]

偏移量：　　0x160
复位值：　　0x00000000
名称：　　HT 总线 P2P 地址窗口 1 使能（外部访问）

表 10-64 HT 总线 P2P 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_image2_en	1	0x0	R/W	HT 总线 P2P 地址窗口 1，使能信号
30	ht_rx_image2_ trans_en	1	0x0	R/W	HT 总线 P2P 地址窗口 1，映射使能信号
29:0	ht_rx_image2_ trans[53:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，转译后地址的[53:24]

偏移量：　　0x164
复位值：　　0x00000000
名称：　　HT 总线 P2P 地址窗口 1 基址（外部访问）

表 10-65 HT 总线 P2P 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_image2_ base[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址基址的[39:24]
15:0	ht_rx_image2_ mask[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址屏蔽的[39:24]

10.5.15 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 10-66 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

位域	位域名称	位宽	复位值	访问	描述
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

10.5.16 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 10-67 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

10.5.17 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 10-68 发送缓存调试寄存器

位域	位域名称	位宽	复位值	访问	描述
31:30	Reserved	2	0x0	R	保留
29	Tx_neg	1	0x0	R/W	发送端缓存调试符号 0：增加相应个数 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 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 当 tx_neg 为 0 时，增加 R_DATA_txadj 个； 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 当 tx_neg 为 0 时，增加 NPC_DATA_txadj 个； 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 当 tx_neg 为 0 时，增加 PC_DATA_txadj 个； 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 当 tx_neg 为 0 时，增加 B_CMD_txadj 个； 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 当 tx_neg 为 0 时，增加 R_CMD_txadj 个； 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 当 tx_neg 为 0 时，增加 NPC_CMD_txadj 个； 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 当 tx_neg 为 0 时，增加 PC_CMD_txadj 个； 当 tx_neg 为 1 时，减少 PC_CMD_txadj+1 个

10.5.18 PHY阻抗匹配控制寄存器

用于控制 PHY 的阻抗匹配使能，发送端和接收端阻抗匹配参数设置

偏移量： 0x10C
复位值： 0x00000000
名称： PHY 阻抗匹配控制寄存器

表 10-69 阻抗匹配控制寄存器

位域	位域名称	位宽	复位值	访问	描述
31	Tx_scanin_en	1	0x0	R/W	TX 阻抗匹配使能
30	Rx_scanin_en	1	0x0	R/W	RX 阻抗匹配使能
27:24	Tx_scanin_ncode	4	0x0	R/W	TX 阻抗匹配扫描输入 ncode
23:20	Tx_scanin_pcode	4	0x0	R/W	TX 阻抗匹配扫描输入 pcode
19:12	Rx_scanin_code	8	0x0	R/W	RX 阻抗匹配扫描输入

10.5.19 Revision ID 寄存器

用于配置控制器版本,配置成新的版本号，通过 Warm Reset 生效。

偏移量： 0x110
复位值： 0x00200000
名称： RevisionID 寄存器

表 10-70 Revision ID 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
23:16	Revision ID	8	0x20	R/W	Revision ID 控制寄存器 0x20：HyperTransport 1.00 0x60：HyperTransport 3.00
15:0	Reserved	16	0x0	R	保留

10.5.20 Error Retry 控制寄存器

用于 HyerTransport 3.0 模式下错误重传使能，配置 Short Retry 的最大次数，显示 Retry 计数器是否翻转。

偏移量： 0x118
复位值： 0x00000000
名称： Error Retry 控制寄存器

表 10-71 Error Retry 控制寄存器

位域	位域名称	位宽	复位值	访问	描述
31:10	Reserved	22	0x0	R	保留
9	Retry Count Rollover	1	0x0	R	Retry 计数器计数翻转
8	Reserved	1	0x0	R	保留
7:6	Short Retry Attempts	2	0x0	R/W	允许的最大 Short Retry 次数

10.5.21 Retry Count 寄存器

用于 HyerTransport 3.0 模式下错误重传计数。

偏移量： 0x11C
复位值： 0x00000000
名称： Retry Count 寄存器

表 10-72 Retry Count 寄存器

位域	位域名称	位宽	复位值	访问	描述
31:20	Reserved	12	0x0	R	保留
19:16	Rrequest delay	4	0x0	R/W	用于在一致性模式下，控制 Rrequest 传输的随机延迟范围 000：0 延迟 001：随机延迟 0-8 010：随机延迟 8-15 011：随机延迟 16-31 100：随机延迟 32-63 101：随机延迟 64-127 110：随机延迟 128-255 111：0 延迟
15:0	Retry Count	16	0x0	R	Retry 计数

10.5.22 Link Train 寄存器

HyperTransport 3.0 链路初始化及链路训练控制寄存器。

偏移量： 0x130
复位值： 0x00000070
名称： Link Train 寄存器

表 10-73 Link Train 寄存器

位域	位域名称	位宽	复位值	访问	描述
31:23	Reserved	9	0x0	R	保留
22：21	Transmitter LS select	2	0x0	R/W	发送端在 Disconnected 或 Inactive 状态下的链路状态： 2’b00 LS0 2’b01 LS1 2’b10 LS2 2’b11 LS3
14	Dsiable Cmd Throttling	1	0x0	R/W	在 HyperTransport 3.0 模式下，默认任意 4 个连续的 DWS 中只能出现一个 Non-info CMD； 1’b0 使能 Cmd Throttling 1’b1 禁用 Cmd Throttling
13:10	Reserved	4	0x0	R	保留
8：7	Receiver LS select	2	0x0	R/W	接收端在 Disconnected 或 Inactive 状态下的链路状态： 2’b00 LS0 2’b01 LS1 2’b10 LS2 2’b11 LS3
6:4	Long Retry Count	3	0x7	R/W	Long Retry 最大次数
3	Scrambling Enable	1	0x0	R/W	是否使能 Scramble 0: 禁用 Scramble 1: 使能 Scramble
2	8B10B Enable	1	0x0	R/W	是否使能 8B10B 0: 禁用 8B10B 1: 使能 8B10B
1	AC	1	0x0	R	是否检测到 AC mode 0: 没有检测到 AC mode 1: 检测到 AC mode
0	Reserved	1	0x0	R	保留

10.5.23 Training 0 超时短计时寄存器

用于配置 HyerTransport 3.0 模式下 Training 0 短计时超时阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x134
复位值： 0x00000080
名称： Training 0 超时短计数寄存器

表 10-74 Training 0 超时短计时寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T0 time	32	0x8	R/W	Training 0 超时短计时寄存器

10.5.24 Training 0 超时长计时寄存器

用于 HyerTransport 3.0 模式下 Training 0 长计数超时阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x138
复位值： 0x000fffff
名称： Training 0 超时长计数寄存器

表 10-75 Training 0 超时长计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T0 time	32	0xfffff	R/W	Training 0 超时长计数寄存器

10.5.25 Training 1 计数寄存器

用于 HyerTransport 3.0 模式下 Training 1 计数阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x13C
复位值： 0x0004fffff
名称： Training 1 计数寄存器

表 10-76 Training 1 计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T1 time	32	0x4fffff	R/W	Training 1 计数寄存器

10.5.26 Training 2 计数寄存器

用于 HyerTransport 3.0 模式下 Training 2 计数阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x144
复位值： 0x0007fffff
名称： Training 2 计数寄存器

表 10-77 Training 2 计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T2 time	32	0x7fffff	R/W	Training 2 计数寄存器

10.5.27 Training 3 计数寄存器

用于 HyerTransport 3.0 模式下 Training 3 计数阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x13C
名称： Training 3 计数寄存器

表 10-78 Training 3 计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T3 time	32	0x7fffff	R/W	Training 3 计数寄存器

10.5.28 软件频率配置寄存器

用于实现控制器在工作过程中切换到任意协议和 PLL 支持的链路频率及控制器频率；

具体切换方法为：在使能软件配置模式的前提下，置位软件频率配置寄存器第 1 位，并写入新的时钟相关的参数，包括决定 PLL 输出频率的 div_refc 和 div_loop ,链路上的分频系数 phy_hi_div 和 phy_lo_div，以及控制器的分频系数 core_div 。之后进入 warm reset 或 LDT disconnect，控制器将会自动复位 PLL，配置新的时钟参数。

时钟频率的计算公式为：

HyperTransport 1.0:

PHY_LINK__CLK = 50MHz×div_loop / div_refc /phy_div

HT_CORE_CLK = 100MHz×div_loop /div_refc /core_div

HyperTransport 3.0:

PHY_LINK__CLK = 100MHz×div_loop/div_refc /phy_div

HT_CORE_CLK = 100MHz×div_loop /div_refc /core_div

等待 PLL 重新锁定的时间在缺省情况下，system clk 为 33M 时约为 30us；也可以在寄存器中写入自定义的等待计数上限；

偏移量： 0x178
复位值： 0x00000000
名称： 软件频率配置寄存器

表 10-79 软件频率配置寄存器

位域	位域名称	位宽	复位值	访问	描述
31：27	PLL relock	5	0x0	R/W	计数器上限配置寄存器，当置位 counter select

	counter				时，计数器计数上限为{PLL_relock_counter ,5'h1f}，否则计数上限为 10'3ff
26	Counter select	1	0x0	R/W	锁定计时器自定义使能： 1'b0 使用默认计数上限； 1'b1 由 PLL_relock_counter 计算得出
25：22	Soft_phy_lo_div	4	0x0	R/W	低位 PHY 分频系数
21：18	Soft_phy_hi_div	4	0x0	R/W	高位 PHY 分频系数
17：16	Soft_div_refc	2	0x0	R/W	PLL 内分频系数，表示分频值
15：9	Soft_div_loop	7	0x0	R/W	PLL 内倍频系数
8：5	Soft_core_div	4	0x0	R/W	控制器时钟分频系数
4：2	Reserved	3	0x0	R	保留
1	Soft cofig enable	1	0x0	R/W	软件配置使能位 1'b0 禁用软件频率配置 1'b1 使能软件频率配置
0	Reserved	1	0x0	R	保留

10.5.29 PHY 配置寄存器

用于配置 PHY 相关的物理参数，当控制器做为两个独立的 8bit 控制器时，高位的 PHY 和低位的 PHY 分别由两个控制器独立控制；当控制器作为 1 个 16bit 的控制器时，高位和低位的 PHY 的配置参数由低位控制器统一控制；

偏移量：　　0x17C
复位值：　　0x83308000
名称：　　PHY 配置寄存器

表 10-80 PHY 配置寄存器

位域	位域名称	位宽	复位值	访问	描述
31	Rx_ckpll_term	1	0x1	R/W	PLL 到 RX 端片上传输线终端阻抗
30	Tx_ckpll_term	1	0x0	R/W	PLL 到 TX 端片上传输线终端阻抗
29	Rx_clk_in_sel_	1	0x0	R/W	时钟 PAD 供给数据 PAD 的时钟选择，HT1 模式下自动选择为 CLKPAD: 1'b0 外来时钟源 1'b1 PLL 时钟
28	Rx_ckdll_sell	1	0x0	R/W	用来锁定 DLL 的时钟选择： 1'b0 PLL 时钟 1'b1 外来时钟源
27:26	Rx_ctle_bitc	2	0x0	R/W	PAD EQD 高频增益
25:24	Rx_ctle_bitr	2	0x3	R/W	PAD EQD 低频增益
23:22	Rx_ctle_bitlim	2	0x0	R/W	PAD EQD 补偿限制

21	Rx_en_ldo	1	0x1	R/W	LDO 控制 1'b0 LDO 禁用 1'b1 LDO 使能
20	Rx_en_by	1	0x1	R/W	BandGap 控制 1'b0 BandGap 禁用 1'b1 BandGap 使能
19：17	Reserved	3	0x0	R	保留
16:12	Tx_preenmp	5	0x08	R/W	PAD 预加重控制信号
11：0	Reserved	12	0x0	R	保留

10.5.30 链路初始化调试寄存器

用于配置在 HyperTransport 3.0 模式下，链路初始化过程中是否使用 PHY 提供的 CDR lock 信号做为链路 CDR 完成的标志；如果忽略该锁定信号，则需要控制器计数等待一定时间后默认 CDR 完成。

偏移量： 0x180
复位值： 0x00000000
名称： 链路初始化调试寄存器

表 10-81 链路初始化调试寄存器

位域	位域名称	位宽	复位值	访问	描述
15	Cdr_ignore_enable	1	0x0	R/W	链路初始化时是否忽略 CRC lock ，通过计数器计数完成等待： 1'b0 等待 CDR lock 1'b1 忽略 CDR lock 信号，通过计数器累加等待
14：0	Cdr_wait_counter	15	0x0	R/W	等待计数器计数上限，基于控制器时钟完成的技术

10.5.31 LDT调试寄存器

软件改变控制器频率后，会导致对 LDT reconnect 阶段计时不准确，需配置该计数器，作为软件配置频率后，LDT 信号无效到控制器开始链路初始化之间的时间,该计时基于控制器时钟。

偏移量： 0x184
复位值： 0x00000000
名称： LDT 调试寄存器

表 10-82 LDT 调试寄存器

位域	位域名称	位宽	复位值	访问	描述
31:16	Rx_wait_time	16	0x0	R/W	RX 端等待计数器的初值

15:0	Tx_wait_time	16	0x0	R/W	TX 端等待计数器的初值

10.6 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 10-5 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 3A3000/3B3000 中采用如下格式实现：

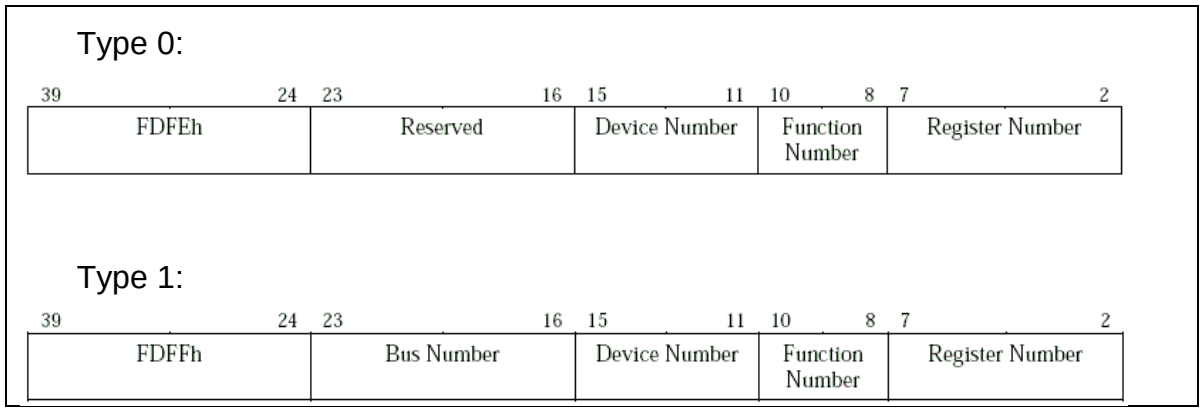


图 10-1 龙芯 3A3000/3B3000 中 HT 协议的配置访问

10.7 HyperTransport多处理器支持

龙芯 3 号处理器使用 HyperTransport 接口进行多处理器互联，并且可以硬件自动维护 4 个芯片之间的一致性请求。下面提供两种多处理器互联方法：

四片龙芯 3 号互联结构

四片 CPU 两两相联构成环状结构。每个 CPU 利用 HT0 的两个 8 位控制器与相邻两片相联，其中 HTx_LO 作为主设备，HTx_HI 作为从设备连接，由此而得到下图的互联结构：

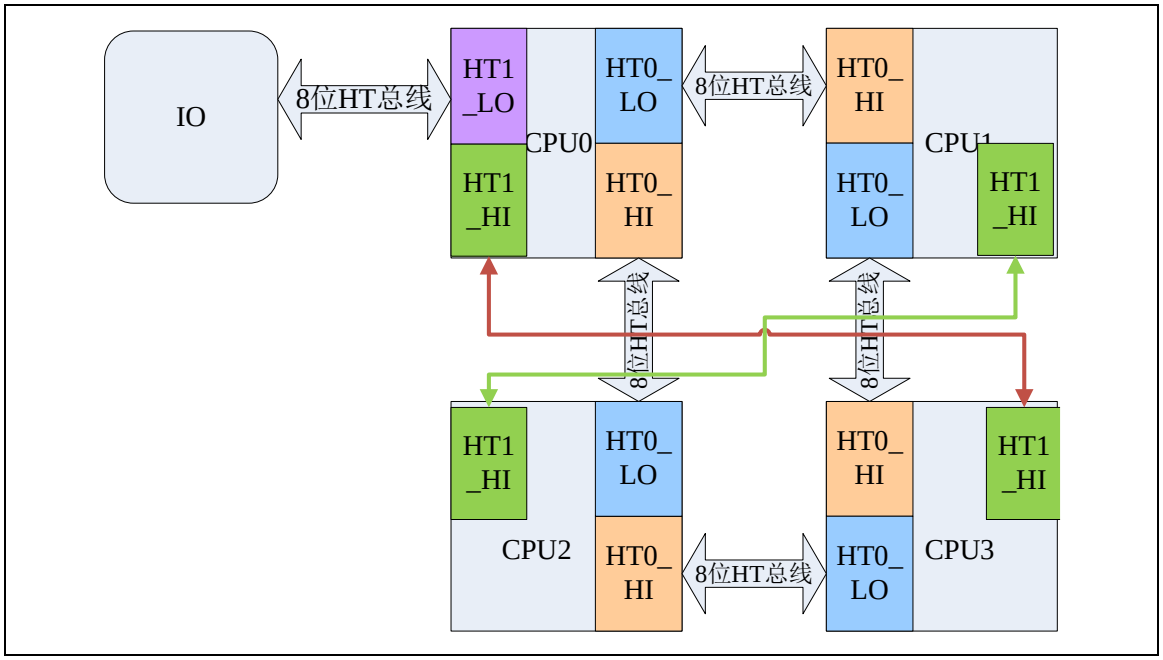


图 10-2 四片龙芯 3 号互联结构

龙芯 3 号互联路由

龙芯 3 号互联路由采用简单 X-Y 路由方法。即路由时，先 X 后 Y，以四片芯片为例，ID 号分别为 00，01，10，11。如果从 11 向 00 发出请求，则为 11 向 00 路由，首先走 X 方向，从 11 走到 10，再走 Y 方向，从 10 走到 00。而在请求的响应从 00 返回 11 时，路由首先走 X 方向，从 00 到 01，再走 Y 方向，从 01 到 11。可以看到，这是两个不同的路由线路。由于这个算法的特征，我们在构建两片芯片互联的时候，将采用不同的办法。

两片龙芯 3 号互联结构

由于固定路由算法的特性，我们在构建两片芯片互联时，有两种不同的方法。首先是采用 8 位 HT 总线互联。这种互联方式下，两个处理器之间只能采用 8 位 HT 互联。两个芯片号分别为 00 与 01，由路由算法，我们可以知道，两个芯片相互访问时都是通过与四片互联时一致的 8 位 HT 总线。如下所示：

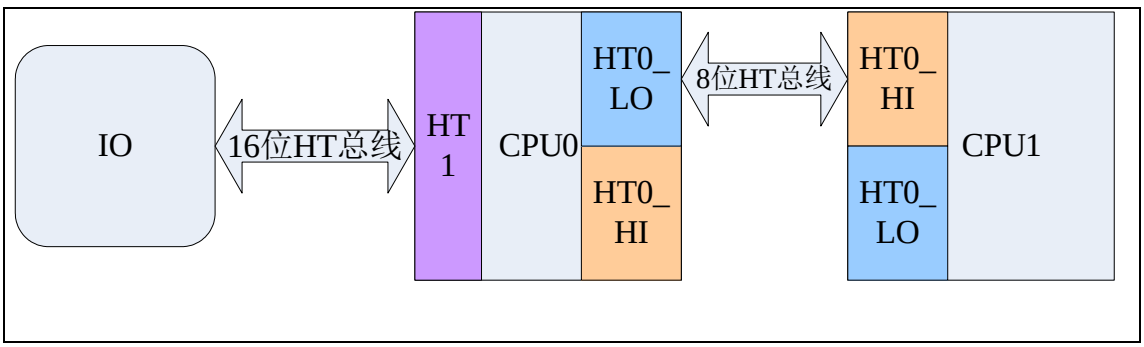


图 10-3 两片龙芯 3 号 8 位互联结构

但是，我们的 HT 总线最宽可以采用 16 位模式，由此最大化带宽的连接方式应该是采用 16 位互联结构。在龙芯三号中，只要把 HT0 控制器设置为 16 位模式，所有发到 HT0 控制器的命令都会被发往 HT0_LO，而不是以前的按照路由表分别发至 HT0_HI 或是 HT0_LO，这样，我们就可以在互联时使用 16 位总线。所以，我们只需要将 CPU0 与 CPU1 的 16 位模式正确配置并将高低位总线正确连接即可使用 16 位 HT 总线互联。而这种互联结构同时也可以使用 8 位的 HT 总线协议进行相互访问。所得到的互联结构如下：

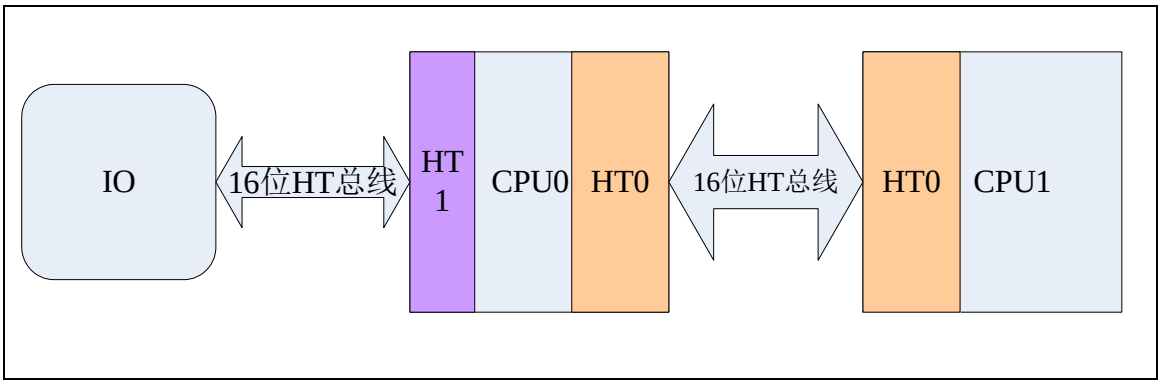


图 10-4 两片龙芯 3 号 16 位互联结构

11 低速 IO 控制器配置

龙芯 3 号 I/O 控制器包括 PCI 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 AXI 端口，CPU 的请求经过地址译码后发送到相应的设备。

11.1 PCI控制器

龙芯 3 号的 PCI 控制器既可以作为主桥控制整个系统，也可以作为普通 PC 设备工作在 PCI 总线上。它的实现符合 PCI 2.3 规范。龙芯 3 号的 PCI 控制器还内置了 PCI 仲裁器。

PCI 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 PCI 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 3A3000/3B3000 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 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 (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 (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 0: 超时后 1: 马上
28	tar_delay_retry	读写	0	target 访问重试策略 0: 根据内部逻辑（见 29 位） 1: 马上重试
27	tar_read_abort_en	读写	0	若 target 对内部的读请求超时，是否让以 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，是否让以 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 000: 8 周期 其它: 不支持
19:16	tar_init_timeout	读写	0000	target 初始延迟超时 PCI 模式下 0: 16 周期 1-7: 禁用计数器 8-15: 8-15 周期 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 0: 8 delay 访问 8: 1 delay 访问 9: 2 delay 访问 10: 3 delay 访问 11: 4 delay 访问 12: 5 delay 访问

				13: 6 delay 访问 14: 7 delay 访问 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） FFF: 64KB 到 16byte FFE: 64KB 到 32byte FF8: 64KB 到 128byte
3	tar_pref_bound_en	读写	0	使用 tar_pref_boundary 的配置 0: 预取到设备边界 1: 使用 tar_pref_boundary
2	Reserved	-	0	
1	tar_splitw_ctrl	读写	0	target split 写控制 0: 阻挡除 Posted Memory Write 以外的访问 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 0: 允许 master 访问超时 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 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 (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 (只对 PCI 有效)
27	mas_io_defer_cnt	读写	0	在外的最大 IO 请求数 0: 由控制 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） 0: 8 1-7: 1-7 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 0:
14	err_module	只读	0	出错的模块 0: target 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 0 最高, 3 最低
27:26	mas_req_priority	读写	0	master 对外的优先级 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) 0: 固定优先级 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 0: 无限重试 1-255: 1-255 次
7:0	mas_trdy_count	读写	00h	master TRDY 超时计数器 0: 禁用 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。设备号由根据 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。

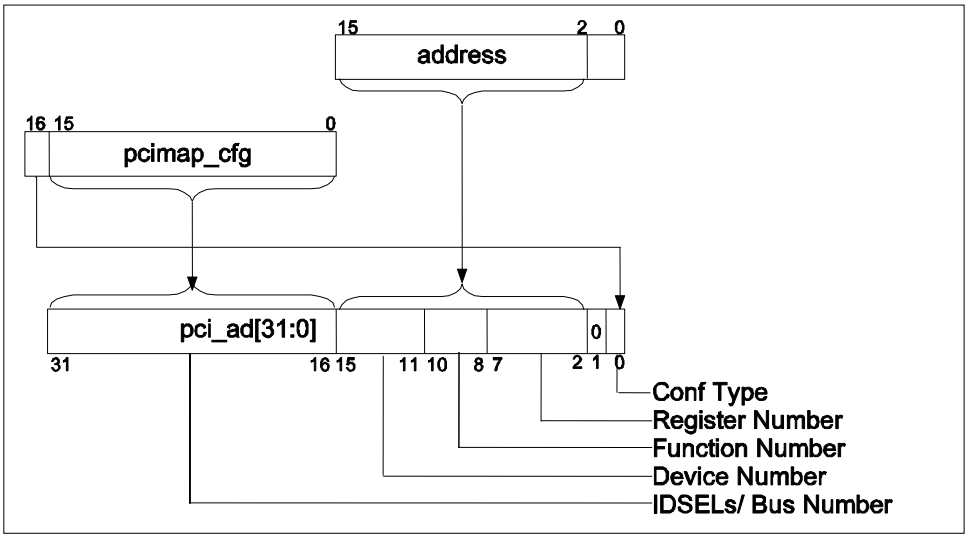


图 11-1 配置读写总线地址生成

PCI 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 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。内部 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_0000-0X1D00_0000	16MByte
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，当 PCI_CONFIG[0]引脚为下拉时，0xBFC00000 的地址被自动路由至 LPC。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。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[23:16]	LPC_MEM_TRANS	读写	0	LPC Memory 空间地址转换控制
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

芯片内部集成两个 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

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

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

位域	位域名称	位宽	访问	描述
7:4	Reserved	4	R	保留
3:1	II	3	R	中断源表示位，详见下表
0	INTp	1	R	中断表示位

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

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 ‘00’ – 1 字节 ‘01’ – 4 字节 ‘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

位域	位域名称	位宽	访问	描述
7	dlab	1	RW	分频锁存器访问位 ‘1’ – 访问操作分频锁存器 ‘0’ – 访问操作正常寄存器
6	bcb	1	RW	打断控制位 ‘1’ – 此时串口的输出被置为 0(打断状态). ‘0’ – 正常操作
5	spb	1	RW	指定奇偶校验位 ‘0’ – 不用指定奇偶校验位 ‘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 ‘0’ – 在每个字符中有奇数个 1（包括数据和奇偶校验位） ‘1’ –在每个字符中有偶数个 1
3	pe	1	RW	奇偶校验位使能 ‘0’ – 没有奇偶校验位 ‘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 ‘0’ – 1 个停止位 ‘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 '00' – 5 位 '01' – 6 位 '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

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 '0' – 正常操作 '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。 DTR ➔ DSR RTS ➔ CTS Out1 ➔ RI 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

位域	位域名称	位宽	访问	描述
7	ERROR	1	R	错误表示位 ‘1’ – 至少有奇偶校验位错误，帧错误或打断中断的一个。 ‘0’ – 没有错误
6	TE	1	R	传输为空表示位 ‘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 ‘0’ – 有数据
5	TFE	1	R	传输 FIFO 位空表示位 ‘1’ – 当前传输 FIFO 为空，给传输 FIFO 写数据时清零 ‘0’ – 有数据
4	BI	1	R	打断中断表示位 ‘1’ –接收到 起始位＋数据＋奇偶位＋停止位都是 0，即有打断中断 ‘0’ – 没有打断
3	FE	1	R	帧错误表示位 ‘1’ – 接收的数据没有停止位 ‘0’ – 没有错误
2	PE	1	R	奇偶校验位错误表示位 ‘1’ – 当前接收数据有奇偶错误 ‘0’ – 没有奇偶错误
1	OE	1	R	数据溢出表示位 ‘1’ – 有数据溢出 ‘0’ – 无溢出
0	DR	1	R	接收数据有效表示位 ‘0’ – 在 FIFO 中无数据

				‘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

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

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

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控制器

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
- 模式故障产生错误标志并发出中断请求
- 双缓冲接收器
- 极性和相位可编程的串行时钟
- 可在等待模式下对 SPI 进行控制
- 支持从 SPI 启动

SPI 控制器寄存器物理地址基址为 0x1FE00220。

表 11-6 SPI 控制器地址空间分布

地址名称	地址范围	大小
SPI Boot	0X1FC0_0000-0X1FD0_0000	1MByte
SPI Memory	0X1D00_0000-0X1E00_0000	16MByte
SPI Register	0X1FE0_0220-0X1FE0_0230	16Byte

SPI Boot 地址空间是系统启动时处理器最先访问的地址空间，当 PCI_CONFIG[0]引脚为上拉时，0xBFC00000 的地址被自动路由至 SPI。

SPI Memory 空间也可以通过 CPU 的读请求直接访问，其最低 1M 字节与 SPI BOOT 空间重叠。

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

位域	位域名称	位宽	访问	描述
7	Spie	1	RW	中断输出使能信号 高有效
6	spe	1	RW	系统工作使能信号高有效
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	时钟相位位 1 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

位域	位域名称	位宽	访问	描述
7	spif	1	RW	中断标志位 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 为 1 表示已经溢出,写 1 则清零
5:4	Reserved	2	RW	保留
3	wffull	1	RW	写寄存器满标志 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

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

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申请信号 00 – 1 字节　　01 - 2 字节 10 - 3 字节　　11 - 3 字节
5:2	Reserved	4	RW	保留
1:0	spre	2	RW	与 Spr 一起设定分频的比率

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

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

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

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 00: 1T 01: 2T 10: 4T 11: 8T

11.5 IO控制器配置

配置寄存器主要用于配置 PCI 控制器的地址窗口、仲裁器以及 GPIO 控制器。表 11-7 列出了这些寄存器，表 11-8 给出寄存器的详细说明。这部分寄存器基地址为 0x1FE00100。

表 11-7 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	Chip Config	芯片配置寄存器
84		
88		
8C		
90	Chip Sample	芯片采样寄存器

表 11-8 寄存器详细描述

位域	字段名	访问	复位值	说明
CR00: PonCfg				
7:0	pcix_bus_dev	只读	lio_ad[7:0]	PCIX Agent 模式下 CPU 取指所使用的总线、设备号
15:8	保留	只读	lio_ad[15:8]	
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 0：停靠到最后一个主设备 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 00: 0 周期 01: 8 周期 10: 32 周期 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 2.7 节）				
CR90: Chip Sample（见 2.7 节）				
CRA0: Chip Sample（见 2.7 节）				
CRB0: PLL config（见 2.7 节）				
CRC0: PLL config（见 2.7 节）				
CRD0: Core config（见 2.7 节）				

12 芯片配置寄存器列表

Name	ADDR	R/W	Description(NULL 表示没有作用)	default value
CPU_WIN0_BASE	0x3ff00000	RW	CPU 窗口 0 的基地址	0x0
CPU_WIN1_BASE	0x3ff00008	RW	CPU 窗口 1 的基地址	0x1000_0000
CPU_WIN2_BASE	0x3ff00010	RW	CPU 窗口 2 的基地址	0x1000_8000_0000
CPU_WIN3_BASE	0x3ff00018	RW	CPU 窗口 3 的基地址	0x0
CPU_WIN4_BASE	0x3ff00020	RW	CPU 窗口 4 的基地址	0x0
CPU_WIN5_BASE	0x3ff00028	RW	CPU 窗口 5 的基地址	0x0
CPU_WIN6_BASE	0x3ff00030	RW	CPU 窗口 6 的基地址	0x0
CPU_WIN7_BASE	0x3ff00038	RW	CPU 窗口 7 的基地址	0x0
CPU_WIN0_MASK	0x3ff00040	RW	CPU 窗口 0 的掩码	0xffff_ffff_f000_0000
CPU_WIN1_MASK	0x3ff00048	RW	CPU 窗口 1 的掩码	0xffff_ffff_f000_0000
CPU_WIN2_MASK	0x3ff00050	RW	CPU 窗口 2 的掩码	0xffff_ffff_f000_0000
CPU_WIN3_MASK	0x3ff00058	RW	CPU 窗口 3 的掩码	0x0
CPU_WIN4_MASK	0x3ff00060	RW	CPU 窗口 4 的掩码	0x0
CPU_WIN5_MASK	0x3ff00068	RW	CPU 窗口 5 的掩码	0x0
CPU_WIN6_MASK	0x3ff00070	RW	CPU 窗口 6 的掩码	0x0
CPU_WIN7_MASK	0x3ff00078	RW	CPU 窗口 7 的掩码	0x0

CPU_WIN0_MMAP	0x3ff00080	RW	CPU 窗口 0 的新基地址	0xf0
CPU_WIN1_MMAP	0x3ff00088	RW	CPU 窗口 1 的新基地址	0x1000_00f2
CPU_WIN2_MMAP	0x3ff00090	RW	CPU 窗口 2 的新基地址	0xf0
CPU_WIN3_MMAP	0x3ff00098	RW	CPU 窗口 3 的新基地址	0x0
CPU_WIN4_MMAP	0x3ff000a0	RW	CPU 窗口 4 的新基地址	0x0
CPU_WIN5_MMAP	0x3ff000a8	RW	CPU 窗口 5 的新基地址	0x0
CPU_WIN6_MMAP	0x3ff000b0	RW	CPU 窗口 6 的新基地址	0x0
CPU_WIN7_MMAP	0x3ff000b8	RW	CPU 窗口 7 的新基地址	0x0
PCI_WIN0_BASE	0x3ff00100	RW	PCI 窗口 0 的基地址	0x8000_0000
PCI_WIN1_BASE	0x3ff00108	RW	PCI 窗口 1 的基地址	0x0
PCI_WIN2_BASE	0x3ff00110	RW	PCI 窗口 2 的基地址	0x0
PCI_WIN3_BASE	0x3ff00118	RW	PCI 窗口 3 的基地址	0x0
PCI_WIN4_BASE	0x3ff00120	RW	PCI 窗口 4 的基地址	0x0
PCI_WIN5_BASE	0x3ff00128	RW	PCI 窗口 5 的基地址	0x0
PCI_WIN6_BASE	0x3ff00130	RW	PCI 窗口 6 的基地址	0x0
PCI_WIN7_BASE	0x3ff00138	RW	PCI 窗口 7 的基地址	0x0
PCI_WIN0_MASK	0x3ff00140	RW	PCI 窗口 0 的掩码	0xffff_ffff_8000_0000
PCI_WIN1_MASK	0x3ff00148	RW	PCI 窗口 1 的掩码	0x0
PCI_WIN2_MASK	0x3ff00150	RW	PCI 窗口 2 的掩码	0x0
PCI_WIN3_MASK	0x3ff00158	RW	PCI 窗口 3 的掩码	0x0

PCI_WIN4_MASK	0x3ff00160	RW	PCI 窗口 4 的掩码	0x0
PCI_WIN5_MASK	0x3ff00168	RW	PCI 窗口 5 的掩码	0x0
PCI_WIN6_MASK	0x3ff00170	RW	PCI 窗口 6 的掩码	0x0
PCI_WIN7_MASK	0x3ff00178	RW	PCI 窗口 7 的掩码	0x0
PCI_WIN0_MMAP	0x3ff00180	RW	PCI 窗口 0 的新基地址	0xf0
PCI_WIN1_MMAP	0x3ff00188	RW	PCI 窗口 1 的新基地址	0x0
PCI_WIN2_MMAP	0x3ff00190	RW	PCI 窗口 2 的新基地址	0x0
PCI_WIN3_MMAP	0x3ff00198	RW	PCI 窗口 3 的新基地址	0x0
PCI_WIN4_MMAP	0x3ff001a0	RW	PCI 窗口 4 的新基地址	0x0
PCI_WIN5_MMAP	0x3ff001a8	RW	PCI 窗口 5 的新基地址	0x0
PCI_WIN6_MMAP	0x3ff001b0	RW	PCI 窗口 6 的新基地址	0x0
PCI_WIN7_MMAP	0x3ff001b8	RW	PCI 窗口 7 的新基地址	0x0
Slock0_addr	0x3ff00200	RW	0 号锁窗口锁地址([63]: valid, [47:0]: addr)	0x0
Slock1_addr	0x3ff00208	RW	1 号锁窗口锁地址([63]: valid, [47:0]: addr)	0x0
Slock2_addr	0x3ff00210	RW	2 号锁窗口锁地址([63]: valid, [47:0]: addr)	0x0
Slock3_addr	0x3ff00218	RW	3 号锁窗口锁地址([63]: valid, [47:0]: addr)	0x0
Slock0_mask	0x3ff00240	RW	0 号锁窗口掩码([47:0]: mask)	0x0
Slock1_mask	0x3ff00248	RW	1 号锁窗口掩码([47:0]: mask)	0x0
Slock2_mask	0x3ff00250	RW	2 号锁窗口掩码([47:0]: mask)	0x0
Slock3_mask	0x3ff00258	RW	3 号锁窗口掩码([47:0]: mask)	0x0

BARRIER_SET	0x3ff00300	WO	barrier 值加 1	
BARRIER_CLR	0x3ff00308	WO	barrier 值减 1	
BARRIER_REF	0x3ff00310	RW	barrier 阈值	0x0
BARRIER_CTRL	0x3ff00318	RW	bit[0]: barrier 值加减使能/barrier 中断使能	0x0
BARRIER_VEC	0x3ff00320	RO	当前 barrier 值	
CONFSIGNAL_CR	0x3ff00400	RW	24：ccsd_en 19:16：ccsd_id 8：xrouter_en 5：x2_pci_rdinterleave 4：x2_cpu_rdinterleave 3:0：scid_sel	0xffff_0000
gs3_HPT	0x3ff00408	RO	每个时钟周期加 1 的计数器	
MTX0_SRC_START_ADDR	0x3ff00600	RW		0x0
MTX0_DST_START_ADDR	0x3ff00608	RW		0x0
MTX0_ORI_LENTH	0x3ff00610	RW		0x0
MTX0_ORI_WIDTH	0x3ff00618	RW		0x0
MTX0_SRC_ROW_STRIDE	0x3ff00620	RW		0x0
MTX0_DST_ROW_STRIDE	0x3ff00628	RW		0x0
MTX0_TRANS_CTRL	0x3ff00630	RW		0x0
MTX1_SRC_START_ADDR	0x3ff00700	RW		0x0

MTX1_DST_START_ADDR	0x3ff00708	RW		0x0
MTX1_ORI_LENTH	0x3ff00710	RW		0x0
MTX1_ORI_WIDTH	0x3ff00718	RW		0x0
MTX1_SRC_ROW_STRIDE	0x3ff00720	RW		0x0
MTX1_DST_ROW_STRIDE	0x3ff00728	RW		0x0
MTX1_TRANS_CTRL	0x3ff00730	RW		0x0
SCache0_perfctrl0	0x3ff00800	RW		
SCache0_perfcnt0	0x3ff00808	RO		
SCache0_perfctrl1	0x3ff00810	RW		
SCache0_perfcnt1	0x3ff00818	RO		
SCache0_perfctrl2	0x3ff00820	RW		
SCache0_perfcnt2	0x3ff00828	RO		
SCache0_perfctrl3	0x3ff00830	RW		
SCache0_perfcnt3	0x3ff00838	RO		
SCache1_perfctrl0	0x3ff00900	RW		
SCache1_perfcnt0	0x3ff00908	RO		
SCache1_perfctrl1	0x3ff00910	RW		
SCache1_perfcnt1	0x3ff00918	RO		
SCache1_perfctrl2	0x3ff00920	RW		
SCache1_perfcnt2	0x3ff00928	RO		

SCache1_perfctrl3	0x3ff00930	RW		
SCache1_perfcnt3	0x3ff00938	RO		
SCache2_perfctrl0	0x3ff00A00	RW		
SCache2_perfcnt0	0x3ff00A08	RO		
SCache2_perfctrl1	0x3ff00A10	RW		
SCache2_perfcnt1	0x3ff00A18	RO		
SCache2_perfctrl2	0x3ff00A20	RW		
SCache2_perfcnt2	0x3ff00A28	RO		
SCache2_perfctrl3	0x3ff00A30	RW		
SCache2_perfcnt3	0x3ff00A38	RO		
SCache3_perfctrl0	0x3ff00B00	RW		
SCache3_perfcnt0	0x3ff00B08	RO		
SCache3_perfctrl1	0x3ff00B10	RW		
SCache3_perfcnt1	0x3ff00B18	RO		
SCache3_perfctrl2	0x3ff00B20	RW		
SCache3_perfcnt2	0x3ff00B28	RO		
SCache3_perfctrl3	0x3ff00B30	RW		
SCache3_perfcnt3	0x3ff00B38	RO		
Core0_IPI_Status	0x3ff01000	RO	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	0x0

Core0_IPI_Set	0x3ff01008	WO	0 号处理器核的 IPI_Set 寄存器	
Core0_IPI_Clear	0x3ff0100c	WO	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	0x0
Core0_MailBox1	0x3ff01028	RW	0 号处理器核的 IPI_MailBox1 寄存器	0x0
Core0_MailBox2	0x3ff01030	RW	0 号处理器核的 IPI_MailBox2 寄存器	0x0
Core0_MailBox3	0x3ff01038	RW	0 号处理器核的 IPI_MailBox3 寄存器	0x0
Core0_int_interval	0x3ff01060	RW		
Core0_int_compare	0x3ff01068	RW		
Core1_IPI_Status	0x3ff01100	RO	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	0x0
Core1_IPI_Set	0x3ff01108	WO	1 号处理器核的 IPI_Set 寄存器	
Core1_IPI_Clear	0x3ff0110c	WO	1 号处理器核的 IPI_Clear 寄存器	
Core1_MailBox0	0x3ff01120	RW	1 号处理器核的 IPI_MailBox0 寄存器	0x0
Core1_MailBox1	0x3ff01128	RW	1 号处理器核的 IPI_MailBox1 寄存器	0x0
Core1_MailBox2	0x3ff01130	RW	1 号处理器核的 IPI_MailBox2 寄存器	0x0
Core1_MailBox3	0x3ff01138	RW	1 号处理器核的 IPI_MailBox3 寄存器	0x0
Core1_int_interval	0x3ff01160	RW		
Core1_int_compare	0x3ff01168	RW		
Core2_IPI_Status	0x3ff01200	RO	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	0x0

Core2_IPI_Set	0x3ff01208	WO	2 号处理器核的 IPI_Set 寄存器	
Core2_IPI_Clear	0x3ff0120c	WO	2 号处理器核的 IPI_Clear 寄存器	
Core2_MailBox0	0x3ff01220	RW	2 号处理器核的 IPI_MailBox0 寄存器	0x0
Core2_MailBox1	0x3ff01228	RW	2 号处理器核的 IPI_MailBox1 寄存器	0x0
Core2_MailBox2	0x3ff01230	RW	2 号处理器核的 IPI_MailBox2 寄存器	0x0
Core2_MailBox3	0x3ff01238	RW	2 号处理器核的 IPI_MailBox3 寄存器	0x0
Core2_int_interval	0x3ff01260	RW		
Core2_int_compare	0x3ff01268	RW		
Core3_IPI_Status	0x3ff01300	RO	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	0x0
Core3_IPI_Set	0x3ff01308	WO	3 号处理器核的 IPI_Set 寄存器	
Core3_IPI_Clear	0x3ff0130c	WO	3 号处理器核的 IPI_Clear 寄存器	
Core3_MailBox0	0x3ff01320	RW	3 号处理器核的 IPI_MailBox0 寄存器	0x0
Core3_MailBox1	0x3ff01328	RW	3 号处理器核的 IPI_MailBox1 寄存器	0x0
Core3_MailBox2	0x3ff01330	RW	3 号处理器核的 IPI_MailBox2 寄存器	0x0
Core3_MailBox3	0x3ff01338	RW	3 号处理器核的 IPI_MailBox3 寄存器	0x0
Core3_int_interval	0x3ff01360	RW		
Core3_int_compare	0x3ff01368	RW		
Int Entry[0--31]	0x3ff01400	RW	32 个 8 位中断路由寄存器	0x0
Intisr	0x3ff01420	RO	32 位中断状态寄存器	

Inten	0x3ff01424	RO	32 位中断使能状态寄存器	
Intenset	0x3ff01428	WO	32 位设置使能寄存器	
Intenclr	0x3ff0142c	WO	32 位清除使能寄存器和脉冲触发的中断	
Intpol	0x3ff01430	WO	无用	0x0
Intedge	0x3ff01434	WO	32 位触发方式寄存器（1：脉冲触发；0：电平触发）	0x0
CORE0_INTISR	0x3ff01440	RO	路由给 CORE0 的 32 位中断状态	
CORE1_INTISR	0x3ff01448	RO	路由给 CORE1 的 32 位中断状态	
CORE2_INTISR	0x3ff01450	RO	路由给 CORE2 的 32 位中断状态	
CORE3_INTISR	0x3ff01458	RO	路由给 CORE3 的 32 位中断状态	

Thsens_int_ctrl_Hi	0x3ff01460	RW	温度传感器高温中断控制寄存器 [7:0]：Hi_gate0：高温阈值 0，超过这个温度将产生中断 [8:8]：Hi_en0：高温中断使能 0 [11:10]：Hi_Sel0：选择高温中断 0 的温度传感器输入源 [23:16]：Hi_gate1：高温阈值 1，超过这个温度将产生中断 [24:24]：Hi_en1：高温中断使能 1 [27:26]：Hi_Sel1：选择高温中断 1 的温度传感器输入源 [39:32]：Hi_gate2：高温阈值 2，超过这个温度将产生中断 [40:40]：Hi_en2：高温中断使能 2 [43:42]：Hi_Sel2：选择高温中断 2 的温度传感器输入源 [55:48]：Hi_gate3：高温阈值 3，超过这个温度将产生中断 [56:56]：Hi_en3：高温中断使能 3 [59:58]：Hi_Sel3：选择高温中断 3 的温度传感器输入源	

Thsens_int_ctrl_Lo	0x3ff01468	RW	温度传感器低温中断控制寄存器 [7:0]：Lo_gate0：低温阈值 0，低于这个温度将产生中断 [8:8]：Lo_en0：低温中断使能 0 [11:10]：Lo_Sel0：选择低温中断 0 的温度传感器输入源 [23:16]：Lo_gate1：低温阈值 1，低于这个温度将产生中断 [24:24]：Lo_en1：低温中断使能 1 [27:26]：Lo_Sel1：选择低温中断 1 的温度传感器输入源 [39:32]：Lo_gate2：低温阈值 2，低于这个温度将产生中断 [40:40]：Lo_en2：低温中断使能 2 [43:42]：Lo_Sel2：选择低温中断 2 的温度传感器输入源 [55:48]：Lo_gate3：低温阈值 3，低于这个温度将产生中断 [56:56]：Lo_en3：低温中断使能 3 [59:58]：Lo_Sel3：选择低温中断 3 的温度传感器输入源	
Thsens_int_status/clr	0x3ff01470	RW	中断状态寄存器，写任意值清除中断 [0]：高温中断触发 [1]：低温中断触发	

Thsens_freq_scale	0x3ff01480	RW	温度传感器高温降频控制寄存器，四组设置优先级由高到低 [7:0]：Scale_gate0：高温阈值 0，超过这个温度将降频 [8:8]：Scale_en0：高温降频使能 0 [11:10]：Scale_Sel0：选择高温降频 0 的温度传感器输入源 [14:12]：Scale_freq0：降频时的分频值 [23:16]：Scale_gate1：高温阈值 1，超过这个温度将降频 [24:24]：Scale_en1：高温降频使能 1 [27:26]：Scale_Sel1：选择高温降频 1 的温度传感器输入源 [30:28]：Scale_freq1：降频时的分频值 [39:32]：Scale_gate2：高温阈值 2，超过这个温度将降频 [40:40]：Scale_en2：高温降频使能 2 [43:42]：Scale_Sel2：选择高温降频 2 的温度传感器输入源 [46:44]：Scale_freq2：降频时的分频值 [55:48]：Scale_gate3：高温阈值 3，超过这个温度将降频 [56:56]：Scale_en3：高温降频使能 3 [59:58]：Scale_Sel3：选择高温降频 3 的温度传感器输入源 [62:60]：Scale_freq3：降频时的分频值	
DFD_PARAM	0x3ff01500	RW	调试触发条件使能 [7:0]：timer，触发延迟，设为 1 表示当条件满足立即触发，设为 0 表示禁止触发，设为其它值表示条件满足后延迟触发的拍数+1 [15:8]：trigger_en，触发条件使能，对应于外部的 8 个触发事件的使能控制	

DFD_TRIGGER	0x3ff01508	WO	软件触发，向这个地址发出写操作，会造成一个软件的触发条件，使得在 timer-1 拍之后触发	
CORE0_AWCOND0	0x3ff01800	RW	CORE0 的 AXI 接口 AW 触发条件 0 设置 [15:0]：awid [19:16]：awlen [22:20]：awsize [24:23]：awburst [26:25]：awlock [30:27]：awcache [33:31]：awprot [37:34]：awcmd [41:38]：awdirqid [43:42]：awstate [47:44]：swscseti [48]：awvalid [49]：awready	

CORE0_AWMASK0	0x3ff01808	RW	CORE0 的 AXI 接口 AW 触发使能 0 设置，最高位为 AW 通道触发使能 [49:0]：awmask [62]：awdata_en：同 wid 的 wdata 触发条件同时满足时，才允许触发 [63]：awchannel_en：触发条件使能 触发条件为 (AW_IN & AWMASK) == (AWCOND & AWMASK)	
CORE0_AWCOND1	0x3ff01810	RW	AW 的触发条件要 COND0 与 COND1 同时满足 [47:0]：awaddr	
CORE0_AWMASK1	0x3ff01818	RW		
CORE0_ARCOND0	0x3ff01820	RW	CORE0 的 AXI 接口 AR 触发条件，与 AW 类似 [15:0]：arid [19:16]：arlen [22:20]：arsize [24:23]：arburst [26:25]：arlock [30:27]：arcache [33:31]：arprot [37:34]：arcmd [47:38]：arcpuno [48]：arvalid [49]：arready	

CORE0_ARMASK0	0x3ff01828	RW	CORE0 的 AXI 接口 AR 触发使能 0 设置，最高位为 AR 通道触发使能 [49:0]：armask [62]：ardata_en：同 rid 的 rdata 触发条件同时满足时，才允许触发 [63]：archannel_en：触发条件使能	
CORE0_ARCOND1	0x3ff01830	RW	[47:0]：araddr	
CORE0_ARMASK1	0x3ff01838	RW		
CORE0_WCOND0	0x3ff01840	RW	CORE0 的 AXI 接口 W 触发条件，与 AW 类似 [15:0]：wid [31:16]：wstrb [32]：wlast [33]：wvalid [34]：wready	
CORE0_WMASK0	0x3ff01848	RW	CORE0 的 AXI 接口 W 触发使能 0 设置，最高位为 W 通道触发使能 [49:0]：wmask [63]：wchannel_en：触发条件使能，当 awdata_en 有效时不需要设置	
CORE0_WCOND1	0x3ff01850	RW		
CORE0_WMASK1	0x3ff01858	RW		
CORE0_WCOND2	0x3ff01860	RW		
CORE0_WMASK2	0x3ff01868	RW		

CORE0_BCOND0	0x3ff01870	RW	CORE0 的 AXI 接口 B 触发条件，与 AW 类似 [15:0]：bid [17:16]：bresp [18]：bvalid [19]：bready	
CORE0_BMASK0	0x3ff01878	RW	CORE0 的 AXI 接口 B 触发使能 0 设置，最高位为 B 通道触发使能 [19:0]：bmask [63]：bchannel_en	
CORE0_RCOND0	0x3ff01880	RW	CORE0 的 AXI 接口 R 触发条件，与 AW 类似 [15:0]：rid [17:16]：rresp [18]：rlast [19]：rrequest [21:20]：rstate [25:22]：rscseti [26]：rvalid [27]：rready	
CORE0_RMASK0	0x3ff01888	RW	CORE0 的 AXI 接口 R 触发使能 0 设置，最高位为 R 通道触发使能 [27:0]：rmask [63]：rchannel_en	
CORE0_RCOND1	0x3ff01890	RW		

CORE0_RMASK1	0x3ff01898	RW		
CORE0_RCOND2	0x3ff018a0	RW		
CORE0_RMASK2	0x3ff018a8	RW		
TUD0_CONF0	0x3ff018e0	RW	TUD0 配置寄存器 0 [47:0]：count_target [55:48]：monitor_enable	
TUD0_CONF1	0x3ff018e8	RW	TUD0 配置寄存器 1 [2:0]：DCDL_sel_signal [5:3]：DCDL_sel_clock [9:6]：signal_sel [13:10]：clok_sel [20:14]：reading_sel [21]：counter_clock_sel [22]：sticky [23]：reset_g [24]：stop [25]：start [26]：cg_en	
TUD0_RESULT	0x3ff018f0	R	TUD0 结果寄存器	

CORE1_AWCOND0	0x3ff01900	RW	CORE1 的 AXI 接口 AW 触发条件 0 设置	
CORE1_AWMASK0	0x3ff01908	RW	CORE1 的 AXI 接口 AW 触发使能 0 设置，最高位为 AW 通道触发使能 触发条件为 (AW_IN & AWMASK) == (AWCOND & AWMASK)	
CORE1_AWCOND1	0x3ff01910	RW	AW 的触发条件要 COND0 与 COND1 同时满足	
CORE1_AWMASK1	0x3ff01918	RW		
CORE1_ARCOND0	0x3ff01920	RW	CORE1 的 AXI 接口 AR 触发条件，与 AW 类似	
CORE1_ARMASK0	0x3ff01928	RW		
CORE1_ARCOND1	0x3ff01930	RW		
CORE1_ARMASK1	0x3ff01938	RW		
CORE1_WCOND0	0x3ff01940	RW	CORE1 的 AXI 接口 W 触发条件，与 AW 类似	
CORE1_WMASK0	0x3ff01948	RW		
CORE1_WCOND1	0x3ff01950	RW		
CORE1_WMASK1	0x3ff01958	RW		
CORE1_WCOND2	0x3ff01960	RW		
CORE1_WMASK2	0x3ff01968	RW		
CORE1_BCOND0	0x3ff01970	RW	CORE1 的 AXI 接口 B 触发条件，与 AW 类似	
CORE1_BMASK0	0x3ff01978	RW		
CORE1_RCOND0	0x3ff01980	RW	CORE1 的 AXI 接口 R 触发条件，与 AW 类似	
CORE1_RMASK0	0x3ff01988	RW		

CORE1_RCOND1	0x3ff01990	RW		
CORE1_RMASK1	0x3ff01998	RW		
CORE1_RCOND2	0x3ff019a0	RW		
CORE1_RMASK2	0x3ff019a8	RW		
TUD1_CONF0	0x3ff019e0	RW	TUD1 配置寄存器 0 [47:0]：count_target [55:48]：monitor_enable	
TUD1_CONF1	0x3ff019e8	RW	TUD0 配置寄存器 1 [2:0]：DCDL_sel_signal [5:3]：DCDL_sel_clock [9:6]：signal_sel [13:10]：clok_sel [20:14]：reading_sel [21]：counter_clock_sel [22]：sticky [23]：reset_g [24]：stop [25]：start [26]：cg_en	
TUD1_RESULT	0x3ff019f0	R	TUD1 结果寄存器	

CORE2_AWCOND0	0x3ff01a00	RW	CORE2 的 AXI 接口 AW 触发条件 0 设置	
CORE2_AWMASK0	0x3ff01a08	RW	CORE2 的 AXI 接口 AW 触发使能 0 设置，最高位为 AW 通道触发使能 触发条件为 (AW_IN & AWMASK) == (AWCOND & AWMASK)	
CORE2_AWCOND1	0x3ff01a10	RW	AW 的触发条件要 COND0 与 COND1 同时满足	
CORE2_AWMASK1	0x3ff01a18	RW		
CORE2_ARCOND0	0x3ff01a20	RW	CORE2 的 AXI 接口 AR 触发条件，与 AW 类似	
CORE2_ARMASK0	0x3ff01a28	RW		
CORE2_ARCOND1	0x3ff01a30	RW		
CORE2_ARMASK1	0x3ff01a38	RW		
CORE2_WCOND0	0x3ff01a40	RW	CORE2 的 AXI 接口 W 触发条件，与 AW 类似	
CORE2_WMASK0	0x3ff01a48	RW		
CORE2_WCOND1	0x3ff01a50	RW		
CORE2_WMASK1	0x3ff01a58	RW		
CORE2_WCOND2	0x3ff01a60	RW		
CORE2_WMASK2	0x3ff01a68	RW		
CORE2_BCOND0	0x3ff01a70	RW	CORE2 的 AXI 接口 B 触发条件，与 AW 类似	
CORE2_BMASK0	0x3ff01a78	RW		
CORE2_RCOND0	0x3ff01a80	RW	CORE2 的 AXI 接口 R 触发条件，与 AW 类似	
CORE2_RMASK0	0x3ff01a88	RW		

CORE2_RCOND1	0x3ff01a90	RW		
CORE2_RMASK1	0x3ff01a98	RW		
CORE2_RCOND2	0x3ff01aa0	RW		
CORE2_RMASK2	0x3ff01aa8	RW		
TUD2_CONF0	0x3ff01ae0	RW	TUD2 配置寄存器 0 [47:0]：count_target [55:48]：monitor_enable	
TUD2_CONF1	0x3ff01ae8	RW	TUD0 配置寄存器 1 [2:0]：DCDL_sel_signal [5:3]：DCDL_sel_clock [9:6]：signal_sel [13:10]：clok_sel [20:14]：reading_sel [21]：counter_clock_sel [22]：sticky [23]：reset_g [24]：stop [25]：start [26]：cg_en	
TUD2_RESULT	0x3ff01af0	R	TUD2 结果寄存器	

CORE3_AWCOND0	0x3ff01b00	RW	CORE3 的 AXI 接口 AW 触发条件 0 设置	
CORE3_AWMASK0	0x3ff01b08	RW	CORE3 的 AXI 接口 AW 触发使能 0 设置，最高位为 AW 通道触发使能 触发条件为 (AW_IN & AWMASK) == (AWCOND & AWMASK)	
CORE3_AWCOND1	0x3ff01b10	RW	AW 的触发条件要 COND0 与 COND1 同时满足	
CORE3_AWMASK1	0x3ff01b18	RW		
CORE3_ARCOND0	0x3ff01b20	RW	CORE3 的 AXI 接口 AR 触发条件，与 AW 类似	
CORE3_ARMASK0	0x3ff01b28	RW		
CORE3_ARCOND1	0x3ff01b30	RW		
CORE3_ARMASK1	0x3ff01b38	RW		
CORE3_WCOND0	0x3ff01b40	RW	CORE3 的 AXI 接口 W 触发条件，与 AW 类似	
CORE3_WMASK0	0x3ff01b48	RW		
CORE3_WCOND1	0x3ff01b50	RW		
CORE3_WMASK1	0x3ff01b58	RW		
CORE3_WCOND2	0x3ff01b60	RW		
CORE3_WMASK2	0x3ff01b68	RW		
CORE3_BCOND0	0x3ff01b70	RW	CORE3 的 AXI 接口 B 触发条件，与 AW 类似	
CORE3_BMASK0	0x3ff01b78	RW		
CORE3_RCOND0	0x3ff01b80	RW	CORE3 的 AXI 接口 R 触发条件，与 AW 类似	
CORE3_RMASK0	0x3ff01b88	RW		

CORE3_RCOND1	0x3ff01b90	RW		
CORE3_RMASK1	0x3ff01b98	RW		
CORE3_RCOND2	0x3ff01ba0	RW		
CORE3_RMASK2	0x3ff01ba8	RW		
TUD3_CONF0	0x3ff01be0	RW	TUD3 配置寄存器 0 [47:0]：count_target [55:48]：monitor_enable	
TUD3_CONF1	0x3ff01be8	RW	TUD0 配置寄存器 1 [2:0]：DCDL_sel_signal [5:3]：DCDL_sel_clock [9:6]：signal_sel [13:10]：clok_sel [20:14]：reading_sel [21]：counter_clock_sel [22]：sticky [23]：reset_g [24]：stop [25]：start [26]：cg_en	
TUD3_RESULT	0x3ff01bf0	R	TUD3 结果寄存器	

TUD4_CONF0	0x3ff01ce0	RW	TUD4 配置寄存器 0 [47:0]：count_target [55:48]：monitor_enable	
TUD4_CONF1	0x3ff01ce8	RW	TUD4 配置寄存器 1 [2:0]：DCDL_sel_signal [5:3]：DCDL_sel_clock [8:6]：signal_sel [11:9]：clock_sel [18:12]：reading_sel [19]：counter_clock_sel [20]：sticky [21]：reset_g [22]：stop [23]：start [24]：cg_en	
TUD4_RESULT	0x3ff01cf0	R	TUD4 结果寄存器	
TUD5_CONF0	0x3ff01de0	RW	TUD5 配置寄存器 0 [47:0]：count_target [55:48]：monitor_enable	

TUD5_CONF1	0x3ff01de8	RW	TUD5 配置寄存器 1 [2:0]：DCDL_sel_signal [5:3]：DCDL_sel_clock [8:6]：signal_sel [11:9]：clock_sel [18:12]：reading_sel [19]：counter_clock_sel [20]：sticky [21]：reset_g [22]：stop [23]：start [24]：cg_en	
TUD5_RESULT	0x3ff01df0	R	TUD5 结果寄存器	
HT0_AWCOND0	0x3ff01e00	RW	HT0 的 AXI 接口 AW 触发条件 0 设置	
HT0_AWMASK0	0x3ff01e08	RW	HT0 的 AXI 接口 AW 触发使能 0 设置，最高位为 AW 通道触发使能 触发条件为 (AW_IN & AWMASK) == (AWCOND & AWMASK)	
HT0_AWCOND1	0x3ff01e10	RW	AW 的触发条件要 COND0 与 COND1 同时满足	
HT0_AWMASK1	0x3ff01e18	RW		
HT0_ARCOND0	0x3ff01e20	RW	HT0 的 AXI 接口 AR 触发条件，与 AW 类似	

HT0_ARMASK0	0x3ff01e28	RW		
HT0_ARCOND1	0x3ff01e30	RW		
HT0_ARMASK1	0x3ff01e38	RW		
HT0_WCOND0	0x3ff01e40	RW	HT0 的 AXI 接口 W 触发条件，与 AW 类似	
HT0_WMASK0	0x3ff01e48	RW		
HT0_WCOND1	0x3ff01e50	RW		
HT0_WMASK1	0x3ff01e58	RW		
HT0_WCOND2	0x3ff01e60	RW		
HT0_WMASK2	0x3ff01e68	RW		
HT0_BCOND0	0x3ff01e70	RW	HT0 的 AXI 接口 B 触发条件，与 AW 类似	
HT0_BMASK0	0x3ff01e78	RW		
HT0_RCOND0	0x3ff01e80	RW	HT0 的 AXI 接口 R 触发条件，与 AW 类似	
HT0_RMASK0	0x3ff01e88	RW		
HT0_RCOND1	0x3ff01e90	RW		
HT0_RMASK1	0x3ff01e98	RW		
HT0_RCOND2	0x3ff01ea0	RW		
HT0_RMASK2	0x3ff01ea8	RW		
HT1_AWCOND0	0x3ff01f00	RW	HT1 的 AXI 接口 AW 触发条件 0 设置	

HT1_AWMASK0	0x3ff01f08	RW	HT1 的 AXI 接口 AW 触发使能 0 设置，最高位为 AW 通道触发使能 触发条件为 (AW_IN & AWMASK) == (AWCOND & AWMASK)	
HT1_AWCOND1	0x3ff01f10	RW	AW 的触发条件要 COND0 与 COND1 同时满足	
HT1_AWMASK1	0x3ff01f18	RW		
HT1_ARCOND0	0x3ff01f20	RW	HT1 的 AXI 接口 AR 触发条件，与 AW 类似	
HT1_ARMASK0	0x3ff01f28	RW		
HT1_ARCOND1	0x3ff01f30	RW		
HT1_ARMASK1	0x3ff01f38	RW		
HT1_WCOND0	0x3ff01f40	RW	HT1 的 AXI 接口 W 触发条件，与 AW 类似	
HT1_WMASK0	0x3ff01f48	RW		
HT1_WCOND1	0x3ff01f50	RW		
HT1_WMASK1	0x3ff01f58	RW		
HT1_WCOND2	0x3ff01f60	RW		
HT1_WMASK2	0x3ff01f68	RW		
HT1_BCOND0	0x3ff01f70	RW	HT1 的 AXI 接口 B 触发条件，与 AW 类似	
HT1_BMASK0	0x3ff01f78	RW		
HT1_RCOND0	0x3ff01f80	RW	HT1 的 AXI 接口 R 触发条件，与 AW 类似	
HT1_RMASK0	0x3ff01f88	RW		
HT1_RCOND1	0x3ff01f90	RW		

HT1_RMASK1	0x3ff01f98	RW		
HT1_RCOND2	0x3ff01fa0	RW		
HT1_RMASK2	0x3ff01fa8	RW		
CORE0_WIN0_BASE	0x3ff02000	RW	一级交叉开关地址窗口	0x0
CORE0_WIN1_BASE	0x3ff02008	RW	一级交叉开关地址窗口	0x0
CORE0_WIN2_BASE	0x3ff02010	RW	一级交叉开关地址窗口	0x0
CORE0_WIN3_BASE	0x3ff02018	RW	一级交叉开关地址窗口	0x0
CORE0_WIN4_BASE	0x3ff02020	RW	一级交叉开关地址窗口	0x0
CORE0_WIN5_BASE	0x3ff02028	RW	一级交叉开关地址窗口	0x0
CORE0_WIN6_BASE	0x3ff02030	RW	一级交叉开关地址窗口	0x0
CORE0_WIN7_BASE	0x3ff02038	RW	一级交叉开关地址窗口	0x0
CORE0_WIN0_MASK	0x3ff02040	RW	一级交叉开关地址窗口	0x0
CORE0_WIN1_MASK	0x3ff02048	RW	一级交叉开关地址窗口	0x0
CORE0_WIN2_MASK	0x3ff02050	RW	一级交叉开关地址窗口	0x0
CORE0_WIN3_MASK	0x3ff02058	RW	一级交叉开关地址窗口	0x0
CORE0_WIN4_MASK	0x3ff02060	RW	一级交叉开关地址窗口	0x0
CORE0_WIN5_MASK	0x3ff02068	RW	一级交叉开关地址窗口	0x0
CORE0_WIN6_MASK	0x3ff02070	RW	一级交叉开关地址窗口	0x0
CORE0_WIN7_MASK	0x3ff02078	RW	一级交叉开关地址窗口	0x0
CORE0_WIN0_MMAP	0x3ff02080	RW	一级交叉开关地址窗口	0x0

CORE0_WIN1_MMAP	0x3ff02088	RW	一级交叉开关地址窗口	0x0
CORE0_WIN2_MMAP	0x3ff02090	RW	一级交叉开关地址窗口	0x0
CORE0_WIN3_MMAP	0x3ff02098	RW	一级交叉开关地址窗口	0x0
CORE0_WIN4_MMAP	0x3ff020a0	RW	一级交叉开关地址窗口	0x0
CORE0_WIN5_MMAP	0x3ff020a8	RW	一级交叉开关地址窗口	0x0
CORE0_WIN6_MMAP	0x3ff020b0	RW	一级交叉开关地址窗口	0x0
CORE0_WIN7_MMAP	0x3ff020b8	RW	一级交叉开关地址窗口	0x0
CORE1_WIN0_BASE	0x3ff02100	RW	一级交叉开关地址窗口	0x0
CORE1_WIN1_BASE	0x3ff02108	RW	一级交叉开关地址窗口	0x0
CORE1_WIN2_BASE	0x3ff02110	RW	一级交叉开关地址窗口	0x0
CORE1_WIN3_BASE	0x3ff02118	RW	一级交叉开关地址窗口	0x0
CORE1_WIN4_BASE	0x3ff02120	RW	一级交叉开关地址窗口	0x0
CORE1_WIN5_BASE	0x3ff02128	RW	一级交叉开关地址窗口	0x0
CORE1_WIN6_BASE	0x3ff02130	RW	一级交叉开关地址窗口	0x0
CORE1_WIN7_BASE	0x3ff02138	RW	一级交叉开关地址窗口	0x0
CORE1_WIN0_MASK	0x3ff02140	RW	一级交叉开关地址窗口	0x0
CORE1_WIN1_MASK	0x3ff02148	RW	一级交叉开关地址窗口	0x0
CORE1_WIN2_MASK	0x3ff02150	RW	一级交叉开关地址窗口	0x0
CORE1_WIN3_MASK	0x3ff02158	RW	一级交叉开关地址窗口	0x0
CORE1_WIN4_MASK	0x3ff02160	RW	一级交叉开关地址窗口	0x0

CORE1_WIN5_MASK	0x3ff02168	RW	一级交叉开关地址窗口	0x0
CORE1_WIN6_MASK	0x3ff02170	RW	一级交叉开关地址窗口	0x0
CORE1_WIN7_MASK	0x3ff02178	RW	一级交叉开关地址窗口	0x0
CORE1_WIN0_MMAP	0x3ff02180	RW	一级交叉开关地址窗口	0x0
CORE1_WIN1_MMAP	0x3ff02188	RW	一级交叉开关地址窗口	0x0
CORE1_WIN2_MMAP	0x3ff02190	RW	一级交叉开关地址窗口	0x0
CORE1_WIN3_MMAP	0x3ff02198	RW	一级交叉开关地址窗口	0x0
CORE1_WIN4_MMAP	0x3ff021a0	RW	一级交叉开关地址窗口	0x0
CORE1_WIN5_MMAP	0x3ff021a8	RW	一级交叉开关地址窗口	0x0
CORE1_WIN6_MMAP	0x3ff021b0	RW	一级交叉开关地址窗口	0x0
CORE1_WIN7_MMAP	0x3ff021b8	RW	一级交叉开关地址窗口	0x0
CORE2_WIN0_BASE	0x3ff02200	RW	一级交叉开关地址窗口	0x0
CORE2_WIN1_BASE	0x3ff02208	RW	一级交叉开关地址窗口	0x0
CORE2_WIN2_BASE	0x3ff02210	RW	一级交叉开关地址窗口	0x0
CORE2_WIN3_BASE	0x3ff02218	RW	一级交叉开关地址窗口	0x0
CORE2_WIN4_BASE	0x3ff02220	RW	一级交叉开关地址窗口	0x0
CORE2_WIN5_BASE	0x3ff02228	RW	一级交叉开关地址窗口	0x0
CORE2_WIN6_BASE	0x3ff02230	RW	一级交叉开关地址窗口	0x0
CORE2_WIN7_BASE	0x3ff02238	RW	一级交叉开关地址窗口	0x0
CORE2_WIN0_MASK	0x3ff02240	RW	一级交叉开关地址窗口	0x0

CORE2_WIN1_MASK	0x3ff02248	RW	一级交叉开关地址窗口	0x0
CORE2_WIN2_MASK	0x3ff02250	RW	一级交叉开关地址窗口	0x0
CORE2_WIN3_MASK	0x3ff02258	RW	一级交叉开关地址窗口	0x0
CORE2_WIN4_MASK	0x3ff02260	RW	一级交叉开关地址窗口	0x0
CORE2_WIN5_MASK	0x3ff02268	RW	一级交叉开关地址窗口	0x0
CORE2_WIN6_MASK	0x3ff02270	RW	一级交叉开关地址窗口	0x0
CORE2_WIN7_MASK	0x3ff02278	RW	一级交叉开关地址窗口	0x0
CORE2_WIN0_MMAP	0x3ff02280	RW	一级交叉开关地址窗口	0x0
CORE2_WIN1_MMAP	0x3ff02288	RW	一级交叉开关地址窗口	0x0
CORE2_WIN2_MMAP	0x3ff02290	RW	一级交叉开关地址窗口	0x0
CORE2_WIN3_MMAP	0x3ff02298	RW	一级交叉开关地址窗口	0x0
CORE2_WIN4_MMAP	0x3ff022a0	RW	一级交叉开关地址窗口	0x0
CORE2_WIN5_MMAP	0x3ff022a8	RW	一级交叉开关地址窗口	0x0
CORE2_WIN6_MMAP	0x3ff022b0	RW	一级交叉开关地址窗口	0x0
CORE2_WIN7_MMAP	0x3ff022b8	RW	一级交叉开关地址窗口	0x0
CORE3_WIN0_BASE	0x3ff02300	RW	一级交叉开关地址窗口	0x0
CORE3_WIN1_BASE	0x3ff02308	RW	一级交叉开关地址窗口	0x0
CORE3_WIN2_BASE	0x3ff02310	RW	一级交叉开关地址窗口	0x0
CORE3_WIN3_BASE	0x3ff02318	RW	一级交叉开关地址窗口	0x0
CORE3_WIN4_BASE	0x3ff02320	RW	一级交叉开关地址窗口	0x0

CORE3_WIN5_BASE	0x3ff02328	RW	一级交叉开关地址窗口	0x0
CORE3_WIN6_BASE	0x3ff02330	RW	一级交叉开关地址窗口	0x0
CORE3_WIN7_BASE	0x3ff02338	RW	一级交叉开关地址窗口	0x0
CORE3_WIN0_MASK	0x3ff02340	RW	一级交叉开关地址窗口	0x0
CORE3_WIN1_MASK	0x3ff02348	RW	一级交叉开关地址窗口	0x0
CORE3_WIN2_MASK	0x3ff02350	RW	一级交叉开关地址窗口	0x0
CORE3_WIN3_MASK	0x3ff02358	RW	一级交叉开关地址窗口	0x0
CORE3_WIN4_MASK	0x3ff02360	RW	一级交叉开关地址窗口	0x0
CORE3_WIN5_MASK	0x3ff02368	RW	一级交叉开关地址窗口	0x0
CORE3_WIN6_MASK	0x3ff02370	RW	一级交叉开关地址窗口	0x0
CORE3_WIN7_MASK	0x3ff02378	RW	一级交叉开关地址窗口	0x0
CORE3_WIN0_MMAP	0x3ff02380	RW	一级交叉开关地址窗口	0x0
CORE3_WIN1_MMAP	0x3ff02388	RW	一级交叉开关地址窗口	0x0
CORE3_WIN2_MMAP	0x3ff02390	RW	一级交叉开关地址窗口	0x0
CORE3_WIN3_MMAP	0x3ff02398	RW	一级交叉开关地址窗口	0x0
CORE3_WIN4_MMAP	0x3ff023a0	RW	一级交叉开关地址窗口	0x0
CORE3_WIN5_MMAP	0x3ff023a8	RW	一级交叉开关地址窗口	0x0
CORE3_WIN6_MMAP	0x3ff023b0	RW	一级交叉开关地址窗口	0x0
CORE3_WIN7_MMAP	0x3ff023b8	RW	一级交叉开关地址窗口	0x0
EAST_WIN0_BASE	0x3ff02400	RW	一级交叉开关地址窗口	0x0

EAST_WIN1_BASE	0x3ff02408	RW	一级交叉开关地址窗口	0x0
EAST_WIN2_BASE	0x3ff02410	RW	一级交叉开关地址窗口	0x0
EAST_WIN3_BASE	0x3ff02418	RW	一级交叉开关地址窗口	0x0
EAST_WIN4_BASE	0x3ff02420	RW	一级交叉开关地址窗口	0x0
EAST_WIN5_BASE	0x3ff02428	RW	一级交叉开关地址窗口	0x0
EAST_WIN6_BASE	0x3ff02430	RW	一级交叉开关地址窗口	0x0
EAST_WIN7_BASE	0x3ff02438	RW	一级交叉开关地址窗口	0x0
EAST_WIN0_MASK	0x3ff02440	RW	一级交叉开关地址窗口	0x0
EAST_WIN1_MASK	0x3ff02448	RW	一级交叉开关地址窗口	0x0
EAST_WIN2_MASK	0x3ff02450	RW	一级交叉开关地址窗口	0x0
EAST_WIN3_MASK	0x3ff02458	RW	一级交叉开关地址窗口	0x0
EAST_WIN4_MASK	0x3ff02460	RW	一级交叉开关地址窗口	0x0
EAST_WIN5_MASK	0x3ff02468	RW	一级交叉开关地址窗口	0x0
EAST_WIN6_MASK	0x3ff02470	RW	一级交叉开关地址窗口	0x0
EAST_WIN7_MASK	0x3ff02478	RW	一级交叉开关地址窗口	0x0
EAST_WIN0_MMAP	0x3ff02480	RW	一级交叉开关地址窗口	0x0
EAST_WIN1_MMAP	0x3ff02488	RW	一级交叉开关地址窗口	0x0
EAST_WIN2_MMAP	0x3ff02490	RW	一级交叉开关地址窗口	0x0
EAST_WIN3_MMAP	0x3ff02498	RW	一级交叉开关地址窗口	0x0
EAST_WIN4_MMAP	0x3ff024a0	RW	一级交叉开关地址窗口	0x0

EAST_WIN5_MMAP	0x3ff024a8	RW	一级交叉开关地址窗口	0x0
EAST_WIN6_MMAP	0x3ff024b0	RW	一级交叉开关地址窗口	0x0
EAST_WIN7_MMAP	0x3ff024b8	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN0_BASE	0x3ff02500	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN1_BASE	0x3ff02508	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN2_BASE	0x3ff02510	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN3_BASE	0x3ff02518	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN4_BASE	0x3ff02520	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN5_BASE	0x3ff02528	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN6_BASE	0x3ff02530	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN7_BASE	0x3ff02538	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN0_MASK	0x3ff02540	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN1_MASK	0x3ff02548	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN2_MASK	0x3ff02550	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN3_MASK	0x3ff02558	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN4_MASK	0x3ff02560	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN5_MASK	0x3ff02568	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN6_MASK	0x3ff02570	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN7_MASK	0x3ff02578	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN0_MMAP	0x3ff02580	RW	一级交叉开关地址窗口	0x0

SOUTH_WIN1_MMAP	0x3ff02588	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN2_MMAP	0x3ff02590	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN3_MMAP	0x3ff02598	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN4_MMAP	0x3ff025a0	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN5_MMAP	0x3ff025a8	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN6_MMAP	0x3ff025b0	RW	一级交叉开关地址窗口	0x0
SOUTH_WIN7_MMAP	0x3ff025b8	RW	一级交叉开关地址窗口	0x0
WEST_WIN0_BASE	0x3ff02600	RW	一级交叉开关地址窗口	0x0
WEST_WIN1_BASE	0x3ff02608	RW	一级交叉开关地址窗口	0x0
WEST_WIN2_BASE	0x3ff02610	RW	一级交叉开关地址窗口	0x0
WEST_WIN3_BASE	0x3ff02618	RW	一级交叉开关地址窗口	0x0
WEST_WIN4_BASE	0x3ff02620	RW	一级交叉开关地址窗口	0x0
WEST_WIN5_BASE	0x3ff02628	RW	一级交叉开关地址窗口	0x0
WEST_WIN6_BASE	0x3ff02630	RW	一级交叉开关地址窗口	0x0
WEST_WIN7_BASE	0x3ff02638	RW	一级交叉开关地址窗口	0x0
WEST_WIN0_MASK	0x3ff02640	RW	一级交叉开关地址窗口	0x0
WEST_WIN1_MASK	0x3ff02648	RW	一级交叉开关地址窗口	0x0
WEST_WIN2_MASK	0x3ff02650	RW	一级交叉开关地址窗口	0x0
WEST_WIN3_MASK	0x3ff02658	RW	一级交叉开关地址窗口	0x0
WEST_WIN4_MASK	0x3ff02660	RW	一级交叉开关地址窗口	0x0

WEST_WIN5_MASK	0x3ff02668	RW	一级交叉开关地址窗口	0x0
WEST_WIN6_MASK	0x3ff02670	RW	一级交叉开关地址窗口	0x0
WEST_WIN7_MASK	0x3ff02678	RW	一级交叉开关地址窗口	0x0
WEST_WIN0_MMAP	0x3ff02680	RW	一级交叉开关地址窗口	0x0
WEST_WIN1_MMAP	0x3ff02688	RW	一级交叉开关地址窗口	0x0
WEST_WIN2_MMAP	0x3ff02690	RW	一级交叉开关地址窗口	0x0
WEST_WIN3_MMAP	0x3ff02698	RW	一级交叉开关地址窗口	0x0
WEST_WIN4_MMAP	0x3ff026a0	RW	一级交叉开关地址窗口	0x0
WEST_WIN5_MMAP	0x3ff026a8	RW	一级交叉开关地址窗口	0x0
WEST_WIN6_MMAP	0x3ff026b0	RW	一级交叉开关地址窗口	0x0
WEST_WIN7_MMAP	0x3ff026b8	RW	一级交叉开关地址窗口	0x0
NORTH_WIN0_BASE	0x3ff02700	RW	一级交叉开关地址窗口	0x0
NORTH_WIN1_BASE	0x3ff02708	RW	一级交叉开关地址窗口	0x0
NORTH_WIN2_BASE	0x3ff02710	RW	一级交叉开关地址窗口	0x0
NORTH_WIN3_BASE	0x3ff02718	RW	一级交叉开关地址窗口	0x0
NORTH_WIN4_BASE	0x3ff02720	RW	一级交叉开关地址窗口	0x0
NORTH_WIN5_BASE	0x3ff02728	RW	一级交叉开关地址窗口	0x0
NORTH_WIN6_BASE	0x3ff02730	RW	一级交叉开关地址窗口	0x0
NORTH_WIN7_BASE	0x3ff02738	RW	一级交叉开关地址窗口	0x0
NORTH_WIN0_MASK	0x3ff02740	RW	一级交叉开关地址窗口	0x0

NORTH_WIN1_MASK	0x3ff02748	RW	一级交叉开关地址窗口	0x0
NORTH_WIN2_MASK	0x3ff02750	RW	一级交叉开关地址窗口	0x0
NORTH_WIN3_MASK	0x3ff02758	RW	一级交叉开关地址窗口	0x0
NORTH_WIN4_MASK	0x3ff02760	RW	一级交叉开关地址窗口	0x0
NORTH_WIN5_MASK	0x3ff02768	RW	一级交叉开关地址窗口	0x0
NORTH_WIN6_MASK	0x3ff02770	RW	一级交叉开关地址窗口	0x0
NORTH_WIN7_MASK	0x3ff02778	RW	一级交叉开关地址窗口	0x0
NORTH_WIN0_MMAP	0x3ff02780	RW	一级交叉开关地址窗口	0x0
NORTH_WIN1_MMAP	0x3ff02788	RW	一级交叉开关地址窗口	0x0
NORTH_WIN2_MMAP	0x3ff02790	RW	一级交叉开关地址窗口	0x0
NORTH_WIN3_MMAP	0x3ff02798	RW	一级交叉开关地址窗口	0x0
NORTH_WIN4_MMAP	0x3ff027a0	RW	一级交叉开关地址窗口	0x0
NORTH_WIN5_MMAP	0x3ff027a8	RW	一级交叉开关地址窗口	0x0
NORTH_WIN6_MMAP	0x3ff027b0	RW	一级交叉开关地址窗口	0x0
NORTH_WIN7_MMAP	0x3ff027b8	RW	一级交叉开关地址窗口	0x0

13 软硬件设计指南

龙芯 3A3000/3B3000 处理器引脚向下兼容龙芯 3A1000 处理器，但是相应的软硬件需要进行一些配置的变更，以使能原有的兼容模式，或者打开龙芯 3A3000/3B3000 的一些新特性，本章重点介绍与龙芯 3A1000/2000 相比，龙芯 3A3000/3B3000 处理器的使用上的软硬件设置区别。

13.1 硬件改动指南

1. 原有 CORE_PLL_AVDD、DDR_PLL_AVDD（2.5v）现为 1.8v。如果使用原有的 3A1000 主板，需要将这两个电源由 2.5v 改为 1.8v。而 3A2000/3B2000 上这些引脚（包括 HT0/1_PLL_AVDD）为 NC，如果考虑与 3A2000/3B2000 的兼容性，可以将这些电源电压修改为 1.8v，或是采用 1.8v/2.5v 可配置的设计；

2. 原有的 MC0/1_COMP_REF_RES 改为 NC pin。如果使用原有的 3A 主板，可以不作修改；（与 3A2000 一致）

3. 原有的 HT0/1_PLL_REF 改为 NC pin。如果使用原有的 3A 主板，可以不作修改；（与 3A2000 一致）

4. 原有的 MC0/1_COMP_REF_GND 改为 MC0/1_A15。如果使用原有的 3A 主板，可以不作修改；但如果连接到内存条，可以支持更大容量内存；（与 3A2000 一致）

5. PCI_CONFIG[0]控制的功能改为 SPI 启动使能，设置为 1 后可以从 SPI FLASH 启动。如果使用原有的 3A 主板，需要设置为 0，从 LPC FLASH 启动；如果主板已有 SPI FLASH，可以将 GPIO[0]作为 SPI_CS 连接，并将 PCI_CONFIG[0]设置为 1，从 SPI FLASH 启动；（与 3A2000 一致）

6. PCI_CONFIG[7]控制的功能改为强制 HT1.0 模式，设置为 1 后 HT 直接采用 1.0 模式启动。如果使用 3A780E 主板，目前需要将其设为 1；如果使用 3A2H 主板，无需特别设置；（与 3A2000 一致）

7. 针对龙芯 3A3000/3B3000A/B/C，CLKSEL[15:10]需要设置为 6’b000000 或 6’b000001，并

在 PMON 中使用软件重新对频率进行配置；

8. CLKSEL[9:5]需要设置为 5’b01111；使用 PMON 进行内存频率设置。针对龙芯 3A3000/3B3000A/B/C，PMON 在频率配置的时候必须使用 NODE 时钟进行分频；（与 3A2000 一致）

9. CLKSEL[4:0]需要设置为 5’b01111；使用 PMON 进行处理器核频率设置。针对龙芯 3A3000/3B3000A/B/C，PMON 在频率配置的时候必须使用 L2 PLL 作为主时钟；（与 3A2000 一致）

10. 对于 3A2H 主板，需要去除 HT0/1_powerok、HT0/1_resetn 上的上拉电阻；（原有的上拉电阻 300 欧姆对于 3A 也不合适，同样可以去除）（与 3A2000 一致）

13.2 频率设置说明

为了与龙芯 3A1000 的频率配置基本兼容，龙芯 3A3000/3B3000 的硬件频率配置范围较窄，为了获得更宽的频率范围和更好的时钟质量，在龙芯 3A3000/3B3000 中主要使用在 PMON 中进行软件配置的方法，配置方法与龙芯 3B1500 相同。具体配置方法请参考 PMON 源代码。

1. 频率设置完全由软件设置，改变频率时不用修改 CLKSEL；

2. 1.25V 核心电压的稳定工作频率：处理器核频率设为 1400MHz，内存频率设为 700MHz，HT 控制器设置为 800MHz，HT 总线 800MHz/1600MHz；

3. 针对龙芯 3A3000/3B3000A/B/C，NODE CLOCK 必须使用 L2 PLL 作为主时钟，DDR CLOCK 必须使用 NODE 时钟进行分频；

13.3 PMON改动指南

以下的 PMON 改动与 3A2000/3B2000 处理器基本一致。

因为从处理器核、内存控制器、HT 控制器到各级交叉开关都进行了不同程度的升级，

因此相比龙芯 3A1000，PMON 需要进行一些改动，主要包括以下必须部分：

1. 去除上电后 L1 Dcache， L1 Icache，Vcache，L2 Cache 的初始化操作（硬件完成）；
2. 在 CPU 刚上电后，关闭所有核的 Store Fill Buffer；
3. 在 CPU 刚上电后，关闭所有核的字写合并功能；
4. 如果需要保持对 3A5 的兼容，设置所有核的 CP0 Diag 寄存器中的 PRID 隐藏位；
5. 将所有汇编代码中 jr rx，rx 不为 31 号寄存器的语句修改为 jr $31；
6. 使用与 3B1500 类似的配置处理器核、内存与节点 PLL 的代码；
7. 使用与 3B1500 类似的内存控制器配置与参数训练的代码；
8. 如果 HT 工作在 1.0 的模式下，HT 只能工作在 8 位模式；
9. 如果使用到 SPI 控制器，基地址由 0xBFE001F0 修改为 0xBFE00220；

除了这些必须的改动，还可以进行以下的改动以增强 PMON 功能：

1. 修改蜂鸣器的 delay 延迟，确保用户能听到蜂鸣声；
2. 添加关闭有缺陷核时钟的支持；
3. 去掉代码中 3A5 对 2h 桥片 HT 控制器的部分 workaround（仍保留部分 workaround）；

13.4 内核改动指南

以下的内核改动与 3A2000/3B2000 基本一致，但需要在内核中对应的部分添加对 3A3000/3B3000 处理器的支持。

1. 修改内核中的 Cache 描述结构，VCache 与 SCache 都使用 16 路组相连；（与 3A2000 一致）

2. 修改温度传感器的计算方式，算法为：结点温度=Thens_out *731/0x4000 - 273；

3. 修改关核时的配置寄存器地址；（与 3A2000 一致）

4. 将刷 ICache/DCache 的操作改为刷 ICache/DCache/VCache；（与 3A2000 一致）

5. 如果使用到 SPI 控制器，基地址由 0xBFE001F0 修改为 0xBFE00220；（与 3A2000 一致）

6. 必须使用 Uncache DMA，采用软件维护 Cache 的数据一致性；（与 3A2000 一致）

7. 增加 store fill buffer 支持：一是需要在所有的 Uncache 请求之前增加一条 SYNC，以保证在 Uncache 请求发生时，store fill buffer 里的内容已经全部写回 Cache；二是需要将所有的在不同核间共享的同步操作中的解锁操作使用 LL/SC 指令实现。（与 3A2000 一致）

8. 不使用设备的 MSI 功能。当必须使用 MSI 功能时，需要将 HT 控制器的 POST 通道的数据接收缓冲区个数设置为 1，并重连 HT 总线；（与 3A2000 一致）

9. 对于硬件自动维护一致性的 DMA 区域不能使用锁 Cache 操作。（与 3A2000 一致）

还可以采用的用于提升性能的修改有：

1. 增加对 FTLB 的支持；（与 3A2000 一致）

2. 增加对 TLB 快速重填的支持；（与 3A2000 一致）

3. 增加 wait 指令支持；（与 3A2000 一致）

4. 增加预取指令支持；（与 3A2000 一致）

5. 使用 DI/EI 实现中断返回。但需要注意的是 EI 指令返回的[31:4]为随机值，与 MIPS 规定有差异。（与 3A2000 一致）