龙芯中科
LOONGSON TECHNOLOGY

LOONGSON

龙芯 3C6000 处理器

寄存器使用手册

多核处理器架构、寄存器描述与系统软件编程指南

V1.0

2026 年 02 月

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

阅读指南

《龙芯 3C6000 处理器寄存器使用手册》介绍龙芯 3C6000 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。

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处理器核以及大量的共享 Cache，还可以通过高速 I/O 接口实现多芯片的互连以组成更大规模的系统。

1.2 龙芯 3C6000 简介

龙芯 3C6000 是基于同一个硅片设计的多种不同封装的处理器系列。包括单硅片的 LS3C6000/S、双硅片的 LS3C6000/D（或 LS3D6000）和四硅片的 LS3C6000/Q（或 LS3E6000）。对于用户软件使用上，主要差异在于多路互连带来的地址空间扩展。以下除了特别说明，主要使用 3C6000 指代单个硅片或结点，适用于整个 3C6000 系列。

龙芯 3C6000 采用 LA664 处理器核，大幅提升计算与访存性能；片上互连网络进行了调整优化；高速 IO 接口使用 PCIE4.0；片间互连接口使用 LCL。

龙芯 3C6000 的整体架构逻辑上基于多级互连实现，如图 1-1 所示。

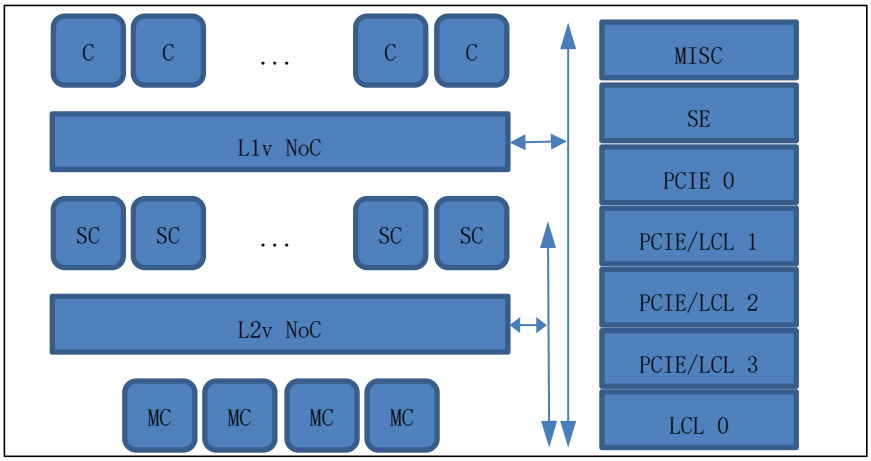


图 1-1 处理器结构示意图

标注说明：

标注	说明	标注	说明
C	处理器核	L1v NoC	一级片上网络
SC	共享高速缓存	L2v NoC	二级片上网络
MC	内存控制器	MISC	启动及配置模块
SE	安全模块	PCIE	外设接口
LCL	龙芯一致性互连接口		

1.2.1 LS3C6000/S

LS3C6000/S 是龙芯单硅片十六核处理器，封装为 FCBGA-2422，单硅片集成 16 个物理核，共包含 32 个逻辑核，基础主频为 2.0GHz - 2.2GHz，主要面向高端服务器领域。其主要技术特征如下：

- 片内集成 16 个 64 位的六发射超标量 LA664 高性能处理器核，每个物理核为两个逻辑核；
- 峰值双精度浮点运算能力 844.8GFLOPS@2.2GHz；
- 峰值单精度浮点运算能力 1689.6GFLOPS@2.2GHz；
- 片内集成 32MB 的分体共享三级 Cache；
- 通过目录协议维护多核及 I/O DMA 访问的 Cache 一致性；
- 内存接口为 4 个 72 位 DDR4 控制器，支持 DDR4-3200；
- 片内集成 4 个 16 位 PCIE 接口，最高速率 16Gbps，其中 1 个可用作 LCL 互连接口；
- 其它接口包括 1 个 SPI、1 个 UART、3 个 I2C、1 个 AVS、16 个 GPIO 接口等。

1.2.2 LS3C6000/D

LS3C6000/D 是基于两个 3C6000 硅片合封的 32 核处理器，也称作 LS3D6000，其封装为 FCLGA-4129，片上共集成 32 个物理核，共包含 64 个逻辑核，基础主频为 2.0GHz - 2.1GHz，主要面向高端服务器领域。其主要技术特征如下：

- 片内集成 32 个 64 位的六发射超标量 LA664 高性能处理器核，每个物理核为两个逻辑核；
- 峰值双精度浮点运算能力 1536GFLOPS@2.0GHz；
- 峰值单精度浮点运算能力 3072GFLOPS@2.0GHz
- 每硅片内集成 32MB 的分体共享三级 Cache；
- 通过目录协议维护多核及 I/O DMA 访问的 Cache 一致性；
- 内存接口为 8 个 72 位 DDR4 控制器，支持 DDR4-3200；
- 片内集成 8 个 16 位 PCIE 接口，最高速率 16Gbps，其中 6 个可用作 LCL 互连接口；

- 其它接口包括 2 个 SPI、1 个 UART、5 个 I2C、1 个 AVS、17 个 GPIO 接口等。

LS3C6000/D 基于两个 3C6000 硅片合封实现，其软件使用方式与多结点的 3C6000 完全一致。其封装互连结构如下图所示。

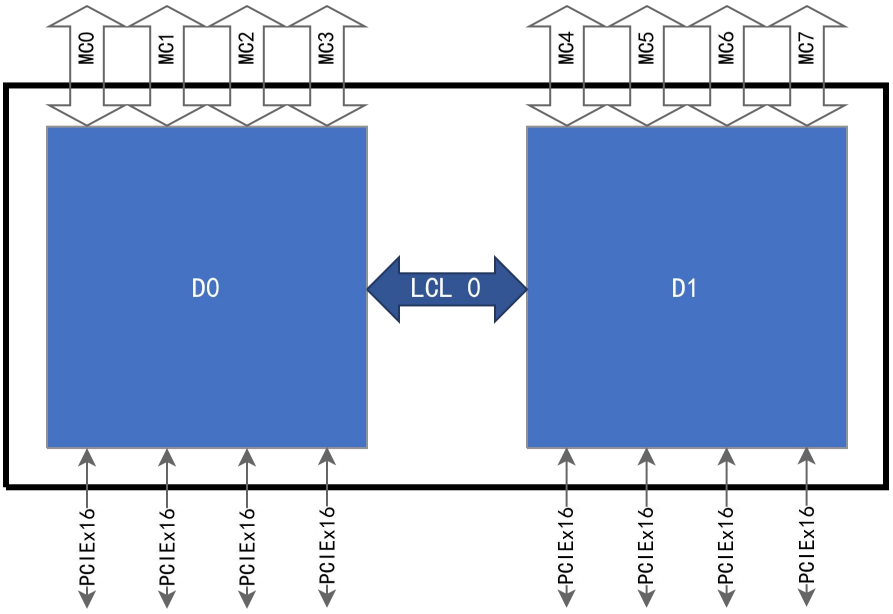


图 1-2LS3C6000/D 处理器结构示意图

1.2.3 LS3C6000/Q

LS3C6000/Q 是基于四个 3C6000 硅片合封的处理器，有 60 核与 64 核两种型号，也称作 LS3E6000，其封装为 FCBGA-6128，片上共集成 60 或 64 个物理核，包含 120 或 128 个逻辑核，基础主频为 1.8-2.0GHz，主要面向高端服务器领域。其主要技术特征如下：

- 片内集成 60 或 64 个 64 位的六发射超标量 LA664 高性能处理器核，每个物理核为两个逻辑核；
- 峰值双精度浮点运算能力 3072GFLOPS@2.0GHz（64 核）；
- 峰值单精度浮点运算能力 6144GFLOPS@2.0GHz（64 核）；
- 每个硅片内集成 32MB 的分体共享三级 Cache；
- 通过目录协议维护多核及 I/O DMA 访问的 Cache 一致性；
- 内存接口为 8 个 72 位 DDR4 控制器，支持 DDR4-3200；
- 片内集成 8 个 16 位 PCIE 接口，最高速率 16Gbps，其中 4 个可用作 LCL 互连接口；
- 其它接口包括 4 个 SPI、4 个 UART、10 个 I2C、2 个 AVS、34 个 GPIO 接口等。

LS3C6000/Q 基于四个 3C6000 硅片合封实现，其软件使用方式与多结点的 3C6000 完全一致。其封装互连结构如下图所示。

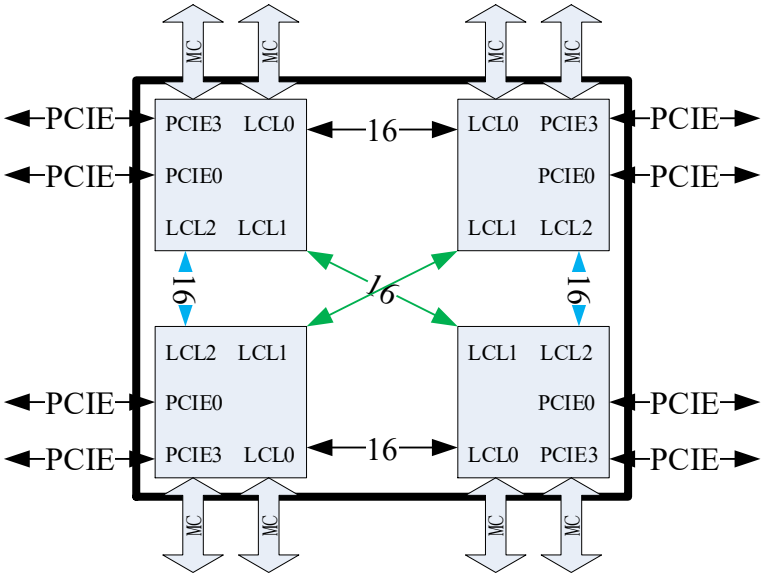


图 1-3LS3C6000/Q 处理器结构示意图

2 系统配置与控制

2.1 芯片工作模式

根据具体型号的不同以及系统规模的需要，芯片的工作模式也有所差异。以下分别进行说明。

2.1.1 LS3C6000/S

LS3C6000/S 主要包括两种工作模式：

- 单芯片模式。系统包含 1 片 LS3C6000/S，是一个单结点十六核处理器系统；
- 双芯片互连模式。系统中包含 2 片，通过 LCL 端口进行互连，构成一个双结点的非均匀访存多处理器系统（CC-NUMA）。

不同的工作模式如下：

（1） 单路服务器，LS3C6000/S 单处理器系统。使用 PCIE0 的低 8 位接口用于 IO 桥片连接。一种常见的连接方式如图 2-1 所示：

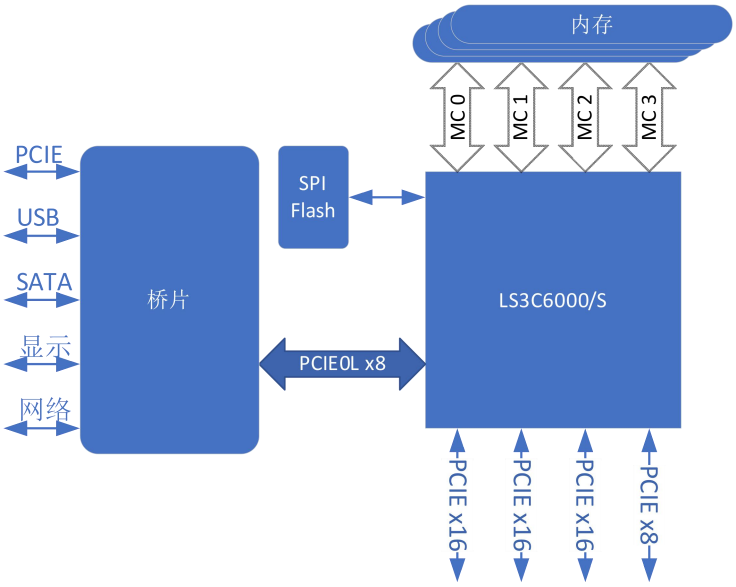


图 2-1LS3C6000/S 单处理器系统

（2） 双路服务器，LS3C6000/S 双处理器系统。使用 PCIE0 的低 8 位接口用于 IO 桥片连接；使用 PCIE/LCL2 接口用于多处理器间互连。一种常见的连接方式如图 2-2 所示：

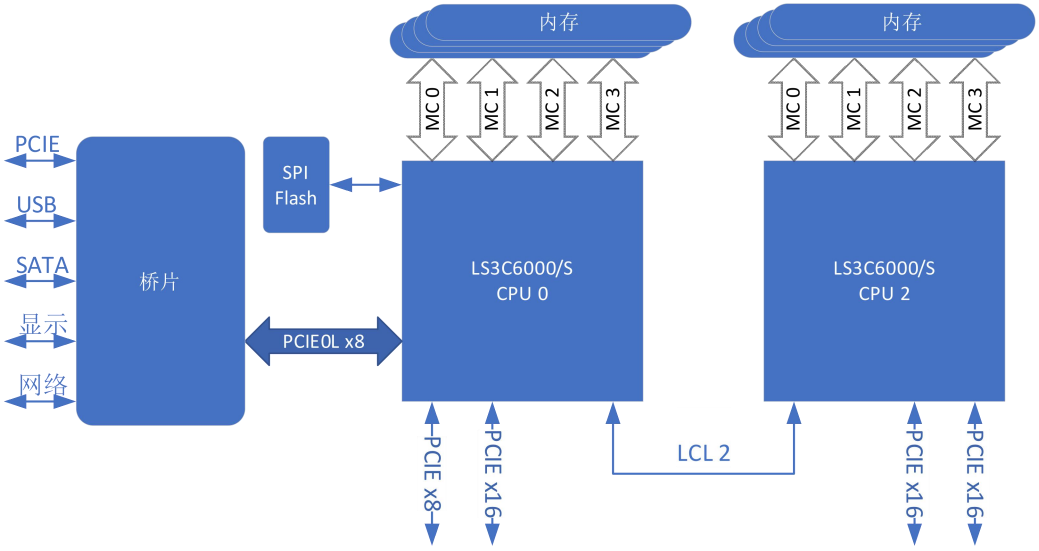


图 2-2LS3C6000/S 双处理器系统

2.1.2 LS3C6000/D

LS3C6000/D 主要分为以下几种工作模式：

- 单芯片模式。系统包含 1 片 LS3C6000/D，是一个双结点 32 核 64 线程 CC-NUMA 处理器系统；
- 双芯片互连模式。系统中包含 2 片 LS3C6000/D，通过 LCL 端口进行互连，构成一个四结点 64 核 128 线程 CC-NUMA 处理器系统；
- 四芯片互连模式。系统中包含 4 片 LS3C6000/D，通过 LCL 端口进行互连，构成一个八结点 128 核 256 线程 CC-NUMA 处理器系统。

根据接口连接硅片的不同，接口命名中增加 D0 或 D1。例如，D0 上的 PCIE0，称为 D0PCIE0 或 D0P0。

不同的工作模式具体如下：

（1） 单路单连服务器，LS3C6000/D 单处理器系统。使用 D0PCIE0 的低 8 位接口用于 IO 桥片连接。其连接方式如图 2-1 所示：

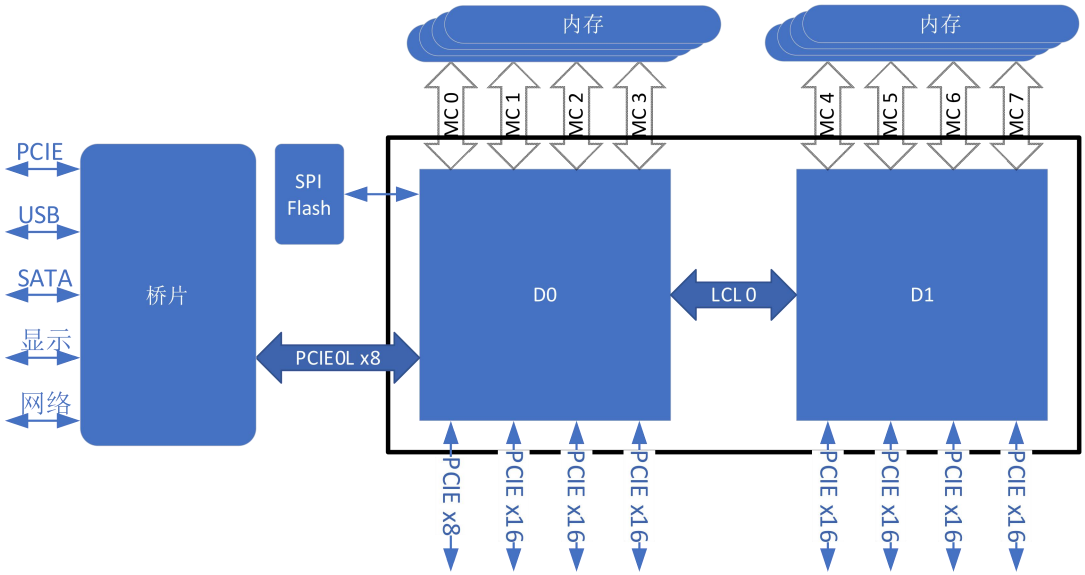


图 2-3LS3C6000/D 单路单连系统结构

（2） 单路双连服务器，LS3C6000/D 单处理器系统。使用 D0PCIE0 的低 8 位接口用于 IO 桥片连接。使用空闲的 PCIE1 或 PCIE2 进行互连，以提高两个硅片之间的连接性能。其连接方式如下图所示：

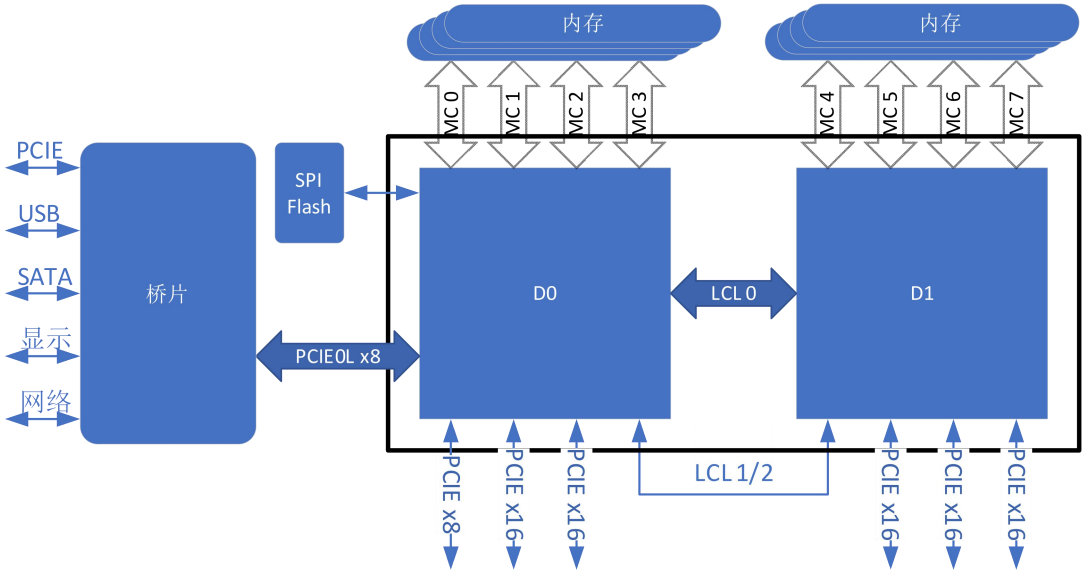


图 2-4LS3C6000/D 单路双连系统结构

（3） 单路三连服务器，LS3C6000/D 单处理器系统。使用 D0PCIE0 的低 8 位接口用于 IO 桥片连接。使用空闲的 PCIE1 和 PCIE2 进行互连，以提高两个硅片之间的连接性能。其连接方式如下图所示：

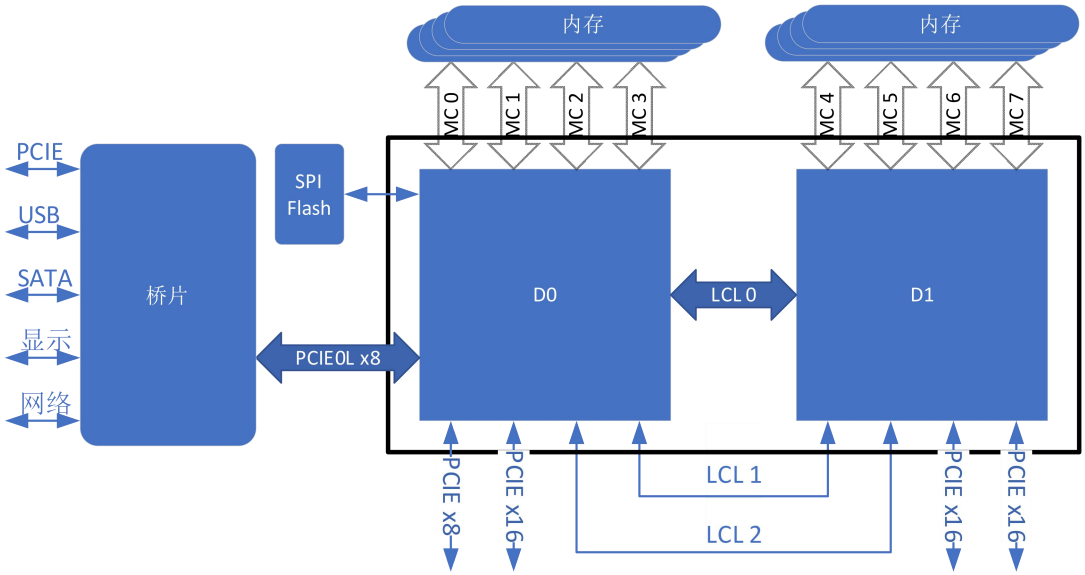


图 2-5LS3C6000/D 单路三连系统结构

（4） 双路服务器，LS3C6000/D 双处理器系统。可以使用 D0PCIE0 的低 8 位接口用于 IO 桥片连接。两种常见的连接方式如下图所示。其中（a）对 D2 的 LCL2/3（图中 02/03）进行了交换；（b）对 D1 的 LCL1/3（图中 11/13）、D3 的 LCL1/3（图中 11/13）进行了交换，以将 LCL1 用作 PCIE 使用。图中其它未连接的接口可作为 PCIE 进行扩展连接。

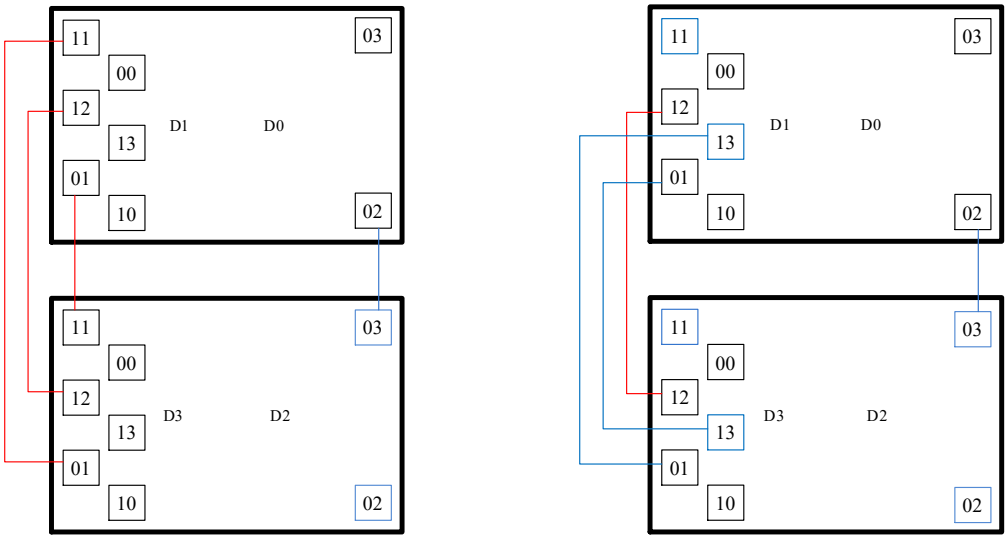


图 2-6LS3C6000/D 双处理器系统

（5） 四路服务器，LS3C6000/D 四处理器系统。使用 D0PCIE0 的低 8 位接口用于 IO 桥片连接。常见的连接方式如下图所示。图中对所有硅片的 LCL2/3（图中的

02/03 和 12/13）都进行了交换。实际设计中是否需要交换可以根据布线需求具体确定。

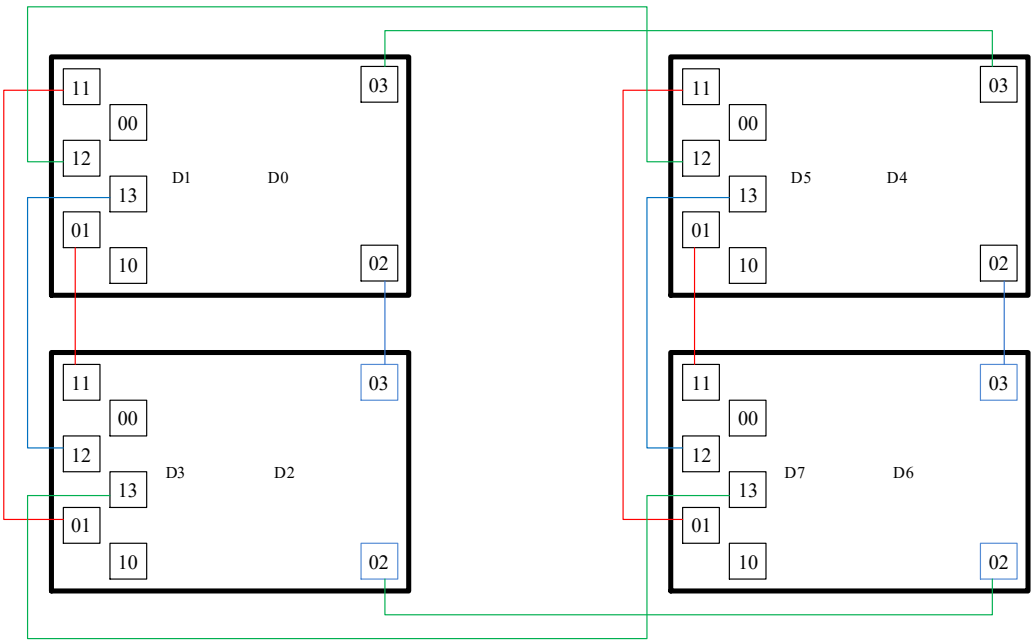


图 2-7LS3C6000/D 四处理器系统

2.1.3 LS3C6000/Q

LS3C6000/Q 的互连与 LS3C6000/D 中的硅片互连模式类似，在此不再赘述。其中单路 LS3C6000/Q 与双路 LS3C6000/D 类似；双路 LS3C6000/Q 与四路 LS3C6000/D 类似。

2.2 控制引脚说明

主要控制引脚包括 DO_TEST、ICCC_EN[1:0]、CHIP_ID[3:0]、CHIP_CONFIG[6:0]。

表 2-1 控制引脚说明

信号	上下拉	作用
DO_TEST	上拉	1'b1 表示功能模式 1'b0 表示测试模式
ICCC_EN[1:0]	下拉	多路一致性模式 2'b00：单硅片模式（对应 LS3C6000/S 单路） 2'b01：双硅片模式（对应 LS3C6000/D 单路，默认启用 LCL0） 2'b10：四硅片模式（对应 LS3C6000/S 和 LS3C6000/D 双路、LS3C6000/Q 单路，默认启用 LCL0/1/2） 2'b11：八硅片模式（对应 LS3C6000/D 四路、LS3C6000/Q 单路，默认启用 LCL0/1/2/3）

CHIP_ID[3:0]	下拉	在多芯片一致性互连模式下表示每个硅片的处理器结点号 需要注意的是在 LS3C6000/S 双路下，使用的 CHIP_ID 分别为 0 和 2
CHIP_CONFIG[6]	下拉	参考时钟选择 1'b1：使用 SYSCLK 作为参考时钟 1'b0：使用差分时钟作为参考时钟
CHIP_CONFIG[5]	上拉	差分时钟选择 1'b1：使用 SYSCLK_I0p/n 作为差分输入 1'b0：使用 SYSCLK_I1p/n 作为差分输入
CHIP_CONFIG[4]	下拉	使用 LCL8 位模式
CHIP_CONFIG[3]	下拉	交换 LCL13 路由
CHIP_CONFIG[2]	下拉	交换 LCL23 路由
CHIP_CONFIG[1]	下拉	本地启动模式
CHIP_CONFIG[0]	下拉	SE 功能使能

3 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。

对于龙芯 3C6000，每个硅片为一个结点，即 44 位地址空间。需要注意的是，LS3C6000/S 双路互连时使用结点号 0 和 2。每个硅片高位的结点地址由其 CHIP_ID 决定。

3.1 结点间的物理地址空间分布

除了单片模式，龙芯 3C6000 处理器还可以采用多个硅片直连构建 CC-NUMA 系统。对于龙芯 3C6000，每个芯片为一个结点，即 44 位地址空间。根据使用的硅片 CHIP_ID 决定可访问的地址空间，其它地址为非法地址。其地址分布具体如下：

表 3-1 系统全局地址分布

起始地址	结束地址	地址[47:44]	硅片号
0x[ID]000_0000_0000	0x[ID]FFF_FFFF_FFFF	[ID]	[ID]

多硅片互连时，应设置扩展路由设置寄存器（IOCSR[0x0400]）的 chipmask 字段，当发生猜测访问时，保证即使没有物理结点的地址也能够得到响应。

设置方法如下：

双硅片：0x1；四硅片：0x3；八硅片：0x7。

需要特别说明的是，LS3C6000/S 双路需要设置为 0x2。

3.2 结点内的物理地址空间分布

龙芯 3C6000 采用每结点 16 核 32 线程配置，龙芯 3C6000 芯片集成的 DDR 内存控制器、PCIE 接口的对应地址都包含在每个结点内从 0x0（含）至 0x1000_0000_0000（不含）的 44 位地址空间内。在结点内部，44 位地址空间又进一步划分给结点内连接的模块，仅当访问类型为 Cache 时，请求会被路由到共享 Cache 模块。

共享 Cache 的交叉寻址方式根据地址散列来确定具体访问目标，并可以通过软件在使用 Cache 之前配置修改。

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。SCID_SEL 位于路由设置寄存器（IOCSR[0x400]）。

表 3-2 SCID_SEL 地址位设置

SCID_SEL	地址位选择
4'b0000	全地址散列
4'b0001	根据[9:6]选择目标位置

龙芯 3C6000 处理器结点的内部 44 位物理地址的默认分布如下表所示。其中目标 MC 由 interleave 和 MEM_MAP 的组合决定，这些配置位位于路由设置寄存器（IOCSR[0x400]）。而 PCI 空间的具体定义可以参考 PCIE 的相关章节。

表 3-3 结点内 44 位物理地址分布

起始地址	结束地址	大小	属性	说明
0x0	0x0FFF_FFFF	256MB	-	内存空间
0x1000_0000	0x17FF_FFFF	128MB	Uncache	PCI Memory 0
0x1800_0000	0x19FF_FFFF	32MB	Uncache	PCI IO
0x1A00_0000	0x1BFF_FFFF	32MB	Uncache	PCI HEADER
0x1000_0000	0x1BFF_FFFF	192MB	Cache	保留
0x1C00_0000	0x1FDF_FFFF	62MB	-	BOOT Flash
0x1FE0_0000	0x1FFF_FFFF	2MB	Uncache	配置空间
0x1FE0_0000	0x1FFF_FFFF	2MB	Cache	保留
0x2000_0000	0x7FFF_FFFF	1.5GB	Uncache	PCI Memory 1
0x2000_0000	0x7FFF_FFFF	1.5GB	Cache	保留
0x8000_0000	0x8FFF_FFFF	256MB	-	保留
0x9000_0000	0xFFFF_FFFF	1.75GB	-	内存空间
0x1_0000_0000	0x9FF_FFFF_FFFF	10236GB	-	内存空间
0xA00_0000_0000	0xA00_0FFF_FFFF	256MB	Uncache	LCL0 配置空间
0xA00_1000_0000	0xA00_1FFF_FFFF	256MB	Uncache	LCL1 配置空间
0xA00_2000_0000	0xA00_2FFF_FFFF	256MB	Uncache	LCL2 配置空间
0xA00_3000_0000	0xA00_3FFF_FFFF	256MB	Uncache	LCL3 配置空间
0xA00_0000_0000	0xA00_3FFF_FFFF	1GB	Cache	保留
0xA00_4000_0000	0xBFF_FFFF_FFFF	2047GB	-	保留
0xC00_0000_0000	0xCFF_FFFF_FFFF	1TB	Uncache	SE
0xD00_0000_0000	0xDFF_FFFF_FFFF	1TB	Uncache	保留
0xC00_0000_0000	0xDFF_FFFF_FFFF	2TB	Cache	保留
0xE00_0000_0000	0xFFF_FFFF_FFFF	2TB	-	PCI 空间

3.2.1 内存地址访问

每个龙芯 3C6000 中所集成的四个内存控制器是通过软件配置路由进行访问的。

在对 MEM_MAP 进行了正确的配置之后，就可以访问到特定的内存控制器。

例如，在内存控制器初始化时，可以先禁用内存地址交错，通过将一个固定的地址（例如 0xFF00000）逐次映射给 MC0/1/2/3，来进行各个内存控制器的初始化。在初始化完成之

后，再使能地址交错，将不同的地址映射到不同的内存上。

3.2.2 PCIE 地址访问

每个龙芯 3C6000 中集成了 8 个 PCIE 控制器，这 8 个控制器与 64 个通道相复用。

软件上，这些 PCIE 控制器位于同一个 PCIE 地址空间内，通过 PCIE 标准的配置空间、IO 空间和 MEM 空间进行访问。

PCIE 地址空间为 0xE00_0000_0000 - 0xEFF_FFFF_FFFF。

3.2.3 配置窗口映射

对于其它地址，可以通过地址窗口映射进行访问。龙芯 3C6000 路由主要通过两级网络结构实现。软件可以对每个主端口接收到的请求进行路由配置，每个从端口都拥有 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 1MB 对齐；MASK 采用类似网络掩码高位为 1 的格式；MMAP 的低四位表示对应目标设备端口的编号，MMAP[4]表示允许取指，MMAP[5]表示允许块读，MMAP[6]表示允许交错访问使能，MMAP[7]表示窗口使能。

在 3C6000 中，对于超过 4 个的相同模块，将 4 个为一组，作为一个内部结点。例如，16 个处理器核，共 32 个逻辑核，以 4 个为一组，共分为 8 个内部结点；16 个 SCache，以 4 个为一组，共分为 4 个内部结点。具体的对应关系如表 3-6 所示。

在 3C6000 中，因为内部结点的引入，窗口配置时，需要增加内部结点号，放在 MMAP 寄存器的[9:8]两位上，同时将第[10]位供结点交错使能位使用。

这里要特别注意的是，3C6000 中地址窗口映射空间的最小单位为 1MB。

表 3-4 MMAP 寄存器位域说明

[63:48]	[47:20]	[19:17]	[16:11]	[10:4]	[3:0]
保留	转换后地址	PCIE 映射配置	保留	窗口使能配置	从设备号

其中“窗口使能配置”中的具体定义见下表。

表 3-5 MMAP 字段对应的空间访问属性

[10]	[9:8]	[7]	[6]	[5]	[4]	[3:0]
内部结点交错使能	内部结点号	窗口使能	允许对 Scache/DDR 进行交错访问，当从设备号为 0/4 时有效，按照对应的“交错选择位”配置进行路由。	允许块读	允许取指	设备号

对于设备的 ID 号，定义如下：

表 3-6 内部结点号和设备号

模块	设备号	内部结点号
Core	-	0 - 7
Scache	0 - 3	0 - 3
MC	4	0 - 3
LCL L1	A	0 - 3
LCL L2	B	0 - 3
SE	C	0
MISC	D	0 - 3（供中断使用）
PCIE0/2/4/6	E	0
PCIE1/3/5/7	F	0

其中，Core/Scache/MC/MISC 都有 4 个对应的内部结点。SE 只有内部结点 0。

对于命中 PCIE 空间的请求，则需要按照 MMAP 的[16:11]进行二次路由，其定义如下：

表 3-7 PCIE 映射配置定义

位	名称	说明
11	PCIE CTRL0 config	对应 PCIE Group 的 0 号控制器的配置空间
12	Base header	基础配置空间（不支持扩展寄存器的访问）
13	PCI IO	PCI IO 空间
14	PCIE MEM	PCI MEM 空间（当所有位为 0 号默认为 PCI MEM）
15	EXT header 0	扩展配置空间 0 格式的访问
16	EXT header 1	扩展配置空间 1 格式的访问

对于共享 Cache 的访问，不同于地址窗口的映射关系，龙芯 3C6000 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。共享 Cache 模块所对应的地址空间根据地址位的特定位确定，并可以通过软件在使用 Cache 前动态配置修改。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 3.2 节所述。

芯片缺省采用固定路由，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。

SCACHE/内存交错访问配置使能后，Slave 号仅为 0 或 4 时有效。为 0 表示路由到 SCACHE，并由 SCID_SEL 决定如何在 4 个 SCACHE 进行交错访问。为 4 表示路由到内存，由 interleave 和 MEM_MAP 决定如何在 4 个 MC 进行交错访问。

每个结点的地址窗口转换寄存器如下表所示。其中 SE、MISC 窗口只在结点 0 上有效。

例如：内部结点 0 寄存器基地址为 0x1FE0_0000；内部结点 4 寄存器基地址为 0x1FE4_0000。

表 3-8 地址窗口寄存器表

地址偏移	寄存器	地址偏移	寄存器
0x2000	CORE0_WIN0_BASE	0x2100	CORE1_WIN0_BASE
0x2008	CORE0_WIN1_BASE	0x2108	CORE1_WIN1_BASE
0x2010	CORE0_WIN2_BASE	0x2110	CORE1_WIN2_BASE
0x2018	CORE0_WIN3_BASE	0x2118	CORE1_WIN3_BASE
0x2020	CORE0_WIN4_BASE	0x2120	CORE1_WIN4_BASE
0x2028	CORE0_WIN5_BASE	0x2128	CORE1_WIN5_BASE
0x2030	CORE0_WIN6_BASE	0x2130	CORE1_WIN6_BASE
0x2038	CORE0_WIN7_BASE	0x2138	CORE1_WIN7_BASE
0x2040	CORE0_WIN0_MASK	0x2140	CORE1_WIN0_MASK
0x2048	CORE0_WIN1_MASK	0x2148	CORE1_WIN1_MASK
0x2050	CORE0_WIN2_MASK	0x2150	CORE1_WIN2_MASK
0x2058	CORE0_WIN3_MASK	0x2158	CORE1_WIN3_MASK
0x2060	CORE0_WIN4_MASK	0x2160	CORE1_WIN4_MASK
0x2068	CORE0_WIN5_MASK	0x2168	CORE1_WIN5_MASK
0x2070	CORE0_WIN6_MASK	0x2170	CORE1_WIN6_MASK
0x2078	CORE0_WIN7_MASK	0x2178	CORE1_WIN7_MASK
0x2080	CORE0_WIN0_MMAP	0x2180	CORE1_WIN0_MMAP
0x2088	CORE0_WIN1_MMAP	0x2188	CORE1_WIN1_MMAP
0x2090	CORE0_WIN2_MMAP	0x2190	CORE1_WIN2_MMAP
0x2098	CORE0_WIN3_MMAP	0x2198	CORE1_WIN3_MMAP
0x20a0	CORE0_WIN4_MMAP	0x21a0	CORE1_WIN4_MMAP
0x20a8	CORE0_WIN5_MMAP	0x21a8	CORE1_WIN5_MMAP
0x20b0	CORE0_WIN6_MMAP	0x21b0	CORE1_WIN6_MMAP
0x20b8	CORE0_WIN7_MMAP	0x21b8	CORE1_WIN7_MMAP
0x2200	CORE2_WIN0_BASE	0x2300	CORE3_WIN0_BASE
0x2208	CORE2_WIN1_BASE	0x2308	CORE3_WIN1_BASE
0x2210	CORE2_WIN2_BASE	0x2310	CORE3_WIN2_BASE
0x2218	CORE2_WIN3_BASE	0x2318	CORE3_WIN3_BASE
0x2220	CORE2_WIN4_BASE	0x2320	CORE3_WIN4_BASE
0x2228	CORE2_WIN5_BASE	0x2328	CORE3_WIN5_BASE
0x2230	CORE2_WIN6_BASE	0x2330	CORE3_WIN6_BASE
0x2238	CORE2_WIN7_BASE	0x2338	CORE3_WIN7_BASE
0x2240	CORE2_WIN0_MASK	0x2340	CORE3_WIN0_MASK
0x2248	CORE2_WIN1_MASK	0x2348	CORE3_WIN1_MASK
0x2250	CORE2_WIN2_MASK	0x2350	CORE3_WIN2_MASK
0x2258	CORE2_WIN3_MASK	0x2358	CORE3_WIN3_MASK
0x2260	CORE2_WIN4_MASK	0x2360	CORE3_WIN4_MASK
0x2268	CORE2_WIN5_MASK	0x2368	CORE3_WIN5_MASK

0x2270	CORE2_WIN6_MASK	0x2370	CORE3_WIN6_MASK
0x2278	CORE2_WIN7_MASK	0x2378	CORE3_WIN7_MASK
0x2280	CORE2_WIN0_MMAP	0x2380	CORE3_WIN0_MMAP
0x2288	CORE2_WIN1_MMAP	0x2388	CORE3_WIN1_MMAP
0x2290	CORE2_WIN2_MMAP	0x2390	CORE3_WIN2_MMAP
0x2298	CORE2_WIN3_MMAP	0x2398	CORE3_WIN3_MMAP
0x22a0	CORE2_WIN4_MMAP	0x23a0	CORE3_WIN4_MMAP
0x22a8	CORE2_WIN5_MMAP	0x23a8	CORE3_WIN5_MMAP
0x22b0	CORE2_WIN6_MMAP	0x23b0	CORE3_WIN6_MMAP
0x22b8	CORE2_WIN7_MMAP	0x23b8	CORE3_WIN7_MMAP
0x2400	SCACHE0_WIN0_BASE	0x2500	SCACHE1_WIN0_BASE
0x2408	SCACHE0_WIN1_BASE	0x2508	SCACHE1_WIN1_BASE
0x2410	SCACHE0_WIN2_BASE	0x2510	SCACHE1_WIN2_BASE
0x2418	SCACHE0_WIN3_BASE	0x2518	SCACHE1_WIN3_BASE
0x2420	SCACHE0_WIN4_BASE	0x2520	SCACHE1_WIN4_BASE
0x2428	SCACHE0_WIN5_BASE	0x2528	SCACHE1_WIN5_BASE
0x2430	SCACHE0_WIN6_BASE	0x2530	SCACHE1_WIN6_BASE
0x2438	SCACHE0_WIN7_BASE	0x2538	SCACHE1_WIN7_BASE
0x2440	SCACHE0_WIN0_MASK	0x2540	SCACHE1_WIN0_MASK
0x2448	SCACHE0_WIN1_MASK	0x2548	SCACHE1_WIN1_MASK
0x2450	SCACHE0_WIN2_MASK	0x2550	SCACHE1_WIN2_MASK
0x2458	SCACHE0_WIN3_MASK	0x2558	SCACHE1_WIN3_MASK
0x2460	SCACHE0_WIN4_MASK	0x2560	SCACHE1_WIN4_MASK
0x2468	SCACHE0_WIN5_MASK	0x2568	SCACHE1_WIN5_MASK
0x2470	SCACHE0_WIN6_MASK	0x2570	SCACHE1_WIN6_MASK
0x2478	SCACHE0_WIN7_MASK	0x2578	SCACHE1_WIN7_MASK
0x2480	SCACHE0_WIN0_MMAP	0x2580	SCACHE1_WIN0_MMAP
0x2488	SCACHE0_WIN1_MMAP	0x2588	SCACHE1_WIN1_MMAP
0x2490	SCACHE0_WIN2_MMAP	0x2590	SCACHE1_WIN2_MMAP
0x2498	SCACHE0_WIN3_MMAP	0x2598	SCACHE1_WIN3_MMAP
0x24a0	SCACHE0_WIN4_MMAP	0x25a0	SCACHE1_WIN4_MMAP
0x24a8	SCACHE0_WIN5_MMAP	0x25a8	SCACHE1_WIN5_MMAP
0x24b0	SCACHE0_WIN6_MMAP	0x25b0	SCACHE1_WIN6_MMAP
0x24b8	SCACHE0_WIN7_MMAP	0x25b8	SCACHE1_WIN7_MMAP
0x2600	SCACHE2_WIN0_BASE	0x2700	SCACHE3_WIN0_BASE
0x2608	SCACHE2_WIN1_BASE	0x2708	SCACHE3_WIN1_BASE
0x2610	SCACHE2_WIN2_BASE	0x2710	SCACHE3_WIN2_BASE
0x2618	SCACHE2_WIN3_BASE	0x2718	SCACHE3_WIN3_BASE
0x2620	SCACHE2_WIN4_BASE	0x2720	SCACHE3_WIN4_BASE
0x2628	SCACHE2_WIN5_BASE	0x2728	SCACHE3_WIN5_BASE

0x2630	SCACHE2_WIN6_BASE	0x2730	SCACHE3_WIN6_BASE
0x2638	SCACHE2_WIN7_BASE	0x2738	SCACHE3_WIN7_BASE
0x2640	SCACHE2_WIN0_MASK	0x2740	SCACHE3_WIN0_MASK
0x2648	SCACHE2_WIN1_MASK	0x2748	SCACHE3_WIN1_MASK
0x2650	SCACHE2_WIN2_MASK	0x2750	SCACHE3_WIN2_MASK
0x2658	SCACHE2_WIN3_MASK	0x2758	SCACHE3_WIN3_MASK
0x2660	SCACHE2_WIN4_MASK	0x2760	SCACHE3_WIN4_MASK
0x2668	SCACHE2_WIN5_MASK	0x2768	SCACHE3_WIN5_MASK
0x2670	SCACHE2_WIN6_MASK	0x2770	SCACHE3_WIN6_MASK
0x2678	SCACHE2_WIN7_MASK	0x2778	SCACHE3_WIN7_MASK
0x2680	SCACHE2_WIN0_MMAP	0x2780	SCACHE3_WIN0_MMAP
0x2688	SCACHE2_WIN1_MMAP	0x2788	SCACHE3_WIN1_MMAP
0x2690	SCACHE2_WIN2_MMAP	0x2790	SCACHE3_WIN2_MMAP
0x2698	SCACHE2_WIN3_MMAP	0x2798	SCACHE3_WIN3_MMAP
0x26a0	SCACHE2_WIN4_MMAP	0x27a0	SCACHE3_WIN4_MMAP
0x26a8	SCACHE2_WIN5_MMAP	0x27a8	SCACHE3_WIN5_MMAP
0x26b0	SCACHE2_WIN6_MMAP	0x27b0	SCACHE3_WIN6_MMAP
0x26b8	SCACHE2_WIN7_MMAP	0x27b8	SCACHE3_WIN7_MMAP
0x2a00	LCL_L1_WIN0_BASE	0x2b00	LCL_L2_WIN0_BASE
0x2a08	LCL_L1_WIN1_BASE	0x2b08	LCL_L2_WIN1_BASE
0x2a10	LCL_L1_WIN2_BASE	0x2b10	LCL_L2_WIN2_BASE
0x2a18	LCL_L1_WIN3_BASE	0x2b18	LCL_L2_WIN3_BASE
0x2a20	LCL_L1_WIN4_BASE	0x2b20	LCL_L2_WIN4_BASE
0x2a28	LCL_L1_WIN5_BASE	0x2b28	LCL_L2_WIN5_BASE
0x2a30	LCL_L1_WIN6_BASE	0x2b30	LCL_L2_WIN6_BASE
0x2a38	LCL_L1_WIN7_BASE	0x2b38	LCL_L2_WIN7_BASE
0x2a40	LCL_L1_WIN0_MASK	0x2b40	LCL_L2_WIN0_MASK
0x2a48	LCL_L1_WIN1_MASK	0x2b48	LCL_L2_WIN1_MASK
0x2a50	LCL_L1_WIN2_MASK	0x2b50	LCL_L2_WIN2_MASK
0x2a58	LCL_L1_WIN3_MASK	0x2b58	LCL_L2_WIN3_MASK
0x2a60	LCL_L1_WIN4_MASK	0x2b60	LCL_L2_WIN4_MASK
0x2a68	LCL_L1_WIN5_MASK	0x2b68	LCL_L2_WIN5_MASK
0x2a70	LCL_L1_WIN6_MASK	0x2b70	LCL_L2_WIN6_MASK
0x2a78	LCL_L1_WIN7_MASK	0x2b78	LCL_L2_WIN7_MASK
0x2a80	LCL_L1_WIN0_MMAP	0x2b80	LCL_L2_WIN0_MMAP
0x2a88	LCL_L1_WIN1_MMAP	0x2b88	LCL_L2_WIN1_MMAP
0x2a90	LCL_L1_WIN2_MMAP	0x2b90	LCL_L2_WIN2_MMAP
0x2a98	LCL_L1_WIN3_MMAP	0x2b98	LCL_L2_WIN3_MMAP
0x2aa0	LCL_L1_WIN4_MMAP	0x2ba0	LCL_L2_WIN4_MMAP
0x2aa8	LCL_L1_WIN5_MMAP	0x2ba8	LCL_L2_WIN5_MMAP
0x2ab0	LCL_L1_WIN6_MMAP	0x2bb0	LCL_L2_WIN6_MMAP

0x2ab8	LCL_L1_WIN7_MMAP	0x2bb8	LCL_L2_WIN7_MMAP
0x2c00	SE_WIN0_BASE	0x2d00	MISC_WIN0_BASE
0x2c08	SE_WIN1_BASE	0x2d08	MISC_WIN1_BASE
0x2c10	SE_WIN2_BASE	0x2d10	MISC_WIN2_BASE
0x2c18	SE_WIN3_BASE	0x2d18	MISC_WIN3_BASE
0x2c20	SE_WIN4_BASE	0x2d20	MISC_WIN4_BASE
0x2c28	SE_WIN5_BASE	0x2d28	MISC_WIN5_BASE
0x2c30	SE_WIN6_BASE	0x2d30	MISC_WIN6_BASE
0x2c38	SE_WIN7_BASE	0x2d38	MISC_WIN7_BASE
0x2c40	SE_WIN0_MASK	0x2d40	MISC_WIN0_MASK
0x2c48	SE_WIN1_MASK	0x2d48	MISC_WIN1_MASK
0x2c50	SE_WIN2_MASK	0x2d50	MISC_WIN2_MASK
0x2c58	SE_WIN3_MASK	0x2d58	MISC_WIN3_MASK
0x2c60	SE_WIN4_MASK	0x2d60	MISC_WIN4_MASK
0x2c68	SE_WIN5_MASK	0x2d68	MISC_WIN5_MASK
0x2c70	SE_WIN6_MASK	0x2d70	MISC_WIN6_MASK
0x2c78	SE_WIN7_MASK	0x2d78	MISC_WIN7_MASK
0x2c80	SE_WIN0_MMAP	0x2d80	MISC_WIN0_MMAP
0x2c88	SE_WIN1_MMAP	0x2d88	MISC_WIN1_MMAP
0x2c90	SE_WIN2_MMAP	0x2d90	MISC_WIN2_MMAP
0x2c98	SE_WIN3_MMAP	0x2d98	MISC_WIN3_MMAP
0x2ca0	SE_WIN4_MMAP	0x2da0	MISC_WIN4_MMAP
0x2ca8	SE_WIN5_MMAP	0x2da8	MISC_WIN5_MMAP
0x2cb0	SE_WIN6_MMAP	0x2db0	MISC_WIN6_MMAP
0x2cb8	SE_WIN7_MMAP	0x2db8	MISC_WIN7_MMAP
0x2e00	PCIE_G0_WIN0_BASE	0x2f00	PCIE_G1_WIN0_BASE
0x2e08	PCIE_G0_WIN1_BASE	0x2f08	PCIE_G1_WIN1_BASE
0x2e10	PCIE_G0_WIN2_BASE	0x2f10	PCIE_G1_WIN2_BASE
0x2e18	PCIE_G0_WIN3_BASE	0x2f18	PCIE_G1_WIN3_BASE
0x2e20	PCIE_G0_WIN4_BASE	0x2f20	PCIE_G1_WIN4_BASE
0x2e28	PCIE_G0_WIN5_BASE	0x2f28	PCIE_G1_WIN5_BASE
0x2e30	PCIE_G0_WIN6_BASE	0x2f30	PCIE_G1_WIN6_BASE
0x2e38	PCIE_G0_WIN7_BASE	0x2f38	PCIE_G1_WIN7_BASE
0x2e40	PCIE_G0_WIN0_MASK	0x2f40	PCIE_G1_WIN0_MASK
0x2e48	PCIE_G0_WIN1_MASK	0x2f48	PCIE_G1_WIN1_MASK
0x2e50	PCIE_G0_WIN2_MASK	0x2f50	PCIE_G1_WIN2_MASK
0x2e58	PCIE_G0_WIN3_MASK	0x2f58	PCIE_G1_WIN3_MASK
0x2e60	PCIE_G0_WIN4_MASK	0x2f60	PCIE_G1_WIN4_MASK
0x2e68	PCIE_G0_WIN5_MASK	0x2f68	PCIE_G1_WIN5_MASK
0x2e70	PCIE_G0_WIN6_MASK	0x2f70	PCIE_G1_WIN6_MASK

0x2e78	PCIE_G0_WIN7_MASK	0x2f78	PCIE_G1_WIN7_MASK
0x2e80	PCIE_G0_WIN0_MMAP	0x2f80	PCIE_G1_WIN0_MMAP
0x2e88	PCIE_G0_WIN1_MMAP	0x2f88	PCIE_G1_WIN1_MMAP
0x2e90	PCIE_G0_WIN2_MMAP	0x2f90	PCIE_G1_WIN2_MMAP
0x2e98	PCIE_G0_WIN3_MMAP	0x2f98	PCIE_G1_WIN3_MMAP
0x2ea0	PCIE_G0_WIN4_MMAP	0x2fa0	PCIE_G1_WIN4_MMAP
0x2ea8	PCIE_G0_WIN5_MMAP	0x2fa8	PCIE_G1_WIN5_MMAP
0x2eb0	PCIE_G0_WIN6_MMAP	0x2fb0	PCIE_G1_WIN6_MMAP
0x2eb8	PCIE_G0_WIN7_MMAP	0x2fb8	PCIE_G1_WIN7_MMAP

需要注意的是，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[63:20],20'h0}

根据缺省的路由，芯片启动后，能够自己去对应的 SPI Flash 取指，并能正确访问芯片上的 IO 设备地址。当需要访问其它地址空间时，软件需要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，将返回随机数据，以防止 CPU 死等。

4 芯片配置寄存器

龙芯 3C6000 中的芯片配置寄存器提供了对芯片的各种功能进行读写配置的机制。下面详述各个配置寄存器。

本章各个结点芯片配置寄存器的基地址为 0x1FE00000，也可以使用配置寄存器指令（IOCSR）进行访问。当使用地址进行访问时，各个内部结点的基地址还需要在地址的[19:16]上加上内部结点号。如下表所示：

表 4-1 不同内部结点的基地址

内部结点号	基地址
0	0x1FE0_0000
1	0x1FE1_0000
2	0x1FE2_0000
3	0x1FE3_0000
4	0x1FE4_0000
5	0x1FE5_0000
6	0x1FE6_0000
7	0x1FE7_0000

本文中 IOCSR[A][B]表示偏移地址为 A 的 IOCSR 寄存器中的位 B，其中 B 可以为一个范围。

4.1 版本寄存器（0x0000）

基地址为 0x1FE00000，偏移地址 0x0000。

表 4-2 版本寄存器

位域	字段名	访问	复位值	描述
7:0	Version	R	8'h15	配置寄存器版本号

4.2 芯片特性寄存器（0x0008）

该寄存器标识了一些软件相关的处理器特性，供软件在使能特定功能前查看。寄存器的偏移地址 0x0008。

表 4-3 芯片特性寄存器

位域	字段名	访问	复位值	描述
0	Centigrade	R	1'b1	为 1 时，表示 IOCSR[0x428]有效
1	Node counter	R	1'b1	为 1 时，表示 IOCSR[0x408]有效
2	MSI	R	1'b1	为 1 时，表示 MSI 可用
3	EXT_IOI	R	1'b1	为 1 时，表示 EXT_IOI 可用

4	IPI_percore	R	1'b1	为 1 时，表示通过 IOCSR 私有地址进行 IPI 发送
5	Freq_percore	R	1'b1	为 1 时，表示通过 IOCSR 私有地址调整频率
6	Freq_scale	R	1'b1	为 1 时，表示动态分频功能可用
7	DVFS_v1	R	1'b1	为 1 时，表示动态调频 v1 可用
8	Tsensor	R	1'b1	为 1 时，表示温度传感器可用
9	中断译码	R	1'b1	为 1 时，中断引脚译码模式可用
10	扁平模式	R	1'b1	为 1 时，表示支持扁平模式
11	Guest Mode	WR	1'b0	KVM 虚拟机模式
12	Freq_scale_16	R	1'b0	为 1 时，表示支持 16 分频模式
13	-	R	1'b1	保留
14	SE enabled	WR	1'b0	为 1 时，表示 SE 功能已使能
15	DMSI	R	1'b0	为 1 时，表示直接中断机制可用
16	RMSI	R	1'b0	为 1 时，表示重映射中断机制可用

4.3 厂商名称（0x0010）

该寄存器用于标识厂商名称。偏移地址 0x0010。

表 4-4 厂商名称寄存器

位域	字段名	访问	复位值	描述
63:0	Vendor	R	0x6e6f7367_6e6f6f4c	字符串“Loongson”

4.4 芯片名称（0x0020）

该寄存器用于标识芯片名称。偏移地址 0x0020。

表 4-5 芯片名称寄存器

位域	字段名	访问	复位值	描述
63:0	ID	R	0x00003030_30364333	字符串“3C6000”

4.5 功能设置寄存器（0x0180）

偏移地址 0x0180。每个 MC 位于不同内部结点的寄存器上。例如 MC0 对应的寄存器为内部结点 0 的 0x0180 的 MC0 位域（bit[8:4]），访问地址为 0x1FE0_0180；而 MC1 对应的寄存器为内部结点 1 的 0x0180 的 MC0 位域（bit[8:4]），访问地址为 0x1FE1_0180。其它的 MC 访问地址以此类推。PCIE 控制的分频控制分为两组，都在内部结点 0 上。

表 4-6 功能设置寄存器

位域	字段名	访问	复位值	描述
0		RW	1'b0	

1		RW	1'b0	
3:2		RW	2'b0	保留
4	MC0_disable_confspace	RW	1'b0	是否禁用 MC0 DDR 配置空间
5	MC0_defult_confspace	RW	1'b1	将所有内存访问路由至配置空间
6	MCA0 clock en	RW	1'b1	MCA0 时钟使能
7	MC0_resetn	RW	1'b1	MC0 软件复位（低有效）
8	MC0_clken	RW	1'b1	是否使能 MC0
23:9				
26:24	PCIEG0_freq_scale_ctrl	RW	3'b011	PCIE 控制器 0/2/4/6 的分频
27		RW	1'b1	
30:28	PCIEG1_freq_scale_ctrl	RW	3'b011	PCIE 控制器 1/3/5/7 的分频
39:31				
42:40	Node_freq_ctrl	RW	3'b111	结点分频
43	-	RW	1'b1	
55:44				
63:56	Cpu_version	R	8'h41	CPU ID

4.6 引脚驱动设置寄存器（0x0188）

偏移地址 0x0188。

表 4-7 引脚驱动设置寄存器

位域	字段名	访问	复位值	描述
15:0				（空）
19:16	AVS	RW	4'b0	AVS 控制信号驱动设置 以下为常温下的典型值 0x0：17mA 0x1：23mA 0x2：29mA 0x3：35mA 其它：保留 （下同）
23:20	I2C	RW	4'b0	I2C 信号驱动设置
27:24	UART	RW	4'b0	UART 信号驱动设置
31:28	SPI	RW	4'b0	SPI 信号驱动设置
35:32	GPIO	RW	4'b0	GPIO 信号驱动设置
39:36	SE UART	RW	4'b0	SE UART 信号驱动设置
43:40	SE SPI	RW	4'b0	SE SPI 信号驱动设置
47:44	SE I2C	RW	4'b0	SE I2C 信号驱动设置
51:48		RW	4'b0	
55:52		RW	4'b0	
59:56	SE GPIO	RW	4'b0	SE GPIO 信号驱动设置

4.7 功能采样寄存器（0x0190）

偏移地址 0x0190。

表 4-8 功能采样寄存器

位域	字段名	访问	复位值	描述
31:0		R		保留
38:32	CHIP_CONFIG	R		主板配置控制
47:38		R		
63:48	Bad_ip_core	R		core15-core0 是否坏

4.8 温度采样寄存器（0x0198）

偏移地址 0x0198。

表 4-9 温度采样寄存器

位域	字段名	访问	复位值	描述
7:0		R		保留
11:8	Bad_ip_mc	R		MC3-0 是否坏
19:12		R		保留
20	dotestn	R		Dotestn 引脚状态
22:21	iccc_en	R		iccc_en[1:0]引脚状态
23		R		保留
24	Thsens0_overflow	R		温度传感器 0 上溢
25	Thsens1_overflow	R		温度传感器 1 上溢
31:26				
47:32	Thsens0_out	R		温度传感器 0 摄氏温度 结点温度=Thens0_out *820/0x4000 - 311 温度范围 -40 度 - 125 度
63:48	Thsens1_out	R		温度传感器 1 摄氏温度 结点温度=Thens1_out *820/0x4000 - 311 温度范围 -40 度 - 125 度

4.9 PCIE 配置寄存器（0x01A0）

偏移地址 0x01A0。

表 4-10 PCIE 配置寄存器

位域	字段名	访问	复位值	描述
3:0	PCIE_G0_enable	RW		PCIE0/2/4/6 使能控制
7:4	PCIE_G1_enable	RW		PCIE1/3/5/7 使能控制
11:8	LCL_resetn	RW		LCL0-3 复位控制

12	LCL1_mode	RW		3C6000/D 单路模式的 LCL1 复用控制
13	LCL2_mode	RW		3C6000/D 单路模式的 LCL2 复用控制
14	PCIE0_chipset_mode	RW		PCIE0 控制器桥片连接使能
15	PCIE_stop_linkdown	RW		PCIE 控制器失连保护使能
20:16	V0_dev_num	RW		内部 PCIE 虚拟桥 0 设备号
25:21	V1_dev_num	RW		内部 PCIE 虚拟桥 1 设备号
30:26	iommu0_dev_num	RW		iommu0 设备号
33:32	PCIE_PHY0_mode	RW		PCIE PHY0 模式配置 00：1x16 01：2x8 10：4x4 11：1x8 + 2x4
35:34	PCIE_PHY1_mode	RW		PCIE PHY1 模式配置 00：1x16 01：2x8 10：4x4 11：1x8 + 2x4
37:36	PCIE_PHY2_mode	RW		PCIE_PHY2 模式配置 00：1x16 01：2x8
39:38	PCIE_PHY3_mode	RW		PCIE_PHY3 模式配置 00：1x16 01：2x8
47:40	PCIE_multidev	RW		使能对应控制器的多设备支持
51:48	PCIE_G0_shut	RW		PCIE0/2/4/6 关闭
55:52	PCIE_G1_shut	RW		PCIE1/3/5/7 关闭

4.10 频率配置寄存器（0x01A8 – 0x01C0）

以下几组软件倍频设置寄存器用于设置芯片主时钟、内存控制器时钟和 PCIE 控制器时钟的工作频率。

其中，Node Clock 对应处理器核、片上网络及高速共享缓存的时钟频率。

Mem Clock 的配置支持多种模式。在 2 倍频模式（mem div 为 1）下，Mem Clock 应为内存控制器时钟的 2 倍；在单倍频模式下（mem div 为 0）下，Mem Clock 应为内存控制器时钟频率。

内存总线工作频率为内存控制器时钟的2倍，总线工作速率为内存控制器时钟的4倍。

内存控制器时钟的设置分为两组，分别控制 MC0/1 和 MC2/3。

PCIE 控制器时钟为控制器内与协议处理无关部分的时钟，同样分为两组分别设置，分别控制 PCIE Group0（PCIE0/2/4/6）和 PCIE Group1（PCIE1/3/5/7）。

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

（参考时钟/DIV_REFC * DIV_LOOPC）/ DIV_OUT。

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（100MHz），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。

下面的寄存器为 PCIE Clock 的配置寄存器。偏移地址为 0x1A8：

表 4-11 PCIE 时钟软件倍频设置寄存器

位域	字段名	访问	复位值	描述
[0]	SEL_PCIE_PLL	RW	0x0	时钟输出选择 1：PCIE 时钟选择 PLL 输出 0：PCIE 时钟选择 SYS CLOCK
[1]	SOFT_SET_PCIE_PLL	RW	0x0	允许软件设置 PCIE PLL
[2]	BYPASS_PCIE_PLL	RW	0x0	Bypass PCIE_PLL
[3]		RW	0x1	
[5:4]		RW		
[6]	LOCKED_PCIE_PLL0	R	0x0	PCIE_PLL0 是否锁定
[7]	PD_PCIE_PLL	RW	0x0	关闭 PCIE PLL
[13:8]	PCIE_PLL_DIV_REFC	RW	0x1	PCIE 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]	PCIE_PLL0_DIV_LOOPC	RW	0x41	PCIE PLL0 输入参数
[29:24]	PCIE_PLL0_DIV_OUT	RW	0x0	PCIE PLL0 输入参数
[30]	NODE_CLOCK_SEL	RW	0x0	0：使用 PCIE_PLL 作为 PCIE 时钟 1：使用 NODE_CLOCK 作为分频输入
[31]	USE_SSC	RW	0x0	当 PRG 开启 SSC 时，选用 SSC 时钟
[34:32]	VDDA_LDO_CTRL	RW		
[35]	VDDA_LDO_BYPASS	RW		
[38:36]	VDDD_LDO_CTRL	RW		
[39]	VDDD_LDO_BYPASS	RW		
[40]	VDDA_LDO_EN	RW		
[41]	VDDD_LDO_EN	RW		
[51:42]	PCIE_PLL1_DIV_LOOPC	RW	0x41	PCIE PLL1 输入参数
[57:52]	PCIE_PLL1_DIV_OUT	RW	0x0	PCIE PLL1 输入参数
[58]	LOCKED_PCIE_PLL1	R	0x0	PCIE_PLL1 是否锁定
其它		RW		保留

下面的寄存器为Main Clock的配置寄存器,Main Clock用于产生node clock、core clock

等的最高工作频率。偏移地址为 0x1B0：

表 4-12 结点时钟软件倍频设置寄存器

位域	字段名	访问	复位值	描述
0	SEL_PLL_NODE	RW	0x0	时钟输出选择 1：Node 时钟选择 PLL 输出 0：Node 时钟选择 SYS CLOCK
1		RW	0x0	保留
2	SOFT_SET_PLL	RW	0x0	允许软件设置 PLL
3	BYPASS_L1	RW	0x0	Bypass L1 PLL
4	BYPASS_L2	RW		Bypass L2 PLL
7:5	-	RW	0x0	保留
8	VDDA_LDO_EN	RW	0x0	使能 VDDA LDO
9	VDDD_LDO_EN	RW	0x0	使能 VDDD LDO
10	L2_DSMCLK_SEL	RW		需要设置为 1
11	L2_bypass_reg	RW		需要设置为 0
12	L2_RSTN	RW	0x0	L2 时钟复位
13	L2_CKOUT_EN	RW	0x0	L2 时钟输出使能
14	L2_CP_SEL	RW	0x0	需要设置为 0
15	L2_FRAC_EN	RW		L2 小数分频使能
16	LOCKED_L1	R	0x0	L1 PLL 是否锁定
17	LOCKED_L2	R	0x0	L2 PLL 是否锁定
18				
19	PD_L1	RW	0x0	关闭 L1 PLL
20	PD_L2	RW	0x0	关闭 L2 PLL
21	L2_VCO_START	RW	0x0	需要设置为 0
22	L2_SEL	RW	0x0	选择 L2 时钟输出
23	USE_SSC	RW	0x0	当 PRG 开启 SSC 时，选用 SSC 时钟
25:24				
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
51:48		RW		
53:52				
63:54		RW		
119:64		RW		
122:120	VDDA_LDO_CTRL	RW		
123	VDDA_LDO_BYPASS	RW		
126:124	VDDD_LDO_CTRL	RW		
127	VDDD_LDO_BYPASS	RW		
其它	-	RW		保留

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

PLL 的 clk_ref/div_refc 的结果应该为 25/50/100MHz，推荐使用 100MHz。VCO 频率（上述式中括号内部分）必须在范围 4.4GHz - 6.4GHz 之内。该要求对其它 PLL 同样适用。

此外，对于 div_loopc 的推荐设置为小于 255。对于 div_out 的推荐设置为 1/2/4/6 及 6 以上，3/5 不推荐使用。

下面的寄存器为 Mem Clock 的配置寄存器，Mem Clock 时钟频率应该配置为最终 DDR 总线时钟频率。偏移地址为 0x1C0：

表 4-13 内存时钟软件倍频设置寄存器

位域	字段名	访问	复位值	描述
[0]	SEL_MEM_PLL	RW	0x0	时钟输出选择 1：MEM 时钟选择 PLL 输出 0：MEM 时钟选择 SYS CLOCK
[1]	SOFT_SET_MEM_PLL	RW	0x0	允许软件设置 MEM PLL
[2]	BYPASS_MEM_PLL	RW	0x0	Bypass MEM_PLL
[3]	MEMDIV_RESETn	RW	0x1	复位内部分频器
[5:4]	MEMDIV_MODE	RW		00：单倍频模式 01：2 倍频模式 其它：保留
[6]	LOCKED_MEM_PLL0	R	0x0	MEM_PLL0 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_PLL0_DIV_LOOPC	RW	0x41	MEM PLL0 输入参数
[29:24]	MEM_PLL0_DIV_OUT	RW	0x0	MEM PLL0 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
[31]	USE_SSC	RW	0x0	当 PRG 开启 SSC 时，选用 SSC 时钟
[34:32]	VDDA_LDO_CTRL	RW		
[35]	VDDA_LDO_BYPASS	RW		
[38:36]	VDDD_LDO_CTRL	RW		
[39]	VDDD_LDO_BYPASS	RW		
[40]	VDDA_LDO_EN	RW		
[41]	VDDD_LDO_EN	RW		
[51:42]	MEM_PLL1_DIV_LOOPC	RW	0x41	MEM PLL1 输入参数
[57:52]	MEM_PLL1_DIV_OUT	RW	0x0	MEM PLL1 输入参数
[58]	LOCKED_MEM_PLL1	R	0x0	MEM_PLL1 是否锁定
其它		RW		保留

4.11 处理器核分频设置寄存器（0x01D0）

以下寄存器用于处理器核动态分频使用，使用该寄存器对处理器核进行调频设置，可以在 100ns 内完成变频操作，没有其它额外开销。偏移地址 0x01D0。

需要注意的是，该寄存器与逻辑核相对应，只有偶数号逻辑核的配置才与物理核实际对应，而奇数号核的配置不生效。

由于 3C6000 采用 16 核 32 线程的设计，总共对应 8 个内部结点的空间。

表 4-14 处理器核软件分频设置寄存器

位域	字段名	访问	复位值	描述
2:0	core0_freqctrl	RW	0x7	逻辑核 0（物理核 0）分频控制值
3	core0_en	RW	0x1	逻辑核 0（物理核 0）时钟使能
6:4	core1_freqctrl	RW	0x7	
7	core1_en	RW	0x1	
10:8	core2_freqctrl	RW	0x7	逻辑核 2（物理核 1）分频控制值
11	core2_en	RW	0x1	逻辑核 2（物理核 1）时钟使能
14:12	core3_freqctrl	RW	0x7	
15	core3_en	RW	0x1	
			注:	软件分频后的时钟频率值等于原来的（分频控制值+1）/8

4.12 处理器核复位控制寄存器（0x01D8）

以下寄存器用于处理器核软件控制复位使用。需要复位时，先将对应核的 resetn 置 0，再将 resetn_pre 置 0，等待 500 微秒后，将 resetn_pre 置 1，再将 resetn 置 1 即可完成整个复位过程。偏移地址 0x01D8。

需要注意的是，该寄存器与逻辑核相对应，只有偶数号逻辑核的配置才与物理核实际对应，而奇数号核的配置不生效。

每个 3C6000 采用 16 核 32 线程的设计，总共对应 8 个内部结点的空间。

表 4-15 处理器核软件分频设置寄存器

位域	字段名	访问	复位值	描述
0	Core0_resetn_pre	RW	0x1	逻辑核 0（物理核 0）复位辅助控制
1	Core0_resetn	RW	0x1	逻辑核 0（物理核 0）复位
2	Core1_resetn_pre	RW	0x1	-
3	Core1_resetn	RW	0x1	-
4	Core2_resetn_pre	RW	0x1	逻辑核 2（物理核 1）复位辅助控制
5	Core2_resetn	RW	0x1	逻辑核 2（物理核 1）复位
6	Core3_resetn_pre	RW	0x1	-
7	Core3_resetn	RW	0x1	-

4.13 路由设置寄存器（0x0400）

以下寄存器用于控制芯片内的部分路由设置。偏移地址 0x0400。

表 4-16 芯片路由设置寄存器

位域	字段名	访问	复位值	描述
3:0	scid_sel	RW	0x0	共享缓存散列位控制
7:4		RW	0xF	
8		RW	0x0	
9		RW	0x0	
10	Fast_path_LCL1_en	RW	0x0	使能 LCL1 快速路径（双路）
11	Fast_path_LCL2_en	RW	0x0	使能 LCL2 间快速路径（双路）
12	mcc_en	RW	0x0	MCC 模式使能
13		RW	0x0	
14		RW		
15	PCI_40bit	RW	0x1	PCI 空间转换仅保留低 40 位
19:16		RW	0x0	
20	PCIE_throt_en	RW	0x0	PCIE 访问空闲压缩使能
21	MISC_throt_en	RW	0x0	MISC 访问空闲压缩使能
22	SE_throt_en	RW	0x0	SE 访问空闲压缩使能
23	MC_throt_en	RW	0x0	MC 访问空闲压缩使能
31:24		RW	0x3	
37:32	interleave_bit	RW	0x0	内存散列位控制
39:38	interleave_en	RW	0x0	内存散列使能
43:40		R		
47:44		RW	0x0	
51:48	Se_control	RW	0x0	
55:52	Se_dma_coherent	RW	0x0	
60:56		RW	0x0	
61	enable_gather_spi	RW	0x1	SPI 访问空闲压缩使能

4.14 扩展路由设置寄存器（0x0410）

以下寄存器用于控制芯片内的部分路由设置。偏移地址 0x0410。

表 4-17 芯片路由设置寄存器

位域	字段名	访问	复位值	描述
3:0		RW	0x0	
7:4	Chip_mask	RW	0x7	芯片掩码，避免猜测到未使用结点的地址时无响应
9:8	interleave0_map	RW	0x0	Interleave 选择的位值为 00 时和 MC 的映射关系 00：映射至 MC0 01：映射至 MC1

				10：映射至 MC2 11：映射至 MC3
11:10	interleave1_map	RW	0x0	Interleave 选择的位值为 01 时和 MC 的映射关系映射规则同上
13:12	interleave2_map	RW	0x0	Interleave 选择的位值为 10 时和 MC 的映射关系映射规则同上
15:14	interleave3_map	RW	0x0	Interleave 选择的位值为 11 时和 MC 的映射关系映射规则同上
23:16	PCIE_throttle	RW	0xfe	
31:24	misc_throttle	RW	0xfe	
39:32	se_throttle	RW	0xfe	
47:40	mc_throttle	RW	0xfe	

4.15 其它功能设置寄存器（0x0420）

以下寄存器用于控制芯片内部分功能使能。偏移地址 0x0420。

表 4-18 其它功能设置寄存器

位域	字段名	访问	复位值	描述
0	disable_jtag	RW	0x0	完全禁用 JTAG 接口
1	disable_jtag_Core	RW	0x0	完全禁用主核 JTAG 调试接口
2	disable_LA132	RW	0x0	完全禁用 LA132
3	disable_jtag_LA132	RW	0x0	完全禁用 LA132 JTAG 调试接口
4	disable_antifuse0	RW	0x0	禁用 fuse
5	disable_antifuse1	RW	0x0	禁用 fuse
6	disable_ID	RW	0x0	禁用 ID 修改
7				保留
8	resetn_LA132	RW	0x0	LA132 复位控制
9	sleeping_LA132	R	0x0	LA132 进入睡眠状态
10	soft_int_LA132	RW	0x0	LA132 核间中断寄存器
15:12	core_int_en_LA132	RW	0x0	LA132 对应每个核的 IO 中断使能
18:16	freqscale_LA132	RW	0x0	LA132 分频控制
19	clken_LA132	RW	0x0	LA132 时钟使能
20		RW	0x0	
21	stable_resetn	RW	0x0	稳定时钟复位控制
22	freqscale_percore	RW	0x0	使能每个核私有的调频寄存器
23	clken_percore	RW	0x0	使能每个核私有的时钟使能
27:24	confbus_timeout	RW	0x8	配置总线超时时间设置，实际时间为 2 的幂次方
28	PCIE_softresetn[0]	RW	0x1	PCIE0/2/4/6 控制器软件复位控制
29	PCIE_softresetn[1]	RW	0x1	PCIE1/3/5/7 控制器软件复位控制
31:30				

35:32	freqscale_mode_core	RW	0x0	每个核的调频模式选择 0：(n+1)/8 1：1/(n+1)
36	freqscale_mode_node	RW	0x0	结点的调频模式选择 0：(n+1)/8 1：1/(n+1)
37	freqscale_mode_LA132	RW	0x0	LA132 的调频模式选择 0：(n+1)/8 1：1/(n+1)
39:38	freqscale_mode_PCIE	RW	0x0	每个 PCIE 的调频模式选择 0：(n+1)/8 1：1/(n+1)
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0：(n+1)/8 1：1/(n+1)
43:41				保留
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能
48	EXT_INT_en	RW	0x0	扩展 IO 中断使能
49	INT_encode	RW	0x0	使能中断引脚编码模式
50	DS_en	RW	0x0	使能电压骤降检测
51	Int_Remap_en	RW	0x0	使能中断重映射机制
53:52		RW	0x0	
54		RW	0x0	
55	Cf_jtag_core0	RW	0x0	仅使能 Core0 JTag 调试
57:56	thsensor_sel	RW	0x0	温度传感器选择
62:60	Auto_scale	R	0x0	自动调频当前值
63	Auto_scale_doing	R	0x0	自动调频正在生效标志

4.16 摄氏温度寄存器（0x0428）

以下寄存器用于观测芯片内部温度传感器数值。偏移地址 0x0428。只有当 IOCSR[0x0008][0]有效时，该寄存器可用。

表 4-19 温度观测寄存器

位域	字段名	访问	复位值	描述
7:0	Centigrade temperature	RO	0x0	摄氏温度
63:8		RW	0x0	

4.17 SRAM 调节寄存器（0x0430）

以下寄存器用于调节处理器核内部 Sram 的工作频率。偏移地址 0x0430。

表 4-20 处理器核 SRAM 调节寄存器

位域	字段名	访问	复位值	描述
31:0	sram_ctrl	RW	0x0	核内 Sram 配置寄存器
63:32		RW	0x0	

4.18 PRG 寄存器（0x0440）

PRG 寄存器用于控制芯片的参考时钟。偏移地址 0x0440。

表 4-21 PRG 寄存器

位域	字段名	访问	复位值	描述
0	PLL_bypass	RW	0x1	
1	REG_bypass	RW	0x1	
2	CLKout_en	RW	0x1	
3	CP_sel	RW	0x0	
4	drive_en	RW	0x1	
5	dsmclk_sel	RW	0x0	
6	frac_en	RW	0x0	
7	ssc_en	RW	0x0	
8	ssc_sprd	RW	0x0	1：中心展频 0：向下展频
9	PLL_pu	RW	0x0	
10	PLL_rstn	RW	0x1	
11	ssc_clk_sel	RW	0x0	
12	PLL_vco_start	RW	0x0	
15:13	ldo_acode	RW	0x3	
19:16	fvco_tune_abs	RW	0x3	
23:20	icp_sel	RW	0x7	
27:24	ssc_ma	RW	0x7	
31:28	PLL_CKin_divn	RW	0x1	
51:32	PLL_div_N	RW	0x0	
75:52	ssc_step	RW	0x106	
99:76	ssc_offset	RW	0x33333	
111:100	ssc_stpsum	RW	0x640	
121:112	PLL_div_M	RW	0x32	
124:122	PLL_CKout_divn	RW	0x1	

4.19 FUSE0 观测寄存器（0x0460）

以下寄存器用于观测部分软件可见的 Fuse0 数值。偏移地址 0x0460。

表 4-22 FUSE 观测寄存器

位域	字段名	访问	复位值	描述
127:0	Fuse_0	RW	0x0	

4.20 FUSE1 观测寄存器（0x0470）

以下寄存器用于观测部分软件可见的 Fuse1 数值。偏移地址 0x0470。

表 4-23 FUSE 观测寄存器

位域	字段名	访问	复位值	描述
127:0	Fuse_1	RW	0x0	

5 芯片时钟分频及使能控制

龙芯 3C6000 的参考时钟可以使用 SYSCLK，也可以使用 SYSCLK_I0p/n 或 SYSCLK_I1p/n，通过 CHIP_CONFIG[6]引脚选择 SYSCLK，或者通过 CHIP_CONFIG[5]引脚选择 SYSCLK_I0p/n，以下统称 SYS_CLOCK。各个时钟的产生都依赖于 SYS_CLOCK，下面章节对这些时钟分别介绍。

龙芯 3C6000 中为处理器核、片上网络及共享缓存、PCIE 控制器及 LA132 核分别设置了分频机制。3C6000 同样支持 1/n 的分频值。

本章各个结点芯片配置寄存器的基地址为 0x1FE00000，也可以使用配置寄存器指令（IOCSR）进行访问。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]上加上内部结点号。

5.1 芯片模块时钟介绍

芯片单端参考时钟 SYSCLK 使用 100MHz 晶振输入。如果参考时钟选用 SYSCLK，需要将 CHIP_CONFIG[6]上拉。

芯片差分参考时钟 SYSCLK_I0p/n 和 SYSCLK_I1p/n 根据主板的设计进行选择。如果选用 SYSCLK_I0p/n，CHIP_CONFIG[6:5]设置为 2'b01；如果选用 SYSCLK_I1p/n，CHIP_CONFIG[6:5]设置为 2'b00。

3C6000 中推荐使用差分参考时钟作为系统参考时钟。

龙芯 3C6000 内部所使用的时钟及其控制方式如下表所示。

表 5-1 处理器内部时钟说明

时钟	时钟来源	倍频方式	分频控制	使能控制	时钟描述
Boot Clock	SYS_CLOCK	*1	不支持	不支持	SPI、UART、I2C、AVS 控制器时钟
Main Clock	SYS PLL	PLL 配置	不支持	不支持	SYS PLL 输出。 Node Clock、Core Clock、LA132 Clock 时钟源 Mem Clock、PCIE Clock、Stable Clock 可选时钟源
Node Clock	Main Clock	*1	支持	不支持	片上网络、共享缓存、结点时钟源
CoreX Clock	Main Clock	*1	支持	支持	CoreX 时钟
LA132 Clock	Main Clock	*1	支持	支持	LA132 时钟，软件需要保证分频后低于 1GHz
Stable Clock	SYS_CLOCK	*1	支持	支持	处理器核恒定计数器时钟
Mem0/1 Clock	MEM PLL0	PLL 配置	不支持	支持	内存控制器 0/1 时钟
	Main Clock	/2、/4、/8	不支持	支持	内存控制器 0/1 备选时钟
Mem2/3 Clock	MEM PLL1	PLL 配置	不支持	支持	内存控制器 2/3 时钟
	Main Clock	/2、/4、/8	不支持	支持	内存控制器 2/3 备选时钟

PCIE Group0 Clock	PCIE PLL0	PLL 配置	支持	不支持	PCIE 控制器 0/2/4/6 时钟
	Main Clock	/2、/4、/8	支持	不支持	PCIE 控制器 0/2/4/6 备选时钟
PCIE Group1 Clock	PCIE PLL1	PLL 配置	支持	不支持	PCIE 控制器 1/3/5/7 时钟
	Main Clock	/2、/4、/8	支持	不支持	PCIE 控制器 1/3/5/7 备选时钟

5.2 处理器核分频及使能控制

处理器核分频有多种模式，一为按地址访问模式，二是处理器配置指令访问模式，以下分别进行介绍。每个处理器核可以分别控制。

5.2.1 按地址访问

每个龙芯 3C6000 中，处理器核支持双线程，32 个逻辑核被分为 8 个内部结点，每个内部结点可以对 2 个物理核进行控制，其基地址规则与其它的配置寄存器一致。

使用该寄存器对处理器核进行调频设置，可以在 100ns 内完成变频操作，没有其它额外开销。偏移地址 0x01D0。

需要注意的是，由于 3C6000 的处理器核采用双线程的设计，偶数号逻辑核实际对应物理核的控制，而奇数号逻辑核对应的寄存器不产生实际效果。

表 5-2 处理器核软件分频设置寄存器

位域	字段名	访问	复位值	描述
2:0	core0_freqctrl	RW	0x7	逻辑核 0（物理核 0）分频控制值
3	core0_en	RW	0x1	逻辑核 0（物理核 0）时钟使能
6:4	core1_freqctrl	RW	0x7	
7	core1_en	RW	0x1	
10:8	core2_freqctrl	RW	0x7	逻辑核 2（物理核 1）分频控制值
11	core2_en	RW	0x1	逻辑核 2（物理核 1）时钟使能
14:12	core3_freqctrl	RW	0x7	
15	core3_en	RW	0x1	
			注:	软件分频后的时钟频率值等于原来的（分频控制值+1）/8

除了分频配置方式，3C6000 中还可以通过寄存器的设置，将分频之后的时钟频率由原来的“（分频控制值+1）/8”调整为“1/（分频控制值+1）”。这个寄存器位于“其它功能设置寄存器”。基地址为 0x1FE00000，偏移地址 0x0420。

表 5-3 其它功能设置寄存器

位域	字段名	访问	复位值	描述
35:32	freqscale_mode_core	RW	0x0	每个核的调频模式选择 0: (n+1)/8 1: 1/(n+1)

5.2.2 配置寄存器指令访问

除了传统的按地址访问模式，还可以使用配置寄存器指令对私有的分频配置寄存器进行访问。

需要注意的是，私有的分频配置寄存器控制与原有的处理器核软件分频设置寄存器控制是互斥的，两者只能选一种使用。选择的方法是通过“其它功能设置寄存器”上的对应位进行控制。该寄存器基地址为 0x1FE00000，偏移地址 0x0420。

表 5-4 其它功能设置寄存器

位域	字段名	访问	复位值	描述
22	freqscale_percore	RW	0x0	使能每个核私有的调频寄存器
23	clken_percore	RW	0x0	使能每个核私有的时钟使能

当 freqscale_percore 被设置为 1 时，使用私有的分频配置寄存器中的 freqscale 位对自己的时钟进行分频设置（包括了 freqscale_mode）；当 clken_percore 被设置为 1 时，使用私有的分频配置寄存器中的 clken 位对时钟使能进行控制。

该配置寄存器定义如下。偏移地址为 0x1050。

需要注意的是，由于 3C6000 的处理器核采用双线程的设计，偶数号逻辑核实际对应处理核的控制，而奇数号逻辑核对应的寄存器不产生实际效果。

表 5-5 处理器核私有分频寄存器

位域	字段名	访问	复位值	描述
4	freqscale_mode	RW	0x0	当前处理器核的分频模式选择 0: (n+1)/8 1: 1/(n+1)
3	clken	RW	0x0	当前处理器核的时钟使能
2:0	freqscale	RW	0x0	当前处理器核的分频设置

5.3 结点时钟分频及使能控制

结点时钟为片上网络与共享缓存所使用的时钟，有两种不同的控制模式，一为软件设置模式，二是硬件自动分频设置。

结点时钟不支持完全关断功能，所以没有对应的 clken 控制位。

5.3.1 软件设置

软件设置方法使用功能设置寄存器中的结点分频位，使用相同的地址进行设置。

该寄存器基地址为 0x1FE00000，偏移地址 0x0180。

表 5-6 功能设置寄存器

位域	字段名	访问	复位值	描述
42:40	Node_freq_ctrl	RW	3'b111	结点分频

与处理器核的分频控制一致，结点时钟也可以通过寄存器的设置，将分频之后的时钟频率由原来的“（分频控制值+1）/8”调整为“1/（分频控制值+1）”。这个寄存器位于“其它功能设置寄存器”。基地址为 0x1FE00000，偏移地址 0x0420。

表 5-7 其它功能设置寄存器

位域	字段名	访问	复位值	描述
36	freqscale_mode_node	RW	0x0	结点的调频模式选择 0: (n+1)/8 1: 1/(n+1)

5.3.2 硬件自动设置

除了由软件进行主动的设置之外，结点时钟还支持由温度传感器触发的自动分频设置。自动分频设置是由软件预先针对不同的温度进行设置，当温度传感器的温度达到对应的预设值时，就会触发相应的自动分频设置。

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：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 3C6000 内部集成 2 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。

FREQ：分频数。当触发分频操作时，这个分频数同样受到 freqscale_mode_node 的影响，当其为 0 时，将频率调整为当前时钟频率的(FREQ+1)/8 倍；为 1 时，将频率调整为当前时钟频率的 1/(FREQ+1)倍。

其基地址为 0x1FE00000。

表 5-8 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	0x1480	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
Thsens_freq_scale_up	0x1490	RW	温度传感器控制寄存器高位 [7:0] Scale_Hi_gate0 高 8 位 [15:8] Scale_Hi_gate1 高 8 位 [23:16] Scale_Hi_gate2 高 8 位 [31:24] Scale_Hi_gate3 高 8 位 [39:32] Scale_Lo_gate0 高 8 位 [47:40] Scale_Lo_gate1 高 8 位 [55:48] Scale_Lo_gate2 高 8 位 [63:56] Scale_Lo_gate3 高 8 位

5.4 PCIE 控制器分频及使能控制

PCIE 控制器的分频机制与其它的类似。PCIE 控制器分为奇偶两组，两组控制器的时钟可以分别控制。使用功能设置寄存器中的对应位进行设置。其基地址为 0x1FE00000，偏移地址 0x0180。

表 5-9 功能设置寄存器

位域	字段名	访问	复位值	描述
26:24	PCIE_G0_freq_scale_ctrl	RW	3'b111	PCIE 控制器 0/2/4/6 的分频
27		RW	1'b1	
30:28	PCIE_G1_freq_scale_ctrl	RW	3'b111	PCIE 控制器 1/3/5/7 的分频
31		RW	1'b1	

与其它分频控制一致，PCIE 控制器时钟也可以通过寄存器的设置，将分频之后的时钟频率由原来的“（分频控制值+1）/8”调整为“1/（分频控制值+1）”。这个寄存器位于“其它功能设置寄存器”。基地址为 0x1FE00000，偏移地址 0x0420。

表 5-10 其它功能设置寄存器

位域	字段名	访问	复位值	描述
38	freqscale_mode_PCIE_G0	RW	0x0	PCIE0/2/4/6 控制器的调频模式选择 0：(n+1)/8 1：1/(n+1)
39	freqscale_mode_PCIE_G1	RW	0x0	PCIE1/3/5/7 控制器的调频模式选择 0：(n+1)/8 1：1/(n+1)

5.5 Stable Counter 分频及使能控制

Stable Counter 的分频机制与其它的类似。使用其它功能设置寄存器中的对应位进行设置。其基地址为 0x1FE00000，偏移地址 0x0420。

表 5-11 其它功能设置寄存器

位域	字段名	访问	复位值	描述
21	stable_reset	RW	0x0	稳定时钟复位控制 1：置为复位状态 0：解除软件复位
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0：(n+1)/8 1：1/(n+1)
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能

需要注意的是，stable_reset 设置为 0 之后，只是解除了软件复位。此时，如果 GPIO_FUNC_en[13]为 1 时，stable counter 的复位还受到 GPIO[13]的控制（低有效）。

GPIO 输出使能寄存器基地址为 0x1FE00000，偏移地址 0x0500。

表 5-12 GPIO 输出使能寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_OEn	RW	32'hffffffff	GPIO 输出使能（低有效）
63:32	GPIO_FUNC_En	RW	32'hffff0000	GPIO 功能使能（低有效）

6 软件时钟系统

龙芯 3C6000 处理器中为系统软件使用的时钟定义了多个不同层次的使用方法。处理器核内部有 stable counter 寄存器，以及芯片级的 node counter 寄存器。

以下对 stable counter 和 node counter 进行介绍。

6.1 Stable Counter

龙芯 3C6000 中恒定时钟源，称之为 stable counter。Stable counter 的时钟有别于处理器核自身的时钟，也有别于结点时钟，是一支独立的主时钟。

在 3C6000 中，处理器核时钟与结点时钟都来源于主时钟，但都可以自由控制分频数（参见前一章的介绍），而 stable counter 的时钟源于输入参考时钟，也可以进行独立分频，不随其它时钟频率的变化而变化。

根据这个时钟源，实现了一个计时器和一个定时器。本章主要介绍 Stable couter 相关的寄存器。

6.1.1 Stable Counter 的时钟控制

Stable counter 使用参考时钟输入，并且可以通过软件分频机制进行分频控制。

以下是 Stable counter 的时钟控制寄存器。该寄存器位于控制芯片其他功能设置寄存器。基地址为 0x1FE00000，偏移地址 0x0420。

表 6-1 其它功能设置寄存器

位域	字段名	访问	复位值	描述
21	stable_reset	RW	0x0	稳定时钟复位控制 1：置为复位状态 0：解除软件复位
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0: (n+1)/8 1: 1/(n+1)
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能

当 BIOS 对 Stable counter 时钟源进行配置后，需要更新每个处理器核中的 MCSR 部分用于控制 CPUCFG.0x4 和 CPUCFG.0x5 的值。参照第 8.1 节描述，CPUCFG.0x4 中应该填写以 Hz 为单位的晶振时钟频率；CPUCFG.0x5[31:16]应填写分频系数；CPUCFG.0x5[15:0]应填写倍频因子。后两者的填写，需要 BIOS 帮助进行计算，从而使得 CCFreq*CFM/CFD 的结果等于 Stable Counter 的实际频率。

6.1.2 Stable Counter 的校准

单芯片情况下，每个核的 Counter 差距在 2 个周期之内，无需特别的校准。在多芯片情况下，不同的芯片之间会有较大的差异，需要由一套专门的软硬件校准机制，将各个核的 counter 差异保持在 100ns 以下。

首先，为了保证每个硅片的主时钟在使用过程中不会产生偏差，需要使用同一个时钟源驱动所有硅片的参考时钟。

其次，为了保证每个芯片的 Stable counter 在同一时刻开始计时，硬件上需要使用两个 GPIO 管脚的复用功能。结点 0 使用 GPIO12 来输出复位信号，其它所有结点（包括结点 0）使用 GPIO13 来输入复位信号（需要配置为 Stable counter 功能）。在主板上需要使用缓冲器件来保证复位时序（主要是信号斜率），复位时序越好，不同芯片间的时钟差异越小。

软件在使用 Stable counter 之前必须对全局的 Stable counter 通过 GPIO12 来进行复位，复位之前需要保证各个芯片的时钟选择一致，各个芯片的复位已经解除。这个工作通常由 BIOS 来完成。系统的连接方案如下图所示。

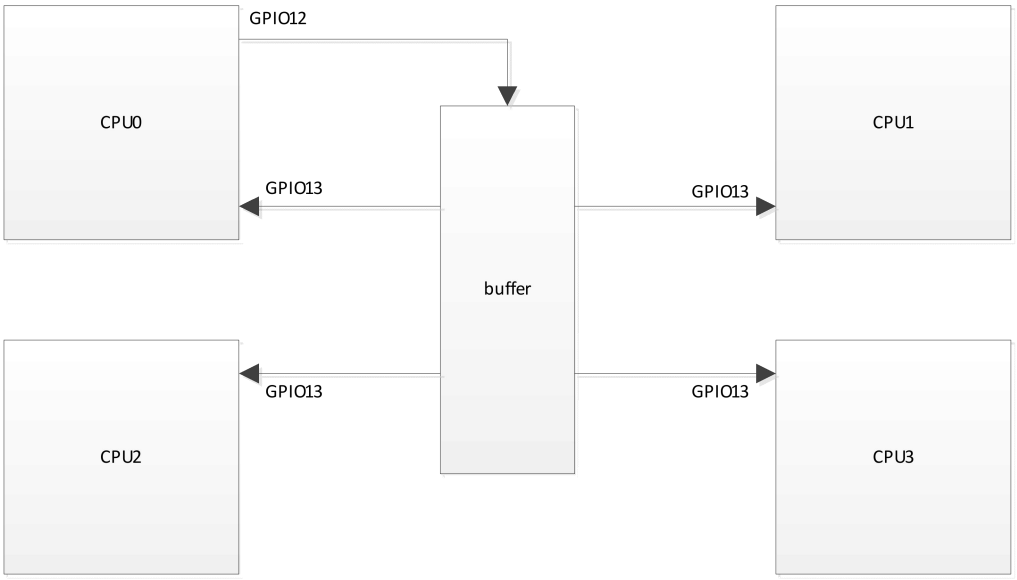


图 6-1 多片互连时的 Stable 复位控制

6.2 Node Counter

需要注意的是，Node counter 的计数频率与 Node clock 完全相同，如果希望使用 Node counter 作为时钟计算依据，就要避免对 Node clock 进行变频。

6.2.1 按地址访问

配置寄存器的基地址为 0x1FE00000，如下表所示。

表 6-2 Node counter 寄存器

名称	偏移地址	权限	描述
Node counter	0x0408	R	64 位结点时钟计数

6.3 时钟系统小结

Stable counter 在稳定性上比起 node counter 更有优势，不会随着其它时钟（node clock 和 core clock）分频的变化而变化。

在易用性上来说 Stable counter 也更方便访问，是软件参考时钟系统的首选方案。

Node clock 更多是考虑传统兼容性的一个设计，是一个时钟系统的备份方案。在以后的芯片设计中将逐步淘汰。

7 GPIO 控制

每个龙芯 3C6000 中提供最多 16 个 GPIO 供系统使用，绝大部分都与其它功能复用。通过寄存器设置，还可以将 GPIO 配置为中断输入功能，并可以设置其中断电平。

本章各个芯片配置寄存器的基地址为 0x1FE00000。

7.1 输出使能寄存器（0x0500）

基地址为 0x1FE00000，偏移地址 0x0500。

表 7-1 输出使能寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_OEn	RW	32'hffffffff	GPIO 输出使能（低有效）
63:32	GPIO_FUNC_En	RW	32'hffff0000	GPIO 功能使能（低有效）

7.2 输入输出寄存器（0x0508）

基地址为 0x1FE00000，偏移地址 0x0508。

表 7-2 输入输出寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_O	RW	32'h0	GPIO 输出设置
63:32	GPIO_I	RO	32'h0	GPIO 输入状态

7.3 中断控制寄存器（0x0510）

基地址为 0x1FE00000，偏移地址 0x0510。

表 7-3 中断控制寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_INT_Pol	RW	32'h0	GPIO 中断有效电平设置 0 - 低电平有效 1 - 高电平有效
63:32	GPIO_INT_en	RW	32'h0	GPIO 中断使能控制，高有效

7.4 功能选择配置寄存器（0x0520）

基地址为 0x1FE00000，偏移地址 0x0520。

表 7-4 功能选择配置寄存器

位域	字段名	访问	复位值	描述
3:0	GPIO_FUNC1	RW	4'h0	复用功能选择 1：PCIE_resetn_sel 0：默认复用功能
9:4	GPIO_FUNC1	RW	4'h0	复用功能选择 1：PCIE_presentn_sel 0：默认复用功能
31:10				
34:32	PCIE_resetn_ch[0]	RW	3'h0	GPIO00 映射到 PCIE0-7 的任一个控制器
37:35	PCIE_resetn_ch[1]	RW	3'h0	GPIO01 映射到 PCIE0-7 的任一个控制器
40:38	PCIE_resetn_ch[2]	RW	3'h0	GPIO02 映射到 PCIE0-7 的任一个控制器
43:41	PCIE_resetn_ch[3]	RW	3'h0	GPIO03 映射到 PCIE0-7 的任一个控制器
46:44	PCIE_presentn_ch[0]	RW	3'h0	GPIO04 映射到 PCIE0-7 的任一个控制器
49:47	PCIE_presentn_ch[1]	RW	3'h0	GPIO05 映射到 PCIE0-7 的任一个控制器
52:50	PCIE_presentn_ch[2]	RW	3'h0	GPIO06 映射到 PCIE0-7 的任一个控制器
55:53	PCIE_presentn_ch[3]	RW	3'h0	GPIO07 映射到 PCIE0-7 的任一个控制器
58:56	PCIE_presentn_ch[4]	RW	3'h0	GPIO08 映射到 PCIE0-7 的任一个控制器
61:59	PCIE_presentn_ch[5]	RW	3'h0	GPIO09 映射到 PCIE0-7 的任一个控制器

将 GPIO 复用为 PCIE presentn 功能时，必须将对应位的 GPIO_FUNC_en 和 GPIO_FUNC1 设置为 1，且只允许将该 GPIO 对应的 PCIE_presentn_ch[x]设置为相应控制器的编号，寄存器中其它 PCIE_presentn_ch[y]的值必须设置为其它控制器的编号，不能与之相同。

7.5 GPIO 引脚功能复用表

3C6000 中 GPIO 引脚与其它功能进行了大量复用，以下列表为芯片功能引脚的引脚功能选择。

需要特别指出的是，GPIO00 – GPIO15 芯片复位时即为 GPIO 功能，默认为输入状态，不驱动 IO。

表 7-5 GPIO 功能复用表

GPIO 寄存器	引脚名称	复用功能 0	复用功能 1
0	GPIO00	SPI_CSn1	PCIE_RESETn0
1	GPIO01	SPI_CSn2	PCIE_RESETn1
2	GPIO02	UART1_RXD	PCIE_RESETn2
3	GPIO03	UART1_TXD	PCIE_RESETn3
4	GPIO04	UART1_RTS	PCIE_PRESENTn0
5	GPIO05	UART1_CTS	PCIE_PRESENTn1

6	GPIO06	UART1_DTR	PCIE_PRESENTn2
7	GPIO07	UART1_DSR	PCIE_PRESENTn3
8	GPIO08	UART1_DCD	PCIE_PRESENTn4
9	GPIO09	UART1_RI	PCIE_PRESENTn5
10	GPIO10	-	-
11	GPIO11	-	-
12	GPIO12	-	-
13	GPIO13	SCNT_RSTn	-
14	GPIO14	PROCHOTn	-
15	GPIO15	THERMTRIPn	-

7.6 GPIO 中断控制

3C6000 中 GPIO 引脚都可以作为中断输入使用。

GPIO00、GPIO08 共享中断控制器的 0 号中断线。

GPIO01、GPIO09 共享中断控制器的 1 号中断线。

GPIO02、GPIO10 共享中断控制器的 2 号中断线。

GPIO03、GPIO11 共享中断控制器的 3 号中断线。

GPIO04、GPIO12 共享中断控制器的 4 号中断线。

GPIO05、GPIO13 共享中断控制器的 5 号中断线。

GPIO06、GPIO14 共享中断控制器的 6 号中断线。

GPIO07、GPIO15 共享中断控制器的 7 号中断线。

每个 GPIO 的中断使能由配置寄存器 GPIO_INT_en 控制，中断电平由 GPIO_INT_POL 控制，寄存器如下：

基地址为 0x1FE00000，偏移地址 0x0510。

表 7-6 中断控制寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_INT_Pol	RW	32'h0	GPIO 中断有效电平设置 0 - 低电平有效 1 - 高电平有效
63:32	GPIO_INT_en	RW	32'h0	GPIO 中断使能控制，高有效

当中断控制器上的每个中断线只使能其中一位 GPIO 时，可以使用边沿触发方式，固定在某个沿（POL 设为 0 时下降沿，为 1 时上升沿）触发中断并在中断控制器中记录。

8 LA664 处理器核

LA664 是六发射 64 位的处理器核。在龙芯 3C6000 中的多个 LA664 核以及共享 Cache 模块通过 AXI 互连网络形成一个分布式共享片上末级 Cache 的多核结构。LA664 的主要特点如下：

- 支持龙芯自主指令系统（LoongArch）；
- 支持同时多线程技术；
- 六发射超标量结构，四个定点、四个向量、四个访存部件；
- 每个向量部件宽度为 256 位，最多支持 8 个单精度或 4 个双精度乘加运算；
- 访存部件支持 256 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；
- 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；
- 一级 Cache 实现奇偶校验，二级、片上末级 Cache 实现 ECC 校验，均支持一位纠正；
- 支持标准的 JTAG 调试接口，方便软硬件调试。

需要注意的是，因 LA664 处理器核支持同时多线程，包含两个逻辑核，通过 csr 0x20 读出的处理器核号为逻辑核号：其中 bit0 表示为线程号，bit[2:1]为物理核号。不论同时多线程机制被软件配置为开启还是关闭，4 个核的 0 号线程返回的逻辑核号（csr 0x20）均为 0/2/4/6。

8.1 3C6000 实现的指令集特性

龙芯 3C6000 具体实现的龙芯指令集功能特性，可以通过龙芯指令集属性识别机制予以动态确认。

龙芯 3C6000 推荐软件使用龙芯自定义的 CPUCFG 指令进行龙芯指令集属性的识别。

CPUCFG 指令为用户态指令，其使用方式为 CPUCFG rd, rj，其中源操作数 rj 寄存器中存放待访问配置信息字的寄存器号，返回的配置字信息写入到 rd 寄存器中，每个配置信息字包含至多 32 位配置信息。例如，1 号配置字中的第 0 位表明是否实现 LA32 架构，那

么这个配置信息就表示为 CPUCFG.1.LA32[bit0]，其中 1 表示配置信息字的字号是 1 号，LA32 表示这个配置信息域的所起的助记名称叫 LA32，bit0 表示 LA32 这个域位于配置字的第 0 位。如果配置信息需要多位表达，那么其位置信息将会记为 bitAA:BB 的形式，表示从配置信息字的第 AA 位到第 BB 位的连续(AA-BB+1) 位。

下表给出 3C6000 实现的指令集功能配置信息列表。最后一列“可能取值”，表示从这个寄存器中有可能读出的值，但不意味着从 3C6000 处理器中读出的就是这个值。具体读出的值，请按照实际硬件执行该指令读出的结果为准，并按照实际读出的值，进行后续的软件判断，尽量不要按本表格最后一列的内容，直接断定某个芯片支持或不支持某功能。

表 8-1 3C6000 实现的指令集功能配置信息列表

寄存器号	位域	字段名	描述	可能取值
0x0	31:0	PRId	处理器标识	32’h14_d001
0x1	1:0	ARCH	2’b00 表示实现 LA32 精简架构；2’b01 表示实现 LA32 架构；2’b10 表示实现 LA64 架构。2’b11 保留。	2’b10
	2	PGMMU	为 1 表示 MMU 支持页映射模式	1’b1
	3	IOCSR	为 1 表示支持 IOCSR 指令	1’b1
	11:4	PALEN	所支持的物理地址位数 PALEN 的值减 1	8’ d47
	19:12	VALEN	所支持的虚拟地址位数 VALEN 的值减 1	8’ d47
	20	UAL	为 1 表示支持非对齐访存	1’b1
	21	RI	为 1 表示支持“读禁止”页属性	1’b1
	22	EP	为 1 表示支持“执行保护”页属性	1’b1
	23	RPLV	为 1 表示支持 RPLV 页属性	1’b1
	24	HP	为 1 表示支持 huge page 页属性	1’b1
	25	CRC32	为 1 表示支持 CRC32 加速指令。	1’b1
	26	MSG_INT	为 1 表示支持消息模式外部中断	1’b1
0x2	0	FP	为 1 表示支持基础浮点数指令	1’b1
	1	FP_SP	为 1 表示支持单精度浮点数	1’b1
	2	FP_DP	为 1 表示支持双精度浮点数	1’b1
	5:3	FP_ver	浮点运算标准的版本号。1 为初始版本号，表示兼容 IEEE 754-2008 标准。	3’h1
	6	LSX	为 1 表示支持 128 位向量扩展	1’b1
	7	LASX	为 1 表示支持 256 位向量扩展	1’b1
	14	LLFTP	为 1 表示支持恒定频率计时器和定时器	1’b1
	17:15	LLFTP_ver	恒定频率计时器和定时器的版本号。1 为初始版本。	3’h1

	21	LSPW	为 1 表示支持软件页表遍历指令	1’b1
	22	LAM	为 1 表示支持 AM*原子访存指令	1’b1
0x3	0	CCDMA	为 1 表示支持硬件 Cache Coherent DMA	1’b1
	1	SFB	为 1 表示支持 Store Fill Buffer（SFB）	1’b1
	3	LLEXC	为 1 表示支持 LL 指令取独占块功能	1’b1
	4	SCDLY	为 1 表示支持 SC 后随机延迟功能	1’b1
	5	LLDBAR	为 1 表示支持 LL 自动带 dbar 功能	1’b1
	6	ITLBHMC	为 1 表示硬件维护 ITLB 与 TLB 之间的一致性	1’b1
	7	ICHMC	为 1 表示硬件维护同一处理器核内 ICache 与 DCache 的数据一致性	1’b1
	10:8	SPW_LVL	page walk 指令所支持的最大目录层数	3’h4
	11	SPW_HP_HF	为 1 表示 page walk 指令在遇到大页时将折半填入 TLB	1’b1
	12	RVA	为 1 表示支持软件配置缩短虚拟地址范围的功能	1’b1
	16:13	RVMAX-1	最大可以配置的虚拟地址缩短位数-1	4’h7
0x4	31:0	CC_FREQ	恒定频率计时器和定时器所用时钟对应的晶振频率，单位 Hz	N/A
0x5	15:0	CC_MUL	恒定频率计时器和定时器所用时钟对应的倍频因子	N/A
	31:16	CC_DIV	恒定频率计时器和定时器所用时钟对应的分频系数	N/A
0x6	0	PMP	为 1 表示支持性能计数器	1’b1
	3:1	PMVER	性能监测器中，架构定义事件的版本号，1 为初始版本。	3’h1
	7:4	PMNUM	性能监测器个数-1	4’h3
	13:8	PMBITS	性能监测计数器位宽-1	6’h3f
	14	UPM	为 1 表示支持用户态读取性能计数器	1’b1
0x10	0	L1 IU_Present	为 1 表示存在一级指令 Cache 或一级统一 Cache	1’b1
	1	L1 IU Unify	为 1 表示 L1 IU_Present 所示的 Cache 是统一 Cache	1’b0
	2	L1 D Present	为 1 表示存在一级数据 Cache	1’b1
	3	L2 IU Present	为 1 表示存在二级指令 Cache 或二级统一 Cache	1’b1
	4	L2 IU Unify	为 1 表示 L2 IU_Present 所示的 Cache 是统一 Cache	1’b1
	5	L2 IU Private	为 1 表示 L2 IU_Present 所示的 Cache 是每个核私有的	1’b1
	6	L2 IU Inclusive	为 1 表示 L2 IU_Present 所示的 Cache 对更低层次（L1）是包含关系	1’b0
	7	L2 D Present	为 1 表示存在二级数据 Cache	1’b0
	8	L2 D Private	为 1 表示二级数据 Cache 是每个核私有的	1’b0

	9	L2 D Inclusive	为 1 表示二级数据 Cache 对更低层次（L1）是包含关系	1'b0
	10	L3 IU Present	为 1 表示存在三级指令 Cache 或三级统一 Cache	1'b1
	11	L3 IU Unify	为 1 表示 L3 IU_Present 所示的 Cache 是统一 Cache	1'b1
	12	L3 IU Private	为 1 表示 L3 IU_Present 所示的 Cache 是每个核私有的	1'b0
	13	L3 IU Inclusive	为 1 表示 L3 IU_Present 所示的 Cache 对更低层次（L1 及 L2）是包含关系	1'b1
	14	L3 D Present	为 1 表示存在三级数据 Cache	1'b0
	15	L3 D Private	为 1 表示三级数据 Cache 是每个核私有的	1'b0
	16	L3 D Inclusive	为 1 表示三级数据 Cache 对更低层次（L1 及 L2）是包含关系	1'b0
0x11	15:0	Way-1	路数-1 （配置字 0x10 中 L1 IU_Present 对应的 Cache）	16'h3
	23:16	Index-log2	log2(每一路 Cache 行数) （配置字 0x10 中 L1 IU_Present 对应的 Cache）	8'h8
	30:24	Linesize-log2	log2(Cache 行字节数) （配置字 0x10 中 L1 IU_Present 对应的 Cache）	8'h6
0x12	15:0	Way-1	路数-1 （配置字 0x10 中 L1 D Present 对应的 Cache）	16'h3
	23:16	Index-log2	log2(每一路 Cache 行数) （配置字 0x10 中 L1 D Present 对应的 Cache）	8'h8
	30:24	Linesize-log2	log2(Cache 行字节数) （配置字 0x10 中 L1 D Present 对应的 Cache）	8'h6
0x13	15:0	Way-1	路数-1 （配置字 0x10 中 L2 IU Present 对应的 Cache）	16'hf
	23:16	Index-log2	log2(每一路 Cache 行数) （配置字 0x10 中 L2 IU Present 对应的 Cache）	8'h8
	30:24	Linesize-log2	log2(Cache 行字节数) （配置字 0x10 中 L2 IU Present 对应的 Cache）	8'h6
0x14	15:0	Way-1	路数-1 （配置字 0x10 中 L3 IU Present 对应的 Cache）	16'hf
	23:16	Index-log2	log2(每一路 Cache 行数) （配置字 0x10 中 L3 IU Present 对应的 Cache）	8'h8
	30:24	Linesize-log2	log2(Cache 行字节数) （配置字 0x10 中 L3 IU Present 对应的 Cache）	8'h6

8.2 配置状态寄存器访问

3C6000 支持配置状态寄存器空间访问，使用 IOCSR 指令对独立的寻址空间进行访问。该空间称为 IOCSR 空间，与现有的寄存器空间、内存空间和 JTAG 空间等互不重叠。

IOCSR 空间通过 IOCSRRD.B/H/W/D 和 IOCSRWR.B/H/W/D 指令进行读、写访问,这些指令只允许在核心态下运行。

IOCSRRD.B/H/W/D 的使用方式为 IOCSRRD.B/H/W/D rd, rj，其中源操作数 rj 寄存器中存放待访问的 IOCSR 地址，读回的内容写入到 rd 寄存器中。

IOCSRWR.B/H/W/D 的使用方式为 IOCSRWR.B/H/W/D rd, rj，其中源操作数 rj 寄存器中存放待访问的 IOCSR 地址，源操作数 rd 寄存器中存放待写入 IOCSR 的值。

使用 IOCSRRD.B/H/W/D 和 IOCSRWR.B/H/W/D 指令可以替代原有的地址映射配置寄存器的方式，即 0x1FE00000 空间，具体的访问方式参考相关章节说明。

9 共享 Cache（SCache）

SCache 模块是每个龙芯 3C6000 处理器内部处理器核所共享的三级 Cache。SCache 模块的主要特征包括：

- 16 项 Cache 访问队列。
- 关键字优先。
- 通过目录支持 Cache 一致性协议。
- 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。
- 支持多种共享 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 时连续区域大小需小于 30MB。

配置窗口基地址为 0x1FE0_0000，或者通过 IOCSR 指令访问。

表 9-1 共享 Cache 锁窗口寄存器配置

名称	偏移地址	位域	描述
Slock0_valid	0x0200	[63:63]	0 号锁窗口有效位
Slock0_addr	0x0200	[47:0]	0 号锁窗口锁地址
Slock0_mask	0x0240	[47:0]	0 号锁窗口掩码
Slock1_valid	0x0208	[63:63]	1 号锁窗口有效位
Slock1_addr	0x0208	[47:0]	1 号锁窗口锁地址

Slock1_mask	0x0248	[47:0]	1 号锁窗口掩码
Slock2_valid	0x0210	[63:63]	2 号锁窗口有效位
Slock2_addr	0x0210	[47:0]	2 号锁窗口锁地址
Slock2_mask	0x0250	[47:0]	2 号锁窗口掩码
Slock3_valid	0x0218	[63:63]	3 号锁窗口有效位
Slock3_addr	0x0218	[47:0]	3 号锁窗口锁地址
Slock3_mask	0x0258	[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。

每结点 4 个 scache 使用同一个配置寄存器，基地址为 0x1FE00000，也可以使用配置寄存器指令（IOCSR）进行访问，偏移地址 0x0280。

表 9-2 共享 Cache 配置寄存器（SC_CONFIG）

位域	字段名	访问	复位值	描述
0	LRU en	RW	1'b1	Scache LRU 替换算法使能
16	Prefetch En	RW	1'b1	Scache 预取功能使能
22:20	Prefetch config	RW	3'h1	当 scache 预取越过配置大小的地址范围时，停止预取 0 - 4KB 1 - 16KB 2 - 64KB 3 - 1MB 7 - 不受限 （注：SCID_SEL==0 时有效）
26:24	Prefetch lookahead	RW	3'h2	scache 预取步长 0 - 保留 1 - 0x100 2 - 0x200 3 - 0x300 4 - 0x400 5 - 0x500 6 - 0x600 7 - 0x700 （注：SCID_SEL==0 时有效）
30:28	Sc stall dirq cycle	RW	3'h2	SC 指令堵住 dirq 的时钟周期数 0 - 1 cycle（nonstall） 1 - 16-31 cycle random 2 - 32-63 cycle random 3 - 64-127 cycle random 4 - 128-255 cycle random 其他 - 无效值

31	MCC storefill en	RW	1'b0	MCC storefill 功能使能
34:32				
35	MCC clean exclusive replace en	RW	1'b0	
36	MCC clean shared replace en	RW	1'b0	

10 处理器核间中断与通信

龙芯 3C6000 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。

龙芯 3C6000 中支持两种不同的访问方式，一种是地址访问模式，另一种是 IOCSR 指令访问模式。推荐使用处理器 IOCSR 指令访问模式。

10.1 按地址访问模式

对于龙芯 3C6000，下列寄存器可以使用基地址 0x1FE0_0000 进行访问。

由于龙芯 3C6000 中包括 32 个逻辑核，连续的两个逻辑核号对应一个物理核，在龙芯 3 号空间规则下，32 个逻辑核被划分为 8 个内部结点。

当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]加上内部结点号。

表 10-1 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。

与处理器核间中断相关的寄存器及其功能描述如下，以下的处理器核都是指逻辑核。

表 10-2 0 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core0_IPI_Status	0x1000	R	0 号处理器核的 IPI_Status 寄存器
Core0_IPI_Enalbe	0x1004	RW	0 号处理器核的 IPI_Enalbe 寄存器
Core0_IPI_Set	0x1008	W	0 号处理器核的 IPI_Set 寄存器
Core0_IPI_Clear	0x100c	W	0 号处理器核的 IPI_Clear 寄存器
Core0_MailBox0	0x1020	RW	0 号处理器核的 IPI_MailBox0 寄存器
Core0_MailBox1	0x1028	RW	0 号处理器核的 IPI_MailBox1 寄存器
Core0_MailBox2	0x1030	RW	0 号处理器核的 IPI_MailBox2 寄存器

Core0_MailBox3	0x1038	RW	0 号处理器核的 IPI_MailBox3 寄存器

表 10-3 1 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core1_IPI_Status	0x1100	R	1 号处理器核的 IPI_Status 寄存器
Core1_IPI_Enalbe	0x1104	RW	1 号处理器核的 IPI_Enalbe 寄存器
Core1_IPI_Set	0x1108	W	1 号处理器核的 IPI_Set 寄存器
Core1_IPI_Clear	0x110c	W	1 号处理器核的 IPI_Clear 寄存器
Core1_MailBox0	0x1120	R	1 号处理器核的 IPI_MailBox0 寄存器
Core1_MailBox1	0x1128	RW	1 号处理器核的 IPI_MailBox1 寄存器
Core1_MailBox2	0x1130	W	1 号处理器核的 IPI_MailBox2 寄存器
Core1_MailBox3	0x1138	W	1 号处理器核的 IPI_MailBox3 寄存器

表 10-4 2 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core2_IPI_Status	0x1200	R	2 号处理器核的 IPI_Status 寄存器
Core2_IPI_Enalbe	0x1204	RW	2 号处理器核的 IPI_Enalbe 寄存器
Core2_IPI_Set	0x1208	W	2 号处理器核的 IPI_Set 寄存器
Core2_IPI_Clear	0x120c	W	2 号处理器核的 IPI_Clear 寄存器
Core2_MailBox0	0x1220	R	2 号处理器核的 IPI_MailBox0 寄存器
Core2_MailBox1	0x1228	RW	2 号处理器核的 IPI_MailBox1 寄存器
Core2_MailBox2	0x1230	W	2 号处理器核的 IPI_MailBox2 寄存器
Core2_MailBox3	0x1238	W	2 号处理器核的 IPI_MailBox3 寄存器

表 10-5 3 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core3_IPI_Status	0x1300	R	3 号处理器核的 IPI_Status 寄存器
Core3_IPI_Enalbe	0x1304	RW	3 号处理器核的 IPI_Enalbe 寄存器
Core3_IPI_Set	0x1308	W	3 号处理器核的 IPI_Set 寄存器
Core3_IPI_Clear	0x130c	W	3 号处理器核的 IPI_Clear 寄存器
Core3_MailBox0	0x1320	R	3 号处理器核的 IPI_MailBox0 寄存器
Core3_MailBox1	0x1328	RW	3 号处理器核的 IPI_MailBox1 寄存器
Core3_MailBox2	0x1330	W	3 号处理器核的 IPI_MailBox2 寄存器
Core3_MailBox3	0x1338	W	3 号处理器核的 IPI_MailBox3 寄存器

上面列出的是龙芯 3C6000 芯片所组成的多处理器系统逻辑核 0 至逻辑核 3 的核间中断相关寄存器列表。由于每个龙芯 3C6000 中包括 32 个逻辑核，连续的两个逻辑核号对应一个物理核，在龙芯 3 号空间规则下，32 个逻辑核被划分为 8 个结点。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]上加上内部结点号。例如，核 4 至核 7 的基地址为 0x1FE10000，具体的寄存器对应于该基址上核 0 至核 3 的偏移。

不同内部结点如下表所示：

表 10-6 不同内部结点的基地址

内部结点号	基地址
0	0x1FE0_0000
1	0x1FE1_0000
2	0x1FE2_0000
3	0x1FE3_0000
4	0x1FE4_0000
5	0x1FE5_0000
6	0x1FE6_0000
7	0x1FE7_0000

10.2 配置寄存器指令访问

使用处理器核 IOCSR 指令，可以通过私有空间对配置寄存器进行访问。

表 10-7 当前处理器核核间中断与通信寄存器列表

名称	偏移地址	权限	描述
perCore_IPI_Status	0x1000	R	当前处理器核的 IPI_Status 寄存器
perCore_IPI_Enalbe	0x1004	RW	当前处理器核的 IPI_Enalbe 寄存器
perCore_IPI_Set	0x1008	W	当前处理器核的 IPI_Set 寄存器
perCore_IPI_Clear	0x100c	W	当前处理器核的 IPI_Clear 寄存器
perCore_MailBox0	0x1020	RW	当前处理器核的 IPI_MailBox0 寄存器
perCore_MailBox1	0x1028	RW	当前处理器核的 IPI_MailBox1 寄存器
perCore_MailBox2	0x1030	RW	当前处理器核的 IPI_MailBox2 寄存器
perCore_MailBox3	0x1038	RW	当前处理器核的 IPI_MailBox3 寄存器

为了向其它核发送核间中断请求及 MailBox 通信，通过以下寄存器进行访问。

表 10-8 处理器核核间通信寄存器

名称	偏移地址	权限	描述
IPI_Send	0x1040	WO	32 位中断分发寄存器 [31] 等待完成标志，置 1 时会等待中断生效 [30:26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 中断向量号，对应 IPI_Status 中的向量
Mail_Send	0x1048	WO	64 位 MailBox 缓存寄存器 [63:32] MailBox 数据 [31] 等待完成标志，置 1 时会等待写入生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留

			[25:16] 处理器核号 [15:5] 保留 [4:2] MailBox 号 0 - MailBox0 低 32 位 1 - MailBox0 高 32 位 2 - MailBox1 低 32 位 3 - MailBox1 高 32 位 4 - MailBox2 低 32 位 5 - MailBox2 高 32 位 6 - MailBox3 低 32 位 7 - MailBox3 高 32 位 [1:0] 保留
FREQ_Send	0x1058	WO	32 位频率使能寄存器 [31] 等待完成标志，置 1 时会等待设置生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 写入对应的处理器核私有频率配置寄存器。IOCSR[0x1050]

需要注意的是，由于 Mail_Send 寄存器一次只可以发送 32 位的数据，当发送 64 位数据时必须拆分为两次发送。因此，目标核在等待 Mail_Box 内容时，需要通过其它的软件手段来确保传输的完整性。例如，发送完 Mail_Box 数据之后，通过核间中断来表示已经发送完成。

10.3 配置寄存器指令调试支持

配置寄存器指令原则上在使用时不跨片访问，但为了满足对跨结点写操作的需求，在此支持使用寄存器地址对跨片访问进行支持。值得注意的是，这类寄存器只能写，不能读。

除了前一节提到的 IPI_Send、Mail Send、Freq Send，还有一个 Any Send 寄存器可供使用，其地址如下。

表 10-9 处理器配置访问寄存器

名称	偏移地址	权限	描述
ANY_Send	0x1158	WO	64 位寄存器访问寄存器 [63:32] 写入数据 [31] 等待完成标志，置 1 时会等待中断生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写

			入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 目标处理器核号 [15:0] 写入的寄存器偏移地址

11 I/O 中断

龙芯 3C6000 芯片支持三种不同的中断方式。第一种为传统中断方式，可以支持处理器上的平台设备；第二种为扩展 IO 中断方式，用于支持 PCI 设备的中断跨片与动态分发功能；第三种为中断映射方式，可以支持中断虚拟化等复杂用法。以下分别对前两种中断方式进行介绍。

龙芯 3C6000 上，每个内部结点都有对应的中断控制器，可以对每个结点的中断控制器分别配置使用。每个中断控制器寄存器的访问方法与前面章节配置寄存器的访问方法一致。

11.1 传统 I/O 中断

龙芯 3C6000 芯片的传统中断支持 32 个中断源，以统一方式进行管理，如下图所示。

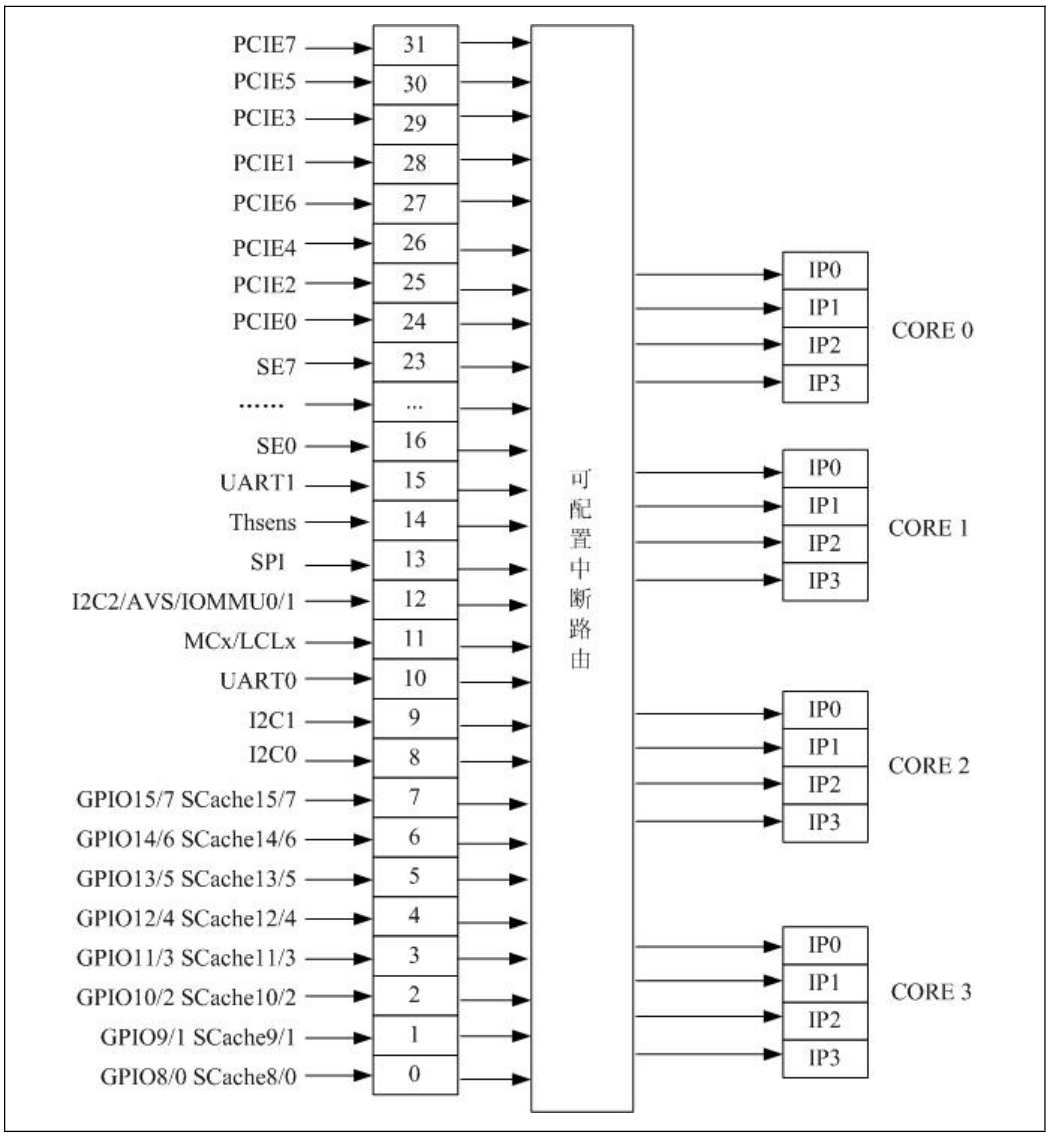


图 11-1 龙芯 3C6000 处理器中断路由示意图

每一个内部结点对应一个中断路由模块，其寄存器的基址与其它的配置寄存器基址同样都是内部结点号位置的变化。每个中断路由模块的中断源，除了 MC[3:0]和 LCL[3:0]的中断分别路由至内部结点[3:0]之外，其它的中断源全部一致。

任意一个 IO 中断源可以被配置为是否使能，可以配置触发的方式以及被路由的目标处理器核中断脚。传统中断不支持中断的跨结点分发，只能中断同一个处理器结点内的处理器核。

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。

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。

边沿触发的中断信号由 Intedge 配置寄存器来选择，写 1 表示边沿触发，写 0 表示电平触发。中断处理程序可以通过 Intenclr 的相应位来清除中断记录，清除中断的同时也会清除中断使能。

表 11-1 中断控制寄存器

位域	访问属性/缺省值				
	Intedge	Inten	Intenset	Intenclr	中断源
0	RW / 0	R / 0	RW / 0	RW / 0	GPIO8/0/SC8/0
1	RW / 0	R / 0	RW / 0	RW / 0	GPIO9/1/SC9/1
2	RW / 0	R / 0	RW / 0	RW / 0	GPIO10/2/SC10/2
3	RW / 0	R / 0	RW / 0	RW / 0	GPIO11/3/SC11/3
4	RW / 0	R / 0	RW / 0	RW / 0	GPIO12/4/SC12/4
5	RW / 0	R / 0	RW / 0	RW / 0	GPIO13/5/SC13/5
6	RW / 0	R / 0	RW / 0	RW / 0	GPIO14/6/SC14/6
7	RW / 0	R / 0	RW / 0	RW / 0	GPIO15/7/SC15/7
8	RW / 0	R / 0	RW / 0	RW / 0	I2C0
9	RW / 0	R / 0	RW / 0	RW / 0	I2C1
10	RW / 0	R / 0	RW / 0	RW / 0	UART0
11	RW / 0	R / 0	RW / 0	RW / 0	MCx/LCLx
12	RW / 0	R / 0	RW / 0	RW / 0	I2C2/AVS/IOMMU0/1
13	RW / 0	R / 0	RW / 0	RW / 0	SPI
14	RW / 0	R / 0	RW / 0	RW / 0	Thsens
15	RW / 0	R / 0	RW / 0	RW / 0	UART1
23 : 16	RW / 0	R / 0	RW / 0	RW / 0	SE[7:0]
31 : 24	RW / 0	R / 0	RW / 0	RW / 0	PCIE7/5/3/1/6/4/2/0

与核间中断类似，IO 中断的基地址同样可以使用 0x1FE00000 进行访问，也可以通过处理器核的 IOCSR 指令进行访问。不同结点使用的基地址规则与之前的配置寄存器基地址规则一致。

11.1.1 按地址访问

按地址访问方式下，基地址可以使用 0x1FE00000。不同结点使用的基地址规则与之前的配置寄存器基地址规则一致。由于每个龙芯 3C6000 中包括 32 个逻辑核，连续的两个逻辑核号对应一个物理核，32 个逻辑核被划分为 8 个内部结点。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]加上内部结点号。例如，核 4 至核 7 的基地址为 0x1FE10000，具体的寄存器对应于该基址上核 0 至核 3 的偏移。

表 11-2 IO 控制寄存器地址

名称	偏移地址	描述
Intisr	0x1420	32 位中断状态寄存器
Inten	0x1424	32 位中断使能状态寄存器
Intenset	0x1428	32 位设置使能寄存器
Intenclr	0x142c	32 位清除使能寄存器
Intedge	0x1434	32 位触发方式寄存器
CORE0_INTISR	0x1440	路由给 CORE0 的 32 位中断状态
CORE1_INTISR	0x1448	路由给 CORE1 的 32 位中断状态
CORE2_INTISR	0x1450	路由给 CORE2 的 32 位中断状态
CORE3_INTISR	0x1458	路由给 CORE3 的 32 位中断状态

龙芯 3C6000 每个内部结点对应 4 个逻辑核，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逻辑核。进一步，中断源可以选择路由到该结点内的逻辑核中断 INT0 到 INT7 中任意一个。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所示。路由寄存器采用向量的方式进行路由选择，如 0x48 表示路由到 3 号逻辑核的 INT2 上。

采用位向量模式时，中断路由能够直接对应低 4 位中断引脚；而使用编码方式，则可以映射到 8 个中断中的任意一个，编码方式由 IOCSR[0x420][49]位控制使能。当该位使能时，下表的[7:4]由位图表示法变为数值编码法。可配置的数值 0-7 表示中断引脚 0-7。例如，在该模式下，0x28 表示路由到 3 号逻辑核的 INT2 上。

表 11-3 中断路由寄存器的说明

位域	说　明
3:0	路由的逻辑核向量号
7:4	路由的逻辑核中断引脚向量号

表 11-4 中断路由寄存器地址

名称	偏移地址	描述	名称	偏移地址	描述
Entry0	0x1400	GPIO8/0/SC8/0	Entry16	0x1410	SE-int0
Entry1	0x1401	GPIO9/1/SC9/1	Entry17	0x1411	SE-int1
Entry2	0x1402	GPIO10/2/SC10/2	Entry18	0x1412	SE-int2
Entry3	0x1403	GPIO11/3/SC11/3	Entry19	0x1413	SE-int3
Entry4	0x1404	GPIO12/4/SC12/4	Entry20	0x1414	SE-int4
Entry5	0x1405	GPIO13/5/SC13/5	Entry21	0x1415	SE-int5
Entry6	0x1406	GPIO14/6/SC14/6	Entry22	0x1416	SE-int6
Entry7	0x1407	GPIO15/7/SC15/7	Entry23	0x1417	SE-int7
Entry8	0x1408	I2C0	Entry24	0x1418	PCIE0
Entry9	0x1409	I2C1	Entry25	0x1419	PCIE2
Entry10	0x140a	UART0	Entry26	0x141a	PCIE4
Entry11	0x140b	MCx/LCLx	Entry27	0x141b	PCIE6
Entry12	0x140c	I2C2/AVS/IOMMU0/1	Entry28	0x141c	PCIE1
Entry13	0x140d	SPI	Entry29	0x141d	PCIE3
Entry14	0x140e	Thsens	Entry30	0x141e	PCIE5
Entry15	0x140f	UART1	Entry31	0x141f	PCIE7

11.1.2 配置寄存器指令访问

在龙芯 3C6000 中，可以使用配置寄存器指令，通过私有空间对配置寄存器进行访问。指令所使用的偏移地址与通过地址访问的方式相同。此外，为了方便用户的使用，对每个核不同的当前中断状态设置了专用的私有中断状态寄存器，如下表所示。

表 11-5 处理器核私有中断状态寄存器

名称	偏移地址	描述
perCore_INTISR	0x1010	路由给当前处理器核的 32 位中断状态

11.2 扩展 I/O 中断

除了兼容原有的传统 IO 中断方式，3C6000 支持扩展 I/O 中断，用于将 PCI 设备的 256 位中断直接分发给各个处理器核，而不再通过传统的中断线进行转发，提升 IO 中断使用的灵活性。需要注意的是，以下的处理器核向量号都是指逻辑核号。

内核在使用扩展 IO 中断前，需要使能“其他功能设置寄存器”中的对应位。该寄存器

基地址为 0x1FE00000，偏移地址 0x0420。

表 11-6 其它功能设置寄存器

位域	字段名	访问	复位值	描述
48	EXT_INT_en	RW	0x0	扩展 IO 中断使能

在扩展 IO 中断模式下，设备中断可以直接进行跨片的转发以及轮转分发等操作。当前版本，最多可以支持 256 个扩展中断向量。

扩展中断配置中，需要按照中断目标的逻辑核号（thread）进行相应的路由配置，配置中主要涉及 IOI_Cluster、IOI_Node、IOI_Core 这三个不同的层次的设置，分别如下。

（1）IOI_Cluster：中断组，每 64 个逻辑核为一个组中。

IOI_Cluster = thread / 64

（2）IOI_Node：组内结点，每组各 16 个 INT_Node，每 4 个逻辑核为一个内部结点。

IOI_Node = (thread / 4) % 16

（3）IOI_Core：结点内核号，每个结点内各 4 个逻辑核。

IOI_Core = thread % 4

11.2.1 按地址访问

以下是相关的扩展 IO 中断寄存器。与其它的配置寄存器一样，基地址可以使用 0x1FE00000，也可以通过处理器核的专用寄存器配置指令进行访问。不同结点使用的基地址规则与之前的配置寄存器基地址规则一致。由于每个龙芯 3C6000 中包括 32 个逻辑核，连续的两个逻辑核号对应一个物理核，32 个逻辑核被划分为 8 内部个结点。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]上加上内部结点号。例如，核 4 至核 7 的基地址为 0x1FE10000，具体的寄存器对应于该基址上核 0 至核 3 的偏移。

表 11-7 扩展 IO 中断使能寄存器

名称	偏移地址	描述
EXT_IOIen[63:0]	0x1600	扩展 IO 中断[63:0]的中断使能配置
EXT_IOIen[127:64]	0x1608	扩展 IO 中断[127:64]的中断使能配置
EXT_IOIen[191:128]	0x1610	扩展 IO 中断[191:128]的中断使能配置
EXT_IOIen[255:192]	0x1618	扩展 IO 中断[255:192]的中断使能配置

表 11-8 扩展 IO 中断自动轮转使能寄存器

名称	偏移地址	描述
EXT_IOIbounce[63:0]	0x1680	扩展 IO 中断[63:0]的自动轮转使能配置
EXT_IOIbounce[127:64]	0x1688	扩展 IO 中断[127:64]的自动轮转使能配置
EXT_IOIbounce[191:128]	0x1690	扩展 IO 中断[191:128]的自动轮转使能配置
EXT_IOIbounce[255:192]	0x1698	扩展 IO 中断[255:192]的自动轮转使能配置

这里需要注意的是，在 bounce 使能的情况下，不能随意修改对应中断位的路由信息。

表 11-9 扩展 IO 中断状态寄存器

名称	偏移地址	描述
EXT_IOIsr[63:0]	0x1700	扩展 IO 中断[63:0]的中断状态
EXT_IOIsr[127:64]	0x1708	扩展 IO 中断[127:64]的中断状态
EXT_IOIsr[191:128]	0x1710	扩展 IO 中断[191:128]的中断状态
EXT_IOIsr[255:192]	0x1718	扩展 IO 中断[255:192]的中断状态

表 11-10 各处理器核的扩展 IO 中断状态寄存器

名称	偏移地址	描述
CORE0_EXT_IOIsr[63:0]	0x1800	路由至处理器核 0 的扩展 IO 中断[63:0]的中断状态
CORE0_EXT_IOIsr[127:64]	0x1808	路由至处理器核 0 的扩展 IO 中断[127:64]的中断状态
CORE0_EXT_IOIsr[191:128]	0x1810	路由至处理器核 0 的扩展 IO 中断[191:128]的中断状态
CORE0_EXT_IOIsr[255:192]	0x1818	路由至处理器核 0 的扩展 IO 中断[255:192]的中断状态
CORE1_EXT_IOIsr[63:0]	0x1900	路由至处理器核 1 的扩展 IO 中断[63:0]的中断状态
CORE1_EXT_IOIsr[127:64]	0x1908	路由至处理器核 1 的扩展 IO 中断[127:64]的中断状态
CORE1_EXT_IOIsr[191:128]	0x1910	路由至处理器核 1 的扩展 IO 中断[191:128]的中断状态
CORE1_EXT_IOIsr[255:192]	0x1918	路由至处理器核 1 的扩展 IO 中断[255:192]的中断状态
CORE2_EXT_IOIsr[63:0]	0x1A00	路由至处理器核 2 的扩展 IO 中断[63:0]的中断状态
CORE2_EXT_IOIsr[127:64]	0x1A08	路由至处理器核 2 的扩展 IO 中断[127:64]的中断状态
CORE2_EXT_IOIsr[191:128]	0x1A10	路由至处理器核 2 的扩展 IO 中断[191:128]的中断状态
CORE2_EXT_IOIsr[255:192]	0x1A18	路由至处理器核 2 的扩展 IO 中断[255:192]的中断状态
CORE3_EXT_IOIsr[63:0]	0x1B00	路由至处理器核 3 的扩展 IO 中断[63:0]的中断状态
CORE3_EXT_IOIsr[127:64]	0x1B08	路由至处理器核 3 的扩展 IO 中断[127:64]的中断状态
CORE3_EXT_IOIsr[191:128]	0x1B10	路由至处理器核 3 的扩展 IO 中断[191:128]的中断状态
CORE3_EXT_IOIsr[255:192]	0x1B18	路由至处理器核 3 的扩展 IO 中断[255:192]的中断状态

与传统 IO 中断类似，扩展 IO 中断的 256 位中断源也可以通过软件配置选择期望中断的目标处理器核。

但中断源并不可以单独选择路由到处理器核中断 INT0 到 INT7 中的任意一个，而是以组为单位进行 INT 中断的路由，以中断对应 ESTAT 的 IS2 到 IS9。下面是按组进行配置的中断引脚路由寄存器。

中断引脚路由位支持编码方式，由 IOCSR[0x420][49]位控制使能。当该位使能时，下表的[3:0]由位图表示法变为数值编码法。可配置的数值 0-7 表示中断引脚 0-7。例如，在该模式下，0x2 表示路由到 INT2 上。

表 11-11 中断引脚路由寄存器的说明

位域	说 明
3:0	路由的处理器核中断引脚向量号
7:4	保留

表 11-12 中断路由寄存器地址

名称	偏移地址	描述
EXT_IOImap0	0x14C0	EXT_IOI[31:0]的引脚路由方式
EXT_IOImap1	0x14C1	EXT_IOI[63:32]的引脚路由方式
EXT_IOImap2	0x14C2	EXT_IOI[95:64]的引脚路由方式
EXT_IOImap3	0x14C3	EXT_IOI[127:96]的引脚路由方式
EXT_IOImap4	0x14C4	EXT_IOI[159:128]的引脚路由方式
EXT_IOImap5	0x14C5	EXT_IOI[191:160]的引脚路由方式
EXT_IOImap6	0x14C6	EXT_IOI[223:192]的引脚路由方式
EXT_IOImap7	0x14C7	EXT_IOI[255:224]的引脚路由方式

每个中断源都另外对应一个 8 位的路由控制器，其格式和地址如表 11-13 所示。其中[7:4]用于在表 11-14 中选择真正的结点路由向量。路由寄存器采用向量的方式进行路由选择，如 0x48 表示路由到 EXT_IOI_node_type4 及 EXT_IOI_cluster_type4 所指结点的 3 号处理器核。

表 11-13 中断目标处理器核路由寄存器的说明

位域	说 明
3:0	路由的处理器核向量号
7:4	路由的内部结点映射方式选择（选择下表中的一项）

需要注意的是，当使用轮转分发模式时（对应的 EXT_IOIbounce 为 1），在结点号与处理器核号的全映射模式上轮转。EXT_IOIbounce 的设置应该在相关的路由映射配置之后。需要注意的是，此处结点的计算为前面给出的 IOI_Node。

IOI_Node = (thread / 4) % 16

例如，当表 11-13 中的设置为 0x27，而表 11-14 中的 EXT_IOI_node_type2 的设置为 0x0013 时，使能轮转分发模式下，该中断将依次在处理器 0 核 0（内部结点 0 核 0，核号 0）、处理器 0 核 1（内部结点 0 核 1，核号 1）、处理器 0 核 2（内部结点 0 核 2，核号 2）、处理器 0 核 4（内部结点 1 核 0，核号 4）、处理器 0 核 5（内部结点 1 核 1，核号 5）、处理器 0 核 6（内部结点 1 核 2，核号 6）、处理器 1 核 0（内部结点 4 核 0，核号 16）、处理器 1 核 1（内部结点 4 核 1，核号 17）、处理器 1 核 2（内部结点 4 核 2，核号 18）上轮转。

当使用固定分发模式时（对应的 EXT_IOIbounce 为 0），结点号的 bitmap 上只允许有一位为 1，或为全 0，对应本地触发。

表 11-14 中断目标处理器核路由寄存器地址

名称	偏移地址	描述
EXT_IOImap_Core0	0x1C00	EXT_IOI[0]的处理器核路由方式
EXT_IOImap_Core1	0x1C01	EXT_IOI[1]的处理器核路由方式
EXT_IOImap_Core2	0x1C02	EXT_IOI[2]的处理器核路由方式
……		
EXT_IOImap_Core254	0x1CFE	EXT_IOI[254]的处理器核路由方式
EXT_IOImap_Core255	0x1CFF	EXT_IOI[255]的处理器核路由方式

表 11-15 中断目标结点映射方式配置

名称	偏移地址	描述
EXT_IOI_node_type0	0x14A0	16 个结点的映射向量类型 0（软件配置）
EXT_IOI_node_type1	0x14A2	16 个结点的映射向量类型 1（软件配置）
EXT_IOI_node_type2	0x14A4	16 个结点的映射向量类型 2（软件配置）
……		
EXT_IOI_node_type15	0x14BE	16 个结点的映射向量类型 15（软件配置）

除了 EXT_IOI_node_type 之外，还需要配置相应的 EXT_IOI_Cluster_type。每个 IOI_Cluster 对应 2 个硅片，包含 16 个结点共 64 核。Cluster_type 与 node_type 一一对应，寄存器地址为 0x14E0 – 0x14F8，每 16 位为一组，共 16 组。具体来说，使用 EXT_IOImap_Core 每 8 位的高 4 位作为索引，使用其指向的 EXT_IOI_node_type 和 EXT_IOI_Cluster_type 选择的一组映射。

与 EXT_IOI_node_type 中采用位向量不同的是，EXT_IOI_Cluster_type 采用编码的方向，指向唯一的映射组。换句话说，如果采用轮转方式进行中断路由，最多只能在同一个 Cluster 内部进行硬件的自动轮转分发。

表 11-16 中断组映射寄存器地址

名称	偏移地址	描述
EXT_IOI_Cluster_type0	0x14E0	4 个 Cluster 的映射编码类型 0（软件配置）
EXT_IOI_Cluster_type1	0x14E2	4 个 Cluster 的映射编码类型 1（软件配置）
EXT_IOI_Cluster_type2	0x14E4	4 个 Cluster 的映射编码类型 2（软件配置）
……		
EXT_IOI_Cluster_type15	0x14FE	4 个 Cluster 的映射编码类型 15（软件配置）

11.2.2 配置寄存器指令访问

使用处理器核的配置寄存器指令进行访问时，最大的不同在于对处理器核的中断状态寄存器的访问成为私有的访问，每个核都只需要向同一个地址发出查询请求就可以得到当前

核的中断状态。

表 11-17 当前处理器核的扩展 IO 中断状态寄存器

名称	偏移地址	描述
perCore_EXT_IOIsr[63:0]	0x1800	路由至当前处理器核的扩展 IO 中断[63:0]的中断状态
perCore_EXT_IOIsr[127:64]	0x1808	路由至当前处理器核的扩展 IO 中断[127:64]的中断状态
perCore_EXT_IOIsr[191:128]	0x1810	路由至当前处理器核的扩展 IO 中断[191:128]的中断状态
perCore_EXT_IOIsr[255:192]	0x1818	路由至当前处理器核的扩展 IO 中断[255:192]的中断状态

11.2.3 扩展 IO 中断触发寄存器

为了支持扩展 IO 中断的动态分发，在配置寄存器中增加了一个扩展 IO 中断触发寄存器，用于将对应的 IO 中断置位。平时可以利用这个寄存器对中断进行调试或测试。

这个寄存器的说明如下：

表 11-18 扩展 IO 中断触发寄存器

名称	偏移地址	权限	描述
EXT_IOI_send	0x1140	WO	扩展 IO 中断设置寄存器 [7:0]为期望设置的中断向量

11.2.4 扩展 IO 中断与传统中断处理的区别

传统的中断处理方式下，中断由高速接口控制器进行内部处理，直接映射到高速 IO 配置寄存器上的 256 个中断向量，再由 256 个中断向量分组产生 4 个或 8 个中断，再路由至各个不同的处理器核。由于采用的是传统的中断线连接，不能直接产生跨片中断，由此所有的 IO 中断都只能直接由单个芯片进行处理。

扩展 IO 中断方式，设备中断由接口控制器直接发给芯片的中断控制器进行处理，中断控制器能直接得到 256 位中断，而不是之前的 1 个中断，这 256 位中断每一位都可以独立路由，独立分发，而且可以实现跨片的分发及轮转。

使用扩展 IO 中断之后，软件处理上与使用传统的中断稍有不同。

传统的接口中断处理时，内核直接到控制器的中断向量上进行查找，然后按位进行处理，只能直接处理芯片内部有中断线连接的中断。

使用扩展 IO 中断之后，内核直接到扩展 IO 状态寄存器（配置空间 0x1800）上读取中断状态进行处理，每个核只会读到中断自己的中断状态并进行处理，不同核之间不会产生干扰。

12 温度传感器

12.1 实时温度采样

龙芯 3C6000 内部集成多个温度传感器，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 0x1FE00000，偏移地址为 0x0198）：

表 12-1 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
25	Thsens1_overflow	R		温度传感器 1 上溢
47:32	Thsens0_out	R		温度传感器 0 摄氏温度 结点温度=Thens0_out *820/0x4000 - 311 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 *820/0x4000 - 311 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。

此外，还可以使用新增的摄氏温度寄存器直接读取当前的摄氏温度。这个寄存器同样可以使用 0x1FE00000 为基地址的读操作进行访问，也可以使用配置寄存器指令进行直接访问，偏移地址为 0x0428。该寄存器描述如下：

表 12-2 温度观测寄存器

位域	字段名	访问	复位值	描述
7:0	Centigrade temperature	RO	0x0	摄氏温度
63:8		RW	0x0	

12.2 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。需要注意的是，Gate 值的设置应该是与 0x198 寄存器相对应的 16 位数值，而不是摄氏温度；

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 3C6000 内部集成多个温度传感器，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，其基地址为 0x1FE00000：

表 12-3 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	0x1460	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	0x1468	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	0x1470	RW	中断状态寄存器，写 1 清除中断 [0]：高温中断触发 [1]：低温中断触发
高低温中断控制高位 Thsens_int_up	0x1478	RW	[7:0] Hi_gate0 高 8 位 [15:8] Hi_gate1 高 8 位 [23:16] Hi_gate2 高 8 位 [31:24] Hi_gate3 高 8 位 [39:32] Lo_gate0 高 8 位 [47:40] Lo_gate1 高 8 位 [55:48] Lo_gate2 高 8 位 [63:56] Lo_gate3 高 8 位

12.3 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。第 0 组优先级最高，当需要设置多个不同的阈值时，应该按照其阈值从高到低排列。

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：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。3C6000 内部集成多个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入。

FREQ：分频数。当触发分频操作时，使用预设的 FREQ 对时钟进行分频，分频的模式受 freqscale_mode_node 的控制。

其基地址为 0x1FE00000。

表 12-4 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	0x1480	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
Thsens_freq_scale_up	0x1490	RW	温度传感器控制寄存器高位 [7:0] Scale_Hi_gate0 高 8 位 [15:8] Scale_Hi_gate1 高 8 位 [23:16] Scale_Hi_gate2 高 8 位 [31:24] Scale_Hi_gate3 高 8 位

			[39:32] Scale_Lo_gate0 高 8 位 [47:40] Scale_Lo_gate1 高 8 位 [55:48] Scale_Lo_gate2 高 8 位 [63:56] Scale_Lo_gate3 高 8 位

12.4 温度状态检测与控制

引脚 PROCHOTn 和 THERMTRIPn 用于温度状态检测与控制，这两个信号分别与 GPIO14 和 GPIO15 复用。其中 PROCHOTn 既可作为输入也可作为输出，THERMTRIPn 仅有输出功能。

PROCHOTn 作为输入时，芯片受外部温度检测电路的控制，外部温度检测电路需要降低芯片温度时可以置 PROCHOTn 为 0，芯片接收到该低电平后会采取降频措施，降频时的分频值由通过寄存器 prochotn_freq_scale 设置。PROCHOTn 作为输出时，芯片可输出高温中断，通过 prochotn_o_sel 寄存器从高温中断控制寄存器所设置的 4 个中断中选择一个作为对外发出的高温中断。

THERMTRIPn 作为输出，由芯片通过 thermtripn_o_sel 寄存器从高温中断控制寄存器所设置的 4 个中断中选择一个作为对外发出的高温中断。

虽然 THERMTRIPn 和 PROCHOTn 都是对外的高温中断，但是 THERMTRIPn 的紧急程度较 PROCHOTn 更高。PROCHOTn 置位时，外部温度控制电路还可以采取一定的措施，比如提高风扇转速。而 THERMTRIPn 置位时，外部电源控制电路应该直接采取紧急断电措施。

具体的控制寄存器如下：

表 12-5 温度状态检测与控制寄存器说明

寄存器	地址	控制	说明
温度状态检测与控制寄存器 Thsens_hi_ctrl	0x1498	RW	[0:0]：prochotn_oe PROCHOTn 引脚输出使能控制，0 为输出，1 为输入 [5:4]：prochotn_o_sel PROCHOTn 高温中断输出选择 [10:8]：prochotn_freq_scale：PROCHOTn 输入有效时的分频值 [17:16]：thermtripn_o_sel THERMTRIPn 高温中断输出选择

12.5 温度传感器的控制

龙芯 3C6000 内部集成了多个温度传感器，可通过寄存器配置调整监测点配置与监测频率等配置，还可直接观测到每一个温度传感器的输出内容用于调试。共有 2 个传感器，使用“内部结点号 0/1+vtsensor_id 号 0”访问。

基地址为 0x1FE00000，内部结点号的地址计算规则与其它配置寄存器一致，ID 号的计算规则为 0x01580+vtsensor_id<<4。

表 12-6 温度传感器配置寄存器说明

位域	字段名	访问	复位值	描述
0	Thsens_trigger	RW	0	使能温度传感器配置，如置位，可由 thsens_mode 和 thsens_cluster 选择监测模式和监测点；为 0 则默认温度监测模式，且监测点由 temp_cluster 配置。
2	Thsens_mode	RW	0	0：温度模式 1：电压模式
3	Thsens_datarate	RW	0	监测频率： 0 - 10~20Hz 1 - 325~650Hz
6:4	Thsens_cluster	RW	0	传感器监测点配置：0 为本地监测点，1~7 为远程监测点
8	Temp_valid	RW	0	使能温度传感器输出，替换 IOCSR[0x198] 中 Thsens0_out 和 Thsens0_overflow 的值为该温度传感器的温度监测值。
11:9	Temp_cluster	RW	0	温度传感器输出监测点选择，Thsens_trigger 使能时无效

除了监测温度之外，还可以监测电压。电压与温度在每个传感器上只能同时观测一个。读出值的计算方法：

结点温度=data*820/0x4000 - 311（温度范围 -40 度 ~ 125 度）

电压=(data+110)*1.226/0x1000

13 DDR4 SDRAM 控制器配置

龙芯 3C6000 处理器内部集成的内存控制器的设计遵守 DDR4 SDRAM 行业标准（JESD79-4）。

13.1 DDR4 SDRAM 控制器功能概述

龙芯 3C6000 处理器中每个内存控制器最大支持 4 个 CS，一共含有 22 位的地址总线（即：18 位的行列地址总线、2 位逻辑 Bank 总线和 2 位逻辑 Bank Group 总线，其中行列地址总线与 RASn、CASn 和 Wen 复用）。

在具体选择使用不同内存芯片类型时，可以调整 DDR4 控制器参数设置进行支持。其中，支持的最大片选（CS_n）为 4，行地址（ROW）数为 18，列地址（COL）数为 12，逻辑体选择（BANK）数为 2（DDR4），逻辑体组（BANK Group）数为 2。

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 3C6000 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；
- ECC 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 DDR4 SDRAM，且参数配置支持 x4、x8、x16 颗粒；
- 控制器与 PHY 频率比 1/2 ；
- 支持数据传输速率范围为 1600Mbps-3200Mbps。

13.2 DDR4 SDRAM 参数配置格式

13.2.1 内存控制器的参数列表

表 13-1 内存控制器软件可见参数列表

Offset	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
PHY								
0x0000							version(RD)	
0x0008		switch_byte48	x4_mode	ddr3_mode			capability（RD）	
0x0010							dram_init(RD)	init_start
0x0018								
0x0020							preamble2	rdfifo_valid
0x0028		rdfifo_empty(RD)				Overflow(RD)		

0x0030		dll_value(RD)	dll_init_done(RD)	dll_lock_mode	dll_bypass	dll_adj_cnt	dll_increment	dll_start_point
0x0038				dll_dbl_fix			dll_close_disable	dll_ck
0x0040								
0x0048							clken_ckca	
0x0050								
0x0058							clken_ds_0	
0x0060								
0x0068							clken_ds_1	
0x0070								
0x0078							clken_ds_2	
0x0080								
0x0088							clken_ds_3	
0x0090								
0x0098							clken_ds_4	ds4_en
0x00a0								
0x00a8							clken_ds_5	
0x00b0								
0x00b8							clken_ds_6	
0x00c0								
0x00c8							clken_ds_7	
0x00d0								
0x00d8							clken_ds_8	ecc_ds_en
0x00e0								vref_slice_enable
……								
0x0100					dll_1xdly_0	dll_1xgen_0	dll_wrdqs_0	dll_wrdq_0
0x0108		vref_sample_0	vrefclk_inv_0	dll_vref_0	vref_dly_0	dll_gate_0	dll_rddqs1_0	dll_rddqs0_0
0x0110	rdodt_ctrl_0	rdgate_len_0	rdgate_mode_0	rdgate_ctrl_0			dqs_oe_ctrl_0	dq_oe_ctrl_0
0x0118				rd_odt_len_add_0	dly_2x_0		redge_sel_0	rddqs_phase_0(RD)
0x0120	w_bdly0_0[31:28]	w_bdly0_0[27:24]	w_bdly0_0[23:20]	w_bdly0_0[19:16]	w_bdly0_0[15:12]	w_bdly0_0[11:8]	w_bdly0_0[7:4]	w_bdly0_0[3:0]
0x0128		w_bdly0_0[59:56]	w_bdly0_0[55:52]	w_bdly0_0[51:48]	w_bdly0_0[47:44]	w_bdly0_0[43:40]	w_bdly0_0[39:36]	w_bdly0_0[35:32]
0x0130	w_bdly1_0[24:21]	w_bdly1_0[20:18]	w_bdly1_0[17:15]	w_bdly1_0[14:12]	w_bdly1_0[11:9]	w_bdly1_0[8:6]	w_bdly1_0[5:3]	w_bdly1_0[2:0]
0x0138								w_bdly1_0[27:26]
0x0140							rg_bdly_0[7:4]	rg_bdly_0[3:0]
0x0148								
0x0150	rdqsp_bdly_0[31:28]	rdqsp_bdly_0[27:24]	rdqsp_bdly_0[23:20]	rdqsp_bdly_0[19:16]	rdqsp_bdly_0[15:12]	rdqsp_bdly_0[11:8]	rdqsp_bdly_0[7:4]	rdqsp_bdly_0[3:0]
0x0158								rdqsp_bdly_0[35:32]
0x0160	rdqsn_bdly_0[31:28]	rdqsn_bdly_0[27:24]	rdqsn_bdly_0[23:20]	rdqsn_bdly_0[19:16]	rdqsn_bdly_0[15:12]	rdqsn_bdly_0[11:8]	rdqsn_bdly_0[7:4]	rdqsn_bdly_0[3:0]
0x0168								rdqsn_bdly_0[35:32]
0x0170	rdq_bdly_0[24:21]	rdq_bdly_0[20:18]	rdq_bdly_0[17:15]	rdq_bdly_0[14:12]	rdq_bdly_0[11:9]	rdq_bdly_0[8:6]	rdq_bdly_0[5:3]	rdq_bdly_0[2:0]
0x0178								rdq_bdly_0[27:26]
0x0180					dll_1xdly_1	dll_1xgen_1	dll_wrdqs_1	dll_wrdq_1
0x0188		vref_sample_1	vrefclk_inv_1	dll_vref_1	vref_dly_1	dll_gate_1	dll_rddqs1_1	dll_rddqs0_1
0x0190	rdodt_ctrl_1	rdgate_len_1	rdgate_mode_1	rdgate_ctrl_1			dqs_oe_ctrl_1	dq_oe_ctrl_1

0x0198				rd_odt_len_add_1	dly_2x_1		redge_sel_1	rddqs_phase_1(RD)
0x01a0	w_bdly0_1[31:28]	w_bdly0_1[27:24]	w_bdly0_1[23:20]	w_bdly0_1[19:16]	w_bdly0_1[15:12]	w_bdly0_1[11:8]	w_bdly0_1[7:4]	w_bdly0_1[3:0]
0x01a8		w_bdly0_1[59:56]	w_bdly0_1[55:52]	w_bdly0_1[51:48]	w_bdly0_1[47:44]	w_bdly0_1[43:40]	w_bdly0_1[39:36]	w_bdly0_1[35:32]
0x01b0	w_bdly1_1[24:21]	w_bdly1_1[20:18]	w_bdly1_1[17:15]	w_bdly1_1[14:12]	w_bdly1_1[11:9]	w_bdly1_1[8:6]	w_bdly1_1[5:3]	w_bdly1_1[2:0]
0x01b8								w_bdly1_1[27:26]
0x01c0							rg_bdly_1[7:4]	rg_bdly_1[3:0]
0x01c8								
0x01d0	rdqsp_bdly_1[31:28]	rdqsp_bdly_1[27:24]	rdqsp_bdly_1[23:20]	rdqsp_bdly_1[19:16]	rdqsp_bdly_1[15:12]	rdqsp_bdly_1[11:8]	rdqsp_bdly_1[7:4]	rdqsp_bdly_1[3:0]
0x01d8								rdqsp_bdly_1[35:32]
0x01e0	rdqsn_bdly_1[31:28]	rdqsn_bdly_1[27:24]	rdqsn_bdly_1[23:20]	rdqsn_bdly_1[19:16]	rdqsn_bdly_1[15:12]	rdqsn_bdly_1[11:8]	rdqsn_bdly_1[7:4]	rdqsn_bdly_1[3:0]
0x01e8								rdqsn_bdly_1[35:32]
0x01f0	rdq_bdly_1[24:21]	rdq_bdly_1[20:18]	rdq_bdly_1[17:15]	rdq_bdly_1[14:12]	rdq_bdly_1[11:9]	rdq_bdly_1[8:6]	rdq_bdly_1[5:3]	rdq_bdly_1[2:0]
0x01f8								rdq_bdly_1[27:26]
0x0200					dll_lxdly_2	dll_lxgen_2	dll_wrdqs_2	dll_wrdq_2
0x0208		vref_sample_2	vrefclk_inv_2	dll_vref_2	vref_dly_2	dll_gate_2	dll_rddqsl_2	dll_rddqs0_2
0x0210	rdodt_ctrl_2	rdgate_len_2	rdgate_mode_2	rdgate_ctrl_2			dqs_oe_ctrl_2	dq_oe_ctrl_2
0x0218				rd_odt_len_add_2	dly_2x_2		redge_sel_2	rddqs_phase_2(RD)
0x0220	w_bdly0_2[31:28]	w_bdly0_2[27:24]	w_bdly0_2[23:20]	w_bdly0_2[19:16]	w_bdly0_2[15:12]	w_bdly0_2[11:8]	w_bdly0_2[7:4]	w_bdly0_2[3:0]
0x0228		w_bdly0_2[59:56]	w_bdly0_2[55:52]	w_bdly0_2[51:48]	w_bdly0_2[47:44]	w_bdly0_2[43:40]	w_bdly0_2[39:36]	w_bdly0_2[35:32]
0x0230	w_bdly1_2[24:21]	w_bdly1_2[20:18]	w_bdly1_2[17:15]	w_bdly1_2[14:12]	w_bdly1_2[11:9]	w_bdly1_2[8:6]	w_bdly1_2[5:3]	w_bdly1_2[2:0]
0x0238								w_bdly1_2[27:26]
0x0240							rg_bdly_2[7:4]	rg_bdly_2[3:0]
0x0248								
0x0250	rdqsp_bdly_2[31:28]	rdqsp_bdly_2[27:24]	rdqsp_bdly_2[23:20]	rdqsp_bdly_2[19:16]	rdqsp_bdly_2[15:12]	rdqsp_bdly_2[11:8]	rdqsp_bdly_2[7:4]	rdqsp_bdly_2[3:0]
0x0258								rdqsp_bdly_2[35:32]
0x0260	rdqsn_bdly_2[31:28]	rdqsn_bdly_2[27:24]	rdqsn_bdly_2[23:20]	rdqsn_bdly_2[19:16]	rdqsn_bdly_2[15:12]	rdqsn_bdly_2[11:8]	rdqsn_bdly_2[7:4]	rdqsn_bdly_2[3:0]
0x0268								rdqsn_bdly_2[35:32]
0x0270	rdq_bdly_2[24:21]	rdq_bdly_2[20:18]	rdq_bdly_2[17:15]	rdq_bdly_2[14:12]	rdq_bdly_2[11:9]	rdq_bdly_2[8:6]	rdq_bdly_2[5:3]	rdq_bdly_2[2:0]
0x0278								rdq_bdly_2[27:26]
0x0280					dll_lxdly_3	dll_lxgen_3	dll_wrdqs_3	dll_wrdq_3
0x0288		vref_sample_3	vrefclk_inv_3	dll_vref_3	vref_dly_3	dll_gate_3	dll_rddqsl_3	dll_rddqs0_3
0x0290	rdodt_ctrl_3	rdgate_len_3	rdgate_mode_3	rdgate_ctrl_3			dqs_oe_ctrl_3	dq_oe_ctrl_3
0x0298				rd_odt_len_add_3	dly_2x_3		redge_sel_3	rddqs_phase_3(RD)
0x02a0	w_bdly0_3[31:28]	w_bdly0_3[27:24]	w_bdly0_3[23:20]	w_bdly0_3[19:16]	w_bdly0_3[15:12]	w_bdly0_3[11:8]	w_bdly0_3[7:4]	w_bdly0_3[3:0]
0x02a8		w_bdly0_3[59:56]	w_bdly0_3[55:52]	w_bdly0_3[51:48]	w_bdly0_3[47:44]	w_bdly0_3[43:40]	w_bdly0_3[39:36]	w_bdly0_3[35:32]
0x02b0	w_bdly1_3[24:21]	w_bdly1_3[20:18]	w_bdly1_3[17:15]	w_bdly1_3[14:12]	w_bdly1_3[11:9]	w_bdly1_3[8:6]	w_bdly1_3[5:3]	w_bdly1_3[2:0]
0x02b8								w_bdly1_3[27:26]
0x02c0							rg_bdly_3[7:4]	rg_bdly_3[3:0]
0x02c8								
0x02d0	rdqsp_bdly_3[31:28]	rdqsp_bdly_3[27:24]	rdqsp_bdly_3[23:20]	rdqsp_bdly_3[19:16]	rdqsp_bdly_3[15:12]	rdqsp_bdly_3[11:8]	rdqsp_bdly_3[7:4]	rdqsp_bdly_3[3:0]
0x02d8								rdqsp_bdly_3[35:32]
0x02e0	rdqsn_bdly_3[31:28]	rdqsn_bdly_3[27:24]	rdqsn_bdly_3[23:20]	rdqsn_bdly_3[19:16]	rdqsn_bdly_3[15:12]	rdqsn_bdly_3[11:8]	rdqsn_bdly_3[7:4]	rdqsn_bdly_3[3:0]
0x02e8								rdqsn_bdly_3[35:32]

0x02f0	rdq_bdly_3[24:21]	rdq_bdly_3[20:18]	rdq_bdly_3[17:15]	rdq_bdly_3[14:12]	rdq_bdly_3[11:9]	rdq_bdly_3[8:6]	rdq_bdly_3[5:3]	rdq_bdly_3[2:0]
0x02f8								rdq_bdly_3[27:26]
0x0300					dll_lxdly_4	dll_lxgen_4	dll_wrdqs_4	dll_wrdq_4
0x0308		vref_sample_4	vrefclk_inv_4	dll_vref_4	vref_dly_4	dll_gate_4	dll_rddqsl_4	dll_rddqs0_4
0x0310	rdodt_ctrl_4	rdgate_len_4	rdgate_mode_4	rdgate_ctrl_4			dqs_oe_ctrl_4	dq_oe_ctrl_4
0x0318				rd_odt_len_add_4	dly_2x_4		redge_sel_4	rddqs_phase_4(RD)
0x0320	w_bdly0_4[31:28]	w_bdly0_4[27:24]	w_bdly0_4[23:20]	w_bdly0_4[19:16]	w_bdly0_4[15:12]	w_bdly0_4[11:8]	w_bdly0_4[7:4]	w_bdly0_4[3:0]
0x0328		w_bdly0_4[59:56]	w_bdly0_4[55:52]	w_bdly0_4[51:48]	w_bdly0_4[47:44]	w_bdly0_4[43:40]	w_bdly0_4[39:36]	w_bdly0_4[35:32]
0x0330	w_bdly1_4[24:21]	w_bdly1_4[20:18]	w_bdly1_4[17:15]	w_bdly1_4[14:12]	w_bdly1_4[11:9]	w_bdly1_4[8:6]	w_bdly1_4[5:3]	w_bdly1_4[2:0]
0x0338								w_bdly1_4[27:26]
0x0340							rg_bdly_4[7:4]	rg_bdly_4[3:0]
0x0348								
0x0350	rdqsp_bdly_4[31:28]	rdqsp_bdly_4[27:24]	rdqsp_bdly_4[23:20]	rdqsp_bdly_4[19:16]	rdqsp_bdly_4[15:12]	rdqsp_bdly_4[11:8]	rdqsp_bdly_4[7:4]	rdqsp_bdly_4[3:0]
0x0358								rdqsp_bdly_4[35:32]
0x0360	rdqsn_bdly_4[31:28]	rdqsn_bdly_4[27:24]	rdqsn_bdly_4[23:20]	rdqsn_bdly_4[19:16]	rdqsn_bdly_4[15:12]	rdqsn_bdly_4[11:8]	rdqsn_bdly_4[7:4]	rdqsn_bdly_4[3:0]
0x0368								rdqsn_bdly_4[35:32]
0x0370	rdq_bdly_4[24:21]	rdq_bdly_4[20:18]	rdq_bdly_4[17:15]	rdq_bdly_4[14:12]	rdq_bdly_4[11:9]	rdq_bdly_4[8:6]	rdq_bdly_4[5:3]	rdq_bdly_4[2:0]
0x0378								rdq_bdly_4[27:26]
0x0380					dll_lxdly_5	dll_lxgen_5	dll_wrdqs_5	dll_wrdq_5
0x0388		vref_sample_5	vrefclk_inv_5	dll_vref_5	vref_dly_5	dll_gate_5	dll_rddqsl_5	dll_rddqs0_5
0x0390	rdodt_ctrl_5	rdgate_len_5	rdgate_mode_5	rdgate_ctrl_5			dqs_oe_ctrl_5	dq_oe_ctrl_5
0x0398				rd_odt_len_add_5	dly_2x_5		redge_sel_5	rddqs_phase_5(RD)
0x03a0	w_bdly0_5[31:28]	w_bdly0_5[27:24]	w_bdly0_5[23:20]	w_bdly0_5[19:16]	w_bdly0_5[15:12]	w_bdly0_5[11:8]	w_bdly0_5[7:4]	w_bdly0_5[3:0]
0x03a8		w_bdly0_5[59:56]	w_bdly0_5[55:52]	w_bdly0_5[51:48]	w_bdly0_5[47:44]	w_bdly0_5[43:40]	w_bdly0_5[39:36]	w_bdly0_5[35:32]
0x03b0	w_bdly1_5[24:21]	w_bdly1_5[20:18]	w_bdly1_5[17:15]	w_bdly1_5[14:12]	w_bdly1_5[11:9]	w_bdly1_5[8:6]	w_bdly1_5[5:3]	w_bdly1_5[2:0]
0x03b8								w_bdly1_5[27:26]
0x03c0							rg_bdly_5[7:4]	rg_bdly_5[3:0]
0x03c8								
0x03d0	rdqsp_bdly_5[31:28]	rdqsp_bdly_5[27:24]	rdqsp_bdly_5[23:20]	rdqsp_bdly_5[19:16]	rdqsp_bdly_5[15:12]	rdqsp_bdly_5[11:8]	rdqsp_bdly_5[7:4]	rdqsp_bdly_5[3:0]
0x03d8								rdqsp_bdly_5[35:32]
0x03e0	rdqsn_bdly_5[31:28]	rdqsn_bdly_5[27:24]	rdqsn_bdly_5[23:20]	rdqsn_bdly_5[19:16]	rdqsn_bdly_5[15:12]	rdqsn_bdly_5[11:8]	rdqsn_bdly_5[7:4]	rdqsn_bdly_5[3:0]
0x03e8								rdqsn_bdly_5[35:32]
0x03f0	rdq_bdly_5[24:21]	rdq_bdly_5[20:18]	rdq_bdly_5[17:15]	rdq_bdly_5[14:12]	rdq_bdly_5[11:9]	rdq_bdly_5[8:6]	rdq_bdly_5[5:3]	rdq_bdly_5[2:0]
0x03f8								rdq_bdly_5[27:26]
0x0400					dll_lxdly_6	dll_lxgen_6	dll_wrdqs_6	dll_wrdq_6
0x0408		vref_sample_6	vrefclk_inv_6	dll_vref_6	vref_dly_6	dll_gate_6	dll_rddqsl_6	dll_rddqs0_6
0x0410	rdodt_ctrl_6	rdgate_len_6	rdgate_mode_6	rdgate_ctrl_6			dqs_oe_ctrl_6	dq_oe_ctrl_6
0x0418				rd_odt_len_add_6	dly_2x_6		redge_sel_6	rddqs_phase_6(RD)
0x0420	w_bdly0_6[31:28]	w_bdly0_6[27:24]	w_bdly0_6[23:20]	w_bdly0_6[19:16]	w_bdly0_6[15:12]	w_bdly0_6[11:8]	w_bdly0_6[7:4]	w_bdly0_6[3:0]
0x0428		w_bdly0_6[59:56]	w_bdly0_6[55:52]	w_bdly0_6[51:48]	w_bdly0_6[47:44]	w_bdly0_6[43:40]	w_bdly0_6[39:36]	w_bdly0_6[35:32]
0x0430	w_bdly1_6[24:21]	w_bdly1_6[20:18]	w_bdly1_6[17:15]	w_bdly1_6[14:12]	w_bdly1_6[11:9]	w_bdly1_6[8:6]	w_bdly1_6[5:3]	w_bdly1_6[2:0]
0x0438								w_bdly1_6[27:26]
0x0440							rg_bdly_6[7:4]	rg_bdly_6[3:0]

0x0448								
0x0450	rdqsp_bdly_6[31:28]	rdqsp_bdly_6[27:24]	rdqsp_bdly_6[23:20]	rdqsp_bdly_6[19:16]	rdqsp_bdly_6[15:12]	rdqsp_bdly_6[11:8]	rdqsp_bdly_6[7:4]	rdqsp_bdly_6[3:0]
0x0458								rdqsp_bdly_6[35:32]
0x0460	rdqsn_bdly_6[31:28]	rdqsn_bdly_6[27:24]	rdqsn_bdly_6[23:20]	rdqsn_bdly_6[19:16]	rdqsn_bdly_6[15:12]	rdqsn_bdly_6[11:8]	rdqsn_bdly_6[7:4]	rdqsn_bdly_6[3:0]
0x0468								rdqsn_bdly_6[35:32]
0x0470	rdq_bdly_6[24:21]	rdq_bdly_6[20:18]	rdq_bdly_6[17:15]	rdq_bdly_6[14:12]	rdq_bdly_6[11:9]	rdq_bdly_6[8:6]	rdq_bdly_6[5:3]	rdq_bdly_6[2:0]
0x0478								rdq_bdly_6[27:26]
0x0480					dll_1xdly_7	dll_1xgen_7	dll_wrdqs_7	dll_wrdq_7
0x0488		vref_sample_7	vrefclk_inv_7	dll_vref_7	vref_dly_7	dll_gate_7	dll_rddqsl_7	dll_rddqs0_7
0x0490	rdodt_ctrl_7	rdgate_len_7	rdgate_mode_7	rdgate_ctrl_7			dqs_oe_ctrl_7	dq_oe_ctrl_7
0x0498				rd_odt_len_add_7	dly_2x_7		redge_sel_7	rddqs_phase_7(RD)
0x04a0	w_bdly0_7[31:28]	w_bdly0_7[27:24]	w_bdly0_7[23:20]	w_bdly0_7[19:16]	w_bdly0_7[15:12]	w_bdly0_7[11:8]	w_bdly0_7[7:4]	w_bdly0_7[3:0]
0x04a8		w_bdly0_7[59:56]	w_bdly0_7[55:52]	w_bdly0_7[51:48]	w_bdly0_7[47:44]	w_bdly0_7[43:40]	w_bdly0_7[39:36]	w_bdly0_7[35:32]
0x04b0	w_bdly1_7[24:21]	w_bdly1_7[20:18]	w_bdly1_7[17:15]	w_bdly1_7[14:12]	w_bdly1_7[11:9]	w_bdly1_7[8:6]	w_bdly1_7[5:3]	w_bdly1_7[2:0]
0x04b8								w_bdly1_7[27:26]
0x04c0							rg_bdly_7[7:4]	rg_bdly_7[3:0]
0x04c8								
0x04d0	rdqsp_bdly_7[31:28]	rdqsp_bdly_7[27:24]	rdqsp_bdly_7[23:20]	rdqsp_bdly_7[19:16]	rdqsp_bdly_7[15:12]	rdqsp_bdly_7[11:8]	rdqsp_bdly_7[7:4]	rdqsp_bdly_7[3:0]
0x04d8								rdqsp_bdly_7[35:32]
0x04e0	rdqsn_bdly_7[31:28]	rdqsn_bdly_7[27:24]	rdqsn_bdly_7[23:20]	rdqsn_bdly_7[19:16]	rdqsn_bdly_7[15:12]	rdqsn_bdly_7[11:8]	rdqsn_bdly_7[7:4]	rdqsn_bdly_7[3:0]
0x04e8								rdqsn_bdly_7[35:32]
0x04f0	rdq_bdly_7[24:21]	rdq_bdly_7[20:18]	rdq_bdly_7[17:15]	rdq_bdly_7[14:12]	rdq_bdly_7[11:9]	rdq_bdly_7[8:6]	rdq_bdly_7[5:3]	rdq_bdly_7[2:0]
0x04f8								rdq_bdly_7[27:26]
0x0500					dll_1xdly_8	dll_1xgen_8	dll_wrdqs_8	dll_wrdq_8
0x0508		vref_sample_8	vrefclk_inv_8	dll_vref_8	vref_dly_8	dll_gate_8	dll_rddqsl_8	dll_rddqs0_8
0x0510	rdodt_ctrl_8	rdgate_len_8	rdgate_mode_8	rdgate_ctrl_8			dqs_oe_ctrl_8	dq_oe_ctrl_8
0x0518				rd_odt_len_add_8	dly_2x_8		redge_sel_8	rddqs_phase_8(RD)
0x0520	w_bdly0_8[31:28]	w_bdly0_8[27:24]	w_bdly0_8[23:20]	w_bdly0_8[19:16]	w_bdly0_8[15:12]	w_bdly0_8[11:8]	w_bdly0_8[7:4]	w_bdly0_8[3:0]
0x0528		w_bdly0_8[59:56]	w_bdly0_8[55:52]	w_bdly0_8[51:48]	w_bdly0_8[47:44]	w_bdly0_8[43:40]	w_bdly0_8[39:36]	w_bdly0_8[35:32]
0x0530	w_bdly1_8[24:21]	w_bdly1_8[20:18]	w_bdly1_8[17:15]	w_bdly1_8[14:12]	w_bdly1_8[11:9]	w_bdly1_8[8:6]	w_bdly1_8[5:3]	w_bdly1_8[2:0]
0x0538								w_bdly1_8[27:26]
0x0540							rg_bdly_8[7:4]	rg_bdly_8[3:0]
0x0548								
0x0550	rdqsp_bdly_8[31:28]	rdqsp_bdly_8[27:24]	rdqsp_bdly_8[23:20]	rdqsp_bdly_8[19:16]	rdqsp_bdly_8[15:12]	rdqsp_bdly_8[11:8]	rdqsp_bdly_8[7:4]	rdqsp_bdly_8[3:0]
0x0558								rdqsp_bdly_8[35:32]
0x0560	rdqsn_bdly_8[31:28]	rdqsn_bdly_8[27:24]	rdqsn_bdly_8[23:20]	rdqsn_bdly_8[19:16]	rdqsn_bdly_8[15:12]	rdqsn_bdly_8[11:8]	rdqsn_bdly_8[7:4]	rdqsn_bdly_8[3:0]
0x0568								rdqsn_bdly_8[35:32]
0x0570	rdq_bdly_8[24:21]	rdq_bdly_8[20:18]	rdq_bdly_8[17:15]	rdq_bdly_8[14:12]	rdq_bdly_8[11:9]	rdq_bdly_8[8:6]	rdq_bdly_8[5:3]	rdq_bdly_8[2:0]
0x0578								rdq_bdly_8[27:26]
……								
0x0700	ca_train_map	wrdqs_disable	ca_invert	ca_training_mode	leveling_cs	tLVL_DELAY	leveling_req(WR)	leveling_mode
0x0708						mpr_loc	leveling_done(RD)	leveling_ready(RD)
0x0710	leveling_resp_7	leveling_resp_6	leveling_resp_5	leveling_resp_4	leveling_resp_3	leveling_resp_2	leveling_resp_1	leveling_resp_0

0x0718								leveling_resp_8
0x0720								
……								
0x0800	dfe_ctrl_ds	pad_ctrl_ds				pad_ctrl_ck		
0x0808		pad_reset_po	pad_oplen_ca	pad_opdly_ca		pad_ctrl_ca		
0x0810	vref_ctrl_ds_3		vref_ctrl_ds_2		vref_ctrl_ds_1		vref_ctrl_ds_0	
0x0818	vref_ctrl_ds_7		vref_ctrl_ds_6		vref_ctrl_ds_5		vref_ctrl_ds_4	
0x0820							vref_ctrl_ds_8	
0x0828								
0x0830			pad_comp_o(RD)				pad_comp_i	
0x0838								
0x0840	pad_ctrl_ds_3		pad_ctrl_ds_2		pad_ctrl_ds_1		pad_ctrl_ds_0	
0x0848	pad_ctrl_ds_7		pad_ctrl_ds_6		pad_ctrl_ds_5		pad_ctrl_ds_4	
0x0850							pad_ctrl_ds_8	
……								
0x0900		rdedge_soft		rd_phase(RD)			clk_inv	
0x0908							rdedge_inv	
CTL								
0x1000	tMRD	tRP	tWLDQSEN	tMOD	tXPR		tCKE	tRESET
0x1008		tXSFAST	tCKSRX	tCKSRE	tCKOFF	tCKEV	tSTAB	tODTL
0x1010	tREFretention				tRFC		tREF	
0x1018	tCKESR	tXSRD	tXS		tRFC_dlr			tREF_IDLE
0x1020					tRDPDEN	tCPDED	tXPDLL	tXP
0x1028					tZQperiod	tZQCL	tZQCS	tZQ_CMD
……								
0x1040	tRCD	tRRD_S_slr	tRRD_L_slr	tRRD_dlr				tRAS_min
0x1048				tRTP	tWR_CRC_DM	tWR	tFAW_slr	tFAW
0x1050	tWTR_S_CRC_DM	tWTR_L_CRC_DM	tWTR_S	tWTR		tCCD_dlr	tCCD_S_slr	tCCD_L_slr
0x1058								
0x1060			tPHY_WRLAT	tWL		tRDDATA	tPHY_RDLAT	tRL
0x1068				tCAL				tPL
0x1070			tW2P_sameba	tW2W_sameba	tW2R_sameba	tR2P_sameba	tR2W_sameba	tR2R_sameba
0x1078			tW2P_samebg	tW2W_samebg	tW2R_samebg	tR2P_samebg	tR2W_samebg	tR2R_samebg
0x1080			tW2P_samec	tW2W_samec	tW2R_samec	tR2P_samec	tR2W_samec	tR2R_samec
0x1088								
0x1090			tW2P_samecs	tW2W_samecs	tW2R_samecs	tR2P_samecs	tR2W_samecs	tR2R_samecs
0x1098				tW2W_diffcs	tW2R_diffcs		tR2W_diffcs	tR2R_diffcs
……								
0x1100	mirror_dimm_cs_map		cs_ref	cs_resync	cs_zqcl	cs_zq	cs_mrs	cs_enable
0x1108	cke_map				cs_map			
0x1110	mirror_cs_enable	mirror_cs_sta(RO)	mirro_dimm_ctrl	cs2cid				cid_map
0x1118						rcd_cs	two_dimm	rdimm
0x1120	mrs_done(RD)	mrs_req(WR)	pre_all_done(RD)	pre_all_req(WR)	cmd_cmd	status_cmd(RD)	cmd_req(WR)	command_mode

0x1128	cmd_cke	cmd_a			cmd_ba	cmd_bg	cmd_c	cmd_cs
0x1130	cs_mrs_sequence					mpr_rd_done(RO)	cmd_mpr_rd	cmd_pda
0x1138						cmd_dq0		
0x1140	mr_3_cs_0		mr_2_cs_0		mr_1_cs_0		mr_0_cs_0	
0x1148	mr_3_cs_1		mr_2_cs_1		mr_1_cs_1		mr_0_cs_1	
0x1150	mr_3_cs_2		mr_2_cs_2		mr_1_cs_2		mr_0_cs_2	
0x1158	mr_3_cs_3		mr_2_cs_3		mr_1_cs_3		mr_0_cs_3	
0x1160	mr_3_cs_4		mr_2_cs_4		mr_1_cs_4		mr_0_cs_4	
0x1168	mr_3_cs_5		mr_2_cs_5		mr_1_cs_5		mr_0_cs_5	
0x1170	mr_3_cs_6		mr_2_cs_6		mr_1_cs_6		mr_0_cs_6	
0x1178	mr_3_cs_7		mr_2_cs_7		mr_1_cs_7		mr_0_cs_7	
0x1180	mr_3_cs_0_ddr4		mr_2_cs_0_ddr4		mr_1_cs_0_ddr4		mr_0_cs_0_ddr4	
0x1188			mr_6_cs_0_ddr4		mr_5_cs_0_ddr4		mr_4_cs_0_ddr4	
0x1190	mr_3_cs_1_ddr4		mr_2_cs_1_ddr4		mr_1_cs_1_ddr4		mr_0_cs_1_ddr4	
0x1198			mr_6_cs_1_ddr4		mr_5_cs_1_ddr4		mr_4_cs_1_ddr4	
0x11a0	mr_3_cs_2_ddr4		mr_2_cs_2_ddr4		mr_1_cs_2_ddr4		mr_0_cs_2_ddr4	
0x11a8			mr_6_cs_2_ddr4		mr_5_cs_2_ddr4		mr_4_cs_2_ddr4	
0x11b0	mr_3_cs_3_ddr4		mr_2_cs_3_ddr4		mr_1_cs_3_ddr4		mr_0_cs_3_ddr4	
0x11b8			mr_6_cs_3_ddr4		mr_5_cs_3_ddr4		mr_4_cs_3_ddr4	
0x11c0	mr_3_cs_4_ddr4		mr_2_cs_4_ddr4		mr_1_cs_4_ddr4		mr_0_cs_4_ddr4	
0x11c8			mr_6_cs_4_ddr4		mr_5_cs_4_ddr4		mr_4_cs_4_ddr4	
0x11d0	mr_3_cs_5_ddr4		mr_2_cs_5_ddr4		mr_1_cs_5_ddr4		mr_0_cs_5_ddr4	
0x11d8			mr_6_cs_5_ddr4		mr_5_cs_5_ddr4		mr_4_cs_5_ddr4	
0x11e0	mr_3_cs_6_ddr4		mr_2_cs_6_ddr4		mr_1_cs_6_ddr4		mr_0_cs_6_ddr4	
0x11e8			mr_6_cs_6_ddr4		mr_5_cs_6_ddr4		mr_4_cs_6_ddr4	
0x11f0	mr_3_cs_7_ddr4		mr_2_cs_7_ddr4		mr_1_cs_7_ddr4		mr_0_cs_7_ddr4	
0x11f8			mr_6_cs_7_ddr4		mr_5_cs_7_ddr4		mr_4_cs_7_ddr4	
0x1200			nc16_map	nc	channel_width	ba_xor_row_offset	addr_new	cs_place
0x1208						bg_xor_row_offset		addr_mirror
0x1210	addr_base_1				addr_base_0			
0x1218				stb_arprior_cfg				
0x1220	addr_mask_1				addr_mask_0			
0x1228	prior_age7		prior_age6		prior_age5		prior_age4	
0x1230			cs_diff	c_diff	bg_diff	ba_diff	row_diff	col_diff
0x1238				CF_confbus_timeout				
0x1240	WRQthreshold	tRDQidle	wr_pkc_num	rwq_arb	retry	no_dead_inorder	placement_en	stb_en/pbuf
0x1248		WRQ_halfth	WRQthreshold_L	WRQ_near_full	rankarb_age		rankarb_age_en	tRWGNTidle
0x1250	pref_low		tRDQwait	RDQthreshold_L	RDQthreshold	rd_pkg_num	rfifo_age	
0x1258	prior_age3		prior_age2		prior_age1		prior_age0	
0x1260	retry_cnt(RD)					rbuffer_max(RD)	rdfifo_depth	stat_en
0x1268	rdage_low		rdage_up		pref_near_full	hot_en	hot_minus	hot_count
0x1270	stb_page_close_timeout		stb_dbg_cnt_clear	stb_dbg_cnt_sel	stb_dbg_cnt			

0x1278	stb_hot_16	stb_hot_4/8	stb_active_16	stb_active_4/8	stb_distance	stb_context_grant	stb_cold_time_refresh	strdrec_soft_reqb_cold_time_init
0x1280	aw_512_align		rd_before_wr	ecc_enable		int_vector(RD)	int_trigger(RD)	int_enable
0x1288						reread_en	rdrec_interval	rdrec_soft_req
0x1290						int_cnt_fatal(RD)	int_cnt_err(RD)	int_cnt
0x1298	ecc_cnt_cs_7(RD)	ecc_cnt_cs_6(RD)	ecc_cnt_cs_5(RD)	ecc_cnt_cs_4(RD)	ecc_cnt_cs_3(RD)	ecc_cnt_cs_2(RD)	ecc_cnt_cs_1(RD)	ecc_cnt_cs_0(RD)
0x12a0	ecc_data_dir(RD)	ecc_code_dir(RD)		ecc_valid	reread_ecc(RD)	ecc_64[23:16](RD)		ecc_code_64(RD)
0x12a8	ecc_addr(RD)							
0x12b0	ecc_data[63:0](RD)							
0x12b8	ecc_data[127:64] (RD)							
0x12c0	ecc_data[191:128] (RD)							
0x12c8	ecc_data[255:192] (RD)							
0x12d0	ecc_err_set[63:0]							
0c12d8								ecc_err_set[71:64]
……								
0x1300							ref_num	ref_sch_en
0x1308							Status_sref(RD)	srefresh_req
……								
0x1340	hardware_pd_7	hardware_pd_6	hardware_pd_5	hardware_pd_4	hardware_pd_3	hardware_pd_2	hardware_pd_1	hardware_pd_0
0x1348	power_sta_7(RD)	power_sta_6(RD)	power_sta_5(RD)	power_sta_4(RD)	power_sta_3(RD)	power_sta_2(RD)	power_sta_1(RD)	power_sta_0(RD)
0x1350	selfref_age		slowpd_age		fastpd_age		active_age	
0x1358				power_up				Age_step
0x1360	tCONF_IDLE				tLPMC_IDLE			
0x1368					half_frq_auto_intev	half_frq_auto_rw	half_frq_reg_rw	sch_lp_en
0x1370							FORC0A	FORC3x
0x1378								FORC09
0x1380								zq_overlap
0x1388								zq_stat_en
0x1390	zq_cnt_1(RD)				zq_cnt_0(RD)			
0x1398	zq_cnt_3(RD)				zq_cnt_2(RD)			
0x13a0	zq_cnt_5(RD)				zq_cnt_4(RD)			
0x13a8	zq_cnt_6(RD)				zq_cnt_6(RD)			
……								
0x13c0					odt_wr_cs_map			
0x13c8							odt_wr_length	odt_wr_delay
0x13d0					odt_rd_cs_map			
0x13d8							odt_rd_length	odt_rd_delay
……								
0x1400		resync_done	resync_req	tRESYNC_length	tRESYNC_delay	tRESYNC_shift	tRESYNC_max	tRESYNC_min
……								
0x1440					pre_predict		tm_cmdq_num	burst_length
0x1448								ca_timing
0x1450						wr/rd_dbi_en	ca_par_en	crc_en

0x1458							tCA_PAR	tWR_CRC
0x1460	bit_map_7	bit_map_6	bit_map_5	bit_map_6	bit_map_3	bit_map_2	bit_map_1	bit_map_0
0x1468	bit_map_15	bit_map_14	bit_map_13	bit_map_12	bit_map_11	bit_map_10	bit_map_9	bit_map_8
0x1470							bit_map_17	bit_map_16
0x1478								bitmap_mirror
0x1480				alertn_misc(RD)			alertn_cnt	alertn_clr
0x1488	alertn_addr(RD)							
……								
0x1498	mpr_train_ecc							
……								
0x14c0	mpr_train_data_0							
0x14c8	mpr_train_data_1							
0x14d0	mpr_train_data_2							
0x14d8	mpr_train_data_3							
0x14e0	mpr_train_data_4							
0x14e8	mpr_train_data_5							
0x14f0	mpr_train_data_6							
0x14f8	mpr_train_data_7							
0x1500	win0_base							
0x1508	win1_base							
0x1510	win2_base							
0x1518	win3_base							
0x1520	win4_base							
0x1528	win5_base							
0x1530	win6_base							
0x1538	win7_base							
……								
0x1580	win0_mask							
0x1588	win1_mask							
0x1590	win2_mask							
0x1598	win3_mask							
0x15a0	win4_mask							
0x15a8	win5_mask							
0x15b0	win6_mask							
0x15b8	win7_mask							
……								
0x1600	win0_mmap							
0x1608	win1_mmap							
0x1610	win2_mmap							
0x1618	win3_mmap							
0x1620	win4_mmap							
0x1628	win5_mmap							
0x1630	win6_mmap							

0x1638	win7_mmap							
……								
0x1680	write_train_data[63:0]							
0x1688	write_train_data[127:64]							
0x1690	write_train_data[191:128]							
0x1698	write_train_data[255:192]							
0x16a0	write_train_data[319:256]							
0x16a8	write_train_data[383:320]							
0x16b0	write_train_data[447:384]							
0x16b8	write_train_data[511:448]							
0x16c0	Write_train_ecc_data[64:0]							
0x16c8		train_write_n	obj_addr		train_ecc_cmd_len		rw_train_req	rw_train_mode/start
0x16d0	pm_train_base_addr							
……								
0x1700							acc_hp	acc_en
0x1708	acc_fake_b				acc_fake_a			
0x1710								
0x1718								
0x1720	addr_base_acc_1				addr_base_acc_0			
0x1728								
0x1730	addr_mask_acc_1				addr_mask_acc_0			
0x1738								
MON								
0x2000								cmd_monitor
0x2008								
0x2010	cmd_fbck[63:0](RD)							
0x2018	cmd_fbck[127:64] (RD)							
0x2020					rw_switch_cnt(RD)			
……								
0x2100								scheduler_mon
0x2108								
0x2110	sch_cmd_num(RD)							
0x2118	ba_conflict_all(RD)							
0x2120	ba_conflict_last1(RD)							
0x2128	ba_conflict_last2(RD)							
0x2130	ba_conflict_last3(RD)							
0x2138	ba_conflict_last4(RD)							
0x2140	ba_conflict_last5(RD)							
0x2148	ba_conflict_last6(RD)							
0x2150	ba_conflict_last7(RD)							
0x2158	ba_conflict_last8(RD)							
0x2160	rd_conflict(RD)							
0x2168	wr_conflict(RD)							

0x2170	rtw_conflict(RD)							
0x2178	wtr_conflict(RD)							
0x2180	rd_conflict_last1(RD)							
0x2188	wr_conflict_last1(RD)							
0x2190	rtw_conflict_last1(RD)							
0x2198	wtr_conflict_last1(RD)							
0x21a0	wr_rd_turnaround(RD)							
0x21a8	cs_turnaround(RD)							
0x21b0	bg_conflict(RD)							
……								
0x2300						sm_leveling		sm_init
0x2308								
0x2310		sm_rank_03		sm_rank_02		sm_rank_01		sm_rank_00
0x2318		sm_rank_07		sm_rank_06		sm_rank_05		sm_rank_04
0x2320		sm_rank_11		sm_rank_10		sm_rank_09		sm_rank_08
0x2328		sm_rank_15		sm_rank_14		sm_rank_13		sm_rank_12
0x2330		sm_rank_19		sm_rank_18		sm_rank_17		sm_rank_16
0x2338		sm_rank_23		sm_rank_22		sm_rank_21		sm_rank_20
0x2340		sm_rank_27		sm_rank_26		sm_rank_25		sm_rank_24
0x2348		sm_rank_31		sm_rank_30		sm_rank_29		sm_rank_28
……								
TST								
0x3000						lpbk_mode	lpbk_start	lpbk_en
0x3008	lpbk_correct(RD)				lpbk_counter(RD)			lpbk_error(RD)
0x3010	lpbk_data_en[63:0]							
0x3018								lpbk_data_en[71:64]
0x3020							lpbk_data_mask_en	
0x3028								
0x3030	Lpbk_dat_w0[63:0]							
0x3038	Lpbk_dat_w0[127:64]							
0x3040	Lpbk_dat_w1[63:0]							
0x3048	Lpbk_dat_w1[127:64]							
0x3050		lpbk_ecc_mask_w0	lpbk_dat_mask_w0				lpbk_ecc_w0	
0x3058		lpbk_ecc_mask_w1	lpbk_dat_mask_w1				lpbk_ecc_w1	
0x3060								prbs_23
0x3068						prbs_init		
……								
0x3100					fix_data_pattern_index	bus_width	page_size	test_engine_en
0x3108			cs_diff_tst	c_diff_tst	bg_diff_tst	ba_diff_tst	row_diff_tst	col_diff_tst
0x3120	addr_base_tst							
0x3128								
0x3130	user_data_pattern							
0x3138								

0x3140	valid_bits[63:0]							
0x3148								valid_bits[71:64]
0x3150	ctrl[63:0]							
0x3158	ctrl[127:64]							
0x3160	obs[63:0] (RD)							
0x3168	obs[127:64] (RD)							
0x3170	obs[191:128] (RD)							
0x3178	obs[255:192] (RD)							
0x3180	obs[319:256] (RD)							
0x3188	obs[383:320] (RD)							
0x3190	obs[447:384] (RD)							
0x3198	obs[511:448] (RD)							
0x31a0	obs[575:512] (RD)							
0x31a8	obs[639:576] (RD)							
0x31b0					obs[671:640](RD)			
……								
0x3200								
0x3208								
0x3220	tud_i0							
0x3228	tud_i1							
0x3230	tud_o(RD)							
……								
0x3300	tst_300							
0x3308	tst_308							
0x3310	tst_310							
0x3318	tst_318							
0x3320	tst_320							
0x3328	tst_328							
0x3330	tst_330							
0x3338	tst_338							
0x3340	tst_340							
0x3348	tst_348							
0x3350	tst_350							
0x3358	tst_358							
0x3360	tst_360							
0x3368	tst_368							
0x3370	tst_370							
0x3378	tst_378							

13.3 软件编程指南

13.3.1 初始化操作

初始化操作由软件向寄存器 Init_start（0x010）写入 0x2 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。

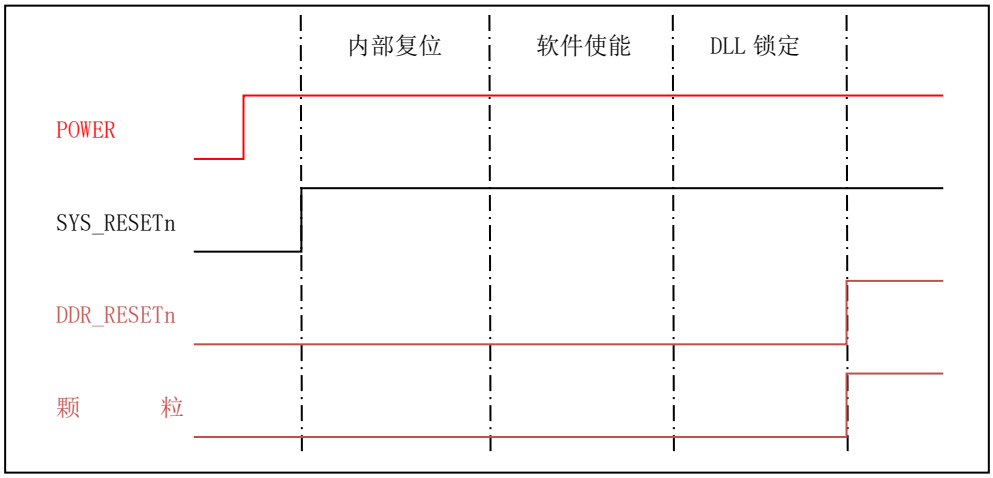
软硬件协同的 DRAM 初始化过程如下：

(1) 设置 pm_clk_sel_ckca 和 pm_clk_sel_ds；
(2) 设置 pm_phy_init_start 为 1，开始初始化 PHY；
(3) 等待 DLL 主控模块锁定，即 pm_dll_init_done 为 1；
(4) 等待所有时钟产生模块的 pm_dll_lock_*或者 pm_pll_lock_*变为 1；
(5) 使能所有的 pm_clken_*；
(6) 将 pm_init_start 设置为 1，内存控制器开始初始化；
(7) 等待内存控制器初始化完成，即 pm_dram_init 的值与 pm_cs_enable 相同。

13.3.2 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 pad_reset_po（0x808）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：

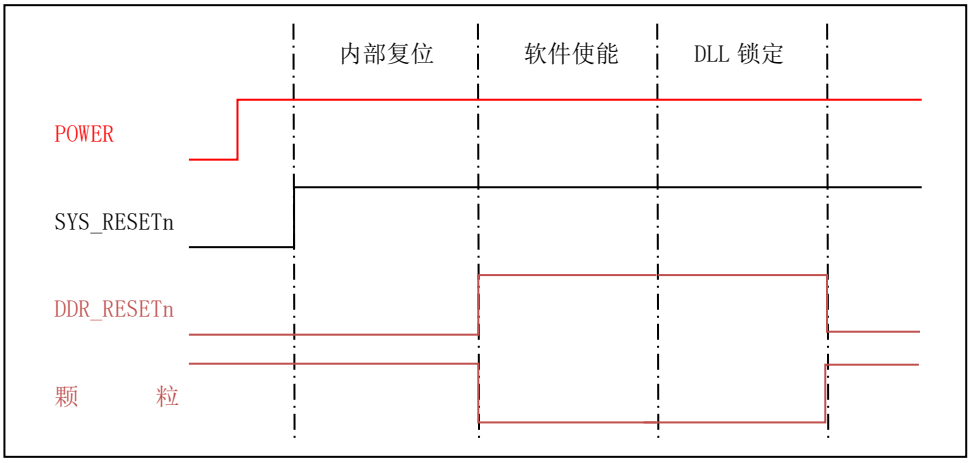
(1) 一般模式，reset_ctrl[1:0] == 2'b00。这种模式下，复位信号引脚的行为与一般的控制模式相兼容。主板上直接将 DDR_RESETn 与内存槽上的对应引脚相连。引脚的行为如下图：



- 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。

(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。引脚的行为如下图：

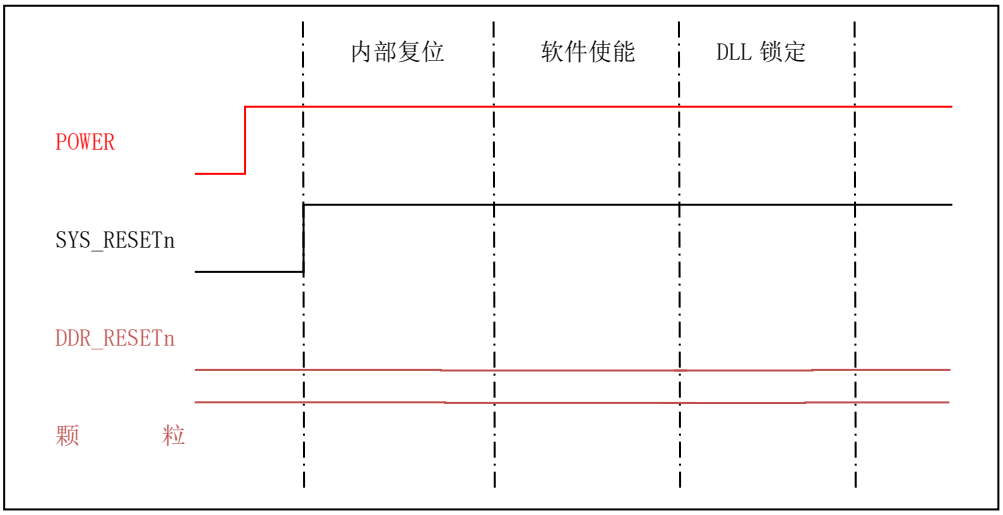


- 未上电时：引脚状态为低；
- 上电时：引脚状态为低；
- 控制器开始配置时：引脚状态为高；
- 控制器开始初始化时：引脚状态为低；
- 正常工作时：引脚状态为低。

(3) 复位禁止模式，pm_pad_reset_o[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。

13.3.3 Leveling

Leveling 操作是在 DDR4 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。此外，本设计还支持 bit-deskew 功能，用于补偿一个 dataslice 内不同 bit 之间的延时差。

13.3.3.1 Write Leveling

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

(1) 完成控制器初始化，参见上一小节内容；
(2) 将 Dll_wrdqs_x（x = 0…8）设置为 0x20；
(3) 将 Dll_wrdq_x（x = 0…8）设置为 0x0；
(4) 设置 Lvl_mode 为 2'b01；
(5) 采样 Lvl_ready 寄存器，如果为 1，表示可以开始 Write Leveling 请求；
(6) 设置 Lvl_req 为 1；
(7) 采样 Lvl_done 寄存器，如果为 1，表示一次 Write Leveling 请求完成；
(8) 采样 Lvl_resp_x 寄存器，如果为 0，则将对应的 Dll_wrdq_x[6:0] 和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 1，然后转向 9；如果为 1，则将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 0，然后继续将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 1，然后转向 9。
(9) 将 Dll_wrdq_x 和 dll_1xdly 的值减 0x40，此时 Dll_wrdq_x 和 dll_1xdly 的值就应该是正确的设置值。
(10) 根据 DIMM 类型设置 pm_dly_2x，对于 0x0 边界右边的颗粒对应的 pm_dly_2x 值增

加 0x010101。

(11)将 Lvl_mode（0x700）设置为 2'b00，退出 Write Leveling 模式。

13.3.3.2 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 为 2'b10；

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

(6) 设置 Lvl_req 为 1；

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

(8) 采样 Lvl_resp_x[0]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)根据 pm_rddqs_phase 的值设置 pm_rdedge_sel

(11)将 Dll_gate_x（x = 0…8）减 0x20；

(12)调整完毕后，再分别进行两次 Lvl_req 操作，观察 Lvl_resp_x[7:5]与 Lvl_resp_x[4:2]的值变化，如果各增加为 Burst_length/2，则继续进行第 13 步操作；如果不为 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；

(13)将 Lvl_mode（0x700）设置为 2'b00，退出 Gate Leveling 模式；

(14)至此 Gate Leveling 操作结束。

13.3.4 功耗控制配置流程

首先需要设置 pm_pad_ctrl_ca[0]为 1，等待内存初始化完成之后，再设置 pm_pad_ctrl_ca[0]为 0。该功能只有 DDR4 模式下使能了 CAL Mode 才可以使用。

13.3.5 单独发起 MRS 命令

对于 DDR4 模式时，内存控制器向内存发出的 MRS 命令次序分别为：

MR3_CS0、MR3_CS1、MR3_CS2、MR3_CS3、MR3_CS4、MR3_CS5、MR3_CS6、MR3_CS7、

MR6_CS0、MR6_CS1、MR6_CS2、MR6_CS3、MR6_CS4、MR6_CS5、MR6_CS6、MR6_CS7、

MR5_CS0、MR5_CS1、MR5_CS2、MR5_CS3、MR5_CS4、MR5_CS5、MR5_CS6、MR5_CS7、

MR4_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR4_CS4、MR4_CS5、MR4_CS6、MR4_CS7、

MR2_CS0、MR2_CS1、MR2_CS2、MR2_CS3、MR2_CS4、MR2_CS5、MR2_CS6、MR2_CS7、

MR1_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR1_CS4、MR1_CS5、MR1_CS6、MR1_CS7、

MR0_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR0_CS4、MR0_CS5、MR0_CS6、MR0_CS7。

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（0x1101）、Mr*_cs*（0x1140 – 0x11f8）设置为正确的值；
(2) 设置 Command_mode（0x0x1120）为 1，使控制器进入命令发送模式；
(3) 采样 Status_cmd（0x1122），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（0x1126）为 1，向 DRAM 发送 MRS 命令；
(5) 采样 Mrs_done（0x1127），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（0x1120）为 0，使控制器退出命令发送模式。

13.3.6 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（0x1128），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（0x1128）设置为正确的值；
(2) 设置 Command_mode（0x1120）为 1，使控制器进入命令发送模式；
(3) 采样 Status_cmd（0x1122），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（0x1121）为 1，向 DRAM 发送命令；
(5) 设置 Command_mode（0x1120）为 0，使控制器退出命令发送模式。

13.3.7 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) 按照初始化流程等待时钟复位稳定；

(3) 将寄存器 Lpbk_en 设为 1；

(4) 将寄存器 Lpbk_start 设为 1；此时自循环测试正式开始。

(5) 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，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。

13.3.8 ECC 功能使用控制

ECC 功能只有在 64 位模式下可以使用。

Ecc_enable（0x1280）包括以下 2 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。

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。

13.3.9 出错状态观测

内存控制器出错后，可通过访问相应的系统配置寄存器来获取相应的出错信息，并进行简单的调试操作。寄存器基地址为 0x1FE00000，同样也可以使用配置寄存器指令进行访问，寄存器及其对应位如下。每个内存控制器的状态观测寄存器位于对应内部结点对应的寄存器空间内，其内部的偏移皆由 0x600 开始，为{0x600 | mc_id<<16}。

表 13-2 0 号内存控制器出错状态观测寄存器

寄存器	偏移地址	控制	说明
0 号内存控制器 ECC 设置寄存器 Mc0_ecc_set	0x0600	RW	0 号内存控制器 ECC 设置寄存器 [5:0]：MC0 int_enable，中断使能 [8]：MC0 int_trigger，中断触发配置 [21:16]：MC0 int_vector(RO)，中断向量（只读） [33:32]：MC0 ecc_enable，ECC 相关功能使能 [40]：MC0 rd_before_wr，读后写功能使能
0 号内存控制器巡检设置寄存器 Mc0_rdrec_soft_reqset	0x0608	RW	[1]：巡检功能使能时，做一次巡检 [15:8]：巡检功能自动使能时的启动时间间隔，单位为 0x200000000000 时钟周期 [16]：重读功能使能
0 号内存控制器 ECC 计数寄存器 Mc0_ecc_cnt	0x0610	RW	0 号内存控制器 ECC 计数寄存器 [7:0]：MC0 int_cnt，配置 ECC 校验触发中断次数阈值 [15:8]：MC0 int_cnt_err(RO)，ECC 校验一位出错次数统计（只读） [23:16]：MC0 int_cnt_fatal(RO)，ECC 校验两位出错次数统计（只读） int_cnt_err 和 int_cnt_fatal 是倒数的计数器，用于触发中断，其起始值是 int_cnt，每次出错减 1，直到为 0。 只有在 int_cnt_xxx 为 0 且有新的出错时，才会触发中断。
0 号内存控制器 ECC 出错次数统计寄存器 Mc0_ecc_cs_cnt	0x0618	RO	0 号内存控制器 ECC 出错次数统计寄存器 [7:0]：MC0 ecc_cnt_cs_0，CS0 出现 ECC 校验错次数统计 [15:8]：MC0 ecc_cnt_cs_1，CS1 出现 ECC 校验错次数统计 [23:16]：MC0 ecc_cnt_cs_2，CS2 出现 ECC 校验错次数统计 [31:24]：MC0 ecc_cnt_cs_3，CS3 出现 ECC 校验错次数统计 [39:32]：MC0 ecc_cnt_cs_4，CS4 出现 ECC 校验错次数统计 [47:40]：MC0 ecc_cnt_cs_5，CS5 出现 ECC 校验错次数统计 [55:48]：MC0 ecc_cnt_cs_6，CS6 出现 ECC 校验错次数统计 [63:56]：MC0 ecc_cnt_cs_7，CS7 出现 ECC 校验错次数统计
0 号内存控制器 ECC 校验码寄存器 Mc0_ecc_code	0x0620	RO	0 号内存控制器 ECC 校验码寄存器 [7:0]：MC0 ecc_code_64，64 位 ECC 校验时的 ECC 校验码，使能内存目录功能时无效 [41:32]：MC0 ecc_code_256，256 位 ECC 校验时的 ECC 校验码，使能内存目录功能时有效

			[52:48]: MC0 ecc_code_dir，内存目录 ECC 校验码，只有使能内存目录功能时有效 [60:56]: MC0 ecc_data_dir，内存目录 ECC 数据，只有使能内存目录功能时有效
0 号内存控制器 ECC 出错地址寄存器 Mc0_ecc_addr	0x0628	RO	0 号内存控制器 ECC 出错地址寄存器 [63:0]: MC0 ecc_addr，ECC 校验出错的地址信息
0 号内存控制器 ECC 出错数据寄存器 0 Mc0_ecc_data0	0x0630	RO	0 号内存控制器 ECC 出错数据寄存器 0 [63:0]:Mc0_ecc_data0，ECC 校验出错时的数据信息，64 位 ECC 模式下的数据，256 位 ECC 模式下的数据[63:0]
0 号内存控制器 ECC 出错数据寄存器 1 Mc0_ecc_data1	0x0638	RO	0 号内存控制器 ECC 出错数据寄存器 1 [63:0]:Mc0_ecc_data1，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[127:64]
0 号内存控制器 ECC 出错数据寄存器 2 Mc0_ecc_data2	0x0640	RO	0 号内存控制器 ECC 出错数据寄存器 2 [63:0]:Mc0_ecc_data2，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[191:128]
0 号内存控制器 ECC 出错数据寄存器 3 Mc0_ecc_data3	0x0648	RO	0 号内存控制器 ECC 出错数据寄存器 3 [63:0]:Mc0_ecc_data3，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[255:192]
0 号内存控制器注错控制 Mc0_ecc_err_set	0x0650	RW	Ecc 注错功能，每一位代表 DDR 总线 64 位数据的每一位
	0x0658	RW	[7:0]:Ecc 注错功能，每一位代表 DDR 总线 ECC 位的每一位

13.3.10 低功耗控制

寄存器基地址为 0x1FE00000，3C6000 每个通道偏移地址为{0x1520 | mc_id<<16}，同样也可以使用配置寄存器指令进行访问，寄存器及其对应位如下。

寄存器	地址	控制	说明
内存控制器调度模块低功耗控制	0x1520	RW	1 为使能低功耗
内存控制器配置寄存器模块低功耗控制	0x1521	RW	1 为使能低功耗
降半频控制	0x1522	RW	[0]: 降半频使能寄存器，1 为使能 [1]: 根据访存空闲自动降半频使能寄存器，1 为使能 [2]: 出错计数溢出时自动降半频使能寄存器，1 为使能
降半频频率选择	0x1523	RW	[1]: 降半频软件控制，0 选择高频，1 选择低频 [7]：降半频切换频率完成标志，该位为只读位
降半频功能关闭完成	0x1524	RO	代表降半频功能关闭完成，1 代表完成（3C6000 没有改寄存器）

自动降频功能需要对低频先初始化，其流程如下：

（1）将 pm_half_frq_reg_rw=1，切换到低频的寄存器读写；

（2）配置完后，再将 0x1fe01522[0]=1；

（3）然后将 0x1fe01523[0]=1；

（4）等待 0x1fe01523[7]；

（5）在低频进行所有读写训练，配置所有参数；

（6）然后将 0x1fe01523[0]=0；

（7）等待 0x1fe01523[7]；

（8）根据需求开启 0x1fe01522[2:1]。

14 PCIE 接口

每个龙芯 3C6000 中，总共集成了 64 位的 PCIE PHY 和 8 个 PCIE 控制器。

其中 64 位 PCIE 分为 4 个独立的 x16 PHY，分别对应芯片接口的 PCIE0/1/2/3。PCIE1/2/3 在不同的情况下可以与 LCL 相复用。

所有控制器和 PHY 分为两组，每组各包含 2 个 x16 PCIE PHY 和 4 个控制器。两组分别为：

（1）PCIE Group0（简称 PCIE G0）：PCIE0/2 PHY 与 PCIE0/2/4/6 控制器

（2）PCIE Group1（简称 PCIE G1）：PCIE1/3 PHY 与 PCIE1/3/5/7 控制器

14.1 控制器复用

每个龙芯 3C6000 集成 4 组 PCIE x16 接口，总共 64 通道，由于存在不同解决方案下进行接口拆分，以连接更多设备的不同需求，总共集成 8 个 PCIE 控制器，按照以下方式进行复用。

其中 PCIE ctrl 0/2/4/6 位于 PCIE Group0，PCIE ctrl 1/3/5/7 位于 PCIE Group1。

表 14-1 PCIE 复用方式

控制器	位置	PHY0	PHY1	PHY2	PHY3
PCIE ctrl0	G0P0	PHY0			
PCIE ctrl1	G1P0		PHY1		
PCIE ctrl2	G0P1	PHY0 4-7		PHY2	
PCIE ctrl3	G1P1		PHY1 4-7		PHY3
PCIE ctrl4	G0P2	PHY0 8-15			
PCIE ctrl5	G1P2		PHY1 8-15		
PCIE ctrl6	G0P3	PHY0 12-15		PHY2 8-15	
PCIE ctrl7	G1P3		PHY1 12-15		PHY3 8-15

通过上述的复用方式，64 个通道在不同的情况下可以拆分为最多 8 个独立的接口，连接最多 8 个不同设备。

复用方式的设置需要在芯片初始化阶段完成，涉及的软件配置寄存器位于 IOCSR[0x1A0]，需要配置的位域如下：

表 14-2 PCIE 配置寄存器

位域	字段名	访问	复位值	描述
3:0	PCIE_G0_enable	RW		PCIE ctrl0/2/4/6 使能控制
7:4	PCIE_G1_enable	RW		PCIE ctrl1/3/5/7 使能控制
33:32	PCIE_PHY0_mode	RW		PCIE PHY0 模式配置 00：1x16 01：2x8 10：4x4 11：1x8 + 2x4
35:34	PCIE_PHY1_mode	RW		PCIE PHY1 模式配置 00：1x16 01：2x8 10：4x4 11：1x8 + 2x4
37:36	PCIE_PHY2_mode	RW		PCIE_PHY2 模式配置 00：1x16 01：2x8
39:38	PCIE_PHY3_mode	RW		PCIE_PHY3 模式配置 00：1x16 01：2x8
51:48	PCIE_G0_shut	RW		PCIE ctrl0/2/4/6 关闭
55:52	PCIE_G1_shut	RW		PCIE ctrl1/3/5/7 关闭

14.2 工作模式

龙芯 3C6000 有多种不同的工作模式。在不同模式下 PCIE 可配置的模式有所不同，以下分别说明。

14.2.1 全 PCIE 模式

LS3C6000/S 单路、LS3C6000/D 单路单连工作模式下，4 个 PCIE 接口全部可用，控制器可使用模式如下。其中标注为“二选一”的只能选用其中一种。

表 14-3 全 PCIE 工作模式

控制器	PHY0 0-3	PHY0 4-7	PHY0 8-11	PHY0 12-15	PHY1 0-3	PHY1 4-7	PHY1 8-11	PHY1 12-15	PHY2 0-7	PHY2 8-15	PHY3 0-7	PHY3 8-15
CTRL 0												
CTRL 1												
CTRL 2												
CTRL 3												
CTRL 4												
CTRL 5												

CTRL 6				二选一						二选一		
CTRL 7								二选一				二选一

由于 PCIE 是按组组织的，两个 PCIE Group 的复用选择方式也相互独立。

表 14-4 全 PCIE 时 PCIE Group0 工作模式

PCIE 接口	第一配置	第二配置
PCIE 0	x8+x8	x8+x4+x4/x8+x8
PCIE 2	x16/x8+x8	x16

表 14-5 全 PCIE 时 PCIE Group1 工作模式

PCIE 接口	第一配置	第二配置
PCIE 1	x16/x8+x8	x16/x8+x4+x4/x8+x8
PCIE 3	x16/x8+x8	x16

14.2.2 LCL1 互连模式

LS3C6000/D 单路双连工作模式下，如果选用 LCL1 作为第二互连通道，则相应硅片上除了 PCIE1 之外的其它的 3 个 PCIE 接口可用，控制器可使用模式如下。其中标注为“二选一”的只能选用其中一种。

表 14-6 LCL1 互连时的 PCIE 复用模式

控制器	PHY0 0-3	PHY0 4-7	PHY0 8-11	PHY0 12-15	PHY1 0-3	PHY1 4-7	PHY1 8-11	PHY1 12-15	PHY2 0-7	PHY2 8-15	PHY3 0-7	PHY3 8-15
CTRL 0												
CTRL 1												
CTRL 2												
CTRL 3												
CTRL 4												
CTRL 5												
CTRL 6				二选一						二选一		
CTRL 7												

由于 PCIE 是按组组织的，每个硅片上两个 PCIE Group 的复用选择方式也相互独立。

表 14-7 LCL1 互连时 PCIE Group0 工作模式

PCIE 接口	第一配置	第二配置
PCIE 0	x8+x8	x8+x4+x4/x8+x8
PCIE 2	x16/x8+x8	x16

表 14-8 LCL1 互连时 PCIE Group1 工作模式

PCIE 接口	第一配置
PCIE 1	-
PCIE 3	x16/x8+x8

14.2.3 LCL2 互连模式

LS3C6000/D 单路双连工作模式下，如果选用 LCL2 作为第二互连通道，则相应硅片上除了 PCIE2 之外的其它的 3 个 PCIE 接口可用，控制器可使用模式如下。其中标注为“二选一”的只能选用其中一种。

表 14-9 LCL2 互连时 PCIE 复用模式

控制器	PHY0 0-3	PHY0 4-7	PHY0 8-11	PHY0 12-15	PHY1 0-3	PHY1 4-7	PHY1 8-11	PHY1 12-15	PHY2 0-7	PHY2 8-15	PHY3 0-7	PHY3 8-15
CTRL 0												
CTRL 1												
CTRL 2												
CTRL 3												
CTRL 4												
CTRL 5												
CTRL 6												
CTRL 7								二选一				二选一

由于 PCIE 是按组组织的，每个硅片上两个 PCIE Group 的复用选择方式也相互独立。

表 14-10 LCL2 互连时 PCIE Group0 工作模式

PCIE 接口	第一配置
PCIE 0	x8+x4+x4/x8+x8
PCIE 2	-

表 14-11 LCL2 互连时 PCIE Group1 工作模式

PCIE 接口	第一配置	第二配置
PCIE 1	x16/x8+x8	x16/x8+x4+x4/x8+x8
PCIE 3	x16/x8+x8	x16

14.2.4 LCL1/2 互连模式

LS3C6000/S 双路、LS3C6000/D 单路三连和双路、LS3C6000/Q 单路工作模式下，PCIE1 与 PCIE2 被设置为 LCL 模式，PCIE0 与 PCIE3 接口可用，控制器可配置模式如下。

表 14-12 LCL1/2 互连 PCIE 复用模式

控制器	PHY0 0-3	PHY0 4-7	PHY0 8-11	PHY0 12-15	PHY1 0-3	PHY1 4-7	PHY1 8-11	PHY1 12-15	PHY2 0-7	PHY2 8-15	PHY3 0-7	PHY3 8-15
CTRL 0												
CTRL 1												
CTRL 2												
CTRL 3												
CTRL 4												
CTRL 5												

CTRL 6												
CTRL 7												

此时两组 PCIE 的可用模式比较确定，如下所示：

表 14-13 LCL1/2 互连 PCIE Group0 工作模式

PCIE 接口	第一配置
PCIE 0	x16/x8+x8/x8+x4+x4/x4+x4+x4+x4
PCIE 2	-

表 14-14 LCL1/2 互连 PCIE Group1 工作模式

PCIE 接口	第一配置
PCIE 1	-
PCIE 3	x16/x8+x8

需要注意的是，将 CHIP_CONFIG[3]上拉可以交换 LCL1 和 LCL3 的使用，或者将 CHIP_CONFIG[2]上拉可以交换 LCL2 和 LCL3 的使用。此时 LCL1 或 LCL2 将与 PCIE3 复用，也就是说 PCIE3 因为作为 LCL1/2 无法使用 PCIE 功能，而 PCIE1/2 则可作为 PCIE 使用。这样做的好处是 PCIE1 的拆分用法更加灵活，或者主板连接更加方便。

需要注意的是，LS3C6000/S 双路默认使用 LCL2（PCIE2）进行互连，LCL1（PCIE1）此时虽然不用也不能作为 PCIE 接口使用。

14.2.5 全互连模式

LS3C6000/D 四路、LS3C6000/Q 双路工作模式下，PCIE1/2/3 都被设置为 LCL 模式，只有 PCIE0 可用，控制器可配置模式如下。

表 14-15 全互连 PCIE 复用模式

控制器	PHY0 0-3	PHY0 4-7	PHY0 8-11	PHY0 12-15	PHY1 0-3	PHY1 4-7	PHY1 8-11	PHY1 12-15	PHY2 0-7	PHY2 8-15	PHY3 0-7	PHY3 8-15
CTRL 0												
CTRL 1												
CTRL 2												
CTRL 3												
CTRL 4												
CTRL 5												
CTRL 6												
CTRL 7												

S 此时只有 PCIE 0 的可用，模式配置如下所示：

表 14-16 全互连 PCIE Group0 工作模式

PCIE 接口	第一配置
PCIE 0	x16/x8+x8/x8+x4+x4/x4+x4+x4+x4
PCIE 2	-

表 14-17 全互连 PCIE Group1 工作模式

PCIE 接口	第一配置
PCIE 1	-
PCIE 3	x16/x8+x8

14.3 PCI 设备树组织结构

龙芯 3C6000 中，每个硅片的 PCIE 控制器被组织在同一个 PCI 设备树型结构中。通过 PCI 扫描的软件流程，能够访问到所有的 PCIE 设备。

除了对应 PCIE 接口的控制器之外，PCIE 树上还有额外的桥接设备（虚拟桥）及地址转换设备（IOMMU）。

此外，龙芯 3C6000 中固定使用 PCIE0 低 8 位连接桥片，在连接桥片时，PCIE0 控制器需要设置为桥片模式，配置位位于 IOCSR[0x1A0]，具体定义如下：

表 14-18 PCIE 配置寄存器

位域	字段名	访问	复位值	描述
14	PCIE0_chipset_mode	RW		PCIE0 控制器桥片连接使能

PCIE0 这两种连接方式决定了 PCI 树的组织形式，不同模式的使用有所不同。

14.3.1 设备连接模式

将 PCIE0 设置为设备连接模式时，PCIE0 连接设备的 P2P 桥模式下，芯片内部的各个设备 ID 固定，如下表所示。

表 14-19 设备连接模式

非桥片模式				位置
BUS 0	DEV0/PCIE 桥 0（3C19）			G0P0
	DEV1/虚拟桥 0（3C09）	BUS N	DEV0/PCIE 桥 2（3C19）	G0P1
			DEV1/PCIE 桥 4（3C29）	G0P2
			DEV2/PCIE 桥 6（3C29）	G0P3
	DEV2/IOMMU 0（3C0F）			
	DEV3/虚拟桥 1（3C09）	BUS M	DEV0/PCIE 桥 1（3C19）	G1P0
			DEV1/PCIE 桥 3（3C19）	G1P1
			DEV2/PCIE 桥 5（3C29）	G1P2
			DEV3/PCIE 桥 7（3C29）	G1P3
			DEV4/IOMMU 1（3C0F）	

14.3.2 桥片连接模式

将 PCIE0 设置为桥片连接模式时，PCIE0 在 PCI 树上不可见，需要使用 0xEFD_FB0x_xxxx 的地址进行其配置头访问，采用 0xEFD_FB1x_xxxx 的地址进行其 BAR 空间的访问。

这个模式下，PCIE0 后的桥片是 0 号总线上的设备，会导致虚拟桥 0、虚拟桥 1 和 IOMMU 0 的 DEV 号进行调整。此时在芯片内部提供一组寄存器，分别用于配置这几个设备号。

其 PCI 树的结构如下表所示。

表 14-20 桥片连接模式

桥片模式				位置
BUS 0	Platform/PCIE 桥 0	BUS 0	桥片设备	G0P0
	DEVx/虚拟桥 0（3C09）	BUS N	DEV0/PCIE 桥 2（3C19）	G0P1
			DEV1/PCIE 桥 4（3C29）	G0P2
			DEV2/PCIE 桥 6（3C29）	G0P3
	DEVy/IOMMU 0（3C0F）			
	DEVz/虚拟桥 1（3C09）	BUS M	DEV0/PCIE 桥 1（3C19）	G1P0
			DEV1/PCIE 桥 3（3C19）	G1P1
			DEV2/PCIE 桥 5（3C29）	G1P2
			DEV3/PCIE 桥 7（3C29）	G1P3
			DEV4/IOMMU 1（3C0F）	

其中 DEVx、DEVy、DEVz 的设备号，在 IOCSR[0x1A0]中进行设置，具体的位置如下。其中 DEVx 对应 V0_dev_num，DEVy 对应 iommu0_dev_num，DEVz 对应 V1_dev_num。

表 14-21 PCIE 配置寄存器

位域	字段名	访问	复位值	描述
20:16	V0_dev_num	RW		PCIE 虚拟桥 0 设备号
25:21	V1_dev_num	RW		PCIE 虚拟桥 1 设备号
30:26	iommu0_dev_num	RW		iommu0 设备号

14.4 地址空间划分

PCI 树以硅片为单位进行组织，每个硅片上的 PCIE 接口总线相互独立，每个硅片上的配置空间、IO 空间、Memory 空间相互独立编址。

龙芯 3C6000 处理器中，默认的 PCIE 接口的地址窗口分布如下：

表 14-22 默认的 PCIE 接口地址

基地址	结束地址	大小	定义
0x0E00_0000_0000	0x0EFF_FFFF_FFFF	1 TB	PCI 地址空间

PCIE 空间为每个硅片上偏移 0x0E00_0000_0000 - 0x0EFF_FFFF_FFFF 的空间内。例如，

硅片 0 上的 PCI 树的基地址为 0x0E00_0000_0000；硅片 3 上的 PCI 树的基地址为 0x3E00_0000_0000。

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 PCIE 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。PCIE 的内部 40 位地址空间其地址窗口分布如下表所示。注意在使用时，应该加上各个结点的基地址（0xnE00_0000_0000）。

表 14-23 PCIE 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 GB	MEM 空间
0xFD_0000_0000	0xFD_0FFF_FFFF	256 MB	中断重映射空间
0xFD_1000_0000	0xFD_F7FF_FFFF	3712 MB	保留
0xFD_F800_0000	0xFD_F8FF_FFFF	16 MB	中断
0xFD_F900_0000	0xFD_FAFF_FFFF	32 MB	保留
0xFD_FB00_0000	0xFD_FBFF_FFFF	16 MB	PCIE0 控制器配置空间
0xFD_FC00_0000	0xFD_FDFF_FFFF	32 MB	I/O 空间
0xFD_FE00_0000	0xFD_FFFF_FFFF	32 MB	PCI 总线配置空间
0xFE_0000_0000	0xFE_1FFF_FFFF	512 MB	PCI 总线扩展配置空间 0
0xFE_2000_0000	0xFE_3FFF_FFFF	512 MB	PCI 总线扩展配置空间 1

14.4.1 配置空间访问

龙芯 3C6000 提供了三种不同的配置空间访问方式供软件使用。这三种空间全部映射到相同的配置空间，只是地址组成方式有所不同，而且标准的配置空间无法访问到扩展的 PCIE 配置空间。

其中标准的配置空间位于 0xFD_FE00_0000 至 0xFD_FFFF_FFFF。该空间可以被默认映射至 0x1A00_0000 至 0x1BFF_FFFF 的 32 位地址空间上。为了能够直接访问 PCIE 的扩展寄存器空间，可以使用另外两个扩展配置空间。

空间		39	28	27	24	23	20	19	16	15	14	12	11	10	8	7	2
配置空间	TYPE0	0xFDFE				0				设备号				功能号		寄存器号	
	TYPE1	0xFDFF				总线号				设备号				功能号		寄存器号	
扩展配置空间 0	TYPE0	0xFE0		扩展寄存器号高位		0				设备号				功能号		扩展寄存器号低位	
	TYPE1	0xFE1		扩展寄存器号高位		总线号				设备号				功能号		扩展寄存器号低位	
扩展配置空间 1	TYPE0	0xFE2		0				设备号		功能号			扩展寄存器号				
	TYPE1	0xFE3		总线号				设备号		功能号			扩展寄存器号				

14.5 配置寄存器

在 PCIE 使用前，需要对 PCIE 进行配置，访问基地址为 0x1FE00000。主要包括以下的

配置寄存器。

14.5.1 PCIE_G0 配置寄存器 0

本组寄存器包含对 PCIE_G0 的控制信号。

地址偏移：3000h

默认值：0000_0000_1919_c000h

位域	名称	访问	描述
63:48	phy2_lane_shut	R/W	从低位到高位分别对应 PHY2 的 lane0～lane15 的关闭控制 0：正常工作模式 1：关闭模式
47:32	phy0_lane_shut	R/W	从低位到高位分别对应 PHY0 的 lane0～lane15 的关闭控制 0：正常工作模式 1：关闭模式
31:16	Reserved	R/W	保留
15	phy2_aux_clk_en	R/W	PHY2 的辅助时钟使能 0：不使用辅助时钟 1：使用辅助时钟
14	phy0_aux_clk_en	R/W	PHY0 的辅助时钟使能 0：不使用辅助时钟 1：使用辅助时钟
13:8	Reserved	R/W	保留
7	axi_awsplit_final_p6	R/W	是否对 PCIE6 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
6	axi_awsplit_final_p4	R/W	是否对 PCIE4 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
5	axi_awsplit_final_p2	R/W	是否对 PCIE2 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
4	axi_awsplit_final_p0	R/W	是否对 PCIE0 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
3:0	Reserved	R/W	保留

14.5.2 PCIE_GROUP0 配置寄存器 1

本组寄存器包含对 PCIE_GROUP0 的控制信号和状态采集信息。

地址偏移：3008h

默认值：0000_0000_0000_00f3h

位域	名称	访问	描述
63	warm_wait_bypass	R/W	PCIE_GROUP0 内的 PCIE 控制器在热复位时是否等待对应 PLL 的锁定状态 0：等待 PLL 锁定 1：不等待 PLL 锁定
62	phy_state_bypass	R/W	PCIE_GROUP0 内的 PCIE 控制器在复位时是否等待对应 PHY 的状态标志 0：等待 PHY 的状态标志 1：不等待 PHY 的状态标志
61	prg_state_bypass	R/W	PCIE_GROUP0 内的 PCIE 控制器在复位时是否等待参考时钟状态 0：等待参考时钟就绪标志 1：不等待参考时钟就绪标志
60	prg_hfc_ready	RO	参考时钟就绪标志 0：未就绪 1：就绪
59:44	PHY2_READY	RO	从低位到高位分别对应 PCIE PHY2 的 lane0～lane15 的 phy_ready 状态
43:28	PHY0_READY	RO	从低位到高位分别对应 PCIE PHY0 的 lane0～lane15 的 phy_ready 状态
27	phy2_cfg_rstn	RO	PCIE PHY2 的复位状态 0：复位有效 1：复位无效
26	phy0_cfg_rstn	RO	PCIE PHY0 的复位状态 0：复位有效 1：复位无效
25	PHY2_HFC_READY_m0	RO	此位为 1 表示 PHY2 的低 8 个 lane 关联的 PLL 时钟已经准备好
24	PHY0_HFC_READY_m0	RO	此位为 1 表示 PHY0 的低 8 个 lane 关联的 PLL 时钟已经准备好
23	phy2_init_cfg_stop	RO	此位为 1 表示 PHY2 的复位后预置初始化配置过程中出现错误
22	phy2_init_cfg_busy	RO	此为 1 表示 PHY2 正在进行其复位后的预置初始化配置
21:18	Reserved	RO	保留
17	phy2_init_cfg_stop	RO	此位为 1 表示 PHY2 的复位后预置初始化配置过程被停止
16	phy2_init_cfg_done	RO	此为 1 表示 PHY2 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 PHY2 进行配置访问
15	phy0_init_cfg_stop	RO	此位为 1 表示 PHY0 的复位后预置初始化配置过程中出现错误
14	phy0_init_cfg_busy	RO	此为 1 表示 PHY0 正在进行其复位后的预置初始化配置
13:10	Reserved	RO	保留
9	phy0_init_cfg_stop	RO	此位为 1 表示 PHY0 的复位后预置初始化配置过程被停止
8	phy0_init_cfg_done	RO	此为 1 表示 PHY0 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 PHY0 进行配置访问
7	link_down_reset_en_p6	R/W	PCIE6 的断链复位使能 当此位为 1 时，PCIE6 如果在完成链路建立后出现断链的情况，将对 PCIE6 产生热复位
6	link_down_reset_en_p4	R/W	PCIE4 的断链复位使能

			当此位为 1 时，PCIE4 如果在完成链路建立后出现断链的情况，将对 PCIE4 产生热复位
5	link_down_reset_en_p2	R/W	PCIE2 的断链复位使能 当此位为 1 时，PCIE2 如果在完成链路建立后出现断链的情况，将对 PCIE2 产生热复位
4	link_down_reset_en_p0	R/W	PCIE0 的断链复位使能 当此位为 1 时，PCIE0 如果在完成链路建立后出现断链的情况，将对 PCIE0 产生热复位
3	phy2_soft_cfg_done	R/W	向此位写入 1，表示软件已完成对 PHY2 的初始化配置 必须在向此位写入 1 后，才能访问使用 PHY2 的 PCIE 控制器
2	phy0_soft_cfg_done	R/W	向此位写入 1，表示软件已完成对 PHY0 的初始化配置 必须在向此位写入 1 后，才能访问使用 PHY0 的 PCIE 控制器
1	phy2_rstn	R/W	PHY2 的软件复位控制 产生 PHY 的软件复位时，需先写入 0，再写入 1 0：复位有效 1：复位无效
0	phy0_rstn	R/W	PHY0 的软件复位控制 产生 PHY 的软件复位时，需先写入 0，再写入 1 0：复位有效 1：复位无效

14.5.3 PCIE_GROUP0 配置寄存器 2

本组寄存器包含对 PCIE_GROP0 的控制和状态采集信息。

地址偏移：3010h

默认值：0000_0000_fff0_0f00h

位域	名称	访问	描述
63	lane0_phystatus_p6	RO	PCIE6 的 lane0 对应的 phystatus 电平
62	rdlh_link_up_p6	RO	PCIE6 的数据链路层 link up 状态标志
61:56	link_state_p6	RO	PCIE6 的链路状态机状态
55	lane0_phystatus_p4	RO	PCIE4 的 lane0 对应的 phystatus 电平
54	rdlh_link_up_p4	RO	PCIE4 的数据链路层 link up 状态标志
53:48	link_state_p4	RO	PCIE4 的链路状态机状态
47	lane0_phystatus_p2	RO	PCIE2 的 lane0 对应的 phystatus 电平
46	rdlh_link_up_p2	RO	PCIE2 的数据链路层 link up 状态标志
45:40	link_state_p2	RO	PCIE2 的链路状态机状态
39	lane0_phystatus_p0	RO	PCIE0 的 lane0 对应的 phystatus 电平
38	rdlh_link_up_p0	RO	PCIE0 的数据链路层 link up 状态标志
37:32	link_state_p0	RO	PCIE0 的链路状态机状态
31	p6_warm_rst_clean	RO	PCIE6 在热复位时的总线状态 0：有未完成请求 1：无未完成请求

30	p4_warm_rst_clean	RO	PCIE4 在热复位时的总线状态 0：有未完成请求 1：无未完成请求
29	p2_warm_rst_clean	RO	PCIE2 在热复位时的总线状态 0：有未完成请求 1：无未完成请求
28	p0_warm_rst_clean	RO	PCIE0 在热复位时的总线状态 0：有未完成请求 1：无未完成请求
27	p6_m_done	RO	PCIE6 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
26	p4_m_done	RO	PCIE4 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
25	p2_m_done	RO	PCIE2 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
24	p0_m_done	RO	PCIE0 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
23	p6_s_done	RO	PCIE6 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
22	p4_s_done	RO	PCIE4 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
21	p2_s_done	RO	PCIE2 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
20	p0_s_done	RO	PCIE0 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
19	force_pcie_cc_p6	R/W	软件强制 PCIE6 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p6 的值设置 DMA 访问的 cache 属性
18	force_pcie_cc_p4	R/W	软件强制 PCIE4 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p4 的值设置 DMA 访问的 cache 属性
17	force_pcie_cc_p2	R/W	软件强制 PCIE2 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p2 的值设置 DMA 访问的 cache 属性
16	force_pcie_cc_p0	R/W	软件强制 PCIE0 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定

			1：使用 conf_pcie_cc_p0 的值设置 DMA 访问的 cache 属性
15	conf_pcie_cc_p6	R/W	PCIE6 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p6 的值为 0x1）的 cache 属性 0：uncached 1：cached
14	conf_pcie_cc_p4	R/W	PCIE4 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p4 的值为 0x1）的 cache 属性 0：uncached 1：cached
13	conf_pcie_cc_p2	R/W	PCIE2 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p2 的值为 0x1）的 cache 属性 0：uncached 1：cached
12	conf_pcie_cc_p0	R/W	PCIE0 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p0 的值为 0x1）的 cache 属性 0：uncached 1：cached
11	p6_axi_prot_en	R/W	PCIE6 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE0 进行系统复位前进行配置
10	p4_axi_prot_en	R/W	PCIE4 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE0 进行系统复位前进行配置
9	p2_axi_prot_en	R/W	PCIE2 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE0 进行系统复位前进行配置
8	p0_axi_prot_en	R/W	PCIE0 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE0 进行系统复位前进行配置
7	p6_soft_rst	R/W	PCIE6 的软件复位控制 0：软件复位撤销 1：软件复位有效
6	p4_soft_rst	R/W	PCIE4 的软件复位控制 0：软件复位撤销 1：软件复位有效
5	p2_soft_rst	R/W	PCIE2 的软件复位控制 0：软件复位撤销 1：软件复位有效
4	p0_soft_rst	R/W	PCIE0 的软件复位控制 0：软件复位撤销 1：软件复位有效
3:0	Reserved	R/W	保留

14.5.4 PCIE_GROUP0 配置寄存器 3

本组寄存器包含对 PCIE_GROP0 的状态采集信息。

地址偏移：3018h

默认值：0000_0000_0f00_0000h

位域	名称	访问	描述
63:32	Reserved	R/W	保留
31	coreclk_ok_p6	RO	PCIE6 的复位状态观测 0：复位进行中 1：复位结束
30	coreclk_ok_p4	RO	PCIE4 的复位状态观测 0：复位进行中 1：复位结束
29	coreclk_ok_p2	RO	PCIE2 的复位状态观测 0：复位进行中 1：复位结束
28	coreclk_ok_p0	RO	PCIE0 的复位状态观测 0：复位进行中 1：复位结束
27:24	lane0_local_pset_index_p0	RO	PCIE0 lane0 TX 使用的 preset_index 值
23:18	lane0_localfs_p0	RO	PCIE0 lane0 TX 的 fullswing 数值
17:0	lane0_tx_coef_p0	RO	PCIE0 lane0 TX 使用的预加重、去加重系数

14.5.5 PCIE_GROUP0 PHY0 配置访问寄存器

本组寄存器用来控制产生 PCIE_GROUP0 PHY0 内部控制寄存器的配置访问操作。

地址偏移：3020h

默认值：0000_0000h

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态

55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

3C6000 中 PCIE PHY 以及 PRG 的内部寄存器均通过以上软件接口（接口定义相同，但是每个 PHY 对应的访问寄存器的地址不同）进行读写配置。具体的配置流程如下：

对于写操作

（1）将 phy_cfg_write 置为 1,要写的数据写入 phy_cfg_data 寄存器,要写的地址写入 phy_cfg_adr 寄存器

（2）将 phy_cfg_trigger 先置 1 再置 0

（3）等待 phy_cfg_done 变为 1

对于读操作

（1）将 phy_cfg_write 置为 0,要读的地址写入 phy_cfg_adr 寄存器

（2）将 phy_cfg_trigger 先置 1 再置 0

（3）等待 phy_cfg_done 变为 1

（4）phy_cfg_data 寄存器中即为从 PHY 中读出的数据

需注意，在配置之前先将 apb_clken 置为 1，且 apb_resetn 置为 0,同时需保证在 APB 配置过程中这两位不变。配置完成后可将 apb_clken 置为 0。

14.5.6 PCIE_GROUP0 PHY2 配置访问寄存器

本组寄存器用来控制产生 PCIE_GROUP0 PHY2 内部控制寄存器的配置访问操作。

地址偏移：3028h

默认值：0000_0000h

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态

55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

14.5.7 PCIE_GROUP1 配置寄存器 0

本组寄存器包含对 PCIE_GROUP1 的控制信号。

地址偏移：3030h

默认值：0000_0000_1919_c000h

位域	名称	访问	描述
63:48	phy3_lane_shut	R/W	从低位到高位分别对应 PHY3 的 lane0～lane15 的关闭控制 0：正常工作模式 1：关闭模式
47:32	phy1_lane_shut	R/W	从低位到高位分别对应 PHY1 的 lane0～lane15 的关闭控制 0：正常工作模式 1：关闭模式
31:16	Reserved	R/W	保留
15	phy3_aux_clk_en	R/W	PHY3 的辅助时钟使能 0：不使用辅助时钟 1：使用辅助时钟
14	phy1_aux_clk_en	R/W	PHY1 的辅助时钟使能 0：不使用辅助时钟 1：使用辅助时钟
13:8	Reserved	R/W	保留
7	axi_awsplit_final_p7	R/W	是否对 PCIE7 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
6	axi_awsplit_final_p5	R/W	是否对 PCIE5 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
5	axi_awsplit_final_p3	R/W	是否对 PCIE3 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
4	axi_awsplit_final_p1	R/W	是否对 PCIE1 内部总线从接口收到的单拍非连续写进行拆分 0：不进行拆分 1：进行拆分
3:0	Reserved	R/W	保留

14.5.8 PCIE_GROUP1 配置寄存器 1

本组寄存器包含对 PCIE_GROUP1 的控制信号和状态采集信息。

地址偏移：3038h

默认值：0000_0000_0000_00f3h

位域	名称	访问	描述
63	warm_wait_bypass	R/W	PCIE_GROUP1 内的 PCIE 控制器在热复位时是否等待对应 PLL 的锁定状态 0：等待 PLL 锁定 1：不等待 PLL 锁定
62	phy_state_bypass	R/W	PCIE_GROUP1 内的 PCIE 控制器在复位时是否等待对应 PHY 的状态标志 0：等待 PHY 的状态标志 1：不等待 PHY 的状态标志
61	prg_state_bypass	R/W	PCIE_GROUP1 内的 PCIE 控制器在复位时是否等待参考时钟状态 0：等待参考时钟就绪标志 1：不等待参考时钟就绪标志
60	prg_hfc_ready	RO	参考时钟就绪标志 0：未就绪 1：就绪
59:44	PHY3_READY	RO	从低位到高位分别对应 PCIE PHY3 的 lane0～lane15 的 phy_ready 状态
43:28	PHY1_READY	RO	从低位到高位分别对应 PCIE PHY1 的 lane0～lane15 的 phy_ready 状态
27	phy3_cfg_rstn	RO	PCIE PHY3 的复位状态 0：复位有效 1：复位无效
26	phy1_cfg_rstn	RO	PCIE PHY1 的复位状态 0：复位有效 1：复位无效
25	PHY3_HFC_READY_m0	RO	此位为 1 表示 PHY3 的低 8 个 lane 关联的 PLL 时钟已经准备好
24	PHY1_HFC_READY_m0	RO	此位为 1 表示 PHY1 的低 8 个 lane 关联的 PLL 时钟已经准备好
23	phy3_init_cfg_stop	RO	此位为 1 表示 PHY3 的复位后预置初始化配置过程中出现错误
22	phy3_init_cfg_busy	RO	此为 1 表示 PHY3 正在进行其复位后的预置初始化配置
21:18	Reserved	RO	保留
17	phy3_init_cfg_stop	RO	此位为 1 表示 PHY3 的复位后预置初始化配置过程被停止
16	phy3_init_cfg_done	RO	此为 1 表示 PHY3 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 PHY3 进行配置访问
15	phy1_init_cfg_stop	RO	此位为 1 表示 PHY1 的复位后预置初始化配置过程中出现错误
14	phy1_init_cfg_busy	RO	此为 1 表示 PHY1 正在进行其复位后的预置初始化配置
13:10	Reserved	RO	保留
9	phy1_init_cfg_stop	RO	此位为 1 表示 PHY1 的复位后预置初始化配置过程被停止
8	phy1_init_cfg_done	RO	此为 1 表示 PHY1 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 PHY1 进行配置访问
7	link_down_reset_en_p7	R/W	PCIE7 的断链复位使能 当此位为 1 时，PCIE7 如果在完成链路建立后出现断链的情况，

			将对 PCIE7 产生热复位
6	link_down_reset_en_p5	R/W	PCIE5 的断链复位使能 当此位为 1 时，PCIE5 如果在完成链路建立后出现断链的情况，将对 PCIE5 产生热复位
5	link_down_reset_en_p3	R/W	PCIE3 的断链复位使能 当此位为 1 时，PCIE3 如果在完成链路建立后出现断链的情况，将对 PCIE3 产生热复位
4	link_down_reset_en_p1	R/W	PCIE1 的断链复位使能 当此位为 1 时，PCIE1 如果在完成链路建立后出现断链的情况，将对 PCIE1 产生热复位
3	phy3_soft_cfg_done	R/W	向此位写入 1，表示软件已完成对 PHY3 的初始化配置 必须在向此位写入 1 后，才能访问使用 PHY3 的 PCIE 控制器
2	phy1_soft_cfg_done	R/W	向此位写入 1，表示软件已完成对 PHY1 的初始化配置 必须在向此位写入 1 后，才能访问使用 PHY1 的 PCIE 控制器
1	phy3_rstn	R/W	PHY3 的软件复位控制 产生 PHY 的软件复位时，需先写入 0，再写入 1 0：复位有效 1：复位无效
0	phy1_rstn	R/W	PHY1 的软件复位控制 产生 PHY 的软件复位时，需先写入 0，再写入 1 0：复位有效 1：复位无效

14.5.9 PCIE_GROUP1 配置寄存器 2

本组寄存器包含对 PCIE_GROUP1 的控制和状态采集信息。

地址偏移：3040h

默认值：0000_0000_fff0_0f00h

位域	名称	访问	描述
63	lane0_phystatus_p7	RO	PCIE7 的 lane0 对应的 phystatus 电平
62	rdlh_link_up_p7	RO	PCIE7 的数据链路层 link up 状态标志
61:56	link_state_p7	RO	PCIE7 的链路状态机状态
55	lane0_phystatus_p5	RO	PCIE5 的 lane0 对应的 phystatus 电平
54	rdlh_link_up_p5	RO	PCIE5 的数据链路层 link up 状态标志
53:48	link_state_p5	RO	PCIE5 的链路状态机状态
47	lane0_phystatus_p3	RO	PCIE3 的 lane0 对应的 phystatus 电平
46	rdlh_link_up_p3	RO	PCIE3 的数据链路层 link up 状态标志
45:40	link_state_p3	RO	PCIE3 的链路状态机状态
39	lane0_phystatus_p1	RO	PCIE1 的 lane0 对应的 phystatus 电平
38	rdlh_link_up_p1	RO	PCIE1 的数据链路层 link up 状态标志
37:32	link_state_p1	RO	PCIE1 的链路状态机状态
31	p7_warm_rst_clean	RO	PCIE7 在热复位时的总线状态

			0：有未完成请求 1：无未完成请求
30	p5_warm_rst_clean	RO	PCIE5 在热复位时的总线状态 0：有未完成请求 1：无未完成请求
29	p3_warm_rst_clean	RO	PCIE3 在热复位时的总线状态 0：有未完成请求 1：无未完成请求
28	p1_warm_rst_clean	RO	PCIE1 在热复位时的总线状态 0：有未完成请求 1：无未完成请求
27	p7_m_done	RO	PCIE7 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
26	p5_m_done	RO	PCIE5 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
25	p3_m_done	RO	PCIE3 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
24	p1_m_done	RO	PCIE1 内部总线主接口请求完成标记 0：有未完成的请求 1：无未完成的请求
23	p7_s_done	RO	PCIE7 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
22	p5_s_done	RO	PCIE5 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
21	p3_s_done	RO	PCIE3 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
20	p1_s_done	RO	PCIE1 内部总线从接口请求完成标记 0：有未完成的请求 1：无未完成的请求
19	force_pcie_cc_p7	R/W	软件强制 PCIE7 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p7 的值设置 DMA 访问的 cache 属性
18	force_pcie_cc_p5	R/W	软件强制 PCIE5 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p5 的值设置 DMA 访问的 cache 属性
17	force_pcie_cc_p3	R/W	软件强制 PCIE3 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p3 的值设置 DMA 访问的 cache 属性

16	force_pcie_cc_p1	R/W	软件强制 PCIE1 的 DMA 访问 cache 属性使能 0：DMA 访问的 cache 属性由 PCIE 包的属性决定 1：使用 conf_pcie_cc_p0 的值设置 DMA 访问的 cache 属性
15	conf_pcie_cc_p7	R/W	PCIE7 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p7 的值为 0x1）的 cache 属性 0：uncached 1：cached
14	conf_pcie_cc_p5	R/W	PCIE5 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p5 的值为 0x1）的 cache 属性 0：uncached 1：cached
13	conf_pcie_cc_p3	R/W	PCIE3 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p3 的值为 0x1）的 cache 属性 0：uncached 1：cached
12	conf_pcie_cc_p1	R/W	PCIE1 在软件强制 DMA 访问 cache 属性时（force_pcie_cc_p0 的值为 0x1）的 cache 属性 0：uncached 1：cached
11	p7_axi_prot_en	R/W	PCIE7 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE7 进行系统复位前进行配置
10	p5_axi_prot_en	R/W	PCIE5 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE5 进行系统复位前进行配置
9	p3_axi_prot_en	R/W	PCIE3 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE3 进行系统复位前进行配置
8	p1_axi_prot_en	R/W	PCIE1 的内部总线保护使能 0：总线保护使能关闭；1：总线保护使能开启； 需要在软件给 PCIE1 进行系统复位前进行配置
7	p7_soft_rst	R/W	PCIE7 的软件复位控制 0：软件复位撤销 1：软件复位有效
6	p5_soft_rst	R/W	PCIE5 的软件复位控制 0：软件复位撤销 1：软件复位有效
5	p3_soft_rst	R/W	PCIE3 的软件复位控制 0：软件复位撤销 1：软件复位有效
4	p1_soft_rst	R/W	PCIE1 的软件复位控制 0：软件复位撤销 1：软件复位有效
3:0	Reserved	R/W	保留

14.5.10 PCIE_GROUP1 配置寄存器 3

本组寄存器包含对 PCIE_GROUP1 的状态采集信息。

地址偏移：3048h

默认值：0000_0000_0f00_0000h

位域	名称	访问	描述
63:32	Reserved	R/W	保留
31	coreclk_ok_p7	RO	PCIE7 的复位状态观测 0：复位进行中 1：复位结束
30	coreclk_ok_p5	RO	PCIE5 的复位状态观测 0：复位进行中 1：复位结束
29	coreclk_ok_p3	RO	PCIE3 的复位状态观测 0：复位进行中 1：复位结束
28	coreclk_ok_p1	RO	PCIE1 的复位状态观测 0：复位进行中 1：复位结束
27:24	lane0_local_pset_index_p1	RO	PCIE1 lane0 TX 使用的 preset_index 值
23:18	lane0_localfs_p1	RO	PCIE1 lane0 TX 的 fullswing 数值
17:0	lane0_tx_coef_p1	RO	PCIE1 lane0 TX 使用的预加重、去加重系数

14.5.11 PCIE_GROUP1 PHY1 配置访问寄存器

本组寄存器用来控制产生 PCIE_GROUP1 PHY1 内部控制寄存器的配置访问操作。

地址偏移：3050h

默认值：0000_0000h

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态

55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

14.5.12 PCIE_GROUP1 PHY3 配置访问寄存器

本组寄存器用来控制产生 PCIE_GROUP1 PHY3 内部控制寄存器的配置访问操作。

地址偏移：3058h

默认值：0000_0000h

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态
55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

14.6 PCI 配置空间

下表为 PCI 桥配置空间分布，相关寄存器为协议标准寄存器，可参考相关的 PCIE 协议手册。

表 14-24 PCI 标准配置空间结构

起始地址偏移	区域
0x00	PCI-Compatible Header
0x40	PCI Power Management
0x50	Message Signaled Interrupt(MSI)
0x70	PCI Express Capabilities
0xd0	VPD
0x100	AER
0x140	SPCIE

0x180	G4PCIE
0x1b0	LMR
0x200	DL feature

下表列出 PCIE 端口的基础配置头缺省值，不同端口的 Device ID 可能不同，其他字段都相同。

表 14-25 PCIE 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	-	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
06h-07h	PCISTS	PCI Status	0010h	RO
08h	RID	Revision ID	01h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	04h	RO
0Bh	BCC	Base Class Code	06h	RO
0Ch	CLS	Cache Line Size	00h	RO
0Dh	PLT	Primary Latency Timer	00h	RO
0Eh	HEADTYP	Header Type	01h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W, RO
18h	PBNUM	Primary Bus Number	00h	R/W
19h	SBNUM	Secondary Bus Number	00h	R/W
1Ah	SuBNUM	Subordinate Bus Number	00h	R/W
1Bh	SLT	Secondary Latency Timer	00h	RO
1Ch	IOBASE	I/O Base	01h	R/W
1Dh	IOLMT	I/O Limit	01h	R/W
1Eh-1Fh	SSTS	Secondary Status	0000h	RO
20h-21h	MBASE	Memory Base	0000h	R/W
22h-23h	MLMT	Memory Limit	0000h	R/W
25h-24h	PMBASE	Prefetchable Memory Base	0000h	R/W
27h-26h	PMLMT	Prefetchable Memory Limit	0000h	R/W
28h-2Bh	PMBU32	Prefetchable Memory Base Upper 32 Bits	00000000h	R/W
2Ch-2Fh	PMLU32	Prefetchable Memory Limit Upper 32 Bits	00000000h	R/W
30h-31h	IOBU	I/O Base Upper 16 Bits	0000h	R/W
32h-33h	IOLMTU	I/O Limit Upper 16 Bits	0000h	R/W
34h	CAPP	Capabilities Pointer	40h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	01h	RO
3Eh-3Fh	BCTRL	Bridge Control Register	0000h	R/W

注：表中未列出的地址空间表示保留。

14.7 PCIE 控制器内部寄存器

PCIE 控制器的内部寄存器用于控制 PCIE 控制器的行为和特性，在 PCIE 配置头中进行地址设置。

该地址空间为 MEM 类型，64 位地址空间，大小为 4KB，基地址等于 64 位 BAR0 的值，该值在初始化时由 PCI 扫描软件分配。

14.7.1 PCIE 端口控制寄存器 0

偏移地址：0x0

默认值：0x40ff0044

位域	复位值	属性	名字	描述
0	0	R/W	Rx_lane_flip_en	PCIE接收线反转
1	0	R/W	Tx_lane_flip_en	PCIE发送线反转
2	1	R/W	Sys_aux_pwr_det	指示拥有辅助电源（Vaux）
3	0	R/W	App_ltssm_enable	PCIE端口链路建立使能
11:4	0x4	R/W	Reserved	Should be set to 0x0 after reset
12	0	R/W	App_req_enter_L1	要求PCIE链路进入L1
13	0	R/W	App_ready_enter_L23	已经准备好让PCIE链路进入L23
14	0	R/W	App_req_exit_L1	要求PCIE端口退出L1
15	0	R/W	Soft_reset_en	软复位使能
23:16	0xff	R/W	Reserved	保留
24	0	R/W	at_en	PCIE端口外部DMA访问地址转换使能
25	0	R/W	bus_error_en	PCIE读返回总线错误的处理方式 0：读数据返回全1，不产生总线错例外 1：产生总线错例外 当PCIE 控制器完成总线扫描和初始化后，此位可以修改为1
29：26	0	R/W	Reserved	保留
30	0x1	R/W	error response select	0：出现错误时，读数据返回全0 1：出现错误时，读数据返回全1
31	0x0	RO	Reserved	保留

14.7.2 PCIE 端口控制寄存器 1

通过对相应位写入 1 产生一个时钟周期的脉冲，控制 PCIE 接口进行相应的操作。写此寄存器时，一次仅能有 1 位被置 1。

偏移地址：0x4

默认值：0x0

位域	复位值	属性	名字	描述
0	0	R/W1P	App_unlock_msg	在PCIE端口上发送解锁消息
1	0	R/W1P	Apps_pm_xmt _turnoff	在PCIE端口上发送PM_Turn_Off消息
2	0	R/W1P	App_init_rst	在PCIE端口上发送热复位消息
3	0	R/W1P	Soft_reset	对PCIE端口进行软复位 仅在PCIE端口控制寄存器0的bit15被置1时产生作用
4	0	R/W1P	Apps_pm_xmt_pme	将PCIE 端口从D1 或者D2 或者D3 状态中唤醒，并发送PM _PME消息
31:5	0x0	RO	Reserved	保留

14.7.3 PCIE 端口状态寄存器 0

偏移地址：0x8

默认值：0x20f

位域	名字	描述
1:0	Cfg_pwr_ind	系统电源状态指示 00b: Reserved 01b: On 10b: Blink 11b: Off
3:2	Cfg_atten_ind	Attention 按钮状态指示 00b: Reserved 01b: On 10b: Blink 11b: Off
4	Cfg_pwr_ctrl	系统电源控制器状态 0: Power On 1: Power Off
5	Pm_xtlh_block_tlp	禁止发出请求指示
6	Cfg_bus_master_en	PCI主设备使能 1: enabled 0: disabled
7	Cfg_mem_space_en	内存映射访问使能 1: enabled 0: disabled
10:8	Cfg_max_rd_req_size	最大读请求大小
13:11	Cfg_max_payload_size	最大数据负载
14	Cfg_rcb	RCB位
15	Rdlh_link_up	数据链路层状态 1: Link is up 0: Link is down
18:16	Pm_curnt_state	当前电源状态

23:19	Cfg_aer_int_msg_num	根错误状态寄存器的31:21位
28:24	Cfg_pcie_cap_int_msg_num	PCIE 能力寄存器13:9位
29	rfc_update0	此位为1表示rfc_data0有更新的流控令牌包的内容。 rfc_update0、rfc_data0被用作观察流控令牌的发放
30	rfc_update1	此位为1表示rfc_data1有更新的流控令牌包的内容。 rfc_update1、rfc_data1被用作观察流控令牌的发放
31	Reserved	保留

14.7.4 PCIE 端口状态寄存器 1

偏移地址：0xc

默认值：0x01000000

位域	名字	描述
5:0	Xmlh_link_state	物理层链路状态机的当前状态
6	Xmlh_link_up	物理链路状态指示 1: Up 0: Down
7	rdlh_link_up	数据链路状态指示 1：Up 0：Down
8	Cfg_eml_control	Electromechanical interlock control. The state of the Electromechanical interlock control bit in the Slot Control register.
16:9	Cfg_pbus_num	设备所属总线号
21:17	Cfg_pbus_dev_num	设备号
24:22	Pm_dstate	电源管理状态指示 000b: D0 001b: D1 010b: D2 011b: D3 100b: Uninitialized Other values: Not applicable
25	Pm_status	电源管理状态位
26	Pm_pme_en	PME使能指示
27	Aux_pm_en	辅助电源使能指示
28	Cfg_link_auto_bw_int	链路带宽状态寄存器更新指示
29	Cfg_bw_mgt_int	链路带宽管理状态寄存器更新指示
30	Reserved	保留
31	cfg_link_eq_req_int	链路均衡训练请求指示 此位为1 表示在暂时无法做链路均衡参数训练的状态下收到了链路均衡训练的请求

14.7.5 用户定义消息 ID 寄存器

偏移地址：0x10

位域	名字	属性	描述
15:0	radm_msg_req_id	RO	访问radm_msg_index 指定的接收用户定义消息组所存储的用户定义消息的ID
17:16	radm_msg_index	RW	访问的接收用户定义消息组的编号 X8控制器允许使用0h、1h X16控制器允许使用0h ~ 3h
31:18	Reserved	RO	保留

14.7.6 流控观察寄存器 0

偏移地址：0x14

位域	名字	属性	描述
31:0	rfc_data1	RO	记录最近一次同一个符号时间里收到2 个流控包时的第二个流控包的内容

14.7.7 PCIE 端口中断状态寄存器

偏移地址：0x18

位域	名字	描述
0	aer_rc_err_int	当RC产生或收到错误消息并且Root Error Command寄存器中对应的错误报告使能被打开时置位。
1	aer_rc_err_msi	当RC产生或收到错误消息并且MSI被使能且Root Error Command寄存器中对应的错误报告使能被打开时置位。
2	sys_err_rc	此位报告检测到系统错误。当Root Control寄存器的对应位使能被打开并且PCIE 总线上如何设备产生：可修复错误、致命错误、非致命错误时，或者RC产生内部错误时置位。
3	pme_int	收到PME中断。当下列条件满足时，此位置位： 1. Command 寄存器中INTx未被禁止； 2. Root Control寄存器中PME中断使能位被打开； 3. 收到PME消息（Root Status寄存器中PME状态位被置位）。
4	pme_msi	收到PME中断。当下列条件满足时，此位置位： 1. Command 寄存器中INTx被禁止,MSI被使能； 2. Root Control寄存器中PME中断使能位被打开； 3. 收到PME消息（Root Status寄存器中PME状态位被置位）。
5	vendor_msg0	用户定义消息组0收到用户定义消息
6	rc_core_wake	从电源管理中唤醒
7	INTA	INTA中断
8	INTB	INTB中断
9	INTC	INTC中断

10	INTD	INTD中断
11	radm_correctable_err	此PCIE 端口收到可修复错误消息
12	radm_nonfatal_err	此PCIE端口收到非致命错误消息
13	radm_fatal_err	此PCIE端口收到致命错误消息
14	pm_pme	收到PM_PME消息
15	pm_to_ack	收到PM_TO_Ack消息
16	hp_pme	当下列条件满足时，此位置位： 1. PCI电源管理控制和状态寄存器中PME使能被打开； 2. Slot Status寄存器中的任意1位由0变1并且对应位的通知使能被打开。
17	hp_int	当下列条件满足时，此位置位： 1. Command 寄存器中INTx未被禁止； 2. Slot Control寄存器中热插拔中断使能被打开； 3. Slot Status寄存器中的任意1位为1，并且对应位的通知使能被打开。
18	hp_msi	当下列条件满足时，此位置位： 1. MSI使能被打开； 2. Slot Control寄存器中热插拔中断使能被打开； 3. Slot Status寄存器中的任意1位从0变1，并且对应位的通知使能被打开。
19	link_auto_bw_int	当链路的带宽状态寄存器被更新并且链路的带宽变化中断被使能时置位。
20	bw_mgt_int	当链路的带宽状态寄存器被更新并且链路的带宽中断被使能时置位。
21	gm_composer_lookup_err	此位被置位表示此PCIE 端口在向外发送数据响应时溢出。这通常表示此PCIE 端口接收端收到的Non-Posted 请求的数量超出了端口流控限制。
22	radmx_composer_lookup_err	此位被置位表示此PCIE 端口接收送数据响应时溢出。这通常表示此PCIE端口发送端发送的Non-Posted请求的数量超出了端口流控限制。
23	phy_int	PHY中断
24	cds_wait_timeout	在数据链路层link up状态下，如果有TLP待发送，且待发送的TLP等待令牌的时间超过设置的时间阈值
25	ltssm_l2_to_detect	LTSSM状态从L2状态退出到设备检测状态
26	pm_turn_off	收到PM_Turn_Off消息
27	Link_req_rst_not_fall	PCIE端口物理链路断裂
28	Link_eq_req_int	此位被置位表面链路重新进行了gen3的equalization参数训练
29	vendor_msg1	用户定义消息组1收到用户定义消息
30	vendor_msg2	用户定义消息组2收到用户定义消息 X8控制器不需要关心此位
31	vendor_msg3	用户定义消息组3收到用户定义消息 X8控制器不需要关心此位

14.7.8 PCIE 端口中断状态清除寄存器

偏移地址：0x1c

R/W1C

在某一位写入 1 则清除 PCIE 端口中断状态寄存器的对应位。

14.7.9 PCIE 端口中断掩码寄存器

偏移地址：0x20

R/W

PCIE 端口中断状态寄存器对应的中断掩码。对应位置为 1 并且中断状态寄存器的相应位也为 1 时，此 PCIE 端口产生中断。

14.7.10 PCIE 端口控制寄存器 2

偏移地址：0x24

默认值： 0x481118

位域	名字	属性	描述
0	addr_trans_bypass	RW	此位为1表示旁路控制器内部的地址转换功能
1	cfg0_busnum_is_zero	RW	此位为1将发送的TYPE0的配置访问中的BUS NUM强制设置为0
2	Header_ro_wen	RW	此位为1，将是header中的只读位变为可写
3	link_down_prot_en	RW	此位为1 时，当链路断开时，由总线保护逻辑接收并处理对外部设备的访问，以防止出现内部总线卡死的情况
4	tlp_extract_protect	RW	此位为1 时，在进入L1 状态的过程中不会看到TLP 的情况下停止对TLP的提取
18:5	Reserved	RW	保留
19	rxstandby_en	RW	此位为1 时，PCIE 控制器在切换速率时会将PIPE 接口上的rxstandby信号置为有效
23:20	rx data queue error mask	RW	从高位到低位分别对par_er 、ecrc_err 、tlp_ablrt 、dllp_abort错误进行屏蔽
24	int_mode0_en	RW	中断路由到中断控制器
25	int_mode1_en	RW	中断路由到IR
31:26	int_offset	RW	新中断地址[27:4] = 旧中断地址[27:4] + {msi data[16:0] << offset

14.7.11 PCIE 端口控制和状态寄存器

偏移地址：0x28

位域	名字	属性	描述
25:0			Reserved
26	max_lane_mode	RO	PCIE端口在最大链路宽度模式工作
27	Is_rc	RO	PCIE端口在RC模式工作
28	L0s	RO	PCIE端口处于L0s功耗状态
29	L1	RO	PCIE端口处于L1功耗状态
30	L2	RO	PCIE端口处于L2功耗状态
31	L2_exit	RO	PCIE端口退出L2功耗状态

14.7.12 用户定制消息发送寄存器 0

偏移地址：0x38

此寄存器用于在发送用户定制 PCIE 消息前存储 PCIE 传输层协议消息包头的头 4 个字节的信息。

位域	名字	描述
1:0	ven_msg_fmt	PCIE协议中传输层协议包头中的Fmt域
6:2	ven_msg_type	PCIE协议中传输层协议包头中的Type域
9:7	ven_msg_tc	PCIE协议中传输层协议包头中的TC域
10	ven_msg_td	PCIE协议中传输层协议包头中的TD域
11	ven_msg_ep	PCIE协议中传输层协议包头中的EP域
13:12	ven_msg_attr	PCIE协议中传输层协议包头中的Attr域
23:14	ven_msg_len	PCIE协议中传输层协议包头中的Length域
31:24	Reserved	保留

14.7.13 用户定制消息发送寄存器 1

此寄存器用于在发送用户定制 PCIE 消息前存储 PCIE 传输层协议消息包头的第 4 到第 7 字节的消息。

偏移地址：0x3c

位域	名字	描述
2:0	ven_msg_fun_num	PCIE协议中传输层协议包头中的Function Number域
10:3	ven_msg_tag	PCIE协议中传输层协议包头中的Tag域
18:11	ven_msg_code	PCIE协议中传输层协议包头中的Message Code域
22:19		Reserved
23	ven_msg_req_valid	此位置1　表示当前用户定制消息的所有参数均已经配置完毕，要求PCIE端口发送此用户定制消息。消息发送完毕后，此位自动清0。
31:24	Reserved	保留

14.7.14 用户定制消息发送寄存器 2

偏移地址：0x40

用于存储待发送用户定制消息所携带数据的低 32 位。

14.7.15 用户定制消息发送寄存器 3

偏移地址：0x44

用于存储待发送用户定制消息所携带数据的高 32 位。

14.7.16 流控观察寄存器 1

偏移地址：0x48

位域	名字	属性	描述
31:0	rfc_data0	RO	记录最近一次同一个符号时间里收到的第一个流控包的内容

14.7.17 MSI 消息发送寄存器

偏移地址：0x5c

用于存储要发送的 MSI 消息的包头参数和触发 MSI 消息的发送。仅在 EP 模式下使用。在发送 MSI 消息前，需要系统软件按照 PCIE 协议为控制器配置 MSI 消息使用的地址并将 EP 的中断方式设置为 MSI 方式。

位域	名字	描述
4:0	ven_msi_vector	PCI协议MSI包中的Vector域
7:5	ven_msg_tc	PCIE协议中传输层协议包头中的TC域
10:8	ven_msg_func_num	PCIE协议中传输层协议包头中的Function Number域
11	ven_msi_valid	此位置1 表示当前MSI 消息的所有参数均已经配置完毕，要求PCIE 端口发送此MSI消息。消息发送完毕后，此位自动清0。
31:12		Reserved

14.7.18 地址译码掩码寄存器 0（addr_mask_l）

偏移地址：0x68

用于存储此 PCIE 端口外部 DMA 访问地址转换的地址掩码的低 32 位。

14.7.19 地址译码掩码寄存器 1（addr_mask_h）

偏移地址：0x6c

R/W

用于存储此 PCIE 端口外部 DMA 访问地址转换的地址掩码的高 32 位。

addr_mask[51:0] = {addr_mask_h[19:0],addr_mask_l[31:0]}

当 at_en 为 1 时，addr_mask 决定来自设备的外部 PCIE 地址的 63～12 位中哪些位需要进行替换。

14.7.20 地址译码转换地址寄存器 0（addr_tran_l）

偏移地址：0x70

R/W

用于存储此 PCIE 端口外部 DMA 访问地址转换的转换地址的低 32 位。

14.7.21 地址译码转换地址寄存器 1（addr_tran_h）

偏移地址：0x74

R/W

用于存储此 PCIE 端口外部 DMA 访问地址转换的转换地址的高 32 位。

addr_tran[51:0] = {addr_tran_h[19:0],addr_tran_l[31:0]}

addr_tran 在 at_en 为 1 时，用来设定来自设备的 PCIE 地址的 63～12 位中需要被替换的值。

14.7.22 接收用户定义消息数据负载读取端口 0

偏移地址：0x78

R/W

用于读取 radm_msg_index 所指定的接收组所存储的用户定义消息数据的低 32 位。

14.7.23 接收用户定义消息数据负载读取端口 1

偏移地址：0x7c

R/W

用于读取 radm_msg_index 所指定的接收组存储的用户定义消息数据的高 32 位。

14.7.24 扩展中断目标地址低位

偏移地址：0x80

位域	名字	描述
0	extioi_addr_trans_en	扩展中断转换使能
31:1	extioi_addr_trans[31:1]	扩展中断命中后转换地址的[31:1]

14.7.25 扩展中断目标地址高位

偏移地址：0x84

位域	名字	描述
31:0	extioi_addr_trans[63:32]	扩展中断命中后转换地址的[63:32]

14.7.26 中断重映射地址空间配置

偏移地址：0x88

位域	名字	描述
0	ir_addr32_en	中断重映射 32 位地址使能
19:0	Rsv	保留
31:20	ir_addr32	中断重映射 32 位地址空间高位[31:20]，相同时命中

14.7.27 PHY 参数观测寄存器 0

偏移地址：0xa0

RO

用于观测由 lane_index 选中 lane 的参数设置。

位域	名字	描述
7:0	ltx_c-1	选中lane本地的TX C-1系数
15:8	ltx_c0	选中lane本地的TX C0系数
23:16	ltx_c+1	选中lane本地的TX C+1系数
27:24	ltx_pset	选中的lane本地的TX preset预设值
31:28	lrx_rhint	选中lane本地的rx preset hint值

14.7.28 PHY 参数观测寄存器 1

偏移地址：0xa4

RO

用于观测由 lane_index 选中 lane 的参数设置。

位域	名字	描述
7:0	ltx_lf	选中lane本地的TX LF参数
15:8	ltx_fs	选中lane本地的TX FS参数
23:16	rtx_lf	选中lane对端的TX LF参数
31:24	rtx_fs	选中lane对端的TX FS参数

14.7.29 PHY 参数观测寄存器 2

偏移地址：0xa8

用于观测由 lane_index 选中 lane 的参数设置。

位域	名字	描述
7:0	rtx_c-1	选中lane对端的TX C-1系数
15:8	rtx_c0	选中lane对端的TX C0系数
23:16	rtx_c+1	选中lane对端的TX C+1系数
31:24	rtx_pset	选中的lane对端的TX preset预设值

14.7.30 PHY 参数观测寄存器 3

偏移地址：0xac

用于观测由 lane_index 选中 lane 的参数设置。

位域	名字	属性	描述
7:0	lane_index	RW	用于配置要观测的lane的编号 编号从0开始
15:8	pset0_fom	RO	选中lane对端在使用preset_0时本地评估的眼宽数值
23:16	pset1_fom	RO	选中lane对端在使用preset_1时本地评估的眼宽数值
31:24	pset2_fom	RO	选中lane对端在使用preset_2时本地评估的眼宽数值

14.7.31 PHY 参数观测寄存器 4

偏移地址：0xb0

用于观测由 lane_index 选中 lane 的参数设置。

位域	名字	属性	描述
7:0	pset3_fom	RO	选中lane对端在使用preset_3时本地评估的眼宽数值
15:8	pset4_fom	RO	选中lane对端在使用preset_4时本地评估的眼宽数值
23:16	pset5_fom	RO	选中lane对端在使用preset_5时本地评估的眼宽数值
31:24	pset6_fom	RO	选中lane对端在使用preset_6时本地评估的眼宽数值

14.7.32 PHY 参数观测寄存器 5

偏移地址：0xb4

用于观测由 lane_index 选中 lane 的参数设置。

位域	名字	属性	描述
7:0	pset7_fom	RO	选中lane对端在使用preset_7时本地评估的眼宽数值
15:8	pset8_fom	RO	选中lane对端在使用preset_8时本地评估的眼宽数值
23:16	pset9_fom	RO	选中lane对端在使用preset_9时本地评估的眼宽数值
31:24	pset10_fom	RO	选中lane对端在使用preset_10时本地评估的眼宽数值

14.7.33 物理层链路观测寄存器 0

偏移地址：0xb8

用于观测物理层链路的状态变化。

位域	名字	属性	描述
5:0		RO	
7:6	link speed d3	RO	0：gen1，1：gen2，2：gen3，3：gen4
13:8		RO	
15:14	link speed d2	RO	
21:16		RO	
23:22	link speed d1	RO	
29:24		RO	
31:30	link speed d0	RO	

14.7.34 物理层链路观测寄存器 1

偏移地址：0xbc

用于观测物理层链路的状态变化。

位域	名字	属性	描述
5:0		RO	
7:6	link speed d7	RO	0：gen1，1：gen2，2：gen3，3：gen4
13:8		RO	
15:14	link speed d6	RO	
21:16		RO	
23:22	link speed d5	RO	
29:24		RO	
31:30	link speed d4	RO	

14.7.35 物理层链路观测寄存器 2

偏移地址：0xc0

用于观测物理层链路退出 L0 状态原因。

位域	名字	属性	描述
8:0	L0 lose case	RO	[8]: rcvd_2_unexpect_ts [7]: xdlh_xmlh_start_link_retrain [6]: directed_recovery [5]: rmlh_deskew_alignment_err [4]: go_recovery_speed_change [3]: rmlh_goto_recovery [2]: directed_equalization [1]: rtlh_req_link_retrain [0]: eidle_inferred_recovery
15:9	Reserved	RO	保留
24:16	last L0 lose case	RO	

31:25	Reserved	RO	保留

14.7.36 数据链路层观测寄存器 0

偏移地址：0xc4

位域	名字	属性	描述
15:0	replay_cnt	RO	
31:16	nack_cnt	RO	

14.7.37 RO 属性的写计数值

偏移地址：0xd0

位域	名字	属性	描述
31:0	p_counter_ro	RO	协议层从 PCIE 链路接收到的带有 RO 属性的写请求的计数

14.7.38 IDO 属性的写计数值

偏移地址：0xd4

位域	名字	属性	描述
31:0	p_counter_ido	RO	协议层从 PCIE 链路接收到的带有 IDO 属性的写请求的计数

14.7.39 接收通道写计数值

偏移地址：0xd8

位域	名字	属性	描述
31:0	p_counter	RO	协议层从 PCIE 链路接收到的写请求的计数

14.7.40 接收通道写计数值

偏移地址：0xdc

默认值：0x00080000

位域	名字	属性	描述
23:0	cdts_timeout_val	RW	发送通道在链路完好情况下获取令牌的超时时间 默认值为 8ms
30:24	Reserved	RW	保留
31	cdts_wait_timeout_en	RW	此位为 1 时，当发送通道的请求在链路完好的情况下超过 cdts_timeout_val 设定的时候还没有获得令牌时将触发总线保护机制工作

14.8 软件编程指南

3C6000 的 PCIE 控制器在使用前，在使用前需要经过以下几个步骤的初始化过程：

- 参考时钟准备
- PCIE Group 工作模式的设定
- 控制器使能
- PHY 参数初始化
- 配置内部网络中的 PCI 虚拟桥以便访问除 PCIE0 外的 PCIE 控制器
- 链路建立初始化

14.8.1 参考时钟准备

芯片的 PCIE 接口使用了由芯片内部的专门模块（PRG）来为芯片内部 PCIE 控制器以及外部的 PCIE 设备提供同源参考时钟的参考时钟方案。在使用 PCIE 控制器前，需正确配置 PRG 的相关工作参数（IOCSR[0x440]、IOCSR[0x448]），确保参考时钟已经就绪。参考时钟的准备分为下列步骤：

（1）要首先根据板卡的情况确定 PRG 模块的 100MHz 时钟选择了正确的输入 PAD：CHIP_CONFIG[5]上拉时选择 SYSCLK_I0p/n,下拉时选择 SYSCLK_I1p/n。

（2）根据需要配置 PRG 是正常工作模式还是 bypass 模式。此模式设置由 IOCSR[0x440][0]（prg_byp）和 IOCSR[0x440][1]（prg_byp_reg）控制：0 正常工作模式，1 bypass 模式；默认值为 1，bypass 工作模式。

（3）如果是正常工作模式，根据需要选择是否开启展频功能。由 IOCSR[0x440][7]（ssc_en）控制：0 不开启 SSC，1 开启 SSC；默认值为 0，不开启 SSC。

（4）如果是正常工作模式，确认 PRG 的输出时钟已经稳定：IOCSR[0x448][63]为 1。

14.8.2 PCIE Group 工作模式设定

芯片的 PCIE 控制器和 PCIE PHY 被划分到 2 个 PCIE GROUP：pcie group0、pcie group1。在使用 PCIE 功能前需根据板卡的应用情况要对这 2 个 PCIE GROUP 的工作方式进行设定。

pcie group0 中的 PHY0 为 PCIE 功能专用，其工作模式由 IOCSR[0x1A0][33:32]的值确定：0x0：1X16（PCIE0），0x1：2X8（PCIE0、PCIE4），0x2：4X4（PCIE0、PCIE2、PCIE4、PCIE6），0x3：1X8+2X4（PCIE0、PCIE4、PCIE6）。

pcie group0 中的 PHY2 为 LCL 与 PCIE 功能复用。当管脚 ICCC_EN1 或 CF_LCL2_mode（IOCSR[0x1A0][13]，默认值为 0）为 1 时，为 LCL 功能。当使用 LCL 功能时，由管脚 CHIP_CONFIG4 的电平值（1：X8， 0：X16）确定是使用 X8 链路还是 X16 链路。当此 PHY 不

使用 LCL 功能时，由 IOCSR[0x1A0][36]的值确定：1：2X8（PCIE2、PCIE6），0：1X16（PCIE2）。

pcie group1 中的 PHY1 为 LCL 与 PCIE 功能复用。当 ICCC_EN1 或 CF_LCL1_mode（IOCSR[0x1A0][12]，默认值为 0）为 1 时，为 LCL 功能。当使用 LCL 功能时，由管脚 CHIP_CONFIG4 的电平值（1：X8， 0：X16）确定是使用 X8 链路还是 X16 链路。当此 PHY 不使用 LCL 功能时，由 IOCSR[0x1A0][35:34]的值确定：0x0: 1X16（PCIE1），0x1: 2X8（PCIE1、PCIE5），0x2: 4X4（PCIE1、PCIE3、PCIE5、PCIE7），0x3: 1X8+2X4（PCIE1、PCIE5、PCIE7）。

pcie group1 中的 PHY3 为 LCL 与 PCIE 功能复用。当管脚 ICCC_EN0 和 ICCC_EN1 均为高电平时为 1 时，为 LCL 功能。当使用 LCL 功能时，由管脚 CHIP_CONFIG4 的电平值（1：X8，0:X16)确定是使用 X8 链路还是 X16 链路。当此 PHY 不使用 LCL 功能时，由 IOCSR[0x1A0][38]的值确定：1：2X8（PCIE3、PCIE7），0：1X16（PCIE3）。

14.8.3 PCIE 控制器使能

使能 PCIE 控制器的的操作包含下列步骤：

（1）使能 PCIE 控制器(IOCSR[0x1A0][7:0]从低位到高位分别对应 PCIE0、PCIE2、PCIE4、PCIE6、PCIE1、PCIE3、PCIE5、PCIE7 的访问使能，默认值为 0）。

（2）将 PCIE GROUP 中配置寄存器 1 的[1:0]配置为 0x0，对 PCIE GROUP 里的 2 个 PCIE PHY 进行复位（pcie group0 是 IOCSR[0x3008]，pcie group1 是 IOCSR[0x3038]）。

（3）将 PCIE GROUP 中配置寄存器 1 的[1:0]配置为 0x3，撤销对 PCIE GROUP 里的 2 个 PCIE PHY 的复位。

（4）将 IOCSR[0x420][29:28]配置为 0x0（bit28 对应 pcie group0，bit29 对应 pcie group1），复位 2 个 PCIE GROUP。

（5）将 IOCSR[0x420][29:28]配置为 0x3，撤销对 2 个 PCIE GROUP 的复位。

14.8.4 PHY 参数初始化

PCIE 控制器在使用前需要完成对 PHY 的参数配置。对 PHY 进行参数配置后才能释放对 PHY 的 PLL 的复位信号，让 PCIE 控制器的复位能够完成。PHY 的参数初始化流程如下：

（1）查询 PCIE GROUP 中配置寄存器 1（pcie group0 是 IOCSR[0x3008]，pcie group1 是 IOCSR[0x3038][8]和 IOCSR[0x3038][16]，等待其值都变为 1（PHY 的内建初始化配置过程结束）。

（2）配置 PHY 的内部寄存器，按照需要，修订 PHY 和 PIPE 接口的工作参数。

（3）将 PCIE GROUP 中配置寄存器 1 的[3:2]置为 0x3，表示控制器所属的 PHY 控制寄存器中的 phy_soft_cfg_done 置 1，释放 PHY 的 PLL 的复位信号。

（4）查询 PCIE GROUP 中配置寄存器 3（pcie group0 是 IOCSR[0x3018]，pcie group1 是 IOCSR[0x3048]）的[31:28]，等待其值变为 0xf（对应的 PCIE 控制器的复位结束）。

（5）PHY 的初始化配置结束。

14.8.5 配置 PCI 虚拟桥

芯片内部的 PCIE 接口中除 PCIE0 外，都需要分别通过内部互联上的 2 个 PCI 虚拟桥（PCI_V0、PCI_V1）才能进行访问。虚拟桥的访问符合 PCI 协议的要求。

在 PCIE0 为设备连接模式时,PCI_V0 和 PCI_V1 分别作为内部虚拟 PCI bus 0 上的 device 1 和 device 3,否则就根据 PCI 配置寄存器（IOCSR[0x1A0]）的设置决定。PCI_V0 下的 device 0 是 PCIE2（PCIE_G0 端口 1）；device 1 是 PCIE4（PCIE_G0 端口 2）；device 2 是 PCIE6（PCIE_G0 端口 3）。PCI_V1 下的 device 0 是 PCIE1（PCIE_G1 端口 0）；device 1 是 PCIE3（PCIE_G1 端口 1）；device 2 是 PCIE5（PCIE_G1 端口 2）；device 3 是 PCIE7（PCIE_G1 端口 3）。

需要注意的是，虚拟桥的配置空间的 Secbus、Subbus 在任何情况下必须正确配置，未经正确初始化时，应该设置 Secbus > Subbus，以防止错误的设备路由。

14.8.6 PCIE 链路建立(Linkup)

链路建立流程如下：

（1）设置链路目标速度。

（2）依据是否要进行 gen3 的链路均衡参数训练，设置是否关闭 gen3 的参数训练

（3）依据是否要进行 gen4 的链路均衡参数训练，设置是否关闭 gen4 的参数训练

（4）配置访问设置 Gen2 Control Register 寄存器中（0x80c）Directed Speed Change 为 1，PHY Tx Swing 为 0。

（5）如果使用 gen3 速率，在 SPCIE 寄存器中为每个 lane 设置 gen3 速率下初始的 PRESET 值（这些 PRESET 值从 header 的地址偏移 0x14c 开始存储）。

（6）如果使用 gen4 速率，在 SPCIE 寄存器中为每个 lane 设置 gen4 速率下初始的 PRESET 值（这些 PRESET 值从 header 的地址偏移 0x1a0 开始存储）。

（7）根据 BAR0 中配的地址通过 MEM 访问设置 PCIE 控制器内部寄存器

app_ltssm_enable(0x0)为 1，开始 link training 过程。

（8）如果目标速率是 gen4，等待链路速率达 gen3 后，将配置访问设置 Gen2 Control Register 寄存器中（0x80c）Directed Speed Change 先写 0 再写 1。

（9）等待链路速率达到设定的速率。

（10）等待内部寄存器 Xmlh_ltssm_state(0xc)为 0xd1。

（11）Linkup 成功。

上述步骤中。步骤 1~6 之间没有顺序要求。

15 LCL 接口

每个龙芯 3C6000 硅片中，集成了 64 位的 PCIE PHY 和 8 个 PCIE 控制器。

其中 64 位 PCIE 分为 4 个独立的 x16 PHY，分别对应芯片接口的 PCIE0/1/2/3。LCL1/2/3 在不同的情况下可以与 PCIE1/2/3 相复用。

LCL0 为独立接口，不与 PCIE 复用。

15.1 地址空间划分

LCL 用于多路互连中进行数据交换，其配置寄存器空间地址使用分布如下：

表 15-1 LCL 配置空间

基地址	结束地址	大小	定义
0xA00_0000_0000	0xA00_0FFF_FFFF	256MB	LCL0 配置空间
0xA00_1000_0000	0xA00_1FFF_FFFF	256MB	LCL1 配置空间
0xA00_2000_0000	0xA00_2FFF_FFFF	256MB	LCL2 配置空间
0xA00_3000_0000	0xA00_3FFF_FFFF	256MB	LCL3 配置空间

15.2 地址交换

LCL 在多路互连中基于位置决定使用的端口，总体来说按照下图的规则决定每个芯片连接使用的 LCL 端口（LCL1/2/3 与 PCIE1/2/3 编号相同）。其中端口号 XY 表示硅片号 X 中的 LCL/PCIE 端口 Y。

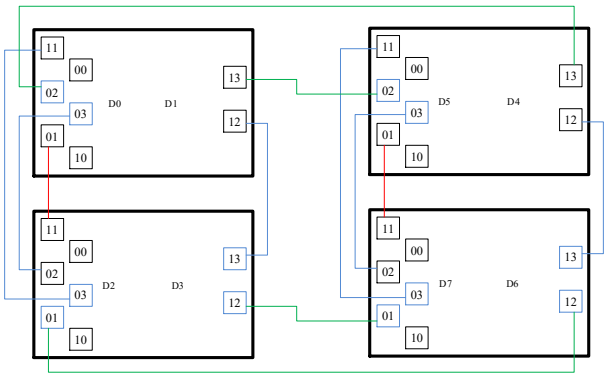


图 15-1 多路互连 LCL 端口选择

在特定情况下，为了方便主板的布局或连线，可以将选用的 LCL1 和 LCL3，或者 LCL2 和 LCL3 进行交换。这个交换是通过引脚 CHIP_CONFIG[3]或 CHIP_CONFIG[2]进行配置的。例

如上图中 D0/1 及 D4/5 的 PCIE13 和 PCIE12 就进行了交换。软件需要注意的是这个交换只改变跨片路由，不改变原有端口的配置空间。

15.3 全局配置寄存器

芯片配置寄存器空间上存在部分对所有 LCL 端口整体控制和状态寄存器，以及针对 LCL0 PHY 的控制和状态寄存器，这些寄存器访问基地址为 0x1FE00000。该部分寄存器在初始化阶段不需要配置，主要用于调试。

15.3.1 LCL 端口配置寄存器 0

本组寄存器包含对 LCL 端口的控制和状态信号。

地址偏移：3060h

默认值：0000_0000_1919_c000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:32	lcl0_phy_lane_shut	R/W	从低位到高位分别对应 LCL PHY 的 lane0～lane15 关闭控制 0：正常工作模式 1：关闭模式
31:16	Reserved	R/W	保留
15	Reserved	R/W	保留
14	lcl0_phy_aux_clk_en	R/W	LCL PHY0 的辅助时钟使能 0：不使用辅助时钟 1：使用辅助时钟
13:0	Reserved	R/W	保留

15.3.2 LCL 端口配置寄存器 1

本组寄存器包含对 LCL 端口的控制和状态信号。

地址偏移：3068h

默认值：0000_0000_0000_00f3h

位域	名称	访问	描述
63	warm_wait_bypass	R/W	四个 LCL 控制器在热复位时是否等待对应 PLL 的锁定状态 0：等待 PLL 锁定 1：不等待 PLL 锁定
62	phy_state_bypass	R/W	四个 LCL 控制器在复位时是否等待对应 PHY 的状态标志

			0：等待 PHY 的状态标志 1：不等待 PHY 的状态标志
61	prg_state_bypass	R/W	四个 LCL 控制器在复位时是否等待参考时钟状态 0：等待参考时钟就绪标志 1：不等待参考时钟就绪标志
60	Reserved	RW	保留
59:44	PHY2_READY	RO	从低位到高位分别对应 PCIE PHY2 的 lane0～lane15 的 phy_ready 状态
44:29	LCL0_PHY_READY	RO	从低位到高位分别对应 LCL0 PHY 的 lane0～lane15 的 phy_ready 状态
28:26	Reserved	RW	保留
25	lcl0_phy_hfc_ready	RO	为 1 表示 LCL0 PHY 关联的 PLL 时钟已准备好
24:17	Reserved	RW	保留
16	phy0_cfg_rstn	RW	LCL0 PHY 的复位控制 0：复位有效 1：复位无效
15	lcl0_phy_init_cfg_stop	RO	此位为 1 表示 LCL0 PHY 复位后预置初始化配置过程中出现错误
14	lcl0_phy_init_cfg_busy	RO	此为 1 表示 LCL0 PHY 正在进行其复位后的预置初始化配置
13:10	Reserved	RO	保留
9	lcl0_phy_init_cfg_stop	RO	此位为 1 表示 LCL0 PHY 的复位后预置初始化配置过程被停止
8	lcl0_phy_init_cfg_done	RO	此为 1 表示 LCL0 PHY 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 LCL0 PHY 进行配置访问
7	Reserved	R/W	保留
6	Reserved	R/W	保留
5	Reserved	R/W	保留
4	Reserved	R/W	保留
3:2	lcl0_pipe_mode	R/W	LCL0 PIPE 工作模式选择
1	lcl0_phy_soft_cfg_done	RW	默认为 1。此位为 1 后，上电配制才可以继续进行。
0	lcl0_phy_rstn	R/W	LCL0 PHY 的软件复位控制 产生 PHY 的软件复位时，需先写入 0，再写入 1 0：复位有效 1：复位无效

15.3.3 LCL 端口控制寄存器 2

本组寄存器包含对 LCL 端口的控制和状态信号。

地址偏移：3070h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63	lane0_phystatus_p3	RO	LCL3 的 lane0 对应的 phystatus 电平
62	rdlh_link_up_p3	RO	LCL3 的数据链路层 link up 状态标志
61:56	link_state_p3	RO	LCL3 的链路状态机状态
55	xmlh_link_up_p2	RO	LCL2 控制器的 mac 层 link up 状态标志
54	lane0_phystatus_p2	RO	LCL2 的 lane0 对应的 phystatus 电平
53	rdlh_link_up_p2	RO	LCL2 的数据链路层 link up 状态标志
52:47	link_state_p2	RO	LCL2 的链路状态机状态
46	xmlh_link_up_p1	RO	LCL1 控制器的 mac 层 link up 状态标志
45	lane0_phystatus_p1	RO	LCL1 的 lane0 对应的 phystatus 电平
44	rdlh_link_up_p1	RO	LCL1 的数据链路层 link up 状态标志
43:38	link_state_p1	RO	LCL1 的链路状态机状态
37	xmlh_link_up_p0	RO	LCL0 控制器的 mac 层 link up 状态标志
36	lane0_phystatus_p0	RO	LCL0 的 lane0 对应的 phystatus 电平
35	rdlh_link_up_p0	RO	LCL0 的数据链路层 link up 状态标志
34:29	link_state_p0	RO	LCL0 的链路状态机状态
28：8	Reserved	R/W	保留
7	p3_soft_rst	R/W	LCL3 控制器的软件复位控制 0：软件复位撤销 1：软件复位有效
6	p2_soft_rst	R/W	LCL2 控制器的软件复位控制 0：软件复位撤销 1：软件复位有效
5	p1_soft_rst	R/W	LCL1 控制器的软件复位控制 0：软件复位撤销 1：软件复位有效
4	p0_soft_rst	R/W	LCL0 控制器的软件复位控制 0：软件复位撤销 1：软件复位有效
3:0	Reserved	R/W	保留

15.3.4 LCL 端口控制寄存器 3

本组寄存器包含对 LCL 端口的状态信号。

地址偏移：3078h

默认值：0000_0000_0f00_0000h

位域	名称	访问	描述
63:32	Reserved	R/W	保留
31	coreclk_ok_p3	RO	LCL3 控制器的复位状态观测 0：复位进行中 1：复位结束
30	coreclk_ok_p2	RO	LCL2 控制器的复位状态观测 0：复位进行中 1：复位结束
29	coreclk_ok_p1	RO	LCL1 控制器的复位状态观测 0：复位进行中 1：复位结束
28	coreclk_ok_p0	RO	LCL0 控制器的复位状态观测 0：复位进行中 1：复位结束
27:24	lane0_local_pset_index_p0	RO	LCL0 PHY lane0 TX 使用的 preset_index 值
23:18	lane0_localfs_p0	RO	LCL0 PHY lane0 TX 的 fullswing 数值
17:0	lane0_tx_coef_p0	RO	LCL0 PHY lane0 TX 使用的预加重、去加重系数

15.3.5 LCL0 PHY 配置访问寄存器

本组寄存器用来控制产生 LCL0 PHY 内部控制寄存器的配置访问操作。

地址偏移：3080h

默认值：0000_0000h

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟

56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态
55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

3C6000 中 LCL0 PHY 的内部寄存器通过以上软件接口，具体的配置流程同 PCIE PHY。

15.4 内部配置寄存器

LCL 内部配置寄存器用于控制链路初始化及内部组包规则等，其寄存器如下表：

表 15-2 LCL 配置寄存器说明

地址	名称	描述
链路性能		
0x0_0078	LINK_ERR	链路错误报告寄存器 Link Error Reporting Register
0x0_007C	LINK_CAPA	链路性能寄存器 Link Capabilities Register
0x0_0080	LINK_CSR	链路控制和状态寄存器 Link Control and Status Register
0x0_0084~0x0_009b	Rsvd	
0x0_009C	LINK_CAPA2	链路性能寄存器 2 Link Capabilities 2 Register
0x0_00a0	LINK_CSR2	链路控制和状态寄存器 2Link Control and Status 2 Register
高级错误报告		
0x0_0104	UESR	不可纠正错误状态寄存器 Uncorrectable Error Status Register
0x0_0108	UEMR	不可纠正错误掩码寄存器 Uncorrectable Error Mask Register
0x0_010C	UESeR	不可纠正错误严重性寄存器 Uncorrectable Error Severity Register
0x0_0110	CESR	可纠正错误状态寄存器 Correctable Error Status Register
0x0_0114	CEMR	可纠正错误掩码寄存器 Correctable Error Mask Register
0x0_0118~0x0_012b	Rsvd	
0x0_012c	ERER	错误报告使能寄存器

		Error Reporting Enable Register
Gen3 物理层控制		
0x0_01d4	G3_EQ_CTRL	Gen3 链路均衡化控制寄存器 Gen3 Link Equalization Control Register
0x0_01d8	LANE_ERR	Lane 错误状态寄存器 Lane Error Status Register
0x0_01dc	G3_EQ_CTRL0	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane0&1 Register
0x0_01e0	G3_EQ_CTRL1	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane2&3 Register
0x0_01e4	G3_EQ_CTRL2	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane4&5 Register
0x0_01e8	G3_EQ_CTRL3	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane6&7 Register
0x0_01ec	G3_EQ_CTRL4	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane8&9 Register
0x0_01f0	G3_EQ_CTRL5	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane10&11 Register
0x0_01f4	G3_EQ_CTRL6	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane12&13 Register
0x0_01f8	G3_EQ_CTRL7	Gen3 均衡化控制寄存器 Gen3 Equalization Control Lane14&15 Register
Gen4 物理层控制		
0x0_0204	G4_CAPA	Gen4 性能寄存器 Gen4 Capabilities Register
0x0_0208	G4_CTRL	Gen4 控制寄存器 Gen4 Control Register
0x0_020c	G4_STAT	Gen4 状态寄存器 Gen4 Status Register
0x0_0210~0x0_021f	Rsvd	
0x0_0220	G4_EQ_CTRL0	Gen4 均衡化控制寄存器 Gen4 Equalization Control Lane0~3 Register
0x0_0224	G4_EQ_CTRL1	Gen4 均衡化控制寄存器 Gen4 Equalization Control Lane4~7 Register
0x0_0228	G4_EQ_CTRL2	Gen4 均衡化控制寄存器 Gen4 Equalization Control Lane8~11 Register
0x0_022c	G4_EQ_CTRL3	Gen4 均衡化控制寄存器 Gen4 Equalization Control Lane12~15 Register
LCL 端口配置		
0x0_0760	FCFC0	流控频率控制寄存器 FC Frequency Control Register 0
0x0_0764	FCFC1	流控频率控制寄存器

		FC Frequency Control Register 1
0x0_0768	FCFE0	流控频率控制使能寄存器 FC Frequency Enable Register
0x0_0900	TX_CMD_AIV0	发送端命令通道防饿死计数器初值寄存器 TX CMD Channel Age Init Value Register 0
0x0_0904	TX_CMD_AIV1	发送端命令通道防饿死计数器初值寄存器 TX CMD Channel Age Init Value Register 1
0x0_0908	TX_DATA_AIV0	发送端数据通道防饿死计数器初值寄存器 TX DATA Channel Age Init Value Register 0
0x0_090c	TX_DATA_AIV1	发送端数据通道防饿死计数器初值寄存器 TX DATA Channel Age Init Value Register 1
0x0_0910	VC_CMD_PRIO	发送端命令通道优先级控制寄存器 CMD Channel Priority Register
0x0_0914	VC_DATA_PRIO	发送端数据通道优先级控制寄存器 DATA Channel Priority Register
0x0_0918	TX_CTRL	发送端通道控制寄存器 TX Channel Control Register
LCL 端口控制与状态		
0x1_0000	PCR0	端口控制寄存器 0 Port Control Register 0
0x1_0004	PCR1	端口控制寄存器 1 Port Control Register 1
0x1_0008	PSR0	端口状态寄存器 0 Port Status Register 0
0x1_000c	PSR1	端口状态寄存器 1 Port Status Register 1
0x1_0010	Rsvd	
0x1_0014	FCPOR0	流控包观测数据寄存器 0 Flow Control Packet Observation Register 0
0x1_0018	PISR	端口中断状态寄存器 Port Interrupt Status Register
0x1_001c	PICR	端口中断清除寄存器 Port Interrupt Clear Register
0x1_0020	PIER	端口中断使能寄存器 Port Interrupt Enable Register
0x1_0024~0x1_0047	Rsvd	
0x1_0048	FCPOR1	流控包观测数据寄存器 1 Flow Control Packet Observation Register 1
0x1_004c~0x1_009f	Rsvd	
0x1_00a0	PHYPOR0	PHY 参数观测寄存器 PHY Parameter Observation Register 0
0x1_00a4	PHYPOR1	PHY 参数观测寄存器

		PHY Parameter Observation Register 1
0x1_00a8	PHYPOR2	PHY 参数观测寄存器 PHY Parameter Observation Register 2
0x1_00ac	PHYPOR3	PHY 参数观测寄存器 PHY Parameter Observation Register 3
0x1_00b0	PHYPOR4	PHY 参数观测寄存器 PHY Parameter Observation Register 4
0x1_00b4	PHYPOR5	PHY 参数观测寄存器 PHY Parameter Observation Register 5
0x1_00b8	LTSSM_DBUG0	LTSSM 调试寄存器 LTSSM Debug Register 0
0x1_00bc	LTSSM_DBUG1	LTSSM 调试寄存器 LTSSM Debug Register 1
0x1_00c0	LTSSM_DBUG2	LTSSM 调试寄存器 LTSSM Debug Register 2
0x1_00c4	DLLOR	数据链路层调试寄存器 Data Link Layer Observation Register

具体寄存器定义如下。

15.4.1 链路性能

15.4.1.1 链路错误报告寄存器(0x0_0078)

位域	名称	属性	默认值	描述
0	CERE	RW	0x0	可纠正错误报告使能
1	NFERE	RW	0x0	非致命错误报告使能
2	FERE	RW	0x0	致命错误报告使能
3	URRE	RW	0x0	不支持的请求报告使能
15:4	Rsvd	-	-	Reserved
16	CED	RW1C	0x0	检测到可纠正错误 无论错误报告是否使能，错误均会被记录在此位
17	NFED	RW1C	0x0	检测到非致命错误 无论错误报告是否使能，错误均会被记录在此位
18	FED	RW1C	0x0	检测到致命错误 无论错误报告是否使能，错误均会被记录在此位
19	URD	RW1C	0x0	检测到不支持的请求 无论错误报告是否使能，错误均会被记录在此位
31:20	Rsvd	-	-	Reserved

15.4.1.2 链路性能寄存器(0x0_007C)

位域	名称	属性	默认值	描述
3:0	MLS	RO	3	最大链路速率 Max Link Speed 指示端口支持的最大链路速率。 0 (Gen1 2.5 GT/s) 1 (Gen2 5.0 GT/s) 2 (Gen3 8.0 GT/s) 3 (Gen4 16.0 GT/s) 其他：reserved
31:4	Rsvd	-	-	Reserved

15.4.1.3 链路控制和状态寄存器(0x0_0080)

位域	名称	属性	默认值	描述
3:0	Rsvd	-	-	Reserved
4	LD	RW	0x0	链路关闭 Link Disable
5	RL	RW	0x0	链路重训练 Retrain Link
15:6	Rsvd	-	-	Reserved
19:16	LS	RO	-	链路速率 Link Speed 指示协商的链路速率。 0 (Gen1 2.5 GT/s) 1 (Gen2 5.0 GT/s) 2 (Gen3 8.0 GT/s) 3 (Gen4 16.0 GT/s) 其他：reserved
25:20	NLW	RO	0x1	协商后的链路宽度 Negotiated Link Width 由硬件在链路初始化后自动设置。当链路没有建立时，该值无效。
26	Rsvd	-	-	Reserved
27	LT	RO	0x0	正在链路训练 Link Training
31:28	Rsvd	-	-	Reserved

15.4.1.4 链路性能寄存器 2(0x0_009c)

位域	名称	属性	默认值	描述
0	Rsvd	-	-	Reserved
7:1	SLSV	RO	0x1	reserved
31:8	Rsvd	-	-	Reserved

15.4.1.5 链路控制和状态寄存器 2(0x0_00a0)

位域	名称	属性	默认值	描述
3:0	TLS	RWS	-	目标链路速率 Target Link Speed 允许的值为: 0 (Gen1 2.5 GT/s) 1 (Gen2 5.0 GT/s) 2 (Gen3 8.0 GT/s) 3 (Gen4 16.0 GT/s) 其他：reserved
4	EC	RWS	0x0	进入一致性测试 Enter Compliance 软件通过设置这一位为 1b 时强制链路在目标链路速率寄存器所规定的速率进入一致性模式
5	HASD	RWS	0x0	关闭硬件自动速率 Hardware Autonomous Speed Disable 当置位时，禁止硬件自行修改链路速率。初始的切换最高速率转换并不受此控制。
6	SDe	ROS	0x0	可选择的去加重值 Selectable De-emphasis 当链路工作在 5.0GT/s 时，去加重水平为: 1: -3.5 dB 0: -6 dB 其余速率下此位无效。
9:7	TM	RWS	0x0	发送裕度 Transmit Margin 该位控制发送端引脚的非去加重电压值 000: 全摆幅 800-1200 mV 半摆幅 400-600 mV 001-010: 值必须是一个非 0 斜率的单调变化值 011: 全摆幅 200-400 mV 半摆幅 100-200 mV 100-111: reserved
10	EMC	RWS	0x0	进入 Modified Compliance 当置位时，进入一致性测试状态时，发送修改的一致性图案。
11	CSOS	RWS	0x0	一致性有序集控制 Compliance SOS 当置位时，LTSSM 要求在发送一致性图案时周期地发送 SKP 有序集
15:12	CPD	RWS	0x0	一致性测试状态时 preset 或去加重值
16	CDeL	RO	0	当前去加重值 l
17	EqC	ROS	0	均衡化完成
18	EqP1S	ROS	0	均衡化阶段 1 完成
19	EqP2S	ROS	0	均衡化阶段 1 完成
20	EqP3S	ROS	0	均衡化阶段 2 完成
21	LER	RW1CS	0	均衡化请求
31:22	Rsvd	-	-	Reserved

15.4.2 高级错误报告

15.4.2.1 不可纠正错误状态寄存器(0x0_0104)

位域	名称	属性	默认值	描述
3:0	Rsvd	-	-	Reserved
4	DLPES	RW1CS	0x0	数据链路层错误标记 Data Link Protocol Error Status
12:5	Rsvd	-	-	Reserved
13	FCPES	RW1CS	0x0	流量控制错误标记 Flow Control Protocol Error Status
31:14	Rsvd	-	-	Reserved

15.4.2.2 不可纠正错误掩码寄存器(0x0_0108)

位域	名称	属性	默认值	描述
3:0	Rsvd	-	-	Reserved
4	DLPES	RWS	0x0	数据链路错误掩码 Data Link Protocol Error Mask
12:5	Rsvd	-	-	Reserved
13	FCPES	RWS	0x0	流量控制错误掩码 Flow Control Protocol Error Mask
31:14	Rsvd	-	-	Reserved

15.4.2.3 不可纠正错误严重性寄存器(0x0_010c)

位域	名称	属性	默认值	描述
3:0	Rsvd	-	-	Reserved
4	DLPESe	RWS	0x1	数据链路错误严重性 Data Link Protocol Error Severity
12:5	Rsvd	-	-	Reserved
13	FCPESe	RWS	0x1	流量控制错误严重性 Flow Control Protocol Error Severity
31:14	Rsvd	-	-	Reserved

15.4.2.4 可纠正错误状态寄存器(0x0_0110)

位域	名称	属性	默认值	描述
0	DLPES	RW1CS	0x0	接收端错误标记
2	BLCLPS	RW1CS	0x0	错误的 LCL 数据包标记
6:2	Rsvd	-	-	Reserved
7	BDLLPS	RW1CS	0x0	错误的 DLLP 标记
8	RNRS	RW1CS	0x0	重传次数回滚标记
11:9	Rsvd	-	-	Reserved

12	RTTS	RW1CS	0x0	重传计数器超时标记
31:13	Rsvd	-	-	Reserved

15.4.2.5 可纠正错误掩码寄存器(0x0_0114)

位域	名称	属性	默认值	描述
0	DLPEM	RWS	0x0	接收端错误掩码
6:1	Rsvd	-	-	Reserved
7	BDLLPM	RWS	0x0	错误的 LCL 数据包掩码
8	RNRM	RWS	0x0	Reserved
11:9	Rsvd	-	-	错误的 DLLP 掩码
12	RTTM	RWS	0x0	重传次数回滚掩码
31:13	Rsvd	-	-	Reserved

15.4.2.6 错误报告使能寄存器(0x0_012c)

位域	名称	属性	默认值	描述
0	CERE	RW(M) WO(S)	0x0	可纠正错误报告使能 此寄存器仅在 Master 端口下能够读出,Slave 端口只能写入
1	NFERE	RW(M) WO(S)	0x0	非致命错误报告使能 此寄存器仅在 Master 端口下能够读出,Slave 端口只能写入
2	FERE	RW(M) WO(S)	0x0	致命错误报告使能 此寄存器仅在 Master 端口下能够读出,Slave 端口只能写入
31:3	Rsvd	-	-	Reserved

15.4.3 Gen3 物理层控制

15.4.3.1 Gen3 链路均衡化控制寄存器(0x0_01d4)

位域	名称	属性	默认值	描述
0	PE	RO	0x0	进行均衡化
1	LERIE	RO	0x0	链路均衡化请求中断使能
31:2	Rsvd	-	-	Reserved

15.4.3.2 Lane 错误状态寄存器(0x0_01d8)

位域	名称	属性	默认值	描述
NL-1:0	LESB	RW1CS	0x0	Lane 错误状态标记 NL：Lane 的数量

31:NL	Rsvd	-	-	Reserved

15.4.3.3 Gen3 均衡化控制寄存器

Offset: 0x0_01dc~0x0_01f8，每条 lane 占用 2 个字节。

位域	名称	属性	默认值	描述
3:0	MPTP	RW（主端口）	1111b	主端口发送端 preset
6:4	MPRPH	RW（主端口）	000	主端口接收端 preset hint
7	Rsvd	-	-	Reserved
11:8	MPTP	RW（主端口） /RO（从端口）	1111b	从端口发送端 preset 主端口侧，可配置从端口发送端使用的 preset；从端口侧，指示当前从端口发送端使用的 preset
14:12	MPRPH	RW（主端口） /RO（从端口）	000	从端口接收 preset hint 主端口侧，可配置从端口接收端使用的 preset hint；从端口侧，指示当前接收到的 preset hint
15	Rsvd	-	-	Reserved

15.4.4 Gen4 物理层控制

15.4.4.1 Gen4 性能寄存器(0x0_0204)

位域	名称	属性	默认值	描述
31:0	Rsvd	-	-	Reserved

15.4.4.2 Gen4 控制寄存器(0x0_0208)

位域	名称	属性	默认值	描述
31:0	Rsvd	-	-	Reserved

15.4.4.3 Gen4 状态寄存器(0x0_020c)

位域	名称	属性	默认值	描述
0	G4EqC	RO	0	Gen4 均衡化完成 当置位时，该位指示 16GT/s 发送端均衡化过程已经完成
1	G4EqP1C	RO	0	Gen4 均衡化阶段 1 完成 当置位时，该位指示 16GT/s 发送端均衡化阶段 1 过程已经完成
2	G4EqP2C	RO	0	Gen4 均衡化阶段 2 完成 当置位时，该位指示 16GT/s 发送端均衡化阶段 2 过程已经完成

3	G4EqP3C	RO	0	Gen4 均衡化阶段 3 完成 当置位时，该位指示 16GT/s 发送端均衡化阶段 3 过程已经完成
4	G4EqR	RW1CS	0	Gen4 链路请求均衡化
31:5	Rsvd	-	-	Reserved

15.4.4.4 Gen4 均衡化控制寄存器

Offset：0x0_0220～0x0_022c，每条 lane 占用 1 个字节。

位域	名称	属性	默认值	描述
3:0	G4MPTP	RW（主端口）	0	Gen4 主端口发送端 preset
7:4	G4SPTP	RW（主端口） /RO（从端口）	0	Gen4 从端口发送端 preset 主端口侧，可配置从端口发送端使用的 preset；从端口侧，指示当前发送端使用的 preset

15.4.5 LCL 端口配置

15.4.5.1 流控频率控制寄存器(0x0_0760~0x0_0767)

位域	名称	属性	默认值	描述
7:0	VC0FCF	RW	0	L1AR 通道流控频率 L1AR 通道 FC 令牌发送频率
15:8	VC1FCF	RW	0	L1R 通道流控频率 L1R 通道 FC 令牌发送频率
23:16	VC2FCF	RW	0	L1W 通道流控频率 L1W 通道 FC 令牌发送频率
31:24	VC3FCF	RW	0	L1B 通道流控频率 L1B 通道 FC 令牌发送频率
39:32	VC4FCF	RW	0	L2AR 通道流控频率 L2AR 通道 FC 令牌发送频率
47:40	VC5FCF	RW	0	L2R C 通道流控频率 L2R 通道 FC 令牌发送频率
55:48	VC6FCF	RW	0	L2W 通道流控频率 L2W 通道 FC 令牌发送频率
63:56	VC7FCF	RW	0	L2B 通道流控频率 L2B 通道 FC 令牌发送频率

15.4.5.2 流控频率控制使能寄存器(0x0_0768)

位域	名称	属性	默认值	描述
1:0	VC0FCFE	RW	0	L1AR 通道流控频率设置使能 00b：使用默认的令牌发送频率 others：L1AR 通道使用 FCFC 设置的令牌发送频率，下同 注：fc 令牌发送计数器的值为 FCFC 设置的发送频率×16
3:2	VC1FCFE	RW	0	L1R 通道流控频率设置使能 L1R 通道使用 FCFC 设置的令牌发送频率
5:4	VC2FCFE	RW	0	L1W 通道流控频率设置使能 L1W 通道使用 FCFC 设置的令牌发送频率
7:6	VC3FCFE	RW	0	L1B 通道流控频率设置使能 L1B 通道使用 FCFC 设置的令牌发送频率
9:8	VC4FCFE	RW	0	L2AR 通道流控频率设置使能 L2AR 通道使用 FCFC 设置的令牌发送频率
11:10	VC5FCFE	RW	0	L2R 通道流控频率设置使能 L2R 通道使用 FCFC 设置的令牌发送频率
13:12	VC6FCFE	RW	0	L2W 通道流控频率设置使能 L2W 通道使用 FCFC 设置的令牌发送频率
15:14	VC7FCFE	RW	0	L2B 通道流控频率设置使能 L2B 通道使用 FCFC 设置的令牌发送频率
31:16	Rsvd	-	-	Reserved

15.4.5.3 发送端命令通道防饿死计数器初值寄存器(0x0_0900~0x0_0907)

位域	名称	属性	默认值	描述
7:0	VC0CCAI	RW	3	L1AR 命令通道防饿死计数器初值
15:8	VC1CCAI	RW	15	L1R 命令通道防饿死计数器初值
23:16	VC2CCAI	RW	5	L1W 命令通道防饿死计数器初值
31:24	VC3CCAI	RW	15	L1B 命令通道防饿死计数器初值
39:32	VC4CCAI	RW	6	L2AR 命令通道防饿死计数器初值
47:40	VC5CCAI	RW	15	L2R 命令通道防饿死计数器初值
55:48	VC6CCAI	RW	8	L2W 命令通道防饿死计数器初值
63:56	VC7CCAI	RW	15	L2B 命令通道防饿死计数器初值

15.4.5.4 发送端数据通道防饿死计数器初值寄存器(0x0_0908~0x0_090f)

位域	名称	属性	默认值	描述
7:0	VC0DCAI	RW	3	L1AR DATA Channel Age Init L1AR 命令通道防饿死计数器初值
15:8	VC1DCAI	RW	15	L1R DATA Channel Age Init L1R 命令通道防饿死计数器初值
23:16	VC2DCAI	RW	5	L1W DATA Channel Age Init L1W 命令通道防饿死计数器初值
31:24	Rsvd	-	-	Reserved
39:32	Rsvd	-	-	Reserved
47:40	VC5DCAI	RW	15	L2R DATA Channel Age Init L2R 命令通道防饿死计数器初值
55:48	VC6DCAI	RW	8	L2W DATA Channel Age Init L2W 命令通道防饿死计数器初值
63:56	Rsvd	-	-	Reserved

15.4.5.5 发送端命令通道优先级控制寄存器(0x0_0910)

位域	名称	属性	默认值	描述
3:0	NCCP1	RW	32	No. CMD Channel of Priority 1 第一优先级命令通道编号，见表 15-3
7:4	NCCP2	RW	32	No. CMD Channel of Priority 2 第二优先级命令通道编号
11:8	NCCP3	RW	32	No. CMD Channel of Priority 3 第三优先级命令通道编号
15:12	NCCP4	RW	32	No. CMD Channel of Priority 4 第四优先级命令通道编号
19:16	NCCP5	RW	32	No. CMD Channel of Priority 5 第五优先级命令通道编号
23:20	NCCP6	RW	32	No. CMD Channel of Priority 6 第六优先级命令通道编号
27:24	NCCP7	RW	32	No. CMD Channel of Priority 7 第七优先级命令通道编号
31:28	NCCP8	RW	32	No. CMD Channel of Priority 8 第八优先级命令通道编号

注意：该寄存器中各个优先级通道编号的值不可以重复，否则更新后的优先级是未定义的状态。

表 15-3 命令通道编号表

命令通道	编号	命令通道	编号
L1AR	0x0	L2AR	0x4
L1R	0x1	L2R	0x5
L1W	0x2	L2W	0x6
L1B	0x3	L2B	0x7

15.4.5.6 发送端数据通道优先级控制寄存器（0x0_0914）

位域	名称	属性	默认值	描述
3:0	NDCP1	RW	32	No. DATA Channel of Priority 1 第一优先级数据通道编号，见表 15-4
7:4	NDCP2	RW	32	No. DATA Channel of Priority 2 第二优先级数据通道编号
11:8	NDCP3	RW	32	No. DATA Channel of Priority 3 第三优先级数据通道编号
15:12	NDCP4	RW	32	No. DATA Channel of Priority 4 第四优先级数据通道编号
19:16	NDCP5	RW	32	No. DATA Channel of Priority 5 第五优先级数据通道编号
31:20	Rsvd	-	-	Reserved

注意：该寄存器中各个优先级通道编号的值不可以重复，否则更新后的优先级是未定义的状态。

表 15-4 数据通道编号表

数据通道	编号	数据通道	编号
L1AR	0x0	L2R	0x3
L1R	0x1	L2W	0x4
L1W	0x2		

15.4.5.7 发送端通道控制寄存器（0x0_0918）

位域	名称	属性	默认值	描述
4:0	Rsvd	-	-	Reserved
5	L2RDCCE	RW	1	L2R Data Channel Compress Enable L2R 通道数据压缩使能
6	L2WDCCE	RW	1	L2W Data Channel Compress Enable L2W 通道数据压缩使能
7	Rsvd	-	-	Reserved
9:8	DCAC	RW	0	Data Compress Algorithm Control

				数据压缩算法控制 01b：禁用数据压缩算法 1 10b：禁用数据压缩算法 2 others：数据压缩算法 1、2 均启用，自动选择高压缩率结果
12:10	Rsvd	-	-	Reserved
13	L2RDCM	RW	1	L2R Data Channel Compress Mode L2R 通道数据压缩模式选择 当此位置 0 后，通过 L2R 通道压缩引擎的数据微包只有压缩长度小于 13DW 时才能进行压缩输出，否则输出原数据微包 当此位置 1 后，通过 L2R 通道压缩引擎的数据微包只要压缩长度小于 16DW 时就会进行压缩输出，否则输出原数据包
14	L2WDCM	RW	1	L2W Data Channel Compress Mode L2W 通道数据压缩模式选择 当此位置 0 后，通过 L2W 通道压缩引擎的数据微包只有压缩长度小于 13DW 时才能进行压缩输出，否则输出原数据微包 当此位置 1 后，通过 L2W 通道压缩引擎的数据微包只要压缩长度小于 16DW 时就会进行压缩输出，否则输出原数据包
15	Rsvd	-	-	Reserved
23:16	CMDBT	RW	5	CMD Buffer Threshold 命令微包缓冲阈值 命令微包获得仲裁许可后，将按照仲裁排序依次进入命令微包缓冲中，当该缓冲中的数据微包数量不超过该寄存器中的阈值时，由数据微包根据组合结果中的数据包剩余空间选择命令微包进行组合；当该缓冲中数量超过阈值时，数据微包被阻塞，发送一个完全由命令微包组合的 LCL 数据包（LCL 数据包 Fmt1Type5，详见 LCL 协议）
24	CPU	W1C	0	Channel Priority Update 通道优先级更新 当向此位写 1 后，在命令通道优先级寄存器和数据通道优先级寄存器中的设置将生效
25	Rsvd	-	-	Reserved
26	CMDAUQC	RW	1	CMD Channel Age Up When Queue Crowded 命令队列拥挤时触发饿死信号 当此位置 1 后，当命令通道发生队列拥挤时，命令通道饿死计数器只要达到设置初值的一半即可触发饿死信号。命令通道的拥挤判断由当前队列中的命令微包数量判断，当队列中累计 4 项时则认为发生队列拥挤。
27	DDTP	RW	1	Disable DLLP Transmit in Packet 当此位置 1 后，LCL 数据维护包（包括应答包和流控包）将禁止在 LCL 数据包内传输，而是仍然通过原有 DLLP 通路传输。
28	DF1T1P			Disable Fmt1Type1 Packet 当此位置 1 后，将禁用 LCL 协议中规定的 Fmt1Type1 格式的数据包组合。

29	CMDBHFM			CMD Buffer Half-full Mode 当此位置 1 后，命令微包缓冲原有的 16 项深度将变为半满模式，即只有 8 项。
31:30	Rsvd	-	-	Reserved

15.4.5.8 命令通道流控信用初值寄存器(0x0_091c~0x0_0923)

该组寄存器与下一组寄存器分别用于配置各个虚通道命令通道和数据通道的流量控制信用初值。默认地，该值与 LCL 控制器接收端各个虚通道的接收队列大小匹配，超过接收队列大小的信用初值可能导致总线卡死。当软件修改后，流控信用不会立即生效，而是需要链路两端重新进行数据链路层初始化才可以生效，建议在配置完成后对链路两端的 LCL 控制器进行一次软复位。当某个通道流控信用配置为 0xff 时，该通道在数据链路层初始化后将会启用无限流控，此时该通道对于到达其发送端的数据包发送不会对进行任何限制。

位域	名称	属性	默认值	描述
7:0	VC0CRI	RW	32	L1AR 命令通道流控信用初值
15:8	VC1CRI	RW	32	L1R 命令通道流控信用初值
23:16	VC2CRI	RW	16	L1W 命令通道流控信用初值
31:24	VC3CRI	RW	32	L1B 命令通道流控信用初值
39:32	VC4CRI	RW	32	L2AR 命令通道流控信用初值
47:40	VC5CRI	RW	64	L2R 命令通道流控信用初值
55:48	VC6CRI	RW	64	L2W 命令通道流控信用初值
63:56	VC7CRI	RW	32	L2B 命令通道流控信用初值

15.4.5.9 数据通道流控信用初值寄存器(0x0_0924~0x0_092f)

位域	名称	属性	默认值	描述
7:0	VC0DRI	RW	64	L1AR 数据通道流控信用初值
15:8	VC1DRI	RW	64	L1R 数据通道流控信用初值
23:16	VC2DRI	RW	16	L1W 数据通道流控信用初值
31:24	Rsvd	-	-	Reserved
39:32	Rsvd	-	-	Reserved
47:40	VC5DRI	RW	64	L2R 数据通道流控信用初值
55:48	VC6DRI	RW	64	L2W 数据通道流控信用初值
63:56	Rsvd	-	-	Reserved

15.4.6 LCL 端口控制与状态

15.4.6.1 端口控制寄存器 0 (0x1_0000)

位域	名称	属性	默认值	描述
0	rx_lane_flip_en	RW	0	LCL 接收线反转 when set to 1, Performs lane reversal for receive lanes.
1	tx_lane_flip_en	RW	1	LCL 发送线反转 when set to 1, initiate lane reversal for transmit lanes.
2	Rsvd	-	-	Reserved
3	app_ltssm_enable	RW	1	LCL 端口链路建立使能 LCL 端口在上电时默认使能端口
14:4	Rsvd	-	-	Reserved
15	soft_reset_en			软复位使能 When set to 1, enables soft reset on this LCL port
31:16	Rsvd	-	-	Reserved

15.4.6.2 端口控制寄存器 1 (0x1_0004)

位域	名称	属性	默认值	描述
0	rx_lane_flip_en	RW	0	LCL 接收线反转 when set to 1, Performs lane reversal for receive lanes.
1	Rsvd	-	-	Reserved
2	app_init_rst	RW1CS	0	在 LCL 端口上发送热复位请求
3	soft_reset	RW	1	对 LCL 端口进行软复位 仅在 PCR0 寄存器的 bit15 被置 1 时产生作用
31:4	Rsvd	-	-	Reserved

15.4.6.3 端口状态寄存器 0 (0x1_0008)

位域	名称	属性	默认值	描述
14:0	Rsvd	-	-	Reserved
15	rdlh_link_up	RW	0	数据链路层状态 Data Link Layer up/down indicator 1: Link is up 0: Link is down

28:16	Rsvd	-	-	Reserved
29	rfc_update0	RO	0	此位为 1 表示 rfc_data0 有更新的流控令牌包的内容。 rfc_update0、rfc_data0 被用作观察流控令牌的发放
30	rfc_update1	RO	0	此位为 1 表示 rfc_data1 有更新的流控令牌包的内容。 rfc_update1、rfc_data1 被用作观察流控令牌的发放
31	Rsvd	-	-	Reserved

15.4.6.4 端口状态寄存器 1 (0x1_000c)

位域	名称	属性	默认值	描述
5:0	xmlh_link_state	RO	0	物理层链路状态机的当前状态
6	xmlh_link_up	RO	0	物理链路状态指示 PHY link up/down indicator 1: Up 0: Down
7	rdlh_link_up	RO	0	数据链路状态指示 1：Up 0：Down
30:8	Rsvd	-	-	Reserved
31	cfg_link_eq_req_int	RO	0	链路均衡训练请求指示 此位为 1 表示在暂时无法做链路均衡参数训练的状态下收到了链路均衡训练的请求

15.4.6.5 流控包观测数据寄存器 0(0x1_0014)

位域	名称	属性	默认值	描述
31:0	rfc_data1	RO	0	记录最近一次同一个符号时间里收到 2 个流控包时的第二个流控包的内容

15.4.6.6 端口中断状态寄存器(0x1_0018)

位域	名称	属性	默认值	描述
0	aer_err_int	RO	0	端口产生错误时，并且 ERER 寄存器中对应的错误报告使能被打开时置位
31:1	Rsvd	-	-	Reserved

15.4.6.7 端口中断清除寄存器(0x1_001c)

位域	名称	属性	默认值	描述
0	aer_err_int_clear	RW1C	0	写 1 则清除 PISR 寄存器中 aer_err_int 的置位

31:1	Rsvd	-	-	Reserved

15.4.6.8 端口中断使能寄存器(0x1_0020)

位域	名称	属性	默认值	描述
0	aer_err_int_en	RW	1	aer_err_int 的中断使能位 当为 1 时产生中断信号，当为 0 时中断线被屏蔽
31:1	Rsvd	-	-	Reserved

15.4.6.9 流控包观测数据寄存器 1 (0x1_0048)

位域	名称	属性	默认值	描述
31:0	rfc_data0	RO	0	记录最近一次同一个符号时间里收到 2 个流控包时的第一个流控包的内容

15.4.6.10 PHY 参数观测寄存器 0(0x1_00a0)

位域	名称	属性	默认值	描述
7:0	ltx_c-1	RO	0	选中 lane 本地的 TX C-1 系数
15:8	ltx_c0	RO	0	选中 lane 本地的 TX C0 系数
23:16	ltx_c+1	RO	0	选中 lane 本地的 TX C+1 系数
27:24	ltx_pset	RO	0	选中 lane 本地的 TX preset 预设值
31:28	lrx_rhint	RO	0	选中 lane 本地的 RX preset hint 值

15.4.6.11 PHY 参数观测寄存器 1(0x1_00a4)

位域	名称	属性	默认值	描述
7:0	ltx_lf	RO	0	选中 lane 本地的 TX LF 参数
15:8	ltx_fs	RO	0	选中 lane 本地的 TX FS 参数
23:16	rtx_lf	RO	0	选中 lane 对端的 TX LF 参数
31:24	rtx_fs	RO	0	选中 lane 对端的 TX FS 参数

15.4.6.12 PHY 参数观测寄存器 2(0x1_00a8)

位域	名称	属性	默认值	描述
7:0	rtx_c-1	RO	0	选中 lane 对端的 TX C-1 系数
15:8	rtx_c0	RO	0	选中 lane 对端的 TX C0 系数
23:16	rtx_c+1	RO	0	选中 lane 对端的 TX C+1 系数
31:24	rtx_pset	RO	0	选中 lane 对端的 TX preset 预设值

15.4.6.13 PHY 参数观测寄存器 3(0x1_00ac)

位域	名称	属性	默认值	描述
7:0	lane_index	RW	0	用于配置要观测的 lane 的编号，编号从 0 开始
15:8	pset0_fom	RO	0	选中 lane 对端使用 preset_0 时本地评估的眼宽数值
23:16	pset1_fom	RO	0	选中 lane 对端使用 preset_1 时本地评估的眼宽数值
31:24	pset2_fom	RO	0	选中 lane 对端使用 preset_2 时本地评估的眼宽数值

15.4.6.14 PHY 参数观测寄存器 4(0x1_00b0)

位域	名称	属性	默认值	描述
7:0	pset3_fom	RO	0	选中 lane 对端使用 preset_3 时本地评估的眼宽数值
15:8	pset4_fom	RO	0	选中 lane 对端使用 preset_4 时本地评估的眼宽数值
23:16	pset5_fom	RO	0	选中 lane 对端使用 preset_5 时本地评估的眼宽数值
31:24	pset6_fom	RO	0	选中 lane 对端使用 preset_6 时本地评估的眼宽数值

15.4.6.15 PHY 参数观测寄存器 5(0x1_00b4)

位域	名称	属性	默认值	描述
7:0	pset7_fom	RO	0	选中 lane 对端使用 preset_7 时本地评估的眼宽数值
15:8	pset8_fom	RO	0	选中 lane 对端使用 preset_8 时本地评估的眼宽数值
23:16	pset9_fom	RO	0	选中 lane 对端使用 preset_9 时本地评估的眼宽数值
31:24	pset10_fom	RO	0	选中 lane 对端使用 preset_10 时本地评估的眼宽数值

15.4.6.16 LTSSM 调试寄存器 0(0x1_00b8)

位域	名称	属性	默认值	描述
5:0	link_state_d3	RO	0	当前 LTSSM 状态机三次跳转之前的状态
7:6	link_speed_d3	RO	0	LTSSM 状态机三次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4
13:8	link_state_d2	RO	0	当前 LTSSM 状态机两次跳转之前的状态
15:14	link_speed_d2	RO	0	LTSSM 状态机两次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4
21:16	link_state_d1	RO	0	当前 LTSSM 状态机一次跳转之前的状态
23:22	link_speed_d1	RO	0	LTSSM 状态机一次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4
29:24	link_state_d0	RO	0	当前 LTSSM 状态机的状态
31:30	link_speed_d0	RO	0	当前速率 0：gen1，1：gen2，2：gen3，3：gen4

15.4.6.17 LTSSM 调试寄存器 1(0x1_00bc)

位域	名称	属性	默认值	描述
5:0	link_state_d7	RO	0	当前 LTSSM 状态机七次跳转之前的状态
7:6	link_speed_d7	RO	0	LTSSM 状态机七次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4
13:8	link_state_d6	RO	0	当前 LTSSM 状态机六次跳转之前的状态
15:14	link_speed_d6	RO	0	LTSSM 状态机六次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4
21:16	link_state_d5	RO	0	当前 LTSSM 状态机五次跳转之前的状态
23:22	link_speed_d5	RO	0	LTSSM 状态机五次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4
29:24	link_state_d4	RO	0	当前 LTSSM 状态机四次跳转之前的状态
31:30	link_speed_d4	RO	0	LTSSM 状态机四次跳转之前的速率 0：gen1，1：gen2，2：gen3，3：gen4

15.4.6.18 LTSSM 调试寄存器 2(0x1_00c0)

位域	名称	属性	默认值	描述
8:0	l0_lose_case	RO	0	LTSSM 状态机从 L0 状态丢失的原因 [0]: rcvd_2_unexpect_ts [1]: xdlh_xmlh_start_link_retrain [2]: directed_recovery [3]: rmlh_deskew_alignment_err [4]: go_recovery_speed_change [5]: rmlh_goto_recovery [6]: directed_equalization [7]: rtlh_req_link_retrain [8]: eidle_inferred_recovery
15:9	Rsvd	-	-	Reserved
24:16	last_l0_lose_case	RO	0	上一次 LTSSM 状态机从 L0 状态丢失的原因，编码同上
31:25	Rsvd	-	-	Reserved

15.4.6.19 数据链路层调试寄存器(0x1_00c4)

位域	名称	属性	默认值	描述
15:0	replay_cnt	RO	0	
31:16	nack_cnt	RO	0	

15.5 错误处理与中断

LCL 控制器支持多种链路上发生错误的识别检测，并且对于可能出现的不同类型错误设置了错误报告使能控制。每个 LCL 控制器都有用一根中断线用于链路的错误处理，且对于不同类型的错误设置了错误中断使能以区分不同严重性和类型的错误。整个控制器的中断产生也可通过使能开关进行控制。

LCL 控制器存在可恢复和不可恢复两种类型的错误。

- 可恢复错误意味着 LCL 控制器能够自行地按照标准流程处理这种类型的错误，且不会造成传输信息的损失。比如，Link CRC 校验出错可以通过重传机制恢复。
- 不可恢复错误意味着 LCL 控制器无法自行处理相关错误，其中可以分成致命和非致命错误。对于 LCL 定义的不可恢复错误，可以通过 UESR 寄存器指定其错误严重性。

15.5.1 可恢复错误

LCL 定义的可恢复错误分为五种，其名称和描述如下表所示。

表 15-5 LCL 可恢复错误类型与描述

名称	所在层次	描述
Receiver Error 接受端解析错误	物理层 （MAC 层、PCS 层）	LCL 数据包的装裱错误；数据包长度不符合协议所定义的三种数据包；deskew 缓冲溢出；Gen3/Gen4 情况下收到不符合协议要求的 Ordered Set；Gen1/Gen2 速率下由 pipe 报告 8b/10b 译码错误（极性错误或码型错误）
Bad LCL Packet LCL 包解析错误	数据链路层	解析出的 LCL 数据包 Link CRC 解析错误；解析出的 LCL 数据包的 sequence num 错误
Bad DLLP DLLP 解析错误	数据链路层	解析出的 DLLP CRC 校验错误；解析出的 DLLP 格式不符合协议要求
REPLAY_NUM Rollover 重传次数超过 3 次	数据链路层	本端口重传缓冲出现了第四次相同 sequence num 的重传操作（该行为将触发链路重建）
Reply Timer Timeout 重传超时	数据链路层	重传缓冲计时器超过了所配置的阈值

当发生上述五种错误时，错误状态可以在任意时刻通过 CESR 寄存器的对应位置读出。CEMR 寄存器的对应位置将作为相应错误位置的掩码以使能该种错误的报告。当 Mask 为 0 时，该错误将作为可恢复错误（Corrected Error）向中断控制报告。

ERER 的第 0 位作为可恢复错误报告使能控制寄存器，当该位为 1 时，未被掩码的可恢复错误将形成可改正错误中断。

15.5.2 不可恢复错误

LCL 定义的不可恢复错误分为两种，其名称和描述如下表所示。

表 15-6 LCL 不可恢复错误类型与描述

名称	描述
Data Link Protocal Error 数据链路层协议错误	收到的相邻 Ack/Nack 包中的数据包 sequence num 的差大于可维护范围
Flow Control Protocol Error 流量控制协议错误	连续 256us 未收到 DLLP（该行为将触发链路重建）

当发生上述两种错误时，错误状态可以在任意时刻通过 UESR 寄存器的对应位置读出。UEMR 寄存器的对应位置将作为相应错误位置的掩码以使能该种错误的报告。UESeR 寄存器可以配置这两种错误的严重程度，当配置为 1 时，该错误将视为一个 Fatal Error 进行报告，否则视为 Non-Fatal Error。

ERER 的第 1、2 位作为不可改正错误的 Non-Fatal Error 和 Fatal Error 报告使能控制寄存器，当该位为 1 时，未被掩码的错误将形成可改正错误中断。

15.5.3 中断控制

PISR 寄存器第 0 位作为 LCL 错误中断状态，ERER 寄存器中使能了对应类型的错误报告后，上述的三种类型错误将会汇聚为该位中断状态。向 PICR 寄存器中的第 0 位写 1 则清除 PISR 寄存器中的置位。PIER 的第 0 位作为中断使能。

15.6 软件编程指南

15.6.1 链路初始化流程

3C6000 最多支持 8 硅片互连，每个硅片的结点号由其对应的 CHIP_ID 决定。

各个硅片所使用的 LCL 端口根据下表规定作为主设备还是从设备。主设备在上电后自行发起链路训练，保证链路能够自行在 Gen1 速率下建立。Gen1 为初始速率，在 Gen1 状态下由固件对各个 PHY 进行物理参数配置等操作并将速率切换到 Gen3，再从 Gen3 切换到 Gen4。

需要注意的是，当进行端口交换时，主从的设置与交换前用于互连的位置一致。例如，进行 CHIP2 进行 23 交换，使用 LCL3 与 CHIP1 互连时，其 LCL3 为从模式。

表 15-7 LCL 端口主从设置

LCL 端口	0	1	2	3
DIE 0	主	主	主	主
DIE 1	从	从	从	主
DIE 2	主	主	从	主
DIE 3	从	从	主	主
DIE 4	主	主	主	从
DIE 5	从	从	从	从
DIE 6	主	主	从	从
DIE 7	从	从	主	从

3C6000 的 LCL 链路初始化需要经过以下几个初始化过程：

（1）上电前准备与工作模式设定（管脚配置）

（2）查询每个启用的 LCL 主端口，等待 LCL 在 Gen1 下建立链路

（3）对每个互连芯片进行阻抗补偿训练

（4）对每个启用的 LCL 端口初始化（PHY 参数初始化）

（5）对每个启用的 LCL 端口配置发送通道参数

（6）对每个启用的 LCL 主端口依次发起 Gen3/Gen4 链路训练

15.6.2 上电前准备与工作模式设定

上电前，确认相关引脚配置能够保证 Gen1 建立的状态，同时确认需要的 LCL 端口所连接的 PHY 工作在 LCL 模式下。

首先根据板卡的情况确定 PRG 模块的 100MHz 时钟选择了正确的输入 PAD：CHIP_CONFIG[5]上拉时选择 SYSCLK_I0p/n,下拉时选择 SYSCLK_I1p/n;确认 PRG 的工作模式以及是否需要开启 SSC 功能，默认为不开启。

龙芯 3C6000 支持多片互连，需要根据不同的互连芯片数量设置 ICCC_EN[1:0]的值。

当使用 LCL 功能时，可由管脚 CHIP_CONFIG[4]的电平值（1：x8，0：x16）确定使用 x8 链路模式还是 x16 链路模式。该管脚将对所有启用的 LCL 端口同时生效。

当启用路由互换功能时，一些 LCL 端口号将被互换，但这些端口与 PHY 的连接关系将保持不变，

在完成上述引脚配置后上电，上电后软件应当等待每个启用的 LCL 端口在 Gen1 速率下建立链路后再进行后续的配置工作。可以通过查询每个启用的 LCL 主端口的 Port Status Register 1(0x1_000c)的第[5:0]位查询当前链路训练状态机的状态是否为 0x11。

15.6.3 阻抗补偿

阻抗补偿模块用于对信道阻抗情况进行自适应训练。上电后，阻抗补偿默认为 bypass 模式，能够保证 LCL 在 Gen1 模式下能够建立链路。随后，通过各个芯片上 PCIE Group0 PHY0 的配置寄存器对阻抗补偿模块进行配置，以保证阻抗补偿能够为 Gen3/Gen4 的链路训练提供合适的参数值。配置流程为：

（1）关闭阻抗补偿 bypass

（2）进行相应的阻抗补偿配置（默认情况下不需要进行参数配置）

（3）重新使能阻抗补偿，此时所有已经建立的 Gen1 链路均会重新建立，因此等待其中一个 LCL 端口重新建立 Gen1 速率下的链路。

对每个芯片均进行上述配置过程。以下为阻抗补偿的示例代码。

```
for(i=0;i<CHIP_NUM;i++){
  lcl_phy_write_reg(CHIP[i],PCIE_G0_PHYCFG0_ADDR,0x1535,0x0);//关闭阻抗补偿 bypass
  lcl_phy_write_reg(CHIP[i],PCIE_G0_PHYCFG0_ADDR,...,...);//阻抗参数配置（默认情况下不需要额外配置）
  lcl_phy_write_reg(CHIP[i],PCIE_G0_PHYCFG0_ADDR,0x1535,0x40);//使能阻抗补偿
  lcl_wait_linkup(CHIP[i],LCL0); // 等待 LCL0 仍然在 Gen1 下建立链路
}
```

15.6.4 端口初始化（PHY 参数初始化）

端口初始化流程主要为配置 Gen3/Gen4 链路训练相关的寄存器，包括控制器部分、PIPE 部分和 PHY 部分。其中，控制器与 PHY 的参数部分与 PCIE 初始化完全相同。

PIPE 部分的设置为可选配置，且对于 PCIE 与 LCL 控制器均可以使用。当使能 LCL PIPE 通路优化后，可以通过减小接受端弹性缓冲的深度和发送端位宽转换上的开销来降低 LCL 通路上的延迟，但有造成链路无法建立的风险。PIPE 中的相关寄存器位于 x8 或者 x16 PIPE 控制寄存器偏移 0xd3，其寄存器描述见下表。

表 15-8 LCL 优化通路控制寄存器描述

位域	名称	属性	默认值	描述
0	lcl_rx_opt_en	RW	0	rx 数据通路优化使能
5:1	lcl_rx_eb_max_depth	RW	0	rx 弹性缓冲的最大深度
6	lcl_tx_opt_en	RW	0	tx 数据通路优化使能
7	*Rsvd*	-	-	*Reserved*

端口初始化需要对所有启用的 LCL 端口进行配置，具体配置流程和参考代码如下：

（1）设置链路目标速度

（2）依据是否要进行 Gen3 的链路均衡参数训练，在 G3_CTRL 寄存器中设置是否关闭 Gen3 的参数训练，配置链路均衡训练模式

（3）依据是否要进行 Gen4 的链路均衡参数训练，在 G4_CTRL 寄存器中设置是否关闭 Gen4 的参数训练，配置链路均衡训练模式

（4）如果使用 Gen3 速率，在 G3_EQ_CTRL 寄存器中为每个 lane 设置 Gen3 速率下初始的 Preset 值

（5）如果使用 Gen4 速率，在 G4_EQ_CTRL 寄存器中为每个 lane 设置 Gen4 速率下初始的 Preset 值

（6）（可选)配置 LCL PIPE 通路优化,LCL0 通过 LCL0 PHY 配置,LCL1 通过 PCIE Group1 PHY0 配置，LCL2 通过 PCIE Group0 PHY1 配置，LCL3 通过 PCIE Group1 PHY1 配置；建议 rx 端 eb 深度参考配置为 8、6 或 4。注：该配置不会影响 LCL/PCIE 的功能使用，只会影响链路延迟

（7）对 PHY 进行的参数配置（请参照 PCIE PHY 的配置流程），PHY 的配置地址同上

```
void lcl_port_init(
    unsigned long long chip_base,
    unsigned long long lcl_offset,
    unsigned int       tgt_spd    //1: gen1, 2: gen2, 3: gen3, 4: gen4
){
    volatile unsigned int read_data, i;
    volatile unsigned long read_data64;
    unsigned int       lane_cnt = 16;

    //wait lcl initial link up (gen 1 will self-build for each lcl after reset)
    //lcl_wait_linkup(chip_base | lcl_offset | 0x10000);
    if (tgt_spd == 1)
      return ;
    else if (tgt_spd == 2) {
    //set target speed
    read_data = LCL_REG(chip_base, lcl_offset, 0xa0);
    LCL_REG(chip_base, lcl_offset, 0xa0) = (read_data & ~0xf | tgt_spd);
    }
    else {
```

```
//set target speed
read_data = LCL_REG(chip_base, lcl_offset, 0xa0);
LCL_REG(chip_base, lcl_offset, 0xa0) = (read_data & ~0xf | tgt_spd);
#ifndef LCL_EQ_23
//disable EQ23 gen3
read_data = LCL_REG(chip_base, lcl_offset, 0x890);
LCL_REG(chip_base, lcl_offset, 0x890) = (read_data | 0x200);
//disable EQ23 gen4
read_data = LCL_REG(chip_base, lcl_offset, 0x894);
LCL_REG(chip_base, lcl_offset, 0x894) = (read_data | 0x200);
#else
// EQ23 search mode:  Direction Change (0x0), Figure Merit (0x1), Customer Mode (0x2)
read_data = LCL_REG(chip_base, lcl_offset, 0x890);
LCL_REG(chip_base, lcl_offset, 0x890) = (read_data & 0xfcffffff);
// EQ23 search mode(gen4) :  Direction Change (0x0), Figure Merit (0x1), Customer Mode (0x2)
read_data = LCL_REG(chip_base, lcl_offset, 0x894);
LCL_REG(chip_base, lcl_offset, 0x894) = (read_data & 0xfcffffff);
#endif
//set gen3 preset
for(i = 0; i < lane_cnt/2; i = i+1){
    LCL_REG(chip_base, lcl_offset, (0x1dc+i*4)) = 0x0;
}
//set gen4 preset
for(i = 0; i < lane_cnt/4; i = i+1){
    LCL_REG(chip_base, lcl_offset, (0x220+i*4)) = 0x66666666;
}
#ifdef LCL_PIPE_OPT  // for decrease LCL transmission delay
  unsigned int lcl_phy_cfg_offset;
  unsigned int pipe_opt_reg_offset;
  #ifdef LCL_x8_mode
    pipe_opt_reg_offset = 0x12d3;
  #else
    pipe_opt_reg_offset = 0x11d3;
  #endif
if (lcl_offset == LCL0_OFFSET)
  lcl_phy_cfg_offset = LCL0_PHY_CFG_REG;
```

```
    else if (lcl_offset == LCL1_OFFSET)
      lcl_phy_cfg_offset = PCIE_G1_PHYCFG0_ADDR;
    else if (lcl_offset == LCL2_OFFSET)
      lcl_phy_cfg_offset = PCIE_G0_PHYCFG1_ADDR;
    else
      lcl_phy_cfg_offset = PCIE_G1_PHYCFG1_ADDR;
    lcl_phy_write_reg(chip_base,lcl_phy_cfg_offset,pipe_opt_reg_offset,0x4d);//  enable  pipe
optimize: eb depth 6
    #endif
    } //tgt_spd == 3 or 4
    //TODO:  need to add combo phy config process, please refer to PCIE func
}
```

15.6.5 发送通道参数配置

本节介绍LCL发送通道的参数配置过程,请注意该部分配置并不会影响LCL的功能使用,只影响性能,且在链路工作的任意时刻对参数进行修改均可立即生效。每个 LCL 端口的发送端都有一套相关寄存器，因此需要对每个启用的 LCL 端口都进行同样的配置流程。

发送通道参数配置主要分为四个方面：通道优先级、通道的防饿死计数值、FC 令牌发送频率、数据压缩使能控制，并给出基于仿真的推荐参数值。

15.6.5.1 配置通道优先级

通道优先级为固定配置,软件可以分别配置 VC_CMD_PRIO 和 VC_DATA_PRIO 寄存器的值,在向 LCL 通道控制寄存器（TX_CTRL）的 Channel Priority Update 位写入 1 后优先级生效。

根据对跨片请求传输的特点，配置原则如下：

（1）L2 网络传输一致性请求，优先级大于 L1 网络

（2）响应通道优先级应当大于请求通道

（3）L1AR 通道传输 DMA 一致性请求在 L1 内部应当具有高优先级

基于上述原则，作为一种可行的配置，硬件设置的命令通道默认优先级为：

L2R>L2B>L2AR>L2W>L1AR>L1B>L1W>L1R

数据通道的默认优先级为：

L2R>L2W>L1AR>L1R>L1W

15.6.5.2 通道防饿死计数值

软件通过配置 TX_CMD_AIV0、TX_CMD_AIV1、TX_DATA_AIV0、TX_DATA_AIV1 寄存器来修改命令通道和数据通道的放饿死计数器初值，该修改立即生效。推荐防饿死计数器值可以按照下表进行配置。

表 15-9 防饿死计数器推荐值

VC	数据通道	命令通道
L1AR	3	3
L1R	15	15
L1W	5	5
L1B	-	15
L2AR	-	4
L2R	15	8
L2W	8	4
L2B	-	4

15.6.5.3 FC 令牌发送频率

FC 令牌的发送频率配置可通过八个虚通道的流控信用频率配置寄存器 FCFC 和 FCFE 来调整。默认情况下，LCL 端口使用默认的流量控制频率进行令牌发送，发送间隔可能较大造成性能瓶颈。当通过 FCFE 寄存器对 FCFC 频率设置使能后，令牌发送频率按照 FCFC 频率设置进行。配置流程为先修改 FCFC 中各个通道的频率设置，再将该设置通过 FCFE 寄存器的对应位置该频率设置，软件配置可以在任意时刻进行，且立即生效。

L2 各个通道推荐的流量控制频率值配置为 0x05，即频率计数器值为 0x50，L1AR 通道的性能可以参照 L2 进行，其余通道可按照默认值即可。

15.6.5.4 数据压缩使能控制

在 TX_CTRL 寄存器中可对 L2R 和 L2W 通道的数据压缩进行控制。默认情况下，L2R 和 L2W 数据压缩均会使能，且两种算法均会启用，且数据压缩模式为至少压缩长度小于 16DW 就会进行压缩输出。bit5、bit6 以及 bit13、bit14 分别控制两个通道的压缩引擎使能和压缩模式，而 bit8、bit9 是对所有数据压缩引擎的控制。软件配置可以在任意时刻进行，且立即生效。

15.6.6 Gen3/Gen4 链路建立

在完成上面几节规定的参数配置后，软件可以向 LCL 主端口发起 Gen3/Gen4 的链路训练过程，流程如下：

（1）如果目标速率是 Gen2 或 Gen3，则直接通过 G2_CTRL 寄存器的 Directed Speed Change 位写 1 控制向 Gen2/Gen3 速率跳转

（2）如果目标速率是 Gen4，先等待链路速率到达 Gen3 后，再向 G2_CTRL 寄存器中（0x80c）Directed Speed Change 位写 1 控制向 Gen4 速率跳转

（3）轮询 LINK_CSR 寄存器的 Link Speed，等待链路速率达到设定的速率

（4）轮询 PSR1 寄存器，直到第 8 位为 0xd1（数据链路 link up、物理链路 link up 和 LTSSM 状态为 L0）。

对所有芯片的 LCL 主端口依次发起上述流程，使得每个启用的 LCL 链路都到达目标速率。对端口初始化和链路建立过程的参考代码如下：

```
//双片
  lcl_port_init(CHIP0_BASE, LCL0_OFFSET, tgt_spd);
  lcl_port_init(CHIP2_BASE, LCL0_OFFSET, tgt_spd);
  lcl_change_speed(CHIP0_BASE, LCL0_OFFSET);

void lcl_change_speed( // one side do this function is enough
    unsigned long long chip_base,
    unsigned long long lcl_offset
)
{
    volatile unsigned int read_data, i;
    unsigned int       tgt_spd;    //1: gen1, 2: gen2, 3: gen3, 4: gen4
   // read target speed
   read_data = LCL_REG(chip_base, lcl_offset, 0xa0);
   tgt_spd = read_data & 0xf;
   if (tgt_spd != 1){
   //set direct speed change
   read_data = LCL_REG(chip_base, lcl_offset, 0x80c);
   LCL_REG(chip_base, lcl_offset, 0x80c) = (read_data | 0x20000);
   }
   if (tgt_spd == 4) {
     lcl_wait_speedchange(chip_base | lcl_offset, 3);// wait speed change to gen3
```

```
        lcl_wait_linkup(chip_base | lcl_offset | 0x10000); // wait link up at gen 3
        read_data = LCL_REG(chip_base, lcl_offset, 0x80c); // direct speed change to gen 4
        LCL_REG(chip_base, lcl_offset, 0x80c) = (read_data | 0x20000);
    }
    lcl_wait_speedchange(chip_base | lcl_offset, tgt_spd);// wait speed change to tgt_spd
    lcl_wait_linkup(chip_base | lcl_offset | 0x10000); // wait link up
}
```

16 低速 IO 控制器配置

低速 I/O 控制器包括 UART 控制器、SPI 控制器、I2C 及 AVS 寄存器。这些 I/O 控制器共享一个 AXI 端口，控制器频率与 SYSCLK 相同，一般情况下为 100MHz。

16.1 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。

针对这两个 UART 还各提供一个物理地址，分别为 0x1FE00100(UART0) 和 0x1FE00110(UART1)。通过这组地址可访问新增的两个寄存器 RFC 和 TFC。

16.1.1 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

16.1.2 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

16.1.3 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

16.1.4 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

16.1.5 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

16.1.6 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

16.1.7 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。

16.1.8 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

16.1.9 接收 FIFO 计数值（RFC）

中文名： 接收 FIFO 计数值

寄存器位宽： [7：0]

偏移量： 0x08

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	RFC	8	R	反映当前接收 FIFO 中有效数据个数

16.1.10 发送 FIFO 计数值（TFC）

中文名： 发送 FIFO 计数值

寄存器位宽： [7：0]

偏移量： 0x09

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	TFC	8	R	反映当前发送 FIFO 中有效数据个数

16.1.11 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

中文名：分频锁存器 3

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	D_DIV	8	RW	存放分频锁存器的小数分频值

16.1.12 新增寄存器的使用

新增的接收 FIFO 计数器（RFC）可供 CPU 检测接收 FIFO 中有效数据的个数，据此 CPU 可在收到一次中断后连续读取多个数据，提高 CPU 处理 UART 接收数据的能力；

发送 FIFO 计数器（TFC）可供 CPU 检测发送 FIFO 中有效数据的个数，据此 CPU 可在保证发送 FIFO 不溢出的前提下连续发送多个数据，提高 CPU 处理 UART 发送数据的能力；

分频锁存器 3（即小数分频寄存器）用于解决仅用整数除法无法精确得到所需波特率的问题。以参考时钟 100MHz 除以 16，再除以波特率，所得商整数部分赋值给由分频器锁存器 MSB 和 LSB，小数部分乘以 256 赋值给分频锁存器 D_DIV。

16.2 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
- 支持 Dual/Quad mode SPI flash

SPI 控制器寄存器物理地址基址为 0x1FE001F0。

表 16-1 SPI 控制器地址空间分布

地址名称	地址范围	大小
SPI Memory	0x1C00_0000-0x1D00_0000	16MB
SPI Register	0x1FE0_01F0-0x1FE0_01FF	16B

SPI Memory 地址空间是系统启动时处理器最先访问的地址空间， 0x1C000000 的地址被自动路由至 SPI。SPI Memory 空间可以通过 CPU 的读请求直接访问。

当需要对 SPI 进行其它操作时，比如发送命令，擦除 SPI Flash 等时，就要使用 SPI Register 空间对控制寄存器进行直接操作。

16.2.1 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

16.2.2 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

16.2.3 数据寄存器（TxFIFO/RxFIFO）

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 Rx FIFO	8	W R	数据发送寄存器 数据接收寄存器

16.2.4 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 - 4 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 spr	00 00	00 01	00 10	00 11	01 00	01 01	01 10	01 11	10 00	10 01	10 10	10 11
分频系数	2	4	16	32	8	64	128	256	512	1024	2048	4096

16.2.5 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。

16.2.6 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

16.2.7 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
7:4	samp_dly	4	RW	采样延迟，用于调整读的时序 1 表示延迟一个单位 2 表示延迟两个单位，以此类推
3	quad_io	1	RW	4 线模式使能，1 有效
2	tFast	1	RW	
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

16.2.8 自定义控制寄存器（CTRL）

中文名：　　SPI Flash 自定义控制寄存器

寄存器位宽：　[7：0]

偏移量：　　0x08

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:4	nbyte	4	RW	一次传输的字节数
3:2	reserve	2	RW	保留
1	nbmode	1	RW	多字节传输模式
0	start	1	RW	开始多字节传输，完成后自动清零

16.2.9 自定义命令寄存器（CMD）

中文名：　　SPI Flash 自定义命令寄存器

寄存器位宽：　[7：0]

偏移量：　　0x09

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	cmd	8	RW	设置发送给 spi flash 的命令

16.2.10 自定义数据寄存器 0（BUF0）

中文名：　　SPI Flash 自定义数据寄存器 0

寄存器位宽：　[7：0]

偏移量：　　0x0a

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	buf0	8	RW	向 SPI 发送写命令时，该寄存器配置发送的第一个字节的数据；向 SPI 发送读命令时，该寄存器存储第一个读回来的数据。

16.2.11 自定义数据寄存器 1（BUF1）

中文名：　　SPI Flash 自定义数据寄存器 1

寄存器位宽：　[7：0]

偏移量：　　0x0b

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	buf1	8	RW	向 SPI 发送写命令时，该寄存器配置发送的第二个字节的数据；向 SPI 发送读命令时，该寄存器存储第二个读回来的数据。

16.2.12 自定义时序寄存器 0（TIMER0）

中文名：　　SPI Flash 自定义时序寄存器 0

寄存器位宽：　[7：0]

偏移量：　　0x0c

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	time0	8	RW	自定义命令所需时间值的低 8 位

16.2.13 自定义时序寄存器 1（TIMER1）

中文名：　　SPI Flash 自定义时序寄存器 1

寄存器位宽：　[7：0]

偏移量：　　0x0d

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	time1	8	RW	自定义命令所需时间值的中间 8 位

16.2.14 自定义时序寄存器 2（TIMER2）

中文名：　　SPI Flash 自定义时序寄存器 2

寄存器位宽：　[7：0]

偏移量：　　0x0e

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	time2	8	RW	自定义命令所需时间值的高 8 位

16.2.15 SPI 双线四线使用指南

除了传统的单线模式，SPI 控制器还支持以双线(dual mode)和四线（quad mode）两种工作模式从 SPI flash 启动。通过设置 dual_io 寄存器可以使 SPI 控制器进入双线模式，设置 quad_io 寄存器可以使 SPI 控制器进入四线模式。可以在 BIOS 代码的前几条指令中增加对这两个寄存器的配置代码，配置完成后控制器即按照配置对应的工作模式进行取指，以此可提高开机速度。

需要注意的是，有的 SPI FLASH 默认并没有使能四线模式，或者在四线模式下需要配置时序相关的参数(如 dummy clocks)。为了增加 SPI 控制器对各种 FLASH 的适用性，本控制器增加自定义的寄存器(0x8-0xe)。其具体使用方法为：

1. 设置自定义命令寄存器（CMD）（0x9），该寄存器为向 SPI FLASH 发送的命令；
2. 如果 SPI FLASH 要求本次发送的命令需要过一段时间才完成，则把等待的时间配置到自定义时序寄存器 TIMER0-TIMER2(0xc-0xe)中，否则这些寄存器保持默认值 0；
3. 如果向 SPI FLASH 写配置信息，则需要把配置信息写入自定义数据寄存器 BUF0-BUF1(0xa-0xb)；如果向 SPI FLASH 读配置信息，则这两个寄存器存储读回来的值；
4. 配置自定义控制寄存器 CTRL[7：1]其中 CTRL[1](nbmode)代表将进行多字节传输模式，此次传输字节数通过 CTRL[7:4](nbyte)给定；

5. 配置自定义控制寄存器 CTRL[0]开始此次传输。

一般来说，所需要配置的寄存器位于 FLASH 的非易失性存储区，所以以上配置仅需要配置一次。

16.3 I2C 控制器

本章给出 I2C 的详细描述和配置使用。本芯片集成了 I2C 接口，主要用于实现与其它器件的数据交换。I2C 总线是由数据线 SDA 和时钟 SCL 构成的串行总线，可发送和接收数据。器件与器件之间进行双向传送，最高传送速率 400kbps。

龙芯 3C6000 中集成的 I2C 控制器既可以作为主设备，也可以作为从设备，这两种模式之间通过配置内部寄存器（偏移 0x7）进行切换。作为从设备时，仅用于读取芯片内部温度，从设备的地址由寄存器 SLV_CTRL[6:0]指定。

I2C0 控制器寄存器物理地址基址为 0x1FE00120。

I2C1 控制器寄存器物理地址基址为 0x1FE00130。

I2C2 控制器寄存器物理地址基址为 0x1FE00138。

下面对具体的内部寄存器进行说明。

16.3.1 分频锁存器低字节寄存器（PRERlo）

中文名： 分频锁存器低字节寄存器

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0xff

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

16.3.2 分频锁存器高字节寄存器（PRERhi）

中文名： 分频锁存器高字节寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0xff

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

假设分频锁存器的值为 prescale，总线时钟输入的频率为固定 100MHz，总线的输出频

率为 clock_s（MHz），则应满足如下关系：

prcescale = 100/(4*clock_s)-1

16.3.3 控制寄存器（CTR）

中文名： 控制寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x20

位域	位域名称	位宽	访问	描述
7	EN	1	RW	模块工作使能位 为 1 正常工作模式, 0 对分频寄存器进行操作
6	IEN	1	RW	中断使能位为 1 则打开中断
5	MST_EN	1	RW	模块主从选择 0：slave 模式 1：master 模式
4:0	Reserved	5	RW	保留

16.3.4 发送数据寄存器（TXR）

中文名： 发送寄存器

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
7:1	DATA	7	W	存放下个将要发送的字节
0	DRW	1	W	当数据传送时，该位保存的是数据的最低位； 当地址传送时，该位指示读写状态

16.3.5 接收数据寄存器（RXR）

中文名： 接收寄存器

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
7:0	RXR	8	R	存放最后一个接收到的字节

16.3.6 命令控制寄存器（CR）

中文名：　　命令寄存器

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	STA	1	W	产生 START 信号
6	STO	1	W	产生 STOP 信号
5	RD	1	W	产生读信号
4	WR	1	W	产生写信号
3	ACK	1	W	产生应答信号
2:1	Reserved	2	W	保留
0	IACK	1	W	产生中断应答信号

该寄存器都是在 I2C 发送数据后硬件自动清零。对这些位读操作时候总是读回 0。

例如，对[3]写 1 时表示此次传输结束时控制器不发送 ack，反之结束时发送 ack。

16.3.7 状态寄存器（SR）

中文名：　　状态寄存器

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	RxACK	1	R	收到应答位 1 没收到应答位 0 收到应答位
6	Busy	1	R	I2c 总线忙标志位 1 总线在忙 0 总线空闲
5	AL	1	R	当 I2C 核失去 I2C 总线控制权时，该位置 1
4:2	Reserved	3	R	保留
1	TIP	1	R	指示传输的过程 1 表示正在传输数据 0 表示数据传输完毕
0	IF	1	R	中断标志位，一个数据传输完，或另外一个器件发起数据传输，该位置 1

16.3.8 从设备控制寄存器（SLV_CTRL）

中文名：　　从设备控制寄存器

寄存器位宽：　[7：0]

偏移量：　　0x07

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7	SLV_EN	1	WR	从模式使能，当 MST_EN 为 0 时起作用，可用于复位从机内部逻辑
6：0	SLV_ADDR	7	WR	从模式 I2C 地址，可通过软件配置

16.4 AVS 控制器

AVS 控制器寄存器物理地址基址为 0x1FE00160。

表 16-2 AVS 控制器地址空间分布

地址名称	地址范围	大小
AVS Register	0x1FE0_0160-0x1FE0_016F	16B

16.4.1 控制寄存器（CSR）

中文名：　　控制寄存器

寄存器位宽：　[31：0]

偏移量：　　0x0

复位值：　　0x0

位域	位域名称	位宽	访问	描述
31	resyn	1	RW	1 表示在对设备进行通信前，先对设备进行重同步；0 则不进行重同步；默认为 0
30:29	mask_ack	2	RW	全写 1 表示 mask alert_ack
28	mask_i	1	RW	1 表示 mask alert_i
27	mask_s	1	RW	1 表示 mask alert_s
26	mask_c	1	RW	1 表示 mask alert_c
25	mask_a	1	RW	1 表示 mask alert_a
24	rx_ctrl	1	RW	控制器采样数据的方式，1 表示在 AVS 时钟的上沿开始采样数据，0 表示在 AVS 时钟的下沿开始采样数据；默认为 0
23:20	rx_delay	4	RW	延迟采样，1 表示延迟一个单位后采样数据；2 表示延迟两个单位后采样数据，以此类推；默认为 0

19:17	clk_div	3	RW	时钟分频系数：0x0，二分频；0x1，四分频；0x2，八分频；0x3，十六分频；0x4，三十二分频；0x5，六十四分频；默认二分频
16	Dmux	1	RW	1 表示接受设备返回的数据，0 反之。默认为 0
15:0	reserved	16	—	—

16.4.2 参数控制寄存器（Mreg）

中文名：　　参数控制寄存器

寄存器位宽：　[31：0]

偏移量：　　0x4

复位值：　　0x0

位域	位域名称	位宽	访问	描述
31	TX_EN	1	RW	写 1 表示开始一次 AVS 通信
30:29	cmd	2	RW	0x0 表示写入设备并且生效；0x1 表示写入并保持，且不立即生效；0x2 保留；0x3 表示读。对于写，建议使 cmd 为 0
28	group	1	RW	0 为 AVS 协议指定的指令类别;1 表示制造厂商自定义的指令类别
27:24	cmd_type	4	RW	0x0 为电压调节命令（1LSB=1mv）；0x1 为压摆率命令，cmd_data 的高八位为上摆率，低八位为下摆率；0x2 为读电流命令（只读）；0x3 为读温度命令（只读）；0x4 为复位电压至默认值命令（只写）；0x5 为功率设置命令；0x6~0xd，保留；0xe 为设备状态清除命令；0xf 为读设备版本命令
23:20	rail_sel	4	RW	轨道选择，如电压轨。如果该值为 0xf，则表示选择全部的轨道
19:4	cmd_data	16	RW	写命令需要设置的值，如设置电压值大小；读命令，该值可设置为 0；状态清除命令需要将该值全部置为 1
3:0	reserved	4	—	—

16.4.3 数据寄存器（Sreg）

中文名：　　返回数据寄存器

寄存器位宽：　[31：0]

偏移量：　　0x8

复位值：　　0x00

位域	位域名称	位宽	访问	描述
31	busy	1	R	1 表示正在进行通信或控制器正在处理数据
30:29	slave_ack	2	RW	0x0 表示 CRC 校验 OK，并且读或写有效；0x1 表示 CRC 校验 OK，数据都 OK，但是设备资源不可用（busy 等）。

				如，超出电压范围的电压设置；0x2 表示 CRC 校验错误；0x3 表示 CRC 校验 OK，但是命令无效等
28	alert_i	1	RW	1 表示出现设备中断
27	alert_s	1	RW	1 表示出现设备状态响应
26	alert_c	1	RW	1 表示 CRC 校验错误
25	alert_a	1	RW	1 表示出现设备状态响应，包含设备自定义的状态响应
24:16	reserved	9	—	—
15:0	sdata	16	R	设备返回的数据。如，读电压命令，该值为设备返回的电压值

16.4.4 使用说明

以设置电压 1.8V 为例，CSR 寄存器设为 0x70000（下沿采样数据，16 分频，参考频率为 100MHz）；然后将 Mreg 设为 0x80007080；等待 busy 为 0，最后读取 slave_ack 寄存器，若为 0，则表示设置成功。读电压需要设置 CSR 寄存器设为 0x70000；然后将 Mreg 设为 0xe0000000；等待 busy 为 0，最后读取 slave_ack 寄存器，若为 0，则表示读取成功，sdata 为设备返回的电压值。假如在一次通信中设备未发送 status frame，会出现 alert_s，alert_c，alert_a 一直被拉高的情况，可将这几位 mask 掉。当时钟分频为 2~8(50~12.5MHz)时，需要设置 rx_delay 的值，经验值为 1 或 2。

修订记录

版本号	更新内容
V1.0	发布版本

技术支持

可通过邮箱向我司提交芯片手册和产品使用的问题，并获取技术支持。

服务邮箱：service@loongson.cn

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