ICH2 硬件
接口手册

2017 年 10 月

成都申威科技有限责任公司

成都市华府大道四段电子科大科技园 D22 栋
Building D22, National University Science and technology park,
Section 4, Huafu Avenue, Chengdu

免责声明

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

成都申威科技有限责任公司

Chengdu Sunway Technology Corporation Limited

地址：成都市华府大道四段电子科大科技园 D22 栋

Building D22, National University Science and technology park,

Section 4, Huafu Avenue, Chengdu

Mail：sales@swcpu.cn

Tel ：028-68769016

Fax：028-68769019

阅读指南

《ICH2 硬件接口手册》主要描述了国产 2 代套片 ICH2 的整体结构、信号说明、外设接口等内容，并对外设接口进行了详细的说明。

文档修订

文档更新记录	文档名	ICH2 硬件接口手册
	版本号	V1.0
	创建人	研发部
	创建日期	2017-10-8

版本更新

版本号	更新内容	更新日期
V1.0	初稿	2017-10-8

技术支持

可通过邮箱或问题反馈网站向我司提交产品使用的问题，并获取技术支持。

售后服务邮箱：sales@swcpu.cn

问题反馈网址：http://www.swcpu.cn/

目　录

1 概述 1
1.1. 芯片总体架构 1
1.1.1. 全片结构 1
1.1.2. 解决方案 2
1.2. 硬件接口简述 3
1.3. 信号说明 3
1.3.1. 管脚数量汇总表 3
1.3.2. 电源汇总表 4
1.3.3. 时钟汇总表 9
1.3.4. AC97 10
1.3.5. DC 11
1.3.6. GMAC 14
1.3.7. 时钟配置 17
1.3.8. 维护/测试/观测 17
1.3.9. SIO（低速 IO：wbsys+lpc） 18
1.3.10. MC 20
1.3.11. PCIE-SW 24
1.3.12. SATA 32
1.3.13. USB 35
1.3.14. 复位/杂项信号 38
1.3.15. ICH2 复位时序 39
2 PCI-E 接口 41
2.1. PCI-E 接口概述 41
2.2. 详细说明 41
2.2.1. 名词解释 42
2.2.2. 信号管脚 42
2.2.3. 热插拔信号 43
2.2.4. 参考电阻信号 44
2.3. 应用要求 44
2.3.1. 参考时钟要求 44
2.3.2. 核心工作时钟要求 44
2.3.3. 延迟和电气要求 45
2.3.4. 端接要求 45
2.3.5. 功耗要求 45
2.4. 初始化与测试 46
2.4.1. 初始化 46
2.4.2. 测试 47
2.5. 交直流电气特性 47
2.5.1. 发送特征参数 47
2.5.2. 接收特征参数 51
3 以太网接口 55
3.1. 以太网接口概述 55
3.2. 管脚说明 56
3.3. 使用方式 57
3.3.1. 板级逻辑示意图 57
3.3.2. 端口类型选择 57
3.3.3. 时钟选择 57
3.3.4. 数据信号 58
3.4. 初始化与测试 58
3.4.1. DMA 初始化–描述符 58

3.4.2. MAC 初始化 59
3.4.3. 正常接收操作 60
3.4.4. 停止和开始操作 61
3.4.5. 测试 61
3.5. 接口时序特性 61
3.6. 交直流电气特性 63
4 SATA 接口 64
4.1. 功能概述 64
4.2. 管脚说明 65
4.3. 使用方式 67
4.3.1. 板级逻辑示意图 67
4.3.2. 数据链路 67
4.3.3. 参考时钟和参考电阻 68
4.3.4. 在位信号 68
4.3.5. 状态 LED 69
4.3.6. 机械在位切换信号 72
4.3.7. 冷在位上电信号 73
4.4. 初始化与测试 73
4.4.1. 软件初始化 73
4.4.2. 端口 DMA 的软件控制 74
4.5. 交直流电气特征 76
5 USB 接口 78
5.1. USB 接口概述 78
5.2. 管脚说明 78
5.3. 初始化与测试 78
5.3.1. XHC 初始化 78
5.3.2. 设备初始化 79
5.3.3. 边界扫描（BSR）测试 83
5.3.4. ATPG（SCAN 链）测试 85
5.3.5. JTAG 测试 86
5.4. 交直流电气特性 87
6 存储器接口 88
6.1. 存储器接口概述 88
6.2. 详细说明 88
6.2.1. 名词解释 88
6.2.2. 信号管脚 90
6.3. 使用方式 90
6.3.1. 地址映射 90
6.4. 初始化与测试 91
6.4.1. 初始化 92
6.4.2. 测试 94
6.5. 交直流电气特性 94
7 显示 DVO 接口 95
7.1. DVO 接口概述 95
7.2. 信号管脚说明 95
7.3. 外接要求 96
7.4. 交直流电气特性 97
8 声卡接口 98
8.1. 声卡接口概述 98
8.2. 详细说明 98
8.2.1. AC97 接口协议 98

8.2.2. 声卡接口信号 100
8.3. 初始化与测试 100
8.3.1. 初始化 100
8.3.2. 从模式传输 100
8.3.3. 主模式传输 101
8.4. 交直流电气特性 101
9 SIO 接口 **102**
9.1. LPC 接口 102
9.1.1. LPC 协议概述 102
9.1.2. 接口信号说明 104
9.1.3. 初始化与测试 104
9.1.4. 布局布线要求 106
9.1.5. 电气特性 106
9.2. SPI 接口 106
9.2.1. SPI 接口概述 106
9.2.2. 详细说明 106
9.2.3. 布局布线要求 109
10 时钟接口 **112**
10.1. 时钟接口概述 112
10.1.1. 工作模式 112
10.1.2. 信号管脚 114
10.2. 使用方式 115
10.2.1. 显示部件时钟配置 115
10.2.2. 存储控制器时钟配置 116
10.2.3. PCIE 交叉开关时钟配置 117
10.2.4. 全局工作时钟配置 118
10.3. 使用说明 119
10.4. 初始化与测试 119
10.4.1. 初始化 119
10.4.2. 测试 120
10.5. 交直流电气特性 123
11 测试接口 **124**
11.1. 测试接口概述 124
11.2. 详细说明 124
11.2.1. TAP 控制器 125
11.2.2. 指令寄存器 127
11.2.3. 数据寄存器 127
11.2.4. 结构和测试流程 130
11.3. 交直流电气特性 148
12 电气特性 **149**
12.1. 电气特性概述 149
12.2. 电地说明 149
12.2.1. DDR3 接口 149
12.2.2. PCI-E 接口 150
12.2.3. USB 接口 151
12.2.4. SATA 接口 151
12.3. 电流和功耗 151
13 封装与管脚 **157**
13.1. 封装与管脚概述 157
13.2. 封装说明 157
13.3. 引脚编号 161

13.4. 焊接要求 171
附录 A（存控训练） 172
附录 B（固件内容） 213
附录 C（DS5 端口训练问题板级解决方案） 221

1 概述

1.1. 芯片总体架构

ICH2 是一款完全自主设计的系统芯片，采用 40nm 工艺生产。ICH2 集成了多层次片上 通信结构、高性能 PCI-E 交叉开关（支持 PCI-E Gen2，共 32Lane，其中 24 Lane 可用于 PCI-E扩展）、图形图像子系统(包括显示控制部件、二维/三维图形加速部件、H.264 编解码部件、MPEG-1/2/4 高清媒体解码加速部件)、DDR2/3 存储控制器、高速输入输出部件(包括 10M/100M/1G 以太网控制器、USB 3.0 多端口控制器、SATA3.0 多端口控制器、多通道高 速 DMA 控制器)、低速输入输出部件(包括 AC‘97 控制器、UART 控制器、I2C 控制器、SPI控制器、PS/2 键盘鼠标控制器、LPC Host 控制器、GPIO 与可编程中断控制器)以及系统控 制部件等。

ICH2 系统接口符合各种规范接口要求，可广泛应用于具有标准 PCI-E 2.0 总线的所有国 产化平台产品中。

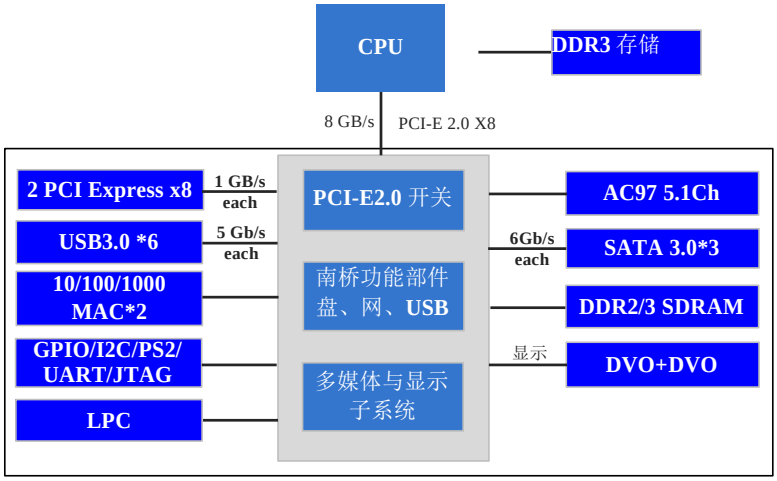


图 1-1 ICH2 结构示意图

1.1.1. 全片结构

ICH2 是一个 IO 控制型芯片，由多层次片上总线将 IO 控制部件互连起来。ICH2 的基本 结构如图 1-2 所示。

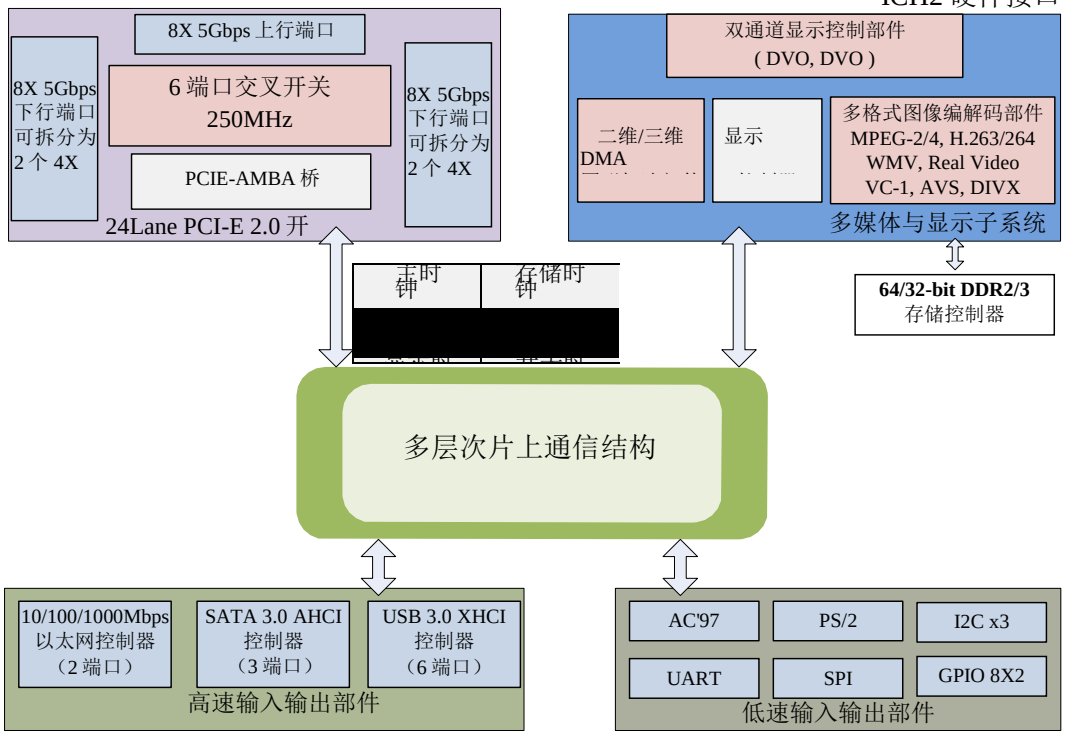


图 1-2 ICH2 基本结构框图

ICH2 芯片的功能如下：

1） 集成 PCI-E 开关，1 个 X8 上行端口，2 个 X8 下行端口（可拆分为 2 个 X4 端口），1 个 X8 下行端口（不可拆分，且仅当芯片配置为当 PCIE-E 开关使用时）；
2） 集成 SATAIII AHCI 控制器，提供 3 个 SATAIII 6Gbps 接口；
3） 图形图像子系统为独立显存架构，包含 DDR3 显存（支持 64/32 位数据宽度、8/16 位颗粒 ）、多通道高速 DMA 控制器（支持主存和显存之间的数据传输）、图形加速单元 GPU（支持 2D/3D 硬件加速）、视频加速单元 VPU（支持 MPEG-2/4、 H.263/264、 WMV、 Real Video、VC-1、AVS、DIVX 视频格式硬件编解码）、双通道标准 DVO 显示输出接口（支持 1080P 高清显示）、双通道 1080P 独立显示；
4） 集成 2 个 10/100/1000M 自适应以太网 GMAC 接口（不集成 PHY），支持 IEEE 802.3 标准和以太网标准，支持 GMII 与/RGMII 两种接口；
5） 集成 6 个 USB 3.0 接口，支持 5Gbps/480Mbps/12Mbps 三种速率；
6） 集成 1 个 64 位或 32 位宽 DDR2/3 存储控制器；
7） 集成 AC‘97 声卡接口，提供 5.1 声道；
8） 集成标准 JTAG 维护接口，可实现运行状态监测与芯片在线维护；
9） 集成自引导部件，可通过 SPI Flash 或 E2PROM 对芯片时钟频率、上电时序、工作模式等进行灵活的上电自引导操作。

1.1.2. 解决方案

ICH2 将为国产安全平台提供高集成度的系统解决方案。如图 1-3 所示的 ICH2 芯片系统 解决方案，其中显示接口 DVO 需要外接 VGA DAC 或 HDMI 接口芯片、LVDS 接口芯片来 适用不同的显示器，AC97 AC-link 需要外接 AC97 CODEC 芯片接音响，以太网控制器要通 过 GMII 接口

接 10/100/1000M MAC PHY 外，其它接口都集成的 PHY，可以直接出相关设备接口。ICH2 出 3 个 SATA III 接口，可以接 SSD、SATA 磁盘、SATA 光驱；最多可以扩 展出 3 个 PCI-E 8X Gen2（或 4 个 PCI-E 4X Gen2+1 个 PCI-E 8X Gen2）接口，用于 PCI-E 显卡、网卡等设备扩展；可扩展出 6 个 USB3.0（兼容 USB2.0/USB1.0）的设备接口，可外 接摄像头、WIFI 模块、蓝牙模块等 USB 外设。

ICH2 采用 PCI-E Gen2 8X 的系统接口，芯片内各设备接口采用商用 IP 实现，符合标准 规范。ICH2 可应用到各种采用 PCI-E Gen2 作为 IO 接口的处理器系统中。

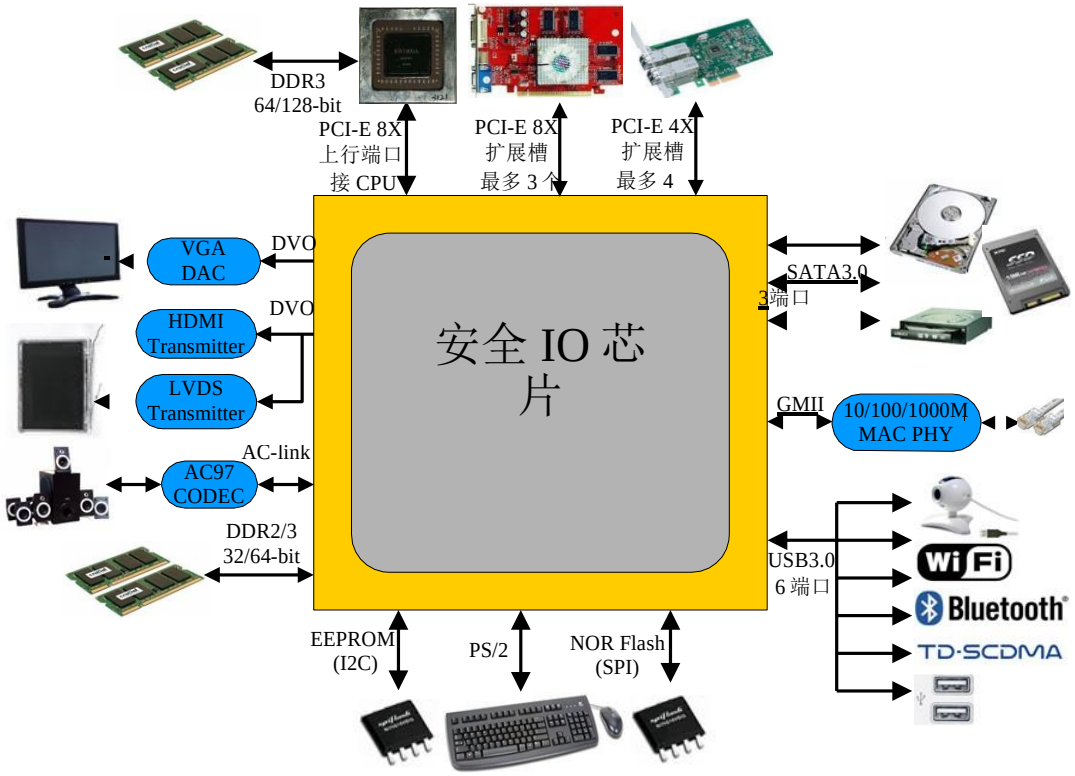


图 1-3 安全 IO 芯片 ICH2 系统解决方案

1.2. 硬件接口简述

ICH2 是一款高性能外围接口，内部集成了 1 个 32/64 位 DDR3 SDRAM 控制器，可实 现 10.664GB/s 峰值访存带宽；四个标准 PCI-E 2.0 ×8 接口，可实现双向 32GB/s 扩展 IO 带 宽；两个 1000/100/10Mb/s 以太网接口，可实现双向 250MB/s 访问带宽；3 个 SATA3.0 接口，可实现双向 4.5GB/s 访问带宽；6 个 USB3.0 接口，可实现双向 6GB/s 访问带宽；集成 2 个 24 位 DVO 接口，最高工作频率 180MHz。另外芯片对外提供标准 JTAG 维护与测试接口。

ICH2 采用 FCBGA（Flip Chip Ball Grid Array）封装，对外接口引脚总数为 1296 根。

1.3. 信号说明

1.3.1. 管脚数量汇总表

表 1-1 管脚汇总表

功能	输	输	双向	模拟电	管脚数
时钟配置	14	0	0	8	22
复位	2	1	0	0	3
维护/测试/观测	7	3	0	0	10
PCIE_SW	81	70	0	40	191
AC97	2	3	0	0	5
DC	4	56	0	0	60
GMAC	32	24	2	0	58
SATA	14	12	3	6	35
USB	38	14	12	15	79
MC(DDR3)	0	30	93	6	129
LPC	3	2	5	0	10
IIC_MASTER	0	0	6	0	6
SPI（flash）	2	3	0	0	5
UART	6	3	0	0	9
PS2	0	0	4	0	4
GPIO	0	0	16	0	16
保留	1	0	3	0	4
核心数字电 VDD=0.9V					128
LVDS_IO 数字电 LVDS_VDDQ18=1.8V					2
IO 数字电 VDDQ=1.8V					35
IO 数字电 VDDQ15=1.5V					19
数字地 VSS					466
总计	206	22	144	75	1296

1.3.2. 电源汇总表

表 1-2 电源汇总表

信号功能	信号名称	管脚数量	工作电压	信号说明	管脚电流	功耗
核心电源	VDD	128	0.95V	设计要求是核心数字 0.9V，但测试发现需要做到0.95V才能保证芯片正常运行	5A	4.5W
数字地	VSS	466	GND	数字地		
PLL 模拟电源	PLL_VDD09A_0	1	0.9V	GPUPLL 的模拟 0.9V 供电，建议 与其他电源之间 用磁珠	2.5mA	0.00225W
	PLL_VDD09A_1	1	0.9V	MCPLL 的模拟 0.9V 供电，建议 与其他电源之间 用磁珠	2.5mA	0.00225W
	PLL_VDD09A_2	1	0.9V	PSWPLL 的模拟 0.9V 供电，建议 与其他电源之间 用磁珠	2.5mA	0.00225W
	PLL_VDD09A_3	1	0.9V	GCLKPLL 的模拟 0.9V 供电，建议 与其他电	2.5mA	0.00225W
	PLL_VSS09A_0	1	GND	连接数字地 VSS	-	-
	PLL_VSS09A_1	1	GND	连接数字地 VSS	-	-
	PLL_VSS09A_2	1	GND	连接数字地 VSS	-	-
	PLL_VSS09A_3	1	GND	连接数字地 VSS	-	-
IO 电源	LVDS_VDDQ18	1	1.8V	LVDS(差分转单端 IOBUF)类型 IO 的 1.8V 供电，建 议与其他电	-	-
	LVDS_VSSQ18	1	GND	连接数字地 VSS。	-	-
	VDDQ15	19	1.5V	DDR3 接口 IO 数	2.5A	3.8W
	VDDQ	35	1.8V	其他 GPIO 接口 IO 数字 1.8V	500mA	0.9W
PSW 电源	PCIe_US_VDDA	4	0.9V	US 端口模拟电压 0.9V，建议与其他电源之间用	203mA	0.18W
	PCIe_US_VDDQA	4	1.8V	US 端口模拟电压 1.8V，建议与其他电源之间用	125mA	0.225W

	PCIe_US_VTT	4	1.5V	US 端口模拟电压 1.5V，建议与其他电源之间用	240mA	0.36W
	PCIe_DS13_VDDA	4	0.9V	DS13 端口模拟电压 0.9V，建议与 其他电源	203mA	0.18W
	PCIe_ DS13_VDDQA	4	1.8V	DS13 端口模拟电压 1.8V，建议与 其他电源	125mA	0.225W
	PCIe_ DS13_VTT	4	1.5V	DS13 端口模拟电压 1.5V，建议与 其他电源	240mA	0.36W
	PCIe_DS24_VDDA	4	0.9V	DS24 端口模拟电压 0.9V，建议与 其他电源	203mA	0.18W
	PCIe_ DS24_VDDQA	4	1.8V	DS24 端口模拟电压 1.8V，建议与 其他电源	125mA	0.225W
	PCIe_ DS24_VTT	4	1.5V	DS24 端口模拟电压 1.5V，建议与 其他电源	240mA	0.36W
	PCIe_DS5_VDDA	4	0.9V	DS5 端口模拟电	203mA	0.18W
				压 0.9V，建议与 其他电源之间		
	PCIe_ DS5_VDDQA	4	1.8V	DS5 端口模拟电压 1.8V，建议与 其他电源之间用 磁珠隔	125mA	0.225W
	PCIe_ DS5_VTT	4	1.5V	DS5 端口模拟电压 1.5V，建议与 其他电源之间用 磁珠隔	240mA	0.36W
SATA	SATA_VP	2	0.9V（+10%，	3 个 SATA 端口的 模拟 0.9V，建议 与其他电	138mA	0.124W
	SATA_VPH	2	1.8V（+10%，	3 个 SATA 端口的 模拟 1.8V，建议 与其他电	24mA	0.043W

	SATA_VPTX	2	0.9V（+10%，-7%	3 个 SATA 端口的发送模拟 0.9V，建议与其他电源之间用磁珠隔开	138mA	0.124W
USB	USB_VP_0	2	0.9V（+10%，	USB_0/1 端口的模拟 0.9V，建议与其他电源之间用磁珠	90mA	0.081W
	USB_VP_1	2	0.9V（+10%，	USB_2/3 端口的模拟 0.9V，建议与其他电源之间用磁珠	90mA	0.081W
	USB_VP_2	2	0.9V（+10%，	USB_4/5 端口的模拟 0.9V，建议与其他电源之间用磁珠	90mA	0.081W
	USB_VDD180_0	1	1.8V（+10%，-	USB_0/1 端口的（VPH/VDD180）模拟 IO 1.8V，建议与其他电	32mA	0.057W
	USB_VDD180_1	1	1.8V（+10%，-	USB_2/3 端口的（VPH/VDD180）模拟 IO 1.8V，建议与其他电	32mA	0.057W
	USB_VDD180_2	1	1.8V（+10	USB_4/5 端口的（VPH/VDD180）	32mA	0.057W
			-7%	议与其他电源之间用磁珠隔开		
	USB_VPTX_0	1	0.9V（+10%，-7%	USB_0/1 端口的 tx 模拟 0.9V，建议与其他电源之间用磁珠隔开（建议与 VP 也用磁珠	90mA	0.081W
	USB_VPTX_1	1	0.9V（+10%，-7%	USB_2/3 端口的 tx 模拟 0.9V，建议与其他电源之间用磁珠隔开（建议与 VP 也用磁珠	90mA	0.081W

	USB_VPTX_2	1	0.9V（+10%，-7%	USB_4/5 端口的 tx 模拟 0.9V，建 议与其他电源之间用磁珠隔开（建议与 VP 也用磁珠	90mA	0.081W
	USB_VDD330_0	1	3.3V（+10%，	USB_0/1 端口的模拟 3.3V，建议 与其他电源之间 用磁珠	32mA	0.105W
	USB_VDD330_1	1	3.3V（+10%，	USB_2/3 端口的模拟 3.3V，建议 与其他电源之间 用磁珠	32mA	0.105W
	USB_VDD330_2	1	3.3V（+10%，	USB_4/5 端口的模拟 3.3V，建议 与其他电源之间 用磁珠	32mA	0.105W
MC（DDR3）	DDR3_VDDA18_0	1	1.8V（+10%，-	DATAX8_0/1/2 的 PLL 模拟 1.8V， 建议与其他电源之间用磁珠隔开	168mA	0.302W
	DDR3_VDDA18_1	1	1.8V（+10%，-	DATAX8_3/8（AC）/4 的 PLL 模拟 1.8V，建议与其他电源之间	168mA	0.302W
	DDR3_VDDA18_2	1	1.8V（+10%，-	DATAX8_5/6/7 的 PLL 模拟 1.8V， 建议与其他电源之间用磁珠隔开	168mA	0.302W
	DDR3_VSSA18_0	1	GN	连接数字地 VSS	-	-
	DDR3_VSSA18_1	1	GN	连接数字地 VSS	-	-
	DDR3_VSSA18_2	1	GN	连接数字地 VSS	-	-
总计		733				~14W

注：PCIE、SATA 和 USB 接口的模拟电源没有对应的模拟地，而是直接采用数字地 VSS 作 为模拟地，因此数据字电源和模拟电源之间需要图 1-4 所示的滤波器。

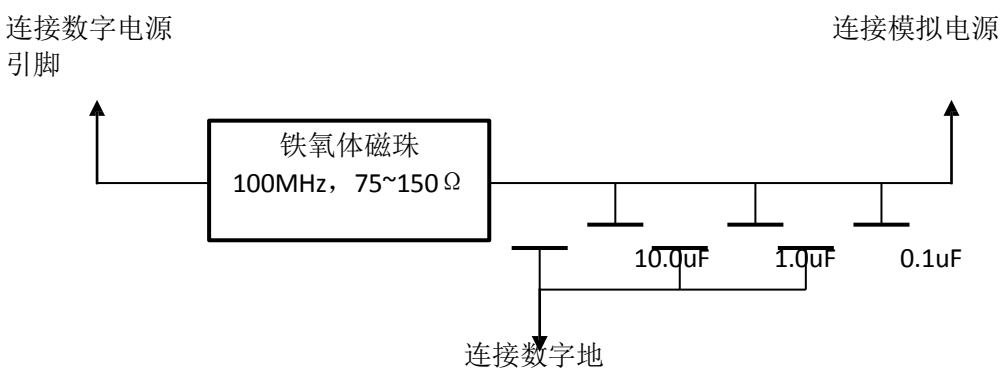


图 1-4 连接模拟电源
引脚的滤波器

1.3.3. 时钟汇总表

表 1-3 时钟汇总表

信号功能	信号名称	管脚	管脚	工作电压	信号频率	信号说明	上下拉/驱动能
全局 PLL	REFCLK_N	I	2	1.8V/2.5V	100MHz	全局工作时钟 PLL 的	
	REFCLK_P						
JTAG	TCK	I	1	1.8V/2.5V	10MHz	JTAG 时钟	输入片内上拉
AC97	AC97_BIT_CLK	I	1	1.8V/2.5V	12.288MHz	AC-link 比特时钟信	输入片内上拉
DC	DC2RGB_DISPLAY0CLK	O	1	1.8V	180MHz	显示通道 0 时钟	输出 24mA 驱
	DC2RGB_DISPLAY1CLK	O	1	1.8V	180MHz	显示通道 1 时钟	输出 24mA 驱
	DC0CLK	I	1	1.8V/2.5V	180MHz	显示通道 0 像素输入	输入片内上拉
	DC1CLK	I	1	1.8V/2.5V	180MHz	显示通道 1 像素输入	输入片内上拉
GMAC	GMAC0_CLK_RX	I	1	1.8V/2.5V	125MHz	GMAC 端口 0 接收	输入片内上拉
	GMAC0_CLK_OSC	I	1	1.8V/2.5V	125MHz	来自晶振的 GMAC	输入片内上拉
	GMAC0_MII_TX_CLK	I	1	1.8V/2.5V	25MHz	来自 PHY 的 GMAC	输入片内上拉
	GMAC1_CLK_RX	I	1	1.8V/2.5V	125MHz	GMAC 端口 1 接收	输入片内上拉
	GMAC1_CLK_OSC	I	1	1.8V/2.5V	125MHz	来自晶振的 GMAC	输入片内上拉
	GMAC1_MII_TX_CLK	I	1	1.8V/2.5V	25MHz	来自 PHY 的 GMAC	输入片内上拉
	GMAC0_CLK_TX	O	1	1.8V	125MHz	输出给 PHY 的 GMAC 端	输出 12mA 驱动
	GMAC1_CLK_TX	O	1	1.8V	125MHz	输出给 PHY	输出 12mA

						GMAC 端口 1 发送	动
LPC	LCLK	I	1	1.8V/2.5V	33MHz	LPC 总线时钟	输入片内上拉
WBSYS	KCLK	IO	1	1.8V	100KHz	键盘时钟	双向，输入片内上拉，
	MCLK	IO	1	1.8V	100KHz	鼠标时钟	双向，输入片内上拉，
	IIC0_CLK	IO	1	1.8V	500KHz	I2C0_Master 时钟	双向，输入片内上拉，
	IIC1_CLK	IO	1	1.8V	500KHz	I2C1_Master 时钟	双向，输入片内上拉，
	IIC2_CLK	IO	1	1.8V	500KHz	I2C2_Master 时钟	双向，输入片内上拉，
	SPI_CLK	O	1	1.8V	50MHz	SPI 接口输出时钟	输出 8mA 驱动
MC	DDR3_CK_N	O	1	1.5V	800MHz	SDRAM 差分时钟信	
	DDR3_CK_P	O	1	1.5V	800MHz		
PSW	PCIe_DS13_CLK_N	I	2	0-0.9V	100MHz	PCIE 下行端口 1 和 3	
	PCIe_DS13_CLK_P						
	PCIe_DS24_CLK_N	I	2	0-0.9V	100MHz	PCIE 下行端口 2 和 4	
	PCIe_DS24_CLK_P						
	PCIe_DS5_CLK_P	I	2	0-0.9V	100MHz	PCIE 下行端口 5 的差	
	PCIe_DS5_CLK_N						
	PCIe_US_CLK_N	I	2	0-0.9V	100MHz	PCIE 上行端口的差	
	PCIe_US_CLK_P						
SATA	SATA_CLK_N	I	2	0-0.9V	100MHz	SATAPHY 差分参考	
	SATA_CLK_P						
USB	USB_0_CLK_P	I	2	0-0.9V	100MHz	USBPHY0/1 差分参	
	USB_0_CLK_N						
	USB_1_CLK_P	I	2	0-0.9V	100MHz	USBPHY2/3 差分参	
	USB_1_CLK_N						
	USB_2_CLK_P	I	2	0-0.9V	100MHz	USBPHY4/5 差分参	
	USB_2_CLK_N						
总计			39				

1.3.4. AC97

表 1-4 AC97 信号表

信号名称	管脚方向	管脚数量	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
AC97_BIT_CLK	I	1	1.8V/2.5V	12.288MHz	AC-link 比特时钟信号	输入片内	
AC97_SDATA_I	I	1	1.8V/2	12.288MHz	AC-link 串行数据	输出	
			.5V		信号	驱动	
AC97_RESET	O	1	1.8V	电平信号	AC-link 复位输出信号	输出 8mA	
AC97_SDATA_O	O	1	1.8V	12.288MHz	AC-link 串行数据输出	输入片内	
AC97_SYNC	O	1	1.8V	12.288MHz	AC-link 同步信号	输出 8mA	
总计		5					

1.3.5. DC

表 1-5 DC 信号表

信号名称	管脚方向	管脚数量	工作电压	信号频率	信号说明	上下拉/驱动	缺省值
DC0CLK	I	1	1.8V/2.5V	180MHz	显示通道0	输入片内上拉	
DC1CLK	I	1	1.8V/2.5V	180MHz	显示通道1	输入片内上拉	
DISPLAY02PAB_INT	I	1	1.8V/2.5V	电平	显示通道	输入片内下拉	
DISPLAY12PAB_INT	I	1	1.8V/2.5V	电平	显示通道	输入片内下拉	
DC2RGB_DISPLAY0BLUE_0	O	8	1.8V	180MHz	显示通道0像素蓝色分量	输出24mA驱动	
DC2RGB_DISPLAY0BLUE_1							
DC2RGB_DISPLAY0BLUE_2							
DC2RGB_DISPLAY0BLUE_3							
DC2RGB_DISPLAY0BLUE_4							
DC2RGB_DISPLAY0BLUE_5							
DC2RGB_DISPLAY0BLUE_6							
DC2RGB_DISPLAY0BLUE_7							
DC2RGB_DISPLAY0CLK	O	1	1.8V	180MHz	显示通道0时钟	输出24mA驱动	
DC2RGB_DISPLAY0EN	O	1	1.8V	180MHz	显示通道0使能	输出24mA驱动	
DC2RGB_DISPLAY0GREEN_0	O	8	1.8V	180MHz	显示通道0像素绿色分量	输出24mA驱动	
DC2RGB_DISPLAY0GREEN_1							
DC2RGB_DISPLAY0GREEN_2							
DC2RGB_DISPLAY0GREEN_3							
DC2RGB_DISPLAY0GREEN_4							
DC2RGB_DISPLAY0GREEN_5							
DC2RGB_DISPLAY0GREEN_6							
DC2RGB_DISPLAY0GREEN_7							
DC2RGB_DISPLAY0HSYNC	O	1	1.8V	180MHz	显示通道0水平	输出24mA驱动	
DC2RGB_DISPLAY0RED_0	O	8	1.8V	180MHz	显示通道0像素红色分量	输出24mA驱动	
DC2RGB_DISPLAY0RED_1							
DC2RGB_DISPLAY0RED_2							
DC2RGB_DISPLAY0RED_3							
DC2RGB_DISPLAY0RED_4							
DC2RGB_DISPLAY0RED_5							
DC2RGB_DISPLAY0RED_6							
DC2RGB_DISPLAY0RED_7							
DC2RGB_DISPLAY0VSYNC	O	1	1.8V	180MHz	显示通道0垂直	输出24mA驱动	

DC2RGB_DISPLAY1BLUE_0	O	8	1.8V	180MHz	显示通道1像素蓝色分量	输出24mA驱动	
DC2RGB_DISPLAY1BLUE_1							
DC2RGB_DISPLAY1BLUE_2							
DC2RGB_DISPLAY1BLUE_3							
DC2RGB_DISPLAY1BLUE_4							
DC2RGB_DISPLAY1BLUE_5							
DC2RGB_DISPLAY1BLUE_6							
DC2RGB_DISPLAY1BLUE_7							
DC2RGB_DISPLAY1CLK	O	1	1.8V	180MHz	显示通道1时钟	输出24mA驱动	
DC2RGB_DISPLAY1EN	O	1	1.8V	180MHz	显示通道1使能	输出24mA驱动	
DC2RGB_DISPLAY1GREEN_0	O	8	1.8V	180MHz	显示通道1像素绿色分量	输出24mA驱动	
DC2RGB_DISPLAY1GREEN_1							
DC2RGB_DISPLAY1 GREEN							
DC2RGB_DISPLAY1 GREEN							
DC2RGB_DISPLAY1 GREEN							
DC2RGB_DISPLAY1 GREEN							
DC2RGB_DISPLAY1 GREEN							
DC2RGB_DISPLAY1 GREEN							
DC2RGB_DISPLAY1HSYNC	O	1	1.8V	180MHz	显示通道1水平	输出24mA驱动	
DC2RGB_DISPLAY1RED_0	O	8	1.8V	180MHz	显示通道1像素红色分量	输出24mA驱动	
DC2RGB_DISPLAY1RED_1							
DC2RGB_DISPLAY1RED_2							
DC2RGB_DISPLAY1RED_3							
DC2RGB_DISPLAY1RED_4							
DC2RGB_DISPLAY1RED_5							
DC2RGB_DISPLAY1RED_6							
DC2RGB_DISPLAY1RED_7							
DC2RGB_DISPLAY1VSYNC	O	1	1.8V	180MHz	显示通道1垂直	输出24mA驱动	
总计		61					

表1-6 GMAC 信号表

1.3.6. GMAC

信号名称	管脚方向	管脚数量	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
GMAC0_CLK_OSC	I	1	1.8V/2.5V	125MHz	来自晶振的GMAC 端口0	输入片内上拉	
GMAC0_CLK_RX	I	1	1.8V/2.5V	125MHz	GMAC 端口0	输入片内上拉	
GMAC0_MII_TX_CLK	I	1	1.8V/2.5V	25MHz	来自 PHY 的GMAC 端口0	输入片内上拉	
GMAC0_PHY_COL	I	1	1.8V/2.5V	125MHz	GMAC 端口 0 CRS	输入片内下拉	
GMAC0_PHY_CRS	I	1	1.8V/2.5V	125MHz	GMAC 端口0	输入片内下拉	
GMAC0_PHY_INTF_SEL	I	1	1.8V/2.5V	电平信号	GMAC 端口0接口类型选择	输入，片内上拉	0
GMAC0_PHY_RXDV	I	1	1.8V/2.5V	125MHz	GMAC 端口0	输入片内下拉	
GMAC0_PHY_RXER	I	1	1.8V/2.5V	125MHz	GMAC 端口0	输入片内下拉	
GMAC0_CLK_TX	O	1	1.8V	125MHz	输出给 PHY 的GMAC 端口	输出12mA驱动	
GMAC0_GMII_MDC	O	1	1.8V	125 Mhz	GMAC 端口0SMA 管理	输出12mA驱动	
GMAC0_GMII_MDIO	IO	1	1.8V	125 Mhz	GMAC 端口0SMA 管理数	双向，输入上拉，输	
GMAC0_PHY_TXEN	O	1	1.8V	125MHz	GMAC 端口0	输出12mA	
GMAC0_PHY_TXER	O	1	1.8V	125MHz	GMAC 端口0	输出12mA	
GMAC0_PHY_RXD_0	I	8	1.8V/2.5V	125MHz	GMAC 端口0接收数据	输入片内上拉	
GMAC0_PHY_RXD_1							
GMAC0_PHY_RXD_2							
GMAC0_PHY_RXD_3							
GMAC0_PHY_RXD_4							
GMAC0_PHY_RXD_5							
GMAC0_PHY_RXD_6							
GMAC0_PHY_RXD_7							
GMAC0_PHY_TXD_0	O	8	1.8V	125MHz	GMAC 端口	输出	

GMAC0_PHY_TXD_1					0	12mA	
GMAC0_PHY_TXD_2					发送数据	驱动	

GMAC0_PHY_TXD_3							
GMAC0_PHY_TXD_4							
GMAC0_PHY_TXD_5							
GMAC0_PHY_TXD_6							
GMAC0_PHY_TXD_7							
GMAC1_CLK_OSC	I	1	1.8V/ 2.5V	125MHz	来自晶振的 GMAC 端口 1 发送时钟	输入片内上拉	
GMAC1_CLK_RX	I	1	1.8V/ 2.5V	125MHz	GMAC 端口 1 接收时钟	输入片内上拉	
GMAC1_MII_TX_CLK	I	1	1.8V/ 2.5V	25MHz	来自 PHY 的 GMAC 端口 1 发送时钟	输入片内上拉	
GMAC1_PHY_COL	I	1	1.8V/ 2.5V	125MHz	GMAC 端口 1 CRS	输入片内下拉	
GMAC1_PHY_CRS	I	1	1.8V/ 2.5V	125MHz	GMAC 端口 1 冲突	输入片内下拉	
GMAC1_PHY_INTF_SEL	I	1	1.8V/ 2.5V	电平信号	GMAC 端口 1 接口类型选择 0:GMII/MII 1:RGMII	输入片内上拉	0
GMAC1_PHY_RXDV	I	1	1.8V/ 2.5V	125MHz	GMAC 端口 1 接收数据有效	输入片内下拉	
GMAC1_PHY_RXER	I	1	1.8V/ 2.5V	125MHz	GMAC 端口 1 接受错误	输入片内下拉	
GMAC1_CLK_TX	O	1	1.8V	125MHz	输出给 PHY 的 GMAC 端口 1 发送时钟	输出 12mA 驱动	
GMAC1_GMII_MDC	O	1	1.8V	125MHZ	GMAC 端口 1 SMA 管理数据 时钟	输出 12mA 驱动	
GMAC1_GMII_MDIO	I O	1	1.8V	125MHZ	GMAC 端口 1 SMA 管理数据	双向，输入 上拉，输	
GMAC1_PHY_TXEN	O	1	1.8V	125MHz	GMAC 端口 1 发送使能	输出 12mA 驱动	
GMAC1_PHY_TXER	O	1	1.8V	125MHz	GMAC 端口 1 发送错误	输出 12mA 驱动	
GMAC1_PHY_RXD_0	I	8	1.8V/ 2.5V	125MHz	GMAC 端口 1 接收数据	输入片内上拉	
GMAC1_PHY_RXD_1							
GMAC1_PHY_RXD_2							
GMAC1_PHY_RXD_3							
GMAC1_PHY_RXD_4							
GMAC1_PHY_RXD_5							
GMAC1_PHY_RXD_6							
GMAC1_PHY_RXD_7							
GMAC1_PHY_TXD_0	O	8	1.8V	125MHz	GMAC 端口 1 发送数据	输出 12mA 驱动	
GMAC1_PHY_TXD_1							
GMAC1_PHY_TXD_2							
GMAC1_PHY_TXD_3							

GMAC1_PHY_TXD_4							
GMAC1_PHY_TXD_5							
GMAC1_PHY_TXD_6							
GMAC1_PHY_TXD_7							
总计		5					

1.3.7. 时钟配置

表 1-7 时钟配置信号表

信号名称	管脚方	管脚数	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
REFCLK_N	I	1	1.8V/2.5V	100MH	参考时钟	输入片内上拉	
REFCLK_P	I	1	1.8V/2.5V	100MH	参考时钟	输入片内上拉	
CFG_GCLKPLL_0	I	3	1.8v/2.5V	电平信号	GCLK 锁相环参数配置引脚	输入片内上拉	
CFG_GCLKPLL_1							
CFG_GCLKPLL_2							
CFG_GPUPLL_0	I	3	1.8v/2.5V	电平信号	GPUCLK 锁相环 参数配置引脚	输入片内上拉	
CFG_GPUPLL_1							
CFG_GPUPLL_2							
CFG_MCPLL_0	I	3	1.8v/2.5V	电平信号	MCCLK 锁相环参 数配置引脚	输入片内上拉	
CFG_MCPLL_1							
CFG_MCPLL_2							
CFG_SWCLKPLL_0	I	3	1.8v/2.5V	电平信号	PSWCLK 锁相环 参数配置引脚	输入片内上拉	
CFG_SWCLKPLL_1							
CFG_SWCLKPLL_2							
PLL_VDD09A_0		4	0.9V	PLL 模拟电	4 个PLL 的模拟电 源		
PLL_VDD09A_1							
PLL_VDD09A_2							
PLL_VDD09A_3							
PLL_VSS09A_0		4	GND	PLL 模拟地	4 个PLL 的模拟电 地		
PLL_VSS09A_1							
PLL_VSS09A_2							
PLL_VSS09A_3							
总计		22					

1.3.8. 维护/测试/观测

表 1-8 维护/测试/观测信号表

信号名称	管脚方向	管脚数	工作电压	信号频率	信号说明	上下拉/驱动能力
JTAG_SEL_0 JTAG_SEL_1 JTAG_SEL_2	I	3	1.8V/2.5V	电平信号	JTAG 链选择	输入片内上拉
TCK	I	1	1.8V/2.5	10MHz	JTAG 串行时钟	输入片内上拉
TDI	I	1	1.8V/2.5	10MHz	JTAG 串行数据输	输入片内上拉
TMS	I	1	1.8V/2.5	10MHz	JTAG 状态控制	输入片内上拉
TRSTN	I	1	1.8V/2.5	10MHz	JTAG 复位，低有	输入片内上拉
TDO	O	1	1.8V	10MHz	JTAG 串行数据输	带三态的输出，12mA 驱
CPU_LINKU	O	1	1.8V	电平	PCIE 链路建立观	输出 8mA 驱动
WATCH_OU	O	1	1.8V	50MHz	片内信号观测	输出 8mA 驱动
总计		10				

1.3.9. SIO（低速 IO：wbsys+lpc）

表 1-9 低速 IO 信号表

信号名称	管脚方向	管脚数	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
LPC 接口							
LCLK	I	1	1.8V/	33MHz	LPC 时钟：33MHz	输入，片内上拉	
CFG_LDRQ1N	I	1	1.8V/	33MHz	LPC 设备 DMA 申	输入，片内上拉	
CFG_LDRQ2N	I	1	1.8V/	33MHz	LPC 设备 DMA 申	输入，片内上拉	
LFRAMEN	O	1	1.8V	33MHz	LPC 帧起始信号：host 用此信号来开 始一个传输周期，或结束一个异常周期，peripheral 不得驱动此信	输出，12mA 驱动	
LRSTN	O	1	1.8V	电平	LPC-HOST 输出的 LPC 复位信号（可用于 LPC 设备的复位），低电平有效	输出，12mA 驱动	
SERIRQ	IO	1	1.8V	33MHz	LPC 串行中断	双向，输入上拉，输出 OD 门，	
LAD_0	IO	4	1.8V	33MHz	LPC 命令/地址/数据复用总线	双向，输入上拉，输出 OD 门，12mA 驱动	
LAD_1							
LAD_2							
LAD_3							

SPI-flash 接口							
FLASH_ADDRSEL	I	1	1.8V/2.5V	50MHz	16/24 位地址格式 选择：0—16	输入，片内上拉	
SPI_RXD	I	1	1.8V/	50MHz	spi 输入数据	输入，片内上拉	
SPI_CLK	O	1	1.8V	50MHz	spi 串行输出线上时钟	输出，8mA 驱动	
SPI_CS_N	O	1	1.8	50MH	片选	输出，8mA 驱	
SPI_TXD	O	1	1.8	50MH	spi 输出数据	输出，8mA 驱	
PS2 键鼠							
KCLK	I	1	1.8	100KH	键盘时钟	双向，输入上	
						输出 8mA 驱动	
KDAT	IO	1	1.8V	100KHz	键盘数据	双向，输入上拉，	
MCLK	IO	1	1.8V	100KHz	鼠标时钟	双向，输入上拉，	
MDAT	IO	1	1.8V	100KHz	鼠标数据	双向，输入上拉，	
串口							
UART_CTS	I	1	1.8V/	5MHz	允许发送	输入，片内下拉	
UART_DCD	I	1	1.8V/	5MHz	载波检测	输入，片内下拉	
UART_DSR	I	1	1.8V/	5MHz	数据通信设备准备好	输入，片内下拉	
UART_HDPLX	I	1	1.8V/2.5V	5MHz	编程选择跳线，0- 通过 uart 口对 spi flash 编	输入，片内上拉	
UART_RI	I	1	1.8V/	5MHz	振铃指示	输入，片内下拉	
UART_RX	I	1	1.8V/	5MHz	串行输入数据	输入，片内上拉	
UART_DTR	O	1	1.8	5MHz	数据终端准备好	输出，8mA 驱	
UART_RTS	O	1	1.8	5MHz	发送请求	输出，8mA 驱	
UART_TX	O	1	1.8	5MHz	串行输出数据	输出，8mA 驱	
GPIO 控制器							
PORTA_0	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTA_1	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTA_2	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTA_3	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTA_4	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTA_5	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	

PORTA_6	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTA_7	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_0	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_1	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_2	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_3	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_4	I	1	1.8		GPIO 可配置分	双向，输入上	
					B 引脚	输出 12mA 驱动	
PORTB_5	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_6	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
PORTB_7	IO	1	1.8V		GPIO 可配置分组	双向，输入上拉，	
I2C_Master							
IIC0_CLK	IO	1	1.8V	500KHz	I2C0 时钟	双向，输入上拉，输出 OD 门，12mA 驱动	
IIC0_DATA	IO	1	1.8V	500KHz	I2C0 数据	双向，输入上拉，输	
IIC1_CLK	IO	1	1.8V	500KHz	I2C1 时钟	双向，输入上拉，输	
IIC1_DATA	IO	1	1.8V	500KHz	I2C1 数据	双向，输入上拉，输	
IIC2_CLK	IO	1	1.8V	500KHz	I2C2 时钟	双向，输入上拉，输	
IIC2_DATA	IO	1	1.8V	500KHz	I2C2 数据	双向，输入上拉，输	
总计		5					

1.3.10. MC

表 1-10 MC 信号表

信号名称	管脚方	管脚数	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
DDR3_DTO_0	O	2	1.5V		延迟线数字测试输出		
DDR3_DTO_1							
DDR3_A0	O	16	1.5v	(466x2) MHz	地址信号		
DDR3_A1							
DDR3_A2							
DDR3_A3							
DDR3_A4							
DDR3_A5							
DDR3_A6							
DDR3_A7							
DDR3_A8							
DDR3_A9							
DDR3_A10							
DDR3_A11							
DDR3_A12							
DDR3_A13							
DDR3_A14							
DDR3_A15							
DDR3_BA0	O	3	1.5v	(466x2) MHz	BANK 信号		
DDR3_BA1							
DDR3_BA2							
DDR3_CAS_N	O	1	1.5v	(466x2) MHz	列地址选通信号		
DDR3_CK_P	O	1	1.5v	(466x2) MHz			
DDR3_CKE	O	1	1.5v	(466x2) MHz	SDRAM 时钟使能信号		
DDR3_CK_N	O	1	1.5v	(466x2) MHz	SDRAM 差分时钟信号		
DDR3_CS_N	O	1	1.5v	(466x2) MHz	片选信号		
DDR3_ODT	O	1	1.5v	(466x2) MHz	SDRAM 颗粒 ODT 信号		
DDR3_RAS_N	O	1	1.5v	(466x2) MHz	行地址选通信号		
DDR3_RST_N	O	1	1.5v	电平信号	SDRAM 颗粒复位信号		

DDR3_WE_N	O	1	2.5v	(466x2) MHz	写命令信号		
DDR3_ATO	IO	1	1.5v		PLL 模拟测试		
DDR3_VREF	IO	1	0.75 v		电压参考信号，通常为 0.5*VDDQ15		
DDR3_PVT0	IO	3	1.5v		外接 240om 正 负 1%的精确 电阻		
DDR3_PVT1							
DDR3_PVT2							
DDR3_DM0	IO	8	1.5v	(466x4) MHz	数据掩码信号		
DDR3_DM1							
DDR3_DM2							
DDR3_DM3							
DDR3_DM4							
DDR3_DM5							
DDR3_DM6							
DDR3_DM7							
DDR3_DQ0	IO	64	1.5v	(466x4)	数据信号		
DDR3_DQ1				MHz			
DDR3_DQ2							
DDR3_DQ3							
DDR3_DQ4							
DDR3_DQ5							
DDR3_DQ6							
DDR3_DQ7							
DDR3_DQ8							
DDR3_DQ9							
DDR3_DQ10							
DDR3_DQ11							
DDR3_DQ12							
DDR3_DQ13							
DDR3_DQ14							
DDR3_DQ15							
DDR3_DQ16							
DDR3_DQ17							
DDR3_DQ18							
DDR3_DQ19							
DDR3_DQ20							
DDR3_DQ21							
DDR3_DQ22							
DDR3_DQ23							

DDR3_DQ24							
DDR3_DQ25							
DDR3_DQ26							
DDR3_DQ27							
DDR3_DQ28							
DDR3_DQ29							
DDR3_DQ30							
DDR3_DQ31							
DDR3_DQ32							
DDR3_DQ33							
DDR3_DQ34							
DDR3_DQ35							
DDR3_DQ36							
DDR3_DQ37							
DDR3_DQ38							
DDR3_DQ39							
DDR3_DQ40							
DDR3_DQ41							
DDR3_DQ42							
DDR3_DQ43							
DDR3_DQ44							
DDR3_DQ45							
DDR3_DQ46							
DDR3_DQ47							
DDR3_DQ48							
DDR3_DQ49							
DDR3_DQ50							
DDR3_DQ51							
DDR3_DQ52							
DDR3_DQ53							
DDR3_DQ54							
DDR3_DQ55							
DDR3_DQ56							
DDR3_DQ57							
DDR3_DQ58							
DDR3_DQ59							
DDR3_DQ60							
DDR3_DQ61							
DDR3_DQ62							
DDR3_DQ63							
DDR3_DQS_0_N	IO	16	1.5v	(466x2) MHz	差分 DQS 信号		
DDR3_DQS_0_P							
DDR3_DQS_1_N							
DDR3_DQS_1_P							

DDR3_DQS_2_N							
DDR3_DQS_2_P							
DDR3_DQS_3_N							
DDR3_DQS_3_P							
DDR3_DQS_4_N							
DDR3_DQS_4_P							
DDR3_DQS_5_N							
DDR3_DQS_5_P							
DDR3_DQS_6_N							
DDR3_DQS_6_P							
DDR3_DQS_7_N							
DDR3_DQS_7_P							
DDR3_VDD18A		3	1.8v	电地	1.8V 模拟电地引脚		
DDR3_VSS18A		3	1.8v	电地	1.8V 模拟地引脚		
总计		129					

1.3.11. PCIE-SW

表 1-11 PCIE 交叉开关信号表

信号名称	管	管脚	工作	信号	信号说明	上下	缺省

	脚方	数量	电压	频率		拉/驱动	值
DS1_ATTN_BTN_PRESD	I	1	1.8v/2.5v	电平信号	插槽按压事件通知给 ds1 内部 逻辑	输入片内	
DS1_SENS_MRL_SENS_STATE	I	1	1.8v/2.5v	电平信号	板上 MRT 传感器状态通知给 ds1 内部逻辑	输入片内	
DS1_SENS_PRES_DET_STATE	I	1	1.8v/2.5v	电平信号	插槽按压状态通知给 ds1 内部 逻辑	输入片内	
DS1_SET_MRL_SENS_CHANGED	I	1	1.8v/2.5v	电平信号	板上 MRT 传感器状态变换通知给 ds1 内部逻辑	输入片内下	
DS1_SET_PWR_FAULT_DETECTED	I	1	1.8v/2.5v	电平信号	板上电源 fault 通知给 ds1 内部 逻辑	输入片内	
DS1_ATTN_INDICATOR_CNTRL_0	O	1	1.8v	电平信号	PCIE 下行端口 1 插槽提醒点灯 信号：00 保留，01-on，10-闪烁，11-	输出，8m	
DS1_ATTN_INDICATOR_CNTRL_1	O	1	1.8v			输出，8m	
DS1_POWER_CONTROLLER	O	1	1.8v	电平信号	PCIE 下行端口 1 内部逻辑输出 给插槽的电源使能信	输出，8mA 驱	
DS1_PRES_DET_CLEAR	O	1	1.8v	电平信号	指示已经被软件清除了 PCIE下行端口 1 的slot 状态寄存器 中的在位状态变化位	输出，8mA 驱动	
DS1_PWR_INDICATOR_CNTRL_0	O	1	1.8v	电平信号	PCIE 下行端口 1 插槽电源状态 点灯信号：00 保留，01-	输出，8m	
DS1_PWR_INDICATOR_CNTRL_1	O	1	1.8v	电平信号		输出	

						8mA	
PCIe_DS13_CLK_N	I	1		100MHz	PHY 差分参考时钟		
PCIe_DS13_CLK_P	I	1		100MHz	PHY 差分参考时钟		
PCIe_DS24_CLK_N	I	1		100MHz	PHY 差分参考时钟		
PCIe_DS24_CLK_P	I	1		100MHz	PHY 差分参考时钟		
PCIe_DS5_CLK_N	I	1		100MHz	PHY 差分参考时钟		
PCIe_DS5_CLK_P	I	1		100MHz	PHY 差分参考时钟		
PCIe_US_CLK_N	I	1		100MHz	PHY 差分参考时钟		
PCIe_US_CLK_P	I	1		100MHz	PHY 差分参考时钟		
PCIe_US_RESREF	I	1		100MHz	PHY 参考电阻		
PCIe_DS13_RESREF	I	1		100MHz	PHY 参考电阻		
PCIe_DS24_RESREF	I	1		100MHz	PHY 参考电阻		
PCIe_DS5_RESREF	I	1		100MHz	PHY 参考电阻		
PCIe_US_Rx_0_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_1_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_2_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_3_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_4_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_5_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_6_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_7_N	I	1		100MHz	差分接收信号		
PCIe_US_Rx_0_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_1_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_2_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_3_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_4_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_5_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_6_P	I	1		100MHz	差分接收信号		
PCIe_US_Rx_7_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_0_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_1_N	I	1		100MHz	差分接收信号		

PCIe_DS13_Rx_2_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_3_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_4_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_5_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_6_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_7_N	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_0_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_1_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_2_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_3_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_4_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_5_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_6_P	I	1		100MHz	差分接收信号		
PCIe_DS13_Rx_7_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_0_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_1_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_2_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_3_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_4_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_5_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_6_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_7_N	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_0_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_1_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_2_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_3_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_4_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_5_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_6_P	I	1		100MHz	差分接收信号		
PCIe_DS24_Rx_7_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_0_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_1_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_2_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_3_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_4_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_5_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_6_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_7_N	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_0_P	I	1		100MHz	差分接收信号		

PCIe_DS5_Rx_1_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_2_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_3_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_4_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_5_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_6_P	I	1		100MHz	差分接收信号		
PCIe_DS5_Rx_7_P	I	1		100MHz	差分接收信号		
PCIe_US_Tx_0_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_1_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_2_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_3_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_4_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_5_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_6_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_7_N	O	1		100MHz	差分发送信号		
PCIe_US_Tx_0_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_1_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_2_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_3_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_4_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_5_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_6_P	O	1		100MHz	差分发送信号		
PCIe_US_Tx_7_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_0_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_1_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_2_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_3_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_4_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_5_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_6_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_7_N	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_0_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_1_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_2_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_3_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_4_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_5_P	O	1		100MHz	差分发送信号		
PCIe_DS13_Tx_6_P	O	1		100MHz	差分发送信号		

PCIe_DS13_Tx_7_P	O	1		100MHz	差分发送信号		

PCIe_DS24_Tx_0_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_1_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_2_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_3_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_4_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_5_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_6_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_7_N	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_0_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_1_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_2_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_3_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_4_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_5_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_6_P	O	1		100MHz	差分发送信号		
PCIe_DS24_Tx_7_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_0_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_1_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_2_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_3_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_4_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_5_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_6_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_7_N	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_0_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_1_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_2_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_3_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_4_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_5_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_6_P	O	1		100MHz	差分发送信号		
PCIe_DS5_Tx_7_P	O	1		100MHz	差分发送信号		
PCIe_US_VDDA		4	模拟 0.9V				
PCIe_US_VDDQA		2	模拟 1.8V				
PCIe_US_VTT		4	模拟 1.5V				

PCIe_DS13_VDDA		4	模拟 0.9V				

PCIe_ DS13_VDDQA		2	模拟 1.8V				
PCIe_ DS13_VTT		4	模拟 1.5V				
PCIe_DS24_VDDA		4	模拟 0.9V				
PCIe_ DS24_VDDQA		2	模拟 1.8V				
PCIe_ DS24_VTT		4	模拟 1.5V				
PCIe_DS5_VDDA		4	模拟 0.9V				
PCIe_ DS5_VDDQA		2	模拟 1.8V				
PCIe_ DS5_VTT		4	模拟 1.5V				
总计		191					

1.3.12. SATA

表 1-12 SATA 信号表

信号名称	管脚方	管脚数	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
SATA_AHCI0_CP_DET	I	1	1.8V/2.5v	电平	sata 端口 0 冷在位检测信号	输入，片内下拉	
SATA_AHCI0_MP_SWITCH	I	1	1.8V/2.5v	电平	sata 端口 0 机械在位切换信号	输入，片内下拉	
SATA_AHCI1_CP_DET	I	1	1.8V/2.5v	电平	sata 端口 1 冷在位检测信号	输入，片内下拉	
SATA_AHCI1_MP_SWITCH	I	1	1.8V/2.5v	电平	sata 端口 1 机械在位切换信号	输入，片内下拉	
SATA_AHCI2_CP_DET	I	1	1.8V/2.5v	电平	sata 端口 2 冷在位检测信号	输入，片内下拉	
SATA_AHCI2_MP_SWITCH	I	1	1.8V/2.5v	电平	sata 端口 2 机械在位切换信号	输入，片内下拉	
SATA_AHCI0_ACT_LED_N	O	1	1.8V	电平	sata 端口 0 状态指示灯	输出，8mA 驱动	
SATA_AHCI0_CP_POD	O	1	1.8V	电平	sata 端口 0 冷在位上电信号	输出，8mA 驱动	

SATA_AHCI1_ ACT_LED_N	O	1	1.8V	电平	sata 端口 1 状态指示	输出，8mA	

					灯	驱动	
SATA_AHCI1_CP_POD	O	1	1.8V	电平	sata 端口 1 冷在位上	输出，8mA 驱动	
SATA_AHCI2_ACT_LED_N	O	1	1.8V	电平	sata 端口 2 状态指示	输出，8mA 驱动	
SATA_AHCI2_CP_POD	O	1	1.8V	电平	sata 端口 2 冷在位上	输出，8mA 驱动	
SATA_CLK_N	I	1		100MHz	sata phy 差分参考时		
SATA_CLK_P	I	1		100MHz	sata phy 差分参考时		
SATA_0_RESREF	IO	1			sata 端口 0 参考电阻		
SATA_1_RESREF	IO	1			sata 端口 1 参考电阻		
SATA_2_RESREF	IO	1			sata 端口 2 参考电阻		
SATA_0_Rx_N	I	1		6Gbps	sata 端口 0 差分接受		
SATA_0_Rx_P	I	1		6Gbps	sata 端口 0 差分接受		
SATA_1_Rx_N	I	1		6Gbps	sata 端口 1 差分接受		
SATA_1_Rx_P	I	1		6Gbp	sata 端口 1 差分接受		
SATA_2_Rx_N	I	1		6Gbps	sata 端口 2 差分接受		
SATA_2_Rx_P	I	1		6Gbps	sata 端口 2 差分接受		
SATA_0_Tx_N	O	1		6Gbps	sata 端口 0 差分发送		
SATA_0_Tx_P	O	1		6Gbp	sata 端口 0 差分发送		
SATA_1_Tx_N	O	1		6Gbps	sata 端口 1 差分发送		
SATA_1_Tx_P	O	1		6Gbps	sata 端口 1 差分发送		
SATA_2_Tx_N	O	1		6Gbps	sata 端口 2 差分发送		
SATA_2_Tx_P	O	1		6Gbps	sata 端口 2 差分发送		
SATA_VP	I	2	0.9V		sata phy 低电压供		
SATA_VPH	I	2	1.8v/2		sata phy 高电压供		

			.5v				
SATA_VPTX	I	2	0.9V		sata phy 发送电压供电		
总计		35					

1.3.13. USB

表 1-13 USB 信号表

信号名称	管脚方	管脚数	工作电压	信号频率	信号说明	上下拉/驱动能力
USB_OVRCUR_N	I	1	1.8V/2.5v	电平	端口 0/1/2/3 过流保护	输入，片内上拉
USB_OVRCUR_F_N	I	1	1.8V/2.5v	电平	端口 4/5 过流保护	输入，片内上拉
USB_VBUS_CTRL	O	1	1.8V	电平	端口 0/1/2/3VBUS 控制，0 有效	输出，8mA 驱动
USB_VBUS_F_CTRL	O	1	1.8V	电平	端口 4/5VBUS 控制，0 有效	输出，8mA 驱动
USB_0_CLK_N	I	1		100MHz	端口 0/1 参考时钟	
USB_1_CLK_N	I	1		100MHz	端口 2/3 参考时钟	
USB_2_CLK_N	I	1		100MHz	端口 4/5 参考时钟	
USB_0_CLK_P	I	1		100MHz	端口 0/1 参考时钟	
USB_1_CLK_P	I	1		100MHz	端口 2/3 参考时钟	
USB_2_CLK_P	I	1		100MHz	端口 4/5 参考时钟	
USB_0_RESREF	I	1			端口 0 参考电阻	
USB_1_RESREF	I	1			端口 1 参考电阻	
USB_2_RESREF	I	1			端口 2 参考电阻	
USB_3_RESREF	I	1			端口 3 参考电阻	
USB_4_RESREF	I	1			端口 4 参考电阻	
USB_5_RESREF	I	1			端口 5 参考电阻	
USB_0_ID	I	1			端口 n 的 mini-receptacle 指示	
USB_1_ID	I	1			端口 n 的 mini-receptacle 指示	
USB_2_ID	I	1			端口 n 的 mini-receptacle 指示	
USB_3_ID	I	1			端口 n 的 mini-receptacle 指示	

USB_4_ID	I	1			端口 n 的 mini-receptacle 指 示	
USB_5_ID	I	1			端口 n 的 mini-receptacle 指 示	
USB_0_Rx_N	I	1			端口 n 的 SS 差分	
USB_1_Rx_N	I	1			端口 n 的 SS 差分 接收信号	
USB_2_Rx_N	I	1			端口 n 的 SS 差分 接收信号	
USB_3_Rx_N	I	1			端口 n 的 SS 差分 接收信号	
USB_4_Rx_N	I	1			端口 N 的 SS 差分	
USB_5_Rx_N	I	1			端口 N 的 SS 差分	
USB_0_Rx_P	I	1			端口 N 的 SS 差分	
USB_1_Rx_P	I	1			端口 N 的 SS 差分	
USB_2_Rx_P	I	1			端口 N 的 SS 差分	
USB_3_Rx_P	I	1			端口 N 的 SS 差分	
USB_4_Rx_P	I	1			端口 N 的 SS 差分	
USB_5_Rx_P	I	1			端口 N 的 SS 差分	
USB_VBUS0_0	I	1			端口 n 的 5V 供	
USB_VBUS0_1	I	1			端口 n 的 5V 供	
USB_VBUS0_2	I	1			端口 n 的 5V 供	
USB_VBUS0_3	I	1			端口 n 的 5V 供	
USB_VBUS0_4	I	1			端口 n 的 5V 供	
USB_VBUS0_5	I	1			端口 n 的 5V 供	
USB_0_Tx_N	O	1			端口 N 的 SS 差分	
USB_1_Tx_N	O	1			端口 N 的 SS 差分	
USB_2_Tx_N	O	1			端口 N 的 SS 差分	
USB_3_Tx_N	O	1			端口 N 的 SS 差分	
USB_4_Tx_N	O	1			端口 N 的 SS 差分	
USB_5_Tx_N	O	1			端口 N 的 SS 差分	

USB_0_Tx_P	O	1			端口 N 的 SS 差分	
USB_1_Tx_P	O	1			端口 N 的 SS 差分	
USB_2_Tx_P	O	1			端口 N 的 SS 差分	
USB_3_Tx_P	O	1			端口 N 的 SS 差分	
USB_4_Tx_P	O	1			端口 N 的 SS 差分	
USB_5_Tx_P	O	1			端口 N 的 SS 差分	
USB_0_DM	I O	1			端口 n 的 2.0 数据	
USB_1_DM	I O	1			端口 n 的 2.0 数据	
USB_2_DM	I O	1			端口 n 的 2.0 数据	
USB_3_DM	I O	1			端口 n 的 2.0 数据	
USB_4_DM	I O	1			端口 n 的 2.0 数据	
USB_5_DM	I O	1			端口 n 的 2.0 数据	
USB_0_DP	I O	1			端口 n 的 2.0 数据	
USB_1_DP	I O	1			端口 n 的 2.0 数据	
USB_2_DP	I O	1			端口 n 的 2.0 数据	
USB_3_DP	I O	1			端口 n 的 2.0 数据	
USB_4_DP	I O	1			端口 n 的 2.0 数据	
USB_5_DP	I O	1			端口 n 的 2.0 数据	
USB_VP_0		2	0.9 V		端口 0/1 的低电压	
USB_VP_1		2	0.9 V		端口 2/3 的低电压	
USB_VP_2		2	0.9 V		端口 4/5 的低电压	
USB_VDD180_0		1	1.8 V/ 2.5v		端口 0/1 的 ss/hs 接	
USB_VDD180_1		1	1.8 V/ 2.5v		端口 2/3 的 ss/hs 接	

USB_VDD180_2		1	1.8V/2.5v		端口 4/5 的 ss/hs 接口电压供电	
USB_VPTX_0		1	0.9V		端口 0/1 的发送电压供电	
USB_VPTX_1		1	0.9V		端口 2/3 的发送电压供电	
USB_VPTX_2		1	0.9V		端口 4/5 的发送电压供电	
USB_VDD330_0		1	3.3V		端口 0/1 的高电压供电	
USB_VDD330_1		1	3.3V		端口 2/3 的高电压供电	
USB_VDD330_2		1	3.3V		端口 4/5 的高电压供电	
总计		70				

1.3.14. 复位/杂项信号

表 1-14 复位/杂项信号表

信号名称	管脚方向	管脚数量	工作电压	信号频率	信号说明	上下拉/驱动能力	缺省值
DCOK	I	1	1.8V/2.5v	电平	上电信号，1 表示上电	输入，片内下拉	
MTRST_N	I	1	1.8V/2.5v	电平	维护复位信号，0 表示复位	输入，片内上拉	
PWRGD	O	1	1.8V	电平	ICH2 复位完成，0 有效	输出 OD 门，8mA	
SPARE_1	IO	2	1.8v	电平信号	芯片预留管脚	输出 8mA 驱动	
SPARE_2							
总计		5					

1.3.15. ICH2 复位时序

一、 ICH2 外部复位

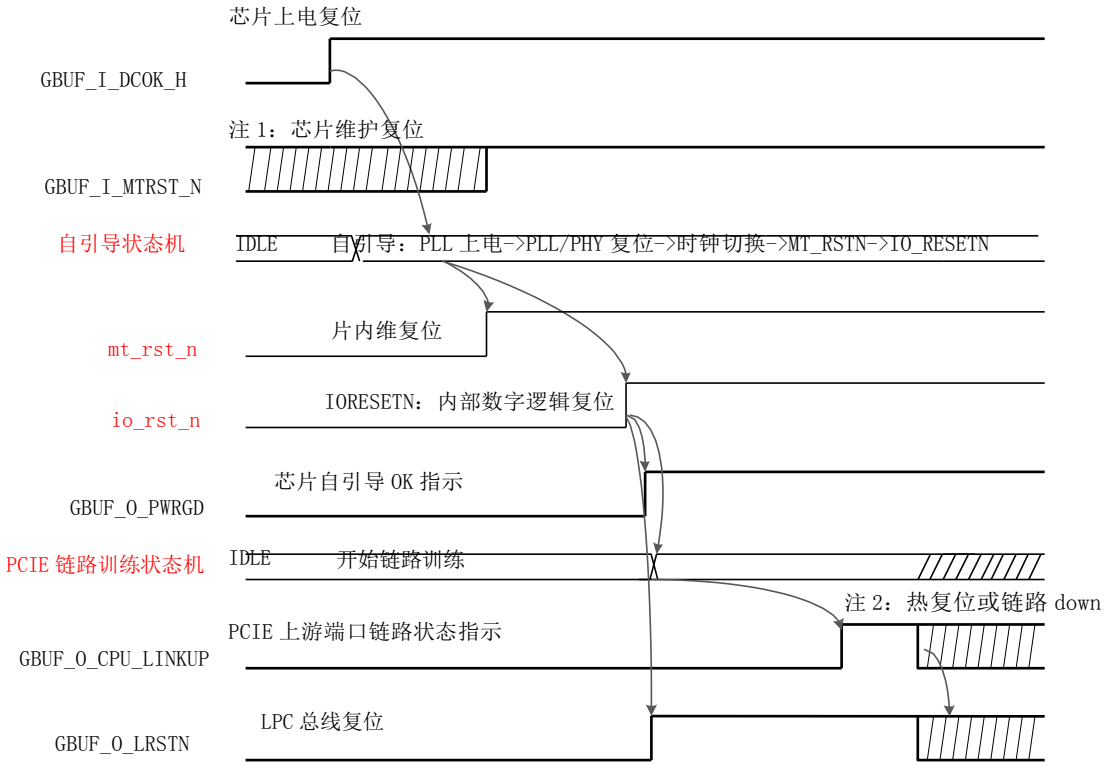


图 1-5 ICH2 外部复位时序图

（上图中黑色粗体字为芯片管脚，红色字体为内部信号）

注 1：芯片维护复位在自引导状态机撤销维护复位之前无作用。

注 2：PCIe 上层指示的热复位或者链路 down 会触发触发 hot_reset 并传导给下游所有设备， 包括给 LPC 的总线复位 GBUF_O_LRSTN。

建议：可根据 PCIE 设备的实际需要选用 GBUF_O_LRSTN 或 GBUF_O_CPU_LINKUP 作为 PCIE_SWITCH 下游 DS13 和 DS24 上的设备复位，如果设备要求上游链路训练完成后才能 复位的建议选用 GBUF_O_CPU_LINKUP 作为复位（此管脚驱动能力为 8mA），否则选用 GBUF_O_LRSTN（此管脚驱动能力为 12mA）。

二、 ICH2 内部复位

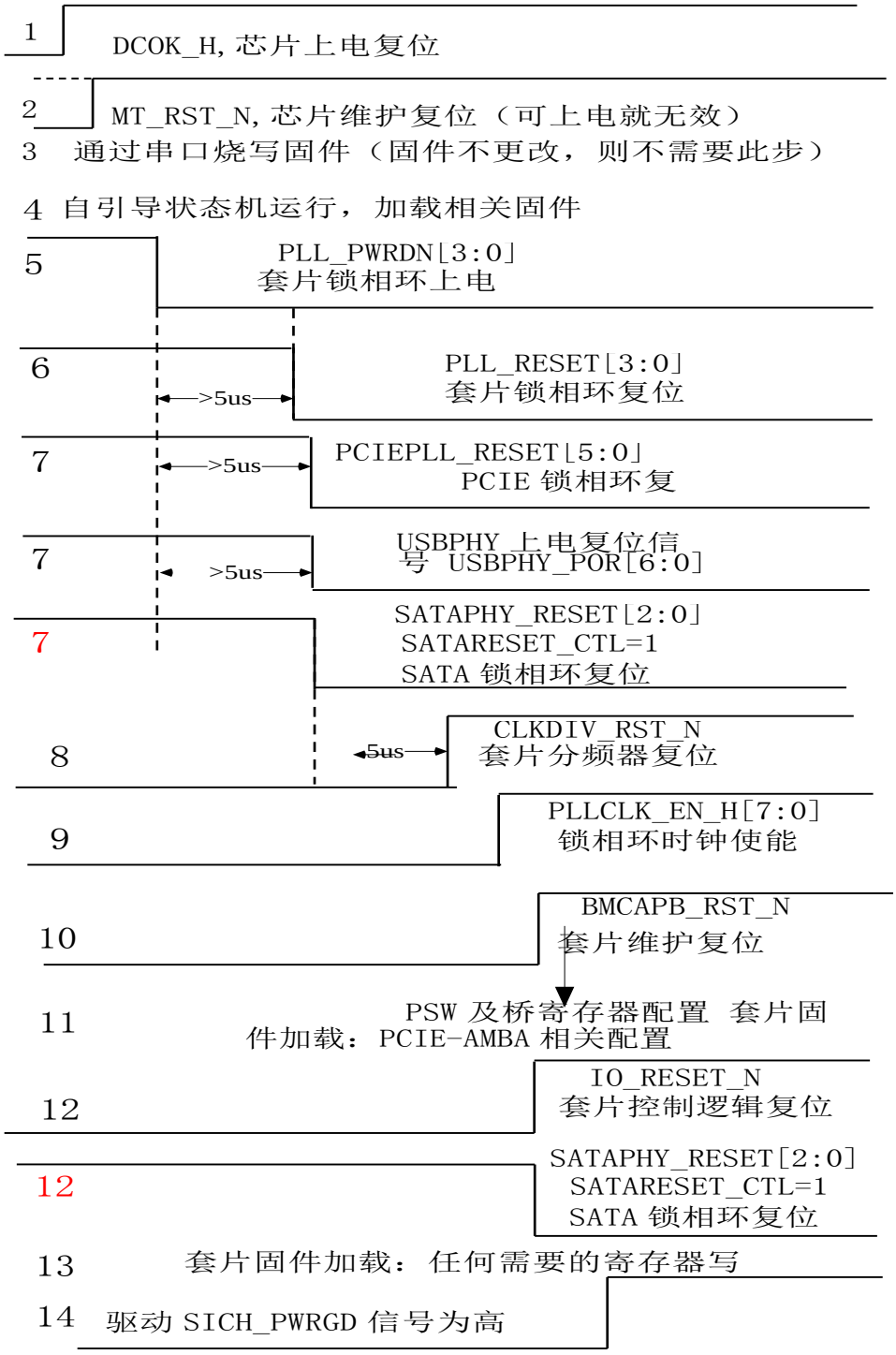


图 1-6 ICH2 内部复位时序图

注 1：可以通过固件配置 SATAPHY 的复位时序为 7 和 12 的其中之一。 固件内容见 15 章附录二。

2 PCI-E 接口

2.1. PCI-E 接口概述

ICH2 集成了一个 6 端口 PCI-E 2.0 交换开关，芯片对外接口为一个 X8 上游端口和 2 个 X8 下游端口或 4 个 X4 下游端口。在纯 switch 模式下（即只作为 switch 使用，不出其他 IO 设备接口，由固件控制），为一个上游端口和最多 5 个下游端口间交换的 PCIE switch，第五个下游端口为 X8 链路。 支持 PCI-E 2.0 规范定义的 switch 设备所有必选功能，每个端口 支持 5Gbps/2.5Gbps 链路传输速率，支持 PCI-E 规范定义的各种 switch 操作，支持 MSI 以 及 INTx 类型中断方式。以 ICH2 为核心的系统可通过该 PCIE switch 扩展 IO 访问能力，实 现高速对外访问。ICH2 集成的 PCI-E switch 有如下主要特性：

- 兼容 PCI Express base specification 2.0 规范
- 兼容 PCI-to-PCI Bridge Architecture specifications Revision1.2
- 非阻塞式传输，任意两个端口之间的通信不会影响其他端口的通信
- 支持 PCIE 序的要求
- 链路两端都实现了流控功能
- 外接口支持 8 位或 16 位 PIPE 接口
- 支持链路重配置，在多 lane 端口支持可编程链路宽度
- 支持 ASPM 规定的两种链路低功耗状态 L0S 和 L1
- 支持对内部端口的配置访问，硬件实现对下游设备配置访问时的 Type1/Type0 转换
- 上游下游端口都支持对 Unexpected Request 和 Unexpected Completion 的处理
- 实现了厂商特定功能寄存器 CSR
- 片内集成 SerDes
- 支持在链路初始化时自协商 X1、X2、X4 和 X8 端口链路带宽
- 支持在链路初始化时自协商 5Gbps/2.5Gbps 链路带宽
- 支持 Lane reversal
- 支持极性翻转
- 支持单个虚通道(VC0)
- 支持基本的和高级的错误报告机制

2.2. 详细说明

2.2.1. 名词解释

- 交换开关(Switch)：PCIE 交换开关是一个具有 2~n 个端口的设备，每个端口连一条PCIE 链路，在系统中用于多个设备的互连。典型的 PCI-E 拓扑结构如下图 2-1 所示。
- ICH2 PCI-E swtich 支持一个上游端口和 5 个下游端口的 6x6 全交换开关。

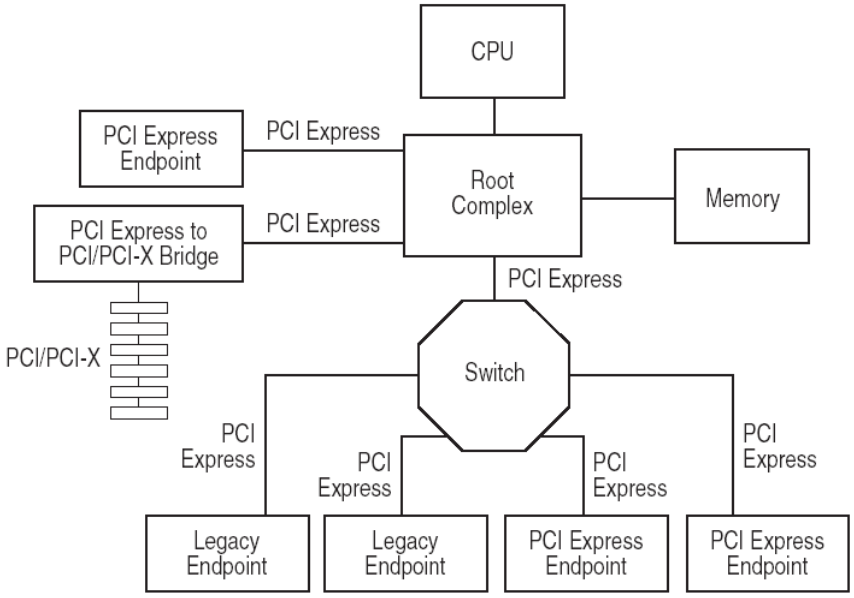


图 2-1 典型 PCI-E 拓扑结构

2.2.2. 信号管脚

2.2.2.1 高速差分信号

ICH2 中集成了 4 个 8X PCIE PHY， 数据信号采用 LVDS 电平，工作频率为 2.5GHz，信号传输带宽可以达到 5Gbps。PCIE 物理链路接口包含 64 对差分信号线 4 对差分参考时钟线，如下表所示

表 2-1 PCI-E 接口高速信号

名称	位宽	IO	描述
PCIE_US_TX[7:0]	8	输出	US 端口差分输出的 8 对数据信号
PCIE_US_TX_P[7:0]	8		
PCIE_US_RX[7:0]	8	输入	US 端口差分输入的 8 对数据信号
PCIE_US_RX_P[7:0]	8		
PCIE_DS13_TX[7:0]	8	输出	DS13 端口差分输出的 8 对数据信号
PCIE_DS13_TX_P[7:0]	8		
PCIE_DS13_RX[7:0]	8	输入	DS13 端口差分输入的 8 对数据信号
PCIE_DS13_RX_P[7:0]	8		
PCIE_DS24_TX[7:0]	8	输出	DS24 端口差分输出的 8 对数据信号
PCIE_DS24_TX_P[7:0]	8		

PCIE_DS24_RX[7:0]	8	输入	DS24 端口差分输入的 8 对数据信号
PCIE_DS24_RX_P[7:0]	8		
PCIE_DS5_TX[7:0]	8	输出	DS5 端口差分输出的 8 对数据信号
PCIE_DS5_TX_P[7:0]	8		
PCIE_DS5_RX[7:0]	8	输入	DS5 端口差分输入的 8 对数据信号
PCIE_DS5_RX_P[7:0]	8		
PCIE_US_CLK	1	输入	US 端口差分输入参考时钟信号
PCIE_US_CLK_P	1		

PCIE_DS13_CLK	1	输入	DS13 端口差分输入参考时钟信号
PCIE_DS13_CLK_P	1		
PCIE_DS24_CLK	1	输入	DS24 端口差分输入参考时钟信号
PCIE_DS24_CLK_P	1		
PCIE_DS5_CLK	1	输入	DS5 端口差分输入参考时钟信号
PCIE_DS5_CLK_P	1		

说明：PCI-E 接口的参考时钟必须采用差分输入，通常采用频率为 100MHz 的时钟信号。

2.2.3. 热插拔信号

芯片内集成的 PCIE Switch，其中 DS1 下游端口支持热拔插，热拔插信号为低速信号，采用 2.5V LVCOMS 电平，以下为热拔插信号说明：

表 2-2 PCI-E 热拔插信号

名称	位宽	方向	含义
DS1_ATTN_BTN_PRESD	1	输入	注意按钮按下
DS1_SENS_MRL_SENS_STATE	1	输入	插槽 MRL 锁扣状态，0：锁扣松开，1：锁扣锁紧
DS1_SENS_PRES_DET_STATE	1	输入	插槽上是否有卡插入：0 有卡，1：
DS1_SET_MRL_SENS_CHANGED	1	输入	插槽 MRL 锁扣状态是否发生过变
DS1_SET_PWR_FAULT_DETECTED	1	输入	电源错误检测，1：检测到电源错误，0：未检测到电源错误
DS1_ATTN_INDICATIOR_CNTRL_0	1	输出	注意指示灯控制信号：00：保留，01：开 10：闪，11： 关
DS1_ATTN_INDICATIOR_CNTRL_1	1	输出	
DS1_POWER_CONTROLLER	1	输出	电源控制信号，0：关电，1：开
DS1_PRES_DET_CLERA	1	输出	按钮检测清除，为 1表示需要清
DS1_PWR_INDICATIOR_CNTRL_0	1	输出	电源指示灯控制信号：

DS1_PWR_INDICATIOR_CNTRL_1	1	输出	00：保留，01：开 10：闪，11： 关

2.2.4. 参考电阻信号

每个 PCIE PHY 都需要端接参考电阻，根据手册要求，都需要通过 200 欧 1% 100ppm/C 精度的电阻连接到地。

表 2-3 PCIE PHY 参考电阻引脚

名称	位宽	方向	含义
PCIE_US_RESREF	1	输入	各个 PHY 参考电阻引脚
PCIE_DS13_PHY0_RESREF	1	输入	
PCIE_DS13_PHY1_RESREF	1	输入	
PCIE_DS24_PHY0_RESREF	1	输入	
PCIE_DS24_PHY1_RESREF	1	输入	
PCIE_DS5_PHY1_RESREF	1	输入	

2.2.4.1. 链路训练成功指示信号

表 2-4 PCIE 链路训练成功指示

名称	位宽	方向	含义
CPU_LINKUP	1	输出	为 1 表示 ICH2 与 CPU 之间的 PCIE 链路已训练成功

2.3. 应用要求

2.3.1. 参考时钟要求

ICH2 的 PCI-E 接口使用第三方 IP 实现，共四个同样的 X8 PHY,它要求从片外（板级）提 供参考时钟。 参考时钟采用差分输入，具体为对应的 PCIE_*_CLK 和 PCIE_*_CLK_P 。为说 明一致， 以下把 PCIE_*_CLK 称为 ref_pad_clk_m，PCIE_*_CLK_P 称为 ref_pad_clk_p。

2.3.2. 核心工作时钟要求

PCIE Switch 要求核心交换带宽必须大于接口数据带宽，因此，对于各个端口来说，在 PIPE 接口为 250Mhz @16bit，内部数据通路为 128 位的情况下，内部核心工作频率不能低于 250Mhz。

2.3.3. 延迟和电气要求

参考时钟必须符合各种应用下的抖动要求，参考时钟频率则可选频率范围为99.52MHz~400MHz。

表 2-5 参考时钟电气特性

参数	最小	最大	单位
频率	99.52	400	MHz
占空比率	40	60	%
输入电压	0	1.8	Vdc
差分输入摆幅	0.300	0.900	Vac_pp
普通模式输入电压	0.150	1.44	V
输入边界比率	0.6	-	V/ns

2.3.4. 端接要求

PCI-E 规范要求差分输入参考时钟的摆幅必须限制在 0～700mV 之间，建议使用高阻驱 动。电阻网络需尽量贴近时钟驱动端以提供良好的匹配电阻，具体如图 2-2 所示：

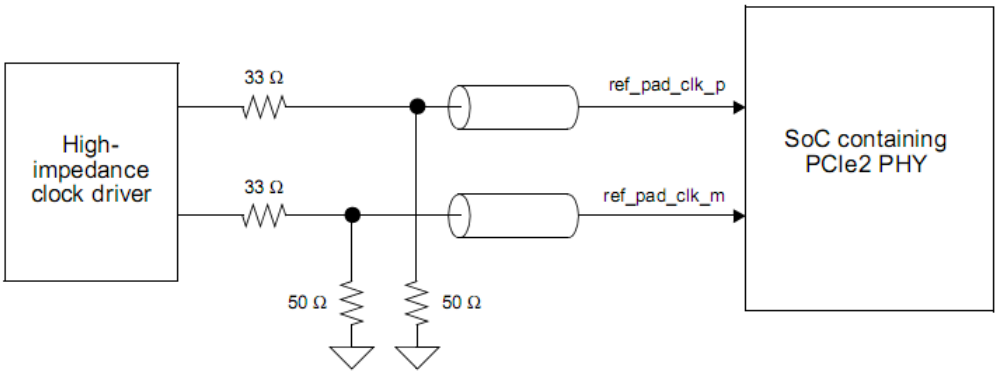


图 2-2 差分时钟输入参考电路

ref_pad_clk_p/ref_pad_clk_m 作为差分时钟信号要求其有较高的斜率，其斜率必须满足1V/ns。由于 5mV 的串扰可导致 5ps 抖动，因此在对差分时钟信号布局布线时需要考虑这些方面的因素，并进行必要的屏蔽。

正常条件下，ref_pad_clk_p/ref_pad_clk_m 由差分时钟源驱动。在测试模式下，可以由单端时钟驱动。在单端测试模式下，未使用的时钟驱动可以接地或悬空。

2.3.5. 功耗要求

表 2-4 列出了 PCI-E 接口 PHY 和其它逻辑在各种运行模式下的功耗情况：

表 2-6 PCI-E 部分功耗要求

工作条件 2.5 or 5 Gbps，1-V TxLVL	功耗（mW） Support/lane	电流 (mA) vp/vp	Lane 电流 (mA) vptx/vp/vph
PCI-E P0 state			
1.8/0.9/1.8/55	18.9/104.3	5.7/7.7	35.55/19.6/12.6
1.5/0.9/1.8/55	18.9/93.7	5.7/7.7	35.55/19.6/12.6
PCI-E P0s state			
1.8/0.9/1.8/55	18.9/47.6	5.7/7.7	5.1/18.5/12.3
1.5/0.9/.18/55	18.9/46	5.7/7.7	5.1/18.5/12.3
PCI-E P1 state			
1.8/0.9/1.8/55	18.9/13.3	5.7/21.7	2.3/2.2/4
1.5/0.9/.18/55	18.9/12.6	4.4/24.2	2.3/2.2/4
PCI-E P2			
1.8/ 0.9/1.8/55	4.9/2.8	4.9/0.3	0/1.6/0.8
1.5/0.9/1.8/55	4.9/2.8	4.9/0.3	0/1.6/0.8

2.4. 初始化与测试

2.4.1. 初始化

ICH2 的 PCI-E Switch 初始化包括基本上电过程和硬件逻辑完成的初始化，以及在 PCI-E 接口正常工作时（接收/发送请求响应）的初始化。

2.4.1.1. PCI-E switch 基本上电过程

- 加电后，板级必须为 PCIE Switch 提供稳定的参考时钟（默认使用 100Mhz）；
- 片内 SerDes 自带的 PLL 根据参考时钟锁定并输出给 Switch 各个端口稳定的工作时钟 (250MHz)；
- PCI-E 内置逻辑在检测到接收器的存在后，接口开始自动与外接设备交互 TS 序列，协 商链路宽度与速度；
- 协商成功后则链路建立，等待软件进行进一步初始化并开始正常工作。

2.4.1.2. PCI-E switch 的软件初始化

在芯片正常工作之前，软件需要按照 PCIE 规范的要求进行系统枚举，即对系统拓扑结 构进行配置，打开各种空间的使能，配置各个端口的基限寄存器等。至少需要配置 Switch上下游端口的一系列配置寄存器：

- **配置各个端口的源级总线号、次级总线号、下属总线号寄存器**根据枚举算法确定的系统拓扑进行配置
- **配置 Command Register**

  Command Register 主要是一些和 PCI 兼容性相关的配置，需要打开 I/O Space, Memory Space 和 Bus Master 的使能
- **配置 Device Control Register**

Device Control Register 主要用来对设备的行为进行配置，其中的 Max_Payload_Size 域 需要根据系统枚举的结果配置为 128B 或 256B

- **配置基限寄存器**

根据对设备申请 BAR 空间的读取和操作系统空间分配原则，对每个端口的存储器基点、 存储器界限寄存器，可预取存储器基点、存储器界限寄存器，IO 基点、IO 界限寄存器进行 配置。

2.4.2. 测试

PCIE Switch 的下游端口提供近端环回链路测试功能，测试发送通道到 PHY 的 PCS 层 环回至自身接收通道的情况下，链路训练状态机是否能正常训练到 L0 状态

1、 维护写被测下游端口的 loopbk_en 寄存器为 1；
2、 等待一段时间；
3、 维护读被测下游端口的 CSR 寄存器 ltssm_state，如果为 2d，这表明近端环回方式下链路训练成功；
4、 维护写被测下游端口的 loopbk_en 寄存器为 0；

2.5. 交直流电气特性

ICH2 PCI-E 接口使用第三方提供的 PCI-E 2.0 PHY，它完全符合 PCI-E 2.0 的标准。PCI-E 工作环境及电气参数在《PCI Express base specification Version2.0》规范中第四章有详细定义和说明，以下部分给出标准中发送器与接收器的特征列表。

2.5.1. 发送特征参数

表 2-7 PCI-E 接口发送参数

名称	参数	2.5GT/s	5GT/s	单	描述
UI	单位间隔	399.88(min)	199.94(min)	ps	1
$V_{TX-DIFF-PP}$	差分 P-P Tx 电压摆幅	0.8(min) 1.2(max)	0.8(min) 1.2(max)	V	$2*\|V_{TXD+}-V_{TXD-}\|$
$V_{TX-DIFF-PP-LOW}$	低功耗差分 P-P Tx 电	0.4(min) 1.2(max)	0.4(min) 1.2(max)	V	$2*\|V_{TXD+}-V_{TXD-}\|$, 9
$V_{TX-DE-RATIO-3.5dB}$	Tx 去加重比例	3.0(min)\ 4.0(max)	3.0(min) 4.0(max)	dB	11
$V_{TX-DE-RATIO-6dB}$	Tx 去加重比例	-	5.5(min) 6.5(max)	dB	11
$T_{MIN-PULSE}$	瞬时独立脉宽	-	0.9(MIN)	UI	2,10
T_{TX-EYE}	发送眼图	0.75(min)	0.75(min)	UI	不包括 SSC 或 refclk 的抖动。2,3,4,10
$T_{TX-EYE-M3DIAN-to-MAX-JITTER}$		0.125(max)	-	UI	2
$T_{TX-HF-DJ-DD}$	大于 1.5MHz	-	0.15(max)	UI	2,10
$T_{TX-LF-RMS}$	Tx RMS 抖动	-	3.0	ps	在 10kHz-1.5MHz
$T_{TX-RISE-FALL}$	发送上升下降时间	0.125(min)	0.15(min)	UI	测量从 20%-80%的摆动。2
$T_{RF-MISMATCH}$	发送上升/下降不匹配	-	0.1(max)	UI	测量从 20%-80%的摆动。2
BW_{TX-PLL}	最大发送 PLL 带宽	22(max)	16(max)	MHz	6
$BW_{TX-PLL-LO-3DB}$	对 3dB 峰值，最小发送	1.5(min)	8(min)	MHz	6,8
$BW_{TX-PLL-LO-1DB}$	对 1dB 峰值，最小发送	-	5(min)	MHz	6,8
$PKG_{TX-PLL1}$	8MHz 峰值 PLL 发送	-	3.0(max)	dB	6,8
$PKG_{TX-PLL2}$	5MHz 峰值 PLL 发送	-	1.0(max)	dB	8
$RL_{TX-DIFF}$	发送封装和 差分返回损失	10(min)	对于 0.05-1.25GHz 10(min); 对于 1.25GHz-	dB	

RL_{TX-CM}	发送封装和共模返回	6(min)	6(min)	dB	
$Z_{TX-DIFF-DC}$	差分发送直流阻抗	80(min)	120(max)	欧姆	
$V_{TX-CM-AC-PP}$	发送交流共模电压(5GT/s)	-	100(max)	mVPP	5
$V_{TX-CM-AC-P}$	发送交流共模电压(2.5GT/s)	20	-	mV	5
$I_{TX-SHORT}$	发送短路电流限制	90(max)	90(max)	mA	当短接地时，发送
$V_{TX-DC-CM}$	发送DC共模电压	0(min) 3.6(m	0(min) 3.6(m	V	在任意情况下发送管脚，允许DC
$V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$	在L0和Electrical Idle时的DC共模电压绝对差值	0(min) 100(max)	0(min) 100(max)	mV	$\lvert V_{TX-CM-DC\ [during\ L0]} - V_{TX-CM-Idle-DC\ [during\ Electrical\ Idle]}\rvert$ <= 100 mV $V_{TX-CM-DC}$ = $DC_{(avg)}$ of $\lvert V_{TX-D+} + V_{TX-D-}\rvert/2$ $V_{TX-CM-Idle-DC}$= DC(avg) of
$V_{TX-CM-DC-LINE-DELTA}$	在D+与D-之间的DC共模电压的绝对差值	0(min) 25(max)	0(min) 25(max)	mV	$\lvert V_{TX-CM-DC-D+\ [during\ L0]} - V_{TX-CM-DC-D-\ [during\ L0.]}\rvert \le$ 25 mV $V_{TX-CM-DC-D+}$ = $DC_{(avg)}$ of $\lvert V_{TX-D+}\rvert$ [during L0] $V_{TX-CM-DC-D-}$ = $DC_{(avg)}$ of $\lvert V_{TX-}$
$V_{TX-IDLE-DIFF-AC-p}$	电气空闲差分峰值输出电压	0(min) 20(max)	0(min) 20(max)	mV	$V_{TX-IDLE-DIFFp}$ = $\lvert V_{TX-Idle-D+} - V_{Tx-Idle-D-}\rvert \le 20$ mV.使用高通滤波器来过滤直流
$V_{TX-IDLE-DIFF-DC}$	直流电气空闲时差分输出电压	-	0(min) 5(max)	mV	$V_{TX-IDLE-DIFF-DC}$ = $\lvert V_{TX-Idle-D+} - V_{Tx-Idle-D-}\rvert \le 5$ mV. 使用低通
$V_{TX-RCV-DETECT}$	接收方检测时允许发送方电压变化	600(max)	600(max)	mV	发送方为了检测是否存在接收方 而改变其正相电 压的

$T_{TX-IDLE-MIN}$	电气空闲时的最小时间	20(min)	20(min)	ns	发送方必须处于电气空闲的最
$T_{TX-IDLE-SET-IDLE}$	从发送序列到电气空闲的最	8(max)	8(max)	ns	
$T_{TX-IDLE-TO-DIFF-DATA}$	从电气空闲转至发送有效差分	8(max)	8(max)	ns	
$T_{CROSSLINK}$	Crosslink 随机超时	1.0(max)	1.0(max)	ms	为了避免在 crosslink 配置中的潜在冲突而设置随
$L_{TX-SKEW}$	Lane-to-Lane 输出偏斜	500ps+2UI (max)	500ps+4UI (max)	ps	一个发送器的任
C_{TX}	AC 去耦电容	75(min)	75(min)	nF	所有的发送器必

说明：

1) SSC 允许进行＋0 至-5000ppm 的时钟频率调制，该调制频率不得超过 33kHz；
2) 对于 5GT/s 的测量需要超过 12.5GHz 带宽的示波器，而对于 2.5GT/s 的测量需要超过 6.2GHz 带宽的示波器。在测量时需要考虑测量管脚的影响；
3) 发送抖动的测量需要把发送器使用低抖动理想时钟，把被测设计作为参考负载；
4) 发送原始抖动数据一定包含一种滤波函数，该函数代表了最差情况下 CDR 的追踪频率范围。2.5GT/s 和 5GT/s 的滤波函数定义可参见 PCI-E 的相关标准；
5) $V_{TX-AC-CM-PP}$ 与 $V_{TX-AC-CM-P}$ 的具体定义为 $V_{TX-AC-CM-PP} = max(V_{D+} + V_{D-})/2 - min(V_{D+} + V_{D-})/2$，$V_{TX-AC-CM-P} = RMS[(V_{D+} + V_{D-})/2 - DC_{AVG}(V_{D+} + V_{D-})/2]$，这些测量是基于 10^6UI；
6) 发送 PLL 带宽必须满足上表的要求，PLL 的顶点不能超越上表规定的值；
7) 在对共模与差分回路损耗进行测量时，被测模块必须上电并且 DC 隔离，其 data+/data-的静态值必须是 low-Z 状态；
8) 对于 2.5GT/s 实现需要考虑 PLL 带宽与峰值的一种组合，对于 5GT/s 的实现必须考虑 PLL 带宽与峰值的两套组合。如果 PLL 的最小带宽大于 8MHz，那么可以允许最大 3.0dB 的峰值；如果 PLL 的最小带宽被放松到 5MHz，那么需要更紧的峰值，比如 1.0dB。对于以上两种情况的最大带宽都是 16MHz；
9) 低摆幅输出电压 VTX-DIFF-PP-LOW 在实现时无需去加重，具体如图 2-所示；

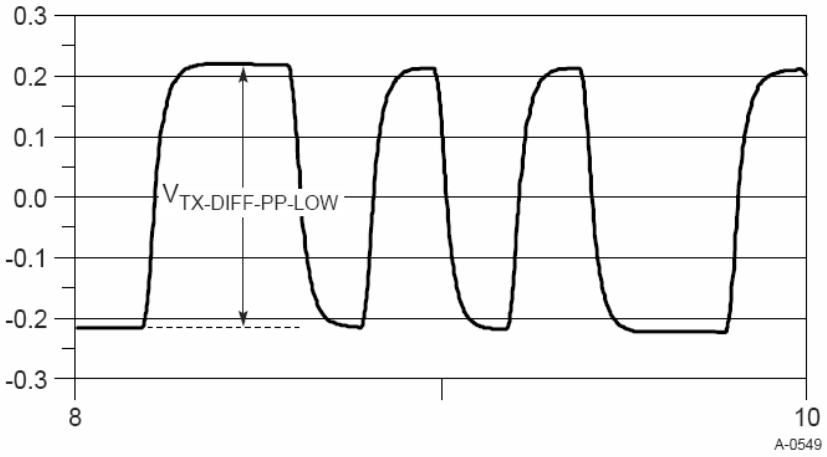


图 2-3 PCI-E 发送去加重

10) 对于 5GT/s 必须除去去加重抖动的影响。此时需要额外增加一个高通滤波函数，具体如图 2-所示

11) RC 的发送去加重参数由其上游的控制器完成，下行的发送去加重参数在设置时要求 PCI-E 链路工作在 2.5GT/s。

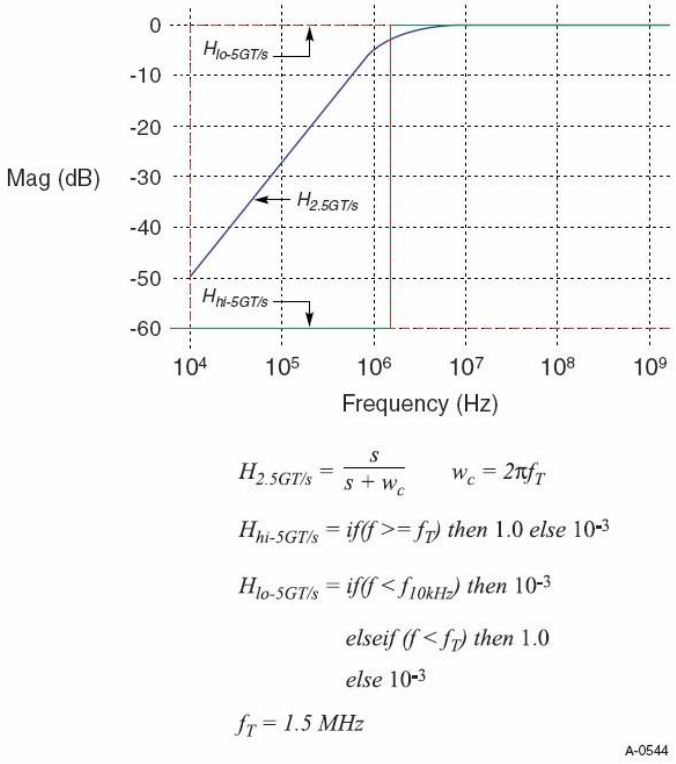


图 2-4 PCI-E 发送侧高通滤波器

2.5.2. 接收特征参数

表 2-8 PCI-E 接口接收参数

名称	参数	2.5GT/s	5GT/s	单位	描述
UI	单位间隔	399.88(min)	199.94(min)	ps	
$V_{RX-DIFF-PP-CC}$	用于参考时钟的接收峰峰电	0.175(min) 1.2(ma	0.120(min) 1.2(ma	V	
$V_{RX-DIFF-PP-DC}$	用于数据的接收	0.175(min) 1.2(ma	0.100(min) 1.2(ma	V	
T_{RX-EYE}	接收眼图宽度	0.40(min)	-	UI	1
$T_{RX-TJ-CC}$	最大接收内在定	-	0.40(max)	UI	2
$T_{RX-TJ-DC}$	最大接收内在定	-	0.34(max)	UI	2
$T_{RX-TJ-DD-CC}$	最大接收内在确定性定时错	-	0.30(max)	UI	2
$T_{RX-DJ-DD-DC}$	最大接收内在确定性定时错	-	0.24(max)	UI	2
$T_{RX-EYE-MEDIAN-to-MAX-JITTER}$	理论中点与实测中点之	0.3(max)	-	UI	仅适用于 2.5GT/s
$T_{RX-MIN-PULSE}$	接收最小脉	-	0.6(min)	UI	根据 10^{-12} 误码率测量得到的最差 Tj
$V_{RX-MAX-MIN-RATIO}$	在连续 UI 时的最	-	5(max)	-	接收眼图必须同时符合 V_{RX_EYE} 范围
$BW_{RX-PLL-HI}$	最大接收	22(max)	16(max)	MHz	二级 PLL 抖动传
$BW_{RX-PLL-LO-3DB}$	最小接收	1.5(min)	8(min)	MHz	二级 PLL 抖动传
$BW_{RX-PLL-LO-1DB}$	最小接收	-	5(min)	MHz	二级 PLL 抖动传
$PKG_{RX-PLL1}$	接收 PLL 最小 8MHz 峰值	-	3.0	dB	二级 PLL 抖动传 输约束函数。3

$PKG_{RX-PLL2}$	接收 PLL 最小 5MHz 峰 值	-	1.0	dB	二级 PLL 抖动传 输约束函数。3
$RL_{RX-DIFF}$	接收差分回路损耗	10(min)	10(min)对于 0.05-1.25GHz 8(min)对	dB	二级 PLL 抖动传 输约束函数。3
RL_{RX-CM}	接收共模	6(min)	6(min)	dB	4
Z_{RX-DC}	接收直流共模阻抗	40(min) 60(m	40(min) 60(m	欧姆	在接收端检测时 需要设置直流
$Z_{RX-DIFF-DC}$	直流差分阻抗	80(min)	-	欧姆	5
$V_{RX-CM-AC-P}$	接收端 AC 共模	150(max)	150(max)	mVP	在 Rx 管脚引入 50 欧姆接地端
$Z_{RX-HIGH-IMP-DC-POS}$	在复位或掉电时直流输	50k(min)	50k(min)	欧姆	7
	抗				
$Z_{RX-HIGH-IMP-DC-NEG}$	在复位或掉电时直 流输入反相共模	1.0k(min)	1.0k(min)	欧姆	7
$V_{RX-IDLE-DET-DIFFP-P}$	电气空闲检测	65(min) 175(m	65(min) 175(m	mV	$V_{RX-IDLE-DET-DIFFp-p}$ = 2*|V_{RX-}
$T_{RX-IDLE-DET-DIFF-ENTERTIME}$	进入不可预见电气 空闲检测门限集	10(max)	10(max)	ms	
$L_{RX-SKEW}$	物理 lane	20(max)	8(max)	ns	8

说明：

1) 接收端眼图范围是在 2×50 欧姆参考负载的条件下定义的；

2) 4 个内在定时错误参数是为了方便接收方设计者而定义的，他们是在接收方接收错误时测量的；

3) 两种 PLL 带宽和峰值的组合都是定义在 5.0GT/s 的，这样可以允许设计者来平衡这 两个参数。如果 PLL 的最小带宽大于等于 8MHz，那么允许 3.0dB 的峰值衰减。如果PLL 的最小带宽被放松到大于等于 5MHz，那么允许的峰值衰减必须满足不小于 1.0dB；4) 在测量共模与差分回路损耗时，被测电路必须上电并且使用直流隔离，差分输入必 须是低阻状态；

5) 当接收方端的接收端接必须连接接收方直流共模阻抗以确保接收端能够被检测到。 这

样就要求在规范要求的检测时间范围内，阻抗补偿必须能够迅速发挥作用；

6) 共模峰值电压由该表达式定义：max{|(Vd+ - Vd-) - V_{-CMDC}|}；

7) $Z_{RX-HIGH-IMP-DC-NEG}$ 和 $Z_{RX-HIGH-IMP-DC-POS}$ 定义了接收方输入反相与正相电压；

8) $L_{RX-SKEW}$ 参数是用来调节转发器的，它规范了同一条链路的多个不同差分物理通道的偏斜大小。

3 以太网接口

3.1. 以太网接口概述

ICH2 片内部集成了 2 个以太网控制器，提供 2 个独立的 PHY 接口，对外接口采用 GMII/MII 和 RGMII 接口。以太网接口控制器支持下面的特征：

- 支持下面的物理层（PHY）接口的 10/100/1000Mbps 数据传输率
  - IEEE 802.3 兼容的 GMII/MII 接口与外部千兆/快速以太网物理层通信
  - RGMII 接口
- 支持全双工和半双工操作
  - 支持半双工的 CSMA/CD 协议
  - 在千兆半双工操作模式下支持包突发和帧扩展
  - 支持全双工的 IEEE 802.3x 流控制
  - 在全双工操作用户应用中可选发送接收暂停控制帧
  - 支持半双工操作的背压
  - 自动传输零量子暂停帧在全双工模式下取消流控制输入
- 附加和开始帧数据（SFD）插入传输通路，在接收通路删除
- 在每个帧基础上可控制产生自动 CRC 校验和填充
- 在接收帧可选自动填充和剥离 CRC
- 可编程的帧长度，支持标准以太网帧，最大 1500B
- 可编程帧间隔(40-96 位时间倍，跨步值为 8)
- 支持多种灵活的地址过滤模式：
  - 最大 31 个额外 48 位预取（DA）地址过滤使用每字节屏蔽码
  - 最大 31 个 48 位 SA 地址比较检查使用每字节屏蔽码
  - 可选 64 位 Hash 过滤对于多播和单一传播（DA）地址
  - 可选传递所有多播寻址帧
  - 杂乱模式支持传递除了网络检测任何过滤的所有帧
  - 传递所有进入包（参考过滤器），带状态报告
- 分离用于传输和接收包返回的 32 位状态
- 支持接收帧 IEEE 802.1Q VLAN 标志检测
- 分离应用层传输，接收和控制接口
- 传输和接收数据通路可配置大小端支持
- 可选完成网络统计使用 RMON/MIB 计数器(RFC2819/RFC2665)
- 可选 MDIO 主接口用于网络设备配置和管理
- 可选模块用于检测 LAN 唤醒帧和 AMD 魔包帧
- 可选接收模块用于校验以太网帧封装的 IPv4 和 TCP 包卸载数据
- 可选增强接收模块用于检查 IPv4 和 IPv6 数据报封装的 IPv4 头校验和 TCP, UDP 或 ICMP 校验

3.2. 管脚说明

ICH2 内部包含以太网 GMAC 处理模块，对外接口采用 GMII/MII 和 RGMII 接口。对应 信号采用 1.8V CMOS 接口（输入兼容 2.5V），输出驱动能力为 12mA。总共 58 根信号，其 相关信号如下表所示：

表 3-1 以太网接口信号定义

名称	位宽	IO	描述
GMAC0_CLK_OSC	1	INPUT	GMAC0 晶振时钟输入
GMAC0_MII_TX_CLK	1	INPUT	GMAC0 的 PHY 时钟输入
GMAC0_CLK_RX	1	INPUT	GMAC0 接收时钟
GMAC0_PHY_INTF_SEL	1	INPUT	GMAC0 PHY 接口类型选择 0，GMII/MII 模式； 1：RGMII 模式。
GMAC0_PHY_RXDV	1	INPUT	GMAC0 PHY 接收数据有效
GMAC0_PHY_RXER	1	INPUT	GMAC0 PHY 接收错误
GMAC0_PHY_RXD[7:0]	8	INPUT	GMAC0 PHY 接收数据。MII 和 RGMII 模式时 ，[3:0]位数据有效，[7:4]数据无效
GMAC0_PHY_CRS	1	INPUT	GMAC0 PHY CRS
GMAC0_PHY_COL	1	INPUT	GMAC0 PHY 冲突
GMAC1_CLK_OSC	1	INPUT	GMAC1 晶振时钟输入
GMAC1_MII_TX_CLK	1	INPUT	GMAC1 的 PHY 时钟输入
GMAC1_CLK_RX	1	INPUT	GMAC1 接收时钟
GMAC1_PHY_INTF_SEL	1	INPUT	GMAC1 PHY 接口类型选择 0，GMII/MII 模式； 1：RGMII 模式。
GMAC1_PHY_RXDV	1	INPUT	GMAC1 PHY 接收数据有效
GMAC1_PHY_RXER	1	INPUT	GMAC1 PHY 接收错误
GMAC1_PHY_RXD[7:0]	8	INPUT	GMAC1 PHY 接收数据。MII 和 RGMII 模式时 ，[3:0]位数据有效，[7:4]数据无效
GMAC1_PHY_CRS	1	INPUT	GMAC1 PHY CRS
GMAC1_PHY_COL	1	INPUT	GMAC1 PHY 冲突
GMAC0_CLK_TX	1	OUTPUT	GMAC0 发送时钟
GMAC0_GMII_MDC	1	OUTPUT	GMAC0 SMA 管理数据时钟
GMAC0_PHY_TXEN	1	OUTPUT	GMAC0 PHY 发送数据使能
GMAC0_PHY_TXER	1	OUTPUT	GMAC0 PHY 发送错误
GMAC0_PHY_TXD[7:0]	8	OUTPUT	GMAC0 PHY 发送数据
GMAC1_CLK_TX	1	OUTPUT	GMAC1 发送时钟
GMAC1_GMII_MDC	1	OUTPUT	GMAC1 SMA 管理数据时钟

GMAC1_PHY_TXEN	1	OUTPUT	GMAC1 PHY 发送数据使能
GMAC1_PHY_TXER	1	OUTPUT	GMAC1 PHY 发送错误
GMAC1_PHY_TXD[7:0]	8	OUTPUT	GMAC1 PHY 发送数据
GMAC0_GMII_MD	1	INOUT	GMAC0 SMA 管理数据
GMAC1_GMII_MD	1	INOUT	GMAC1 SMA 管理数据

3.3. 使用方式

3.3.1. 板级逻辑示意图

以太网板级逻辑使用方法如下图所示：

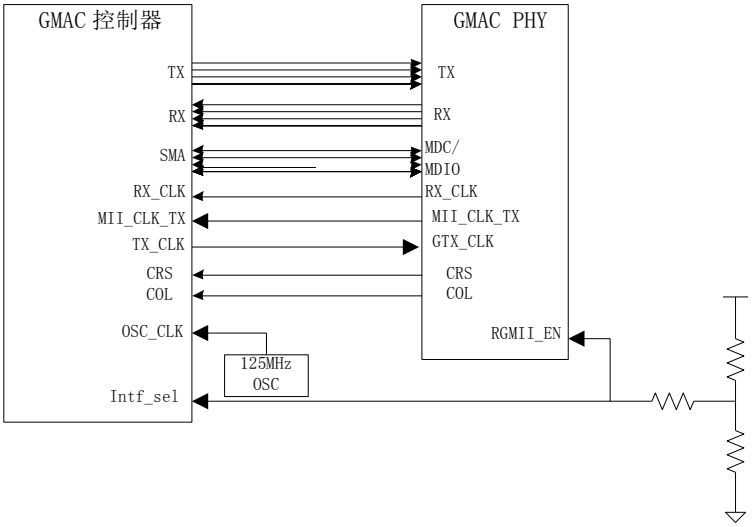


图 3-1 以太网板级逻辑示意图

3.3.2. 端口类型选择

GMAC 控制器支持 GMII/MII/RGMII 3 种以太网 PHY 的接口。通过管脚输入信号 GMAC0_PHY_INTF_SEL 和 GMAC0_PHY_INTF_SEL 选择端口类型：
GMAC0/1_PHY_INTF_SEL：0，GMII/MII 模式；1：RGMII 模式。

3.3.3. 时钟选择

- 当使用 MII 接口且工作在 10Mbps 速率时，PHY 提供给控制器 2.5MHz 的 RX_CLK 和 TX_CLK；
- 当使用 MII 接口且工作在 100Mbps 速率时，PHY 提供给控制器 25MHz 的 RX_CLK 和

TX_CLK；

- 当使用 GMII 或 RGMII 接口时，PHY 使用 GMAC 控制器输出的 125MHz 的 TX_CLK 时钟，并为控制器提供 125MHz 的 RX_CLK；

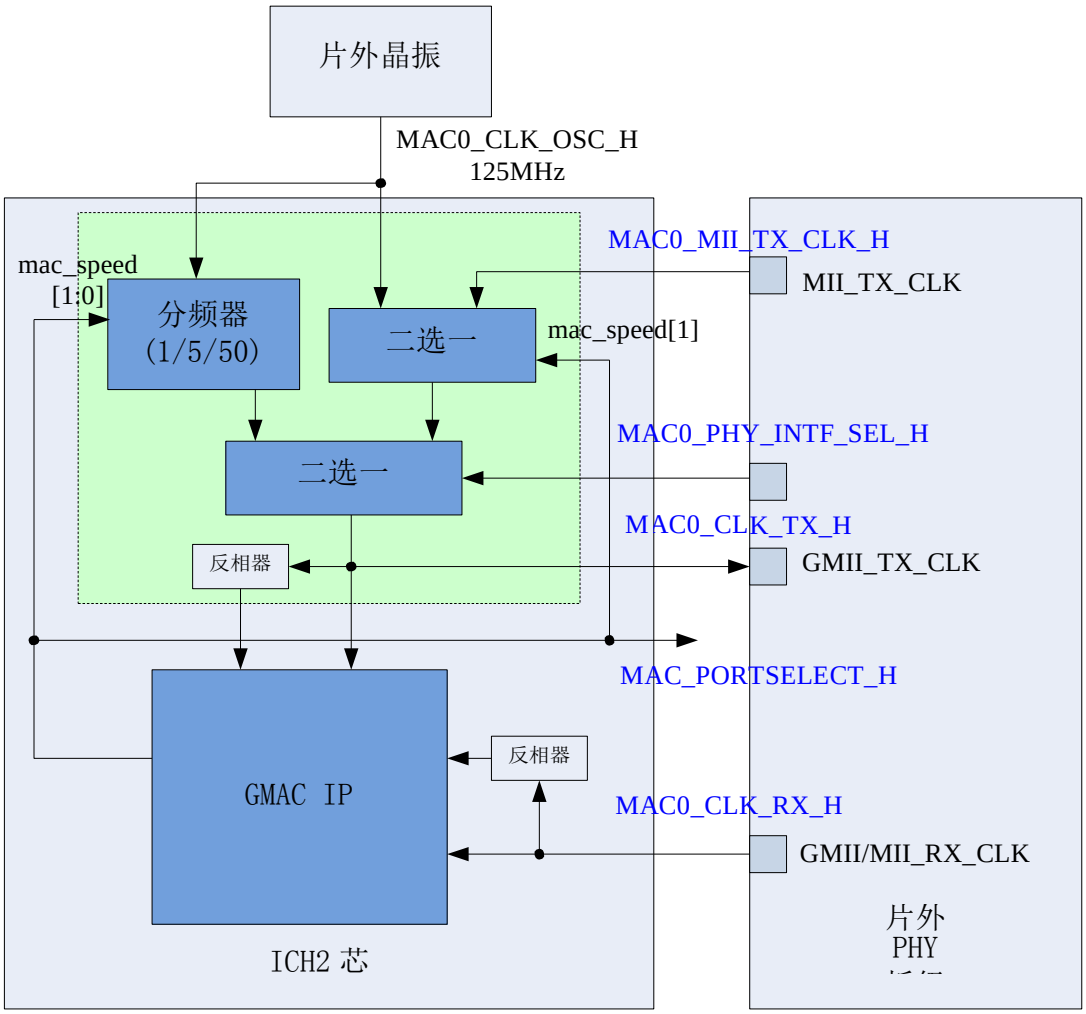


图 3-2 以太网时钟选择示意图

3.3.4. 数据信号

- 当使用 MII 接口或 RGMII 接口时，TX[3:0]/RX[3:0]有效；
- 当使用 GMII 接口时，TX[7:0]/RX[7:0]有效；

3.4. 初始化与测试

3.4.1. DMA 初始化–描述符

执行 DMA 初始化必须执行下面的操作。这个初始化序列只用于 GMAC-AHB、 GMAC-

AXI 和 GMAC-DMA 配置。对于 GMAC-AXI 配置，AXI 定义的控制寄存器也必须 编程。

1. 提供软件复位。这将复位所有的 GMAC 内部寄存器和逻辑。（DMA 寄存器 0 总 线模式寄存器的位 0）。
2. 等待完成复位过程（查询 DMA 寄存器 0 总线模式寄存器的位 0，只有复位操作完 成后这位才清除为 0）。
3. 设置下列域的值，完成对 DMA 寄存器 0 总线模式寄存器的初始化。
   a. 混合突发 MB 和 AAL（只有在 GMAC-AHB/GMAC-AXI 配置下）
   b. 固定突发 FB 或未定义突发
   c. 突发长度值和突发模式值
   d. 描述符长度（只有使用环模式时有效）
   e. TxDMA 和 RxDMA 仲裁策略
4. 在 AXI 总线模式寄存器中设置 AXI 接口选项

   a. 如果固定突发长度是使能的，选择 AXI 总线上最大可能的突发长度（位[7:1]）。5. 创建用于发送和接收的合适的描述符链。也必须确保接收描述符属于 DMA（描述 符第[31]位设置为 1。）参考–章节 7 描述符Ⅱ了解更多信息。当 OSF 模式使用时，需要最少 2 个描述符。
6. 软件重新使用任何描述符时，应该创建 3 个以上不同的发送或者接收描述符在描述 符链上。
7. 初始化接收和发送描述符列表地址，接收描述符基址寄存器是 DMA 寄存器 3，发 送描述符基址寄存器是 DMA 寄存器 4。
8. 设置 DMA 寄存器 6（操作模式寄存器）来编程下面的域，进行操作模式的初始化。 a. 接收和发送存储转发
   b. 接收和发送阈值控制（RTC 和 TTC） c. 硬件流控制使能
   d. 流控制激活和不激活用于 MTL 接收和发送 FIFO 的阈值（RFA 和 RFD） e. 错误帧和过短正确帧传递使能
   f. OSF 模式
9. 清除中断请求，通过写状态寄存器（DMA 寄存器 5）中设置的位（只是中断位）。 例如，通过向位[16]（正常中断汇总）写 1 将清除这位。
10. 通过编程中断使能寄存器（DMA 寄存器 7）使能中断。
11. 通过设置操作模式寄存器（DMA 寄存器 6）的 SR（位[1]）和 ST（位[13]）启动接 收和发送 DMA。

3.4.2. MAC 初始化

下面的 MAC 初始化操作可以在 DMA 初始化序列之后执行。如果 MAC 初始化在 DMA 建立之前完成，MAC 接收端的（下面的最后一步）只能在 DMA 激活后才可以使能。否则 接收帧将装填 RxFIFO 并导致溢出。如果使能了 TBI/SGMII/RTBI 接口则开始进行第 1 步和 第 2 步，否则执行第 3 和第 4 步。

1. 编程自动协商控制寄存器（GMAC 寄存器 48）使能自动协商 ANE（位[12]）。设 置 ELE（位[14]）将使能 PHY 回环发送数据，设置 RAN 位将重新开始自动协商（只 对于

TBI/SGMII/RTBI 的 PHY 接口）。

2. 检查自动协商状态寄存器（GMAC 寄存器 49）来完成自动协商过程。ANC（位[5]） 应该设置为 1，当链路状态（位[2]）设置为 1 时表明链路是连通的（只对于 TBI/SGMII/RTBI 的 PHY 接口）。
3. 编程 GMII 地址寄存器（GMAC 寄存器 4）来控制的用于外部 PHY 的管理周期， 例如物理层地址 PA（位[15-11]）。同样设置位[0]（GMII 忙）进行对外部 PHY 的 读写。
4. 从 PHY 读 GMII 数据寄存器（GMAC 寄存器 5）的 16 位数据，得到链路连接状态， 操作速率和操作模式，通过设置合适的寄存器地址值（GMAC 寄存器 4 位[15-11]）。
5. 提供 MAC 地址寄存器（GMAC 寄存器 16 用于 MAC 地址 0 高位，GMAC 寄存器 17 用于 MAC 地址 0 低位）。如果在你的配置中（在 coreConsultant 配置时）使能 了超过一个 MAC 地址，对他们进行适当的编程。
6. 如果在你的配置中使能了哈希过滤，编程哈希过滤寄存器（GMAC 寄存器 2 哈希 表高位寄存器，GMAC 寄存器 3 哈希表低位寄存器）。
7. 编程 MAC 帧过滤寄存器（GMAC 寄存器 1）中下面的域，用于对接收的帧进行适 当的过滤。
   a. 接收所有
   b. 混杂模式
   c. 哈希或预取过滤
   d. 单播，多播，广播和控制帧过滤设置等
8. 编程流控制寄存器（GMAC 寄存器 6）中下面的域，用于适当的流控制。 a. 暂停时间和其他暂停帧控制位
   b. 接收和发送流控制位
   c. 流控制忙/背压激活
9. 编程中断屏蔽寄存器的位，如果需要，如果对于你的配置是可用的。
10. 编程 MAC 配置寄存器（GMAC 寄存器 0）中合适的域，例如传输过程中的帧间隔，持续发送不使能等。基于自动协商你可以设置双工模式（位[11]），端口选择（位 [15]）等。
11. 设置 MAC 配置寄存器（GMAC 寄存器 0）中的传输使能 TE（位[3]）和接收使能 RE（位[2]）。

3.4.3. 正常接收操作

对于正常操作，可以进行下面的步骤。

1. 对正常的发送和接收中断，读中断状态。然后查询描述符，读属于主机的描述符状 态（发送或者接收）。
2. 在上面步骤完成后，在描述中设置适当的值，确保发送和接收描述符属于 DMA， 以重新开始发送和接收数据。
3. 如果描述符不属于 DMA（或者没有描述符可以使用），DMA 将进入悬挂状态。 发送或接收重新开始需要释放描述符，并且通过向 Tx/RX 查询命令寄存器写 0 来 发出查询命令（DMA 寄存器 1 发送查询命令寄存器，DMA 寄存器 2 接收查询命 令寄存器）。
4. 在调试过程中可以读到当前主机发送和接收描述符地址指针（DMA 寄存器 18 当 前

主机发送描述符寄存器，DMA 寄存器 19 当前主机接收描述符寄存器）。

5. 在调试过程中可以读到当前主机发送和接收缓冲地址指针（DMA 寄存器 18 当前 主机发送缓冲寄存器，DMA 寄存器 19 当前主机接收缓冲寄存器）。

3.4.4. 停止和开始操作

当传输需要暂停一段时间时，执行下面的操作。

1. 通过清除操作模式寄存器（DMA 寄存器 6）的 ST（位[13]），不使能发送 DMA（如果适当的）。
2. 等待前面所有的帧传输结束，这可以通过读 MAC 调试寄存器（GMAC 寄存器 9）的适当的位进行检查。
3. 不使能 MAC 发送和接收通过清除 MAC 配置寄存器（GMAC 寄存器 0）中的发送使能 TE（位[3]）和接收使能 RE（位[2]）。
4. 在确保 RxFIFO 中的数据都传送到主存后，不使能接收 DMA（如果适当的）。

5. 确保 TxFIFO 和 RxFIFO 都是空的。 要重启操作，

先启动 DMA，再使能 MAC 发送和接收。

3.4.5. 测试

ICH2 的以太网接口支持由设备驱动程序发起的片外环回测试。具体流程如下：

1) 使用交叉跳线（TX 与 RX 短接）连接物理连接器，或者在 ICH2 片外把 GMII 的 PHY_TXD [7:0]与 PHY_RXD [7:0]短接，或者通过把外接 PHY 配置成还回模式；
2) 按照正常机制发起发送以太网帧操作；
3) 按照正常接收机制接收以太网帧并写入内存；
4) 由软件读取内存对发送与接收帧进行比对，以确定以太网接口的功能是否正确。

ICH2 同时也支持基于 GMAC 的片内自环，基本环回测试流程与片外环回相同，区别在于软件此时需要配置 GMACR5[LM]为 1 即可。

注意环回测试时最好关闭包过滤功能。

3.5. 接口时序特性

以太网接口大多采用 GMII 和 RGMII 与外部 PHY 芯片进行连接，这两种典型接口的时序特性如下图所示：

表 3-2 GMII 发送时序特性

参数	Min	Typical	Maximum	Unit
发送时钟周期	-	8	-	ns
TX_SETUP	-	6	-	ns
TX_HOLD	-	1.7	-	ns

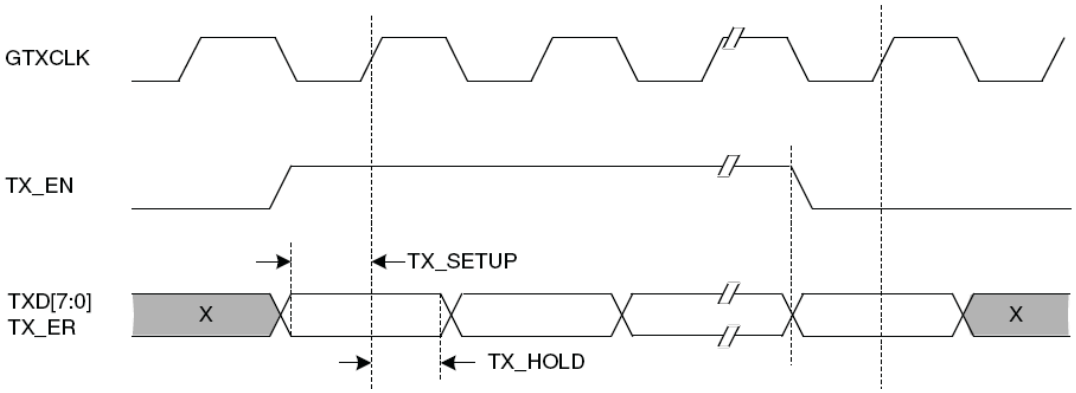


图 3-3 GMII 发送时序

表 3-3 RGMII 发送时序特性

参数	Min	Typical	Maximum	Unit
发送时钟周期	-	8	-	ns
$T_{SETUP}R$	-	-2	-	ns
$T_{HOLD}R$	-	-5.4~-5.8	-	ns

注：由于 GMAC 发送数据相对发送时钟 CLK_TX 上升沿的 setup 和 hold 时间为负值，因此必须根据 PHY 芯片 CLK_TX_to_phy 时钟与发送数据的时序要求在板级进行时钟相位调整。例如，若 PHY 芯片要求发送数据与发送时钟 CLK_TX_to_phy 的建立时间为 Tsetup_phy，则需要在板级将 GMAC 输出的时钟 CLK_TX 退后（2ns+Tsetup_phy）的延迟。

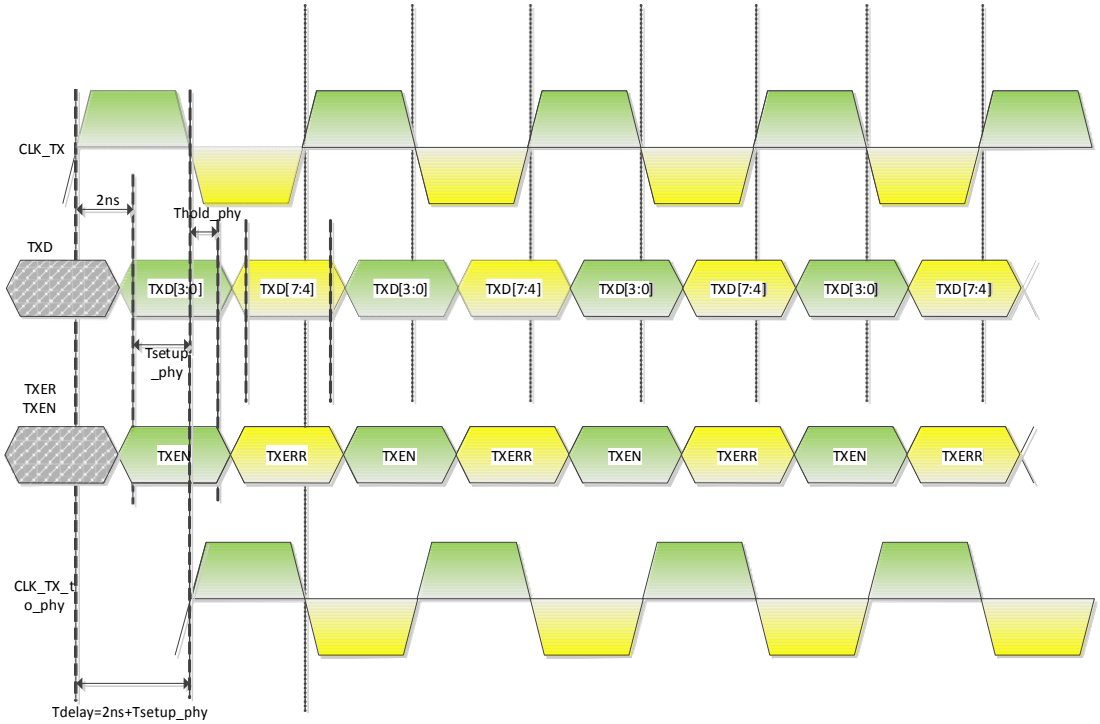


图 3-4 RGMII 接口发送时序

图 3-5 RGMII 接口接

收时序 表 3-4 RGMII

参数	Min	Typic	Maximum	Unit
接收时序特性				
发送时钟周期	-	8	-	ns
$T_{SETUP}R$	1	-	-	ns
$T_{HOLD}R$	1	-	-	ns

3.6. 交直流电气特性

以太网接口管脚使用标准 LVCOMS 1.8V 供电的接口缓冲，其管脚电气特性参见 0 节。

4 SATA 接口

4.1. 功能概述

ICH2 片内部集成了 3 端口 SATA 3.0 总线控制器，SATA 控制器提供下列特征：

- SATA 1.5Gb/s，SATA 3.0Gb/s，SATA 6.0Gb/s 速率
- eSATA（需要外部模拟逻辑支持 eSATA）
- 兼容下面的标准：
  - SATA 3.0
  - AHCI 1.3
- 可配置特征：
  - RX 数据缓冲恢复时钟系统
  - 当包含 RX 数据缓冲时的数据对界电路
  - OOB 信号检测和产生
  - 8b/10b 编码/解码
  - 机械在位切换，冷在位检测，活动 LED 支持
  - 输出端口表明 COMRESET 后协商的速率，用于电源优化
  - 设备睡眠（DevSleep）电源管理
- 当包含 TX OOB 信号时，也包含下面的特征：
  - SATA 1.5Gb/s，SATA 3.0Gb/s，SATA 6.0Gb/s 速率协商
  - 异步信号恢复，包括查询重试
  - 数字信号支持设备热插拔
- 电源管理特征，包括自动部分睡眠到全部睡眠状态转换
- BIST 回环模式
- 3 个 SATA 设备端口
- 每个端口的内部 DMA 引擎
- 硬件支持的最大 32 个条目的原始命令队列
- 下电模式下不使能 RX 和 TX 数据时钟
- 任何扇区大小

SATA 控制器不支持的特征和限制：

- 只支持 3 个端口，而不是 AHCI 规范中允许的最大数量 32 个
- 附件管理
- 远端逻辑响应 BIST 模式（PHY 的限制）
- 多消息 MSI
- 远端逻辑发起 BIST 模式（PHY 的限制）
- 非零缓冲偏移
- NVMHCI 和 BIOS/OS 交接 AHCI 特征
- OOB 突发不包括 D24.3 字符

- 以一种特有的方式实现预取，不能通过 AHCI 规范中定义的命令头 P 位。
- 端口复位（COMRESET）：首选机制用于错误恢复，替代软件复位
- 基于协议的端口选择器在位检测
- SATA 传统模式
- 厂商定义 BIST 模式
- 厂商定义 FIS 类型
- 产生 BIST 特征不兼容 ECN 51

4.2. 管脚说明

表 4-1 SATA 接口信号定义

名称	位宽	IO	描述
SATA_CLK_P	1	INPUT	SATA_PHY 的 差分参考时钟输入
SATA_CLK_N	1	INPUT	SATA_PHY 的 差分参考时钟输入
SATA_0_Rx_P	1	INPUT	sata 端口 0 差分接收信号
SATA_0_Rx_N	1	INPUT	sata 端口 0 差分接收信号
SATA_1_Rx_P	1	INPUT	sata 端口 1 差分接收信号
SATA_1_Rx_N	1	INPUT	sata 端口 1 差分接收信号
SATA_2_Rx_P	1	INPUT	sata 端口 2 差分接收信号
SATA_2_Rx_N	1	INPUT	sata 端口 2 差分接收信号
SATA_VP	2	INPUT	sata 端口低电压供电
SATA_VPH	2	INPUT	sata 端口高电压供电
SATA_VPTX	2	INPUT	Sata 发送电压供电
SATA_AHCI0_MP_SWITCH	1	INPUT	sata 端口 0 机械在位切换信号
SATA_AHCI0_CP_DET	1	INPUT	sata 端口 0 冷在位检测信号
SATA_AHCI1_MP_SWITCH	1	INPUT	sata 端口 1 机械在位切换信号
SATA_AHCI1_CP_DET	1	INPUT	sata 端口 1 冷在位检测信号
SATA_AHCI2_MP_SWITCH	1	INPUT	sata 端口 2 机械在位切换信号
SATA_AHCI2_CP_DET	1	INPUT	sata 端口 2 冷在位检测信号
SATA_0_Tx_P	1	OUTPUT	sata 端口 0 差分发送信号
SATA_0_Tx_N	1	OUTPUT	sata 端口 0 差分发送信号
SATA_1_Tx_P	1	OUTPUT	sata 端口 1 差分发送信号
SATA_1_Tx_N	1	OUTPUT	sata 端口 1 差分发送信号
SATA_2_Tx_P	1	OUTPUT	sata 端口 2 差分发送信号
SATA_2_Tx_N	1	OUTPUT	sata 端口 2 差分发送信号
SATA_AHCI0_ACT_LED	1	OUTPUT	sata 端口 0 状态指示灯
SATA_AHCI0_CP_POD	1	OUTPUT	sata 端口 0 冷在位上电信号
SATA_AHCI1_ACT_LED	1	OUTPUT	sata 端口 1 状态指示灯
SATA_AHCI1_CP_POD	1	OUTPUT	sata 端口 1 冷在位上电信号

SATA_AHCI2_ACT_LED	1	OUTPUT	sata 端口 2 状态指示灯
SATA_AHCI2_CP_POD	1	OUTPUT	sata 端口 2 冷在位上电信号
SATA_PHY_DTB_OUT_0	2	OUTPUT	sata phy 内部测试总线

SATA_0_RESREF	1	INOUT	sata 端口 0 参考电阻
SATA_1_RESREF	1	INOUT	sata 端口 1 参考电阻
SATA_2_RESREF	1	INOUT	sata 端口 2 参考电阻

4.3. 使用方式

4.3.1. 板级逻辑示意图

下图示意了 SATA 板级逻辑示意图：

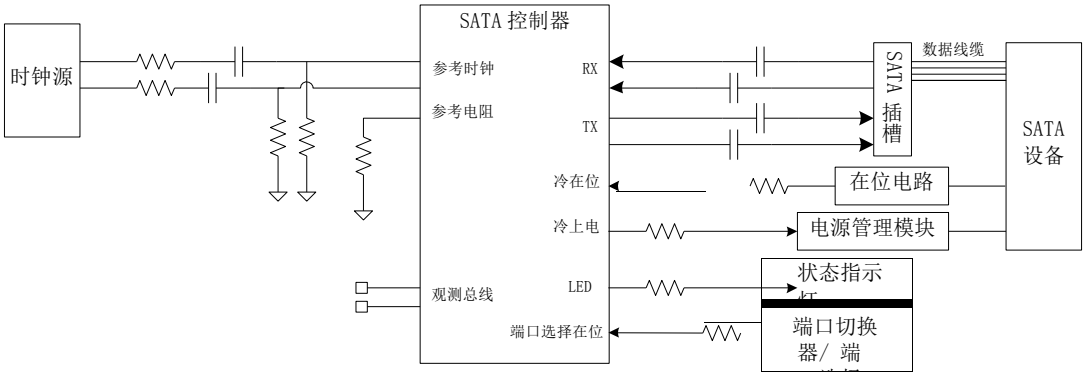


图 4-1 SATA 板级逻辑示意图

4.3.2. 数据链路

SATA 数据差分链路：

SATA_0_Rx_P SATA_0_Rx_N
SATA_1_Rx_P SATA_1_Rx_N
SATA_2_Rx_P SATA_2_Rx_N
SATA_0_Tx_P SATA_0_Tx_N
SATA_1_Tx_P SATA_1_Tx_N
SATA_2_Tx_P SATA_2_Tx_N

使用：

下图示意了 SATA 数据链路的连接方式：

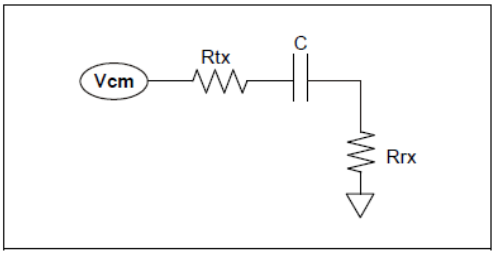


Figure 132 – Example Circuit for Common Mode Transients

0 < Vcm < 2 V
21.25 < Rtx < 40 Ohms
21.25 < Rrx < 40 Ohms
C < 0.024 uF (Two 0.012 uF capacitors in parallel)

图 4-2 SATA 数据链路的连接方式示意图

4.3.3. 参考时钟和参考电阻

SATA 参考时钟： SATA_CLK_P
SATA_CLK_N
SATA 参考电阻：
SATA_0_RESREF
SATA_1_RESREF
SATA_2_RESREF
下表中列出了对板级参考时钟和参考电阻的需求：

表 4-2 板级参考时钟和参考电阻需求

Symbol	Parameter	Min	Typ	Max	Units	Conditions
FREF_OFFSET	Reference clock frequency offset	-300		300	ppm	
RMSJREF_CLK	Reference clock Random Jitter (RMS)			3	ps	1.5MHz to Nyquist frequency. For example, for 100MHz reference clock, the Nyquist frequency is 50MHz.
DJREF_CLK	Reference clock cycle to cycle jitter			150	ps	DJ over all frequency
DCREF_CLK	Duty cycle	40		60	%	
VCMREF_CLk	Common mode input level	0		1.32	V	Differential inputs
VDREF_CLK	Differential input swing	0.3			Vpp	Differential inputs. See Note 1 for extra details
VOLREF_CLK	Single ended input logic Low	-0.3		0.3	V	If single ended input is used
VOHREF_CLK	Single ended input logic High	0.8		1.32	V	If single ended input is used
SWREF_CLK	Input edge rate	0.6			V/ns	
R_resref	Required external reference resistance		200		ohm	+/-1% accuracy
REF_CLK_SKEW	Ref Clock Skew (+/-)			200	pS	

VDREF_CLK/4+VCMREF_CLK<=vp+diode forward biasing voltage and VCMREF_CLK-VDREF_CLK/4>=-diode forward biasing voltage

4.3.4. 在位信号

SATA 在位信号：
SATA_AHCI0_CP_DET

SATA_AHCI1_CP_DET
SATA_AHCI2_CP_DET

使用：

测试板可以简单地使用可以插跳线帽的 2 个跳线针来实现。 协议中给出的参考电路。

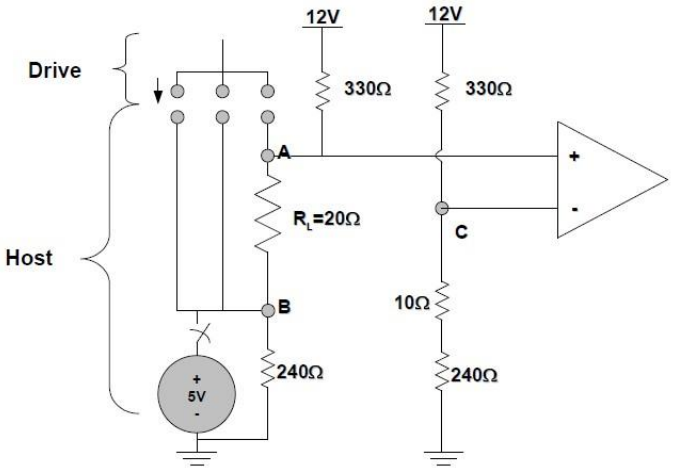


Figure 110 – Example presence detection implementation

Table 28 – Comparator voltages for alternate example presence detection circuit

	Receptacle Powered		Receptacle unpowered	
Device not present	V_A = 5.40 V	V_C = 5.17 V	V_A = 5.29 V	V_C = 5.17 V
Device present	V_A = 5.0 V	V_C = 5.17 V	*V_A = 5.05 V	V_C = 5.17 V

*If the inserted device provides a finite impedance to ground, then V_A should be lower than this value increasing the voltage differential further and increasing the margins.

图 4.3 SATA 在位信号参考电路示意图

4.3.5. 状态 LED

SATA 状态指示灯信号：

SATA_AHCI0_ACT_LED
SATA_AHCI1_ACT_LED
SATA_AHCI2_ACT_LED

使用：

SATA 协议中给出的 SATA 状态指示灯使用示意图。

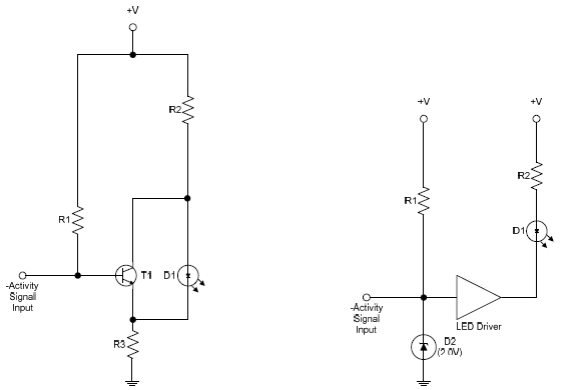


Figure 107 – Example host LED driver circuits

图 4-4 SATA 状态指示灯使用示意图 下面是 3

种 sata 状态指示灯的使用场景：

可以将多个端口的灯分开指示，也可以合并多个端口使用 1 个状态指示，还可以合并其他底板上的设备状态信号统一指示。

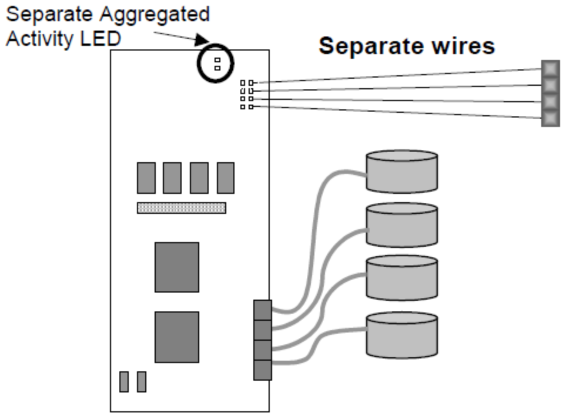


Figure 243 – Device Activity LEDs with Separate Wires

图 4-5 SATA 状态指示灯使用场景 1

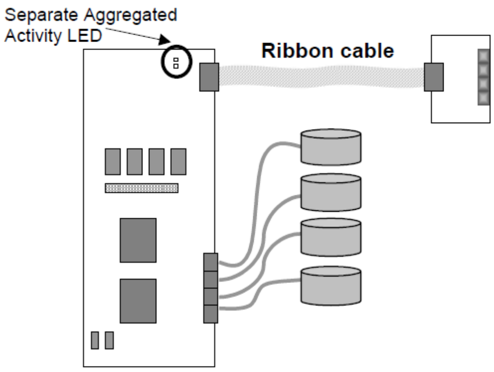


Figure 244 – Device Activity LEDs with Ribbon Cable

图 4-6 SATA 状态指示灯使用场景 2

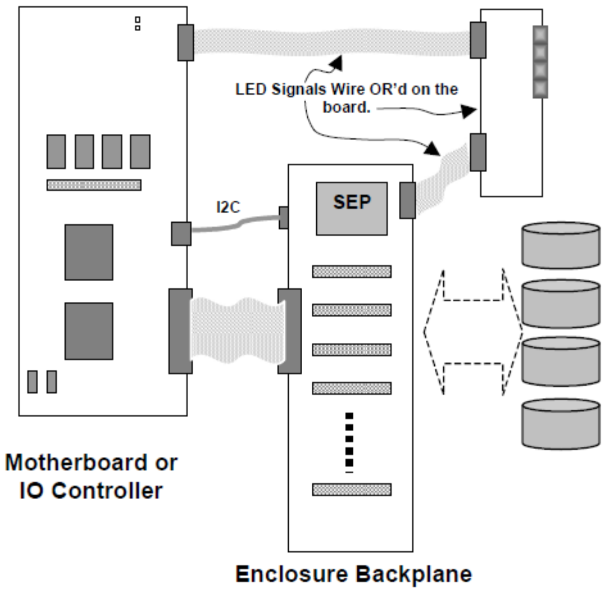


Figure 245 – Device Activity LEDs in a Storage Subsystem

图 4-7 SATA 状态指示灯使用场景 3

4.3.6. 机械在位切换信号

SATA 端口选择在位信号：

SATA_AHCI0_MP_SWITCH

SATA_AHCI1_MP_SWITCH

SATA_AHCI2_MP_SWITCH

使用：

测试板可以简单地使用可以插跳线帽的 2 个跳线针来实现。

端口选择器(Port selector)和端口硬件在位切换器(Mechanical Presence Switch)是类似的 Host 选择控制器。

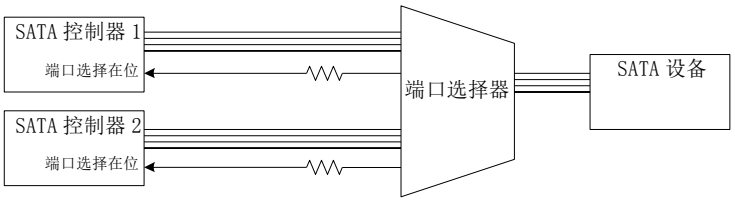


图 4-8 SATA 端口选择器使用示意图

协议中给出的 sata 端口切换/端口选择器的使用使用场景和方法：

在一些场景中，比如 RAID 应用，允许使用端口选择器，选择冗余的多个磁盘控制器对单个 磁盘进行访问，但一个时刻只能有 1 个控制器可以访问设备。协议并没有规定使能控制器的 方法，也没有规定控制器和选择器的协同工作方法。一般可以通过 OOB 信号或者边带信号 进行端口的使能，在我们设计中使用的就是边带控制信号，通过端口切换在位检查信号指示 端口选择器是否打开。

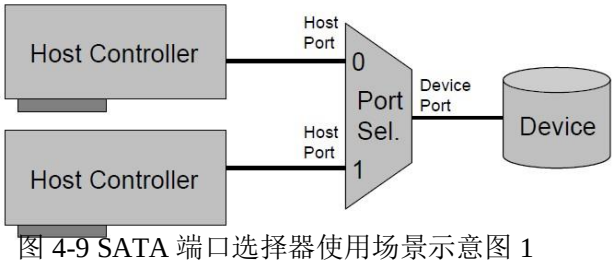


图 4-9 SATA 端口选择器使用场景示意图 1

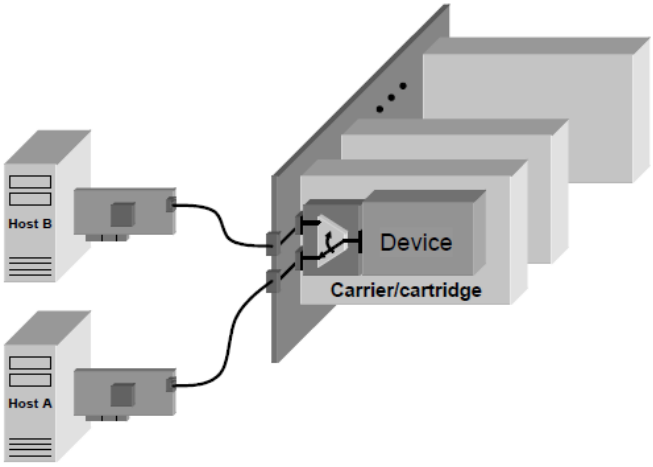


图 4-10 SATA 端口选择器使用场景示意图 2

4.3.7. 冷在位上电信号

SATA 冷上电信号信号：

SATA_AHCI0_CP_POD
SATA_AHCI1_CP_POD
SATA_AHCI2_CP_POD

使用：

这是用于板级电源管理时，给冷在位的 sata 设备上电的信号。如果没有实现板级 sata 设备 电源管理，可以不连接，或者连接到用于信号观测的跳针或电路。

4.4. 初始化与测试

4.4.1. 软件初始化

DWC_ahsata 的软件初始化包含 2 个独立的阶段：固件阶段（平台 BIOS）和系统软件 阶段。

4. 4. 1. 1. 固件定义的初始化

在上电后开始固件初始化。下列的寄存器应该初始化成合适的值，以反映平台的支持能力：

- CAP.SSS：支持交错启动
- CAP.SMPS：支持机械在位切换
- PI：端口实现数量
- P#CMD.HPCP：端口是否有热插拔能力。当端口的 P#CMD.MPSP 或 P#CMD.CPD 设置 为 1 时 P#CMD.HPCP 应该设置为 1。
- P#CMD.MPSP：端口上是否连接到机械在位切换器。
- P#CMD.CPD：端口上是否连接了冷在位检查逻辑。

注意：固件应该位 PI 寄存器定义的平台上的每个端口初始化 HPCP，MPSP 和 CPD。

在固件初始化上面提到的寄存器后，然后应该执行下面的步骤来完成交错启动过程（当可用 于平台）在每个实现的端口（PI 寄存器定义的）

1. 确保 P#CMD.ST=0，P#CMD.CR=0，P#CMD.FRE=0，P#CMD.FR=0，和 P#SCTL.DET=0。
2. 分配命令列表和 FIS 接收区域的存储空间。设置 P#CLB 和 P#CLBU 为分配命令列表的 物理地址。 设置 P#FB 和 P#FBU 为非配 FIS 接收区域的物理地址。 然后设置 P#CMD.PRE=1。
3. 通过设置 P#CMD.SUD=1 初始化连接到端口的 SATA 设备的启动。
4. 等待连接到端口的设备给出的正确提示（等待在位提示的最大时间定义在 SATA 规范 中）。通过查询 P#SSTS.DET 检查这个过程是否结束。当读 P#SSTS.DET 返回值为 1h 或者 3h 时，

固件应该继续下面的步骤，否则查询直到超时，然后移到下一个端口，从 步骤 1 开始对下一个端口的启动。

5. 通过向每个实现的位中写 1，清除 P#SERR 寄存器。
6. 等待 SATA 驱动准备好的提示。通过检查 P#TFD.STS 检查这个过程。当 P#TFD.STS.BSY，P#TFD.STS.DRQ 和 P#TFD.STS.ERR 是全 0，在 ATA/ATAPI-7 规范定义的的最大时间前，表明设备准备好。

4.4.1.2. 系统软件定义的初始化

软件可以在设置 GHC.HR 为 1 之前，执行 DWC_ahsata 全局复位。当固件（BIOS）已经分 配了存储空间，为命令列表和 FIS 接收区域初始化了适当的寄存器，软件可以忽略这个过程的步骤。

下面系统软件的步骤可以让 DWC_ahsata 进入最小初始化状态：

1. 通过读 PI 寄存器，决定 DWC_ahsata 实现了哪些端口。这个按位映射的值指导软件确 定有多少端口是可用的，哪些端口需要初始化。
2. 通过读和检查每个端口的 P#CMD 寄存器，确保 DWC_ahsata 不在运行状态。当 P#CMD.ST，P#CMD.CR，P#CMD.FRE 和 P#CMD.FR 都为 0 时，端口在空闲状态。否 则，端口不在空闲状态，应该在控制 DWC_ahsata 全局和端口定义寄存器前使端口进入 空闲状态。软件使端口进入空闲状态，通过清除 P#CMD.ST 寄存器，等待读 P#CMD.CR 时返回 0。这个过程软件需要等待至少 500ms。当 P#CMD。PRE 设置为 1 是，软件应 该清除该位为 0，然后等待 500ms 再读 P#CMD.FR 是否为 0。当P#CMD.CR 或 P#CMD.FR 没有正确清除为 0，软件应该使用端口复位或者全局复位来恢复。
3. 通过读 CAP.NCS 决定 HBA 支持多少命令槽。
4. 对每个实现的端口，系统软件应该分配存储空间并编程：
   - P#CLB 和 P#CLBU（当 CAPS64A 设置为 1）
   - P#FB 和 P#FBU（当 CAPS64A 设置为 1） 注意：对于系统软件有个好的习惯，分配的存储空间是清 0 的，并通过 P#CLB 和 P#FB指向这个区域。（It is good practice for system software to zero-out the memory allocated and referenced by P#CLB and P#FB.）在设置 P#FB 和 P#FBU 指向 FIS 接收区域的物理地址后，系统软件应该设置 P#CMD.FRE 为 1。
5. 对每个实现的端口，通过向每个实现的位写 1 清除 P#SERR 寄存器。 决定哪些事件应该产生中断，设置每个实现的端口的 P#IE 寄存器为适当的值。为了使 DWC_ahsata 产生中断，系统软件应该设置 GHC.IE 为 1。

   注意：对于多层结构的 DWC_ahsata 的中断结构，系统软件应该确保在编程 P#IE 和 GHC.IE 前清除 P#IS（首先清除它）和 IS.IPS（其次清除它）寄存器为 0。这样可以阻 止这些寄存器中任何残留的设置为 1 的位产生中断。

软件不能设置 P#CMD.ST 为 1，直到确定功能设备是在位，需要确认 P#TFD.STS.BSY，P#TFD.STS.DRQ，P#TFD.STS.ERR 的位都为0，P#SSTS.DET=3h。为了使P#TFD 寄存器能够根据端口初始化寄存器 FIS 进行更新，P#SERR.DIAG_X 位应该清除为 0。

4.4.2. 端口 DMA 的软件控制

4.4.2.1. 开始（P#CMD.ST）

当 P#CMD.ST 设置为 1，软件不允许执行下面的动作：

- 控制 P#CMD.POD 通过冷在位检测逻辑对设备上电或下电（当平台支持并且 DWC_ahsata 使能）；
- 控制 P#SCTL.DET 改变 PHY 的状态；

- 控制 P#CMD.SUD 启动设备（当平台支持）。 上面的动作只允许在端口不在运行状态时执行，运行状态需要确认 P#CMD.ST 和 P#CMD.CR为 0。

软件设置 P#CMD.ST 应该只能在下面条件变成有效之后：

- P#CMD.CR 确认清除为 0，P#CMD.FRE 已经设置为 1；
- 端口上功能设备在位（需要确认 P#TFD.STS.BSY=0， P#TFD.STS.DRQ=0，P#SSTS.DET=3h），P#CLB/P#CLBU 编程为有效值。

4.4.2.2. FIS 接收使能（P#CMD.FRE）

当 P#CMD.FRE 设置时（导致 P#CMD.FR 设置为 1），端口接收来自设备的 FIS，并把他们 拷贝到系统主存中。当 P#CMD.FRE 清除时（导致 P#CMD.FR 清除为 0），接收 FIS 被保留 在 RxFIFO 中，当 RxFIFO 满时，之后的 FIS 接收就被阻塞。

软件允许控制 P#CMD.FRE，这样可以移动 FIS 接收区域到新的位置。当这位清除为 0，软件必须先等待 P#CMD.FR 清除为 0，表明端口 DMA 引擎接收 FIS 在空闲条件下。当P#CMD.FR 和 P#CMD.FRE 都清除为0，软件可以更新P#FB 和 P#FBU 值。在设置P#CMD.FRE 为 1 前，软件应该确保 P#FB 和 P#FBU 设置为有效值。软件不能在 P#CMD.FRE 设置为 1 时写 P#FB 和 P#FBU。

软件应该在 P#CMD.ST 设置为 1 前设置 P#CMD.FRE 为 1。当 P#CMD.ST 或 P#CMD.CR 设 置为 1 时，软件不能清除 P#CMD.FRE 为 0。

在全局或端口复位时，P#CMD.FRE 位清除为 0。包含设备签名的 D2H 寄存器 FIS 被端口接收，签名域也被更新。

注意：当 DWC_ahsata 端口由于某个错误停止时（例如，P#IS.IFS 设置为 1），FIS 不会被 传递直到 P#CMD.ST 清除为 0 以从错误中恢复。

4.5. 交直流电气特征

SATA 控制逻辑相关接口使用标准 LVCMOS 1.8V 供电的接口缓冲，其电气特性参见 0 节。

SATA PHY 相关的电地接口电气特性如下表 4-3 所示：（其中 vph 为 1.8v）

表 4-3 SATA PHY 电地接口电气特性

Power Supply	Signal Name	Description	Value
Analog power supplies	vp vptx	PHY analog and digital supply PHY transmit supply	0.9 V (+ 10%, – 7%) at the macro pins with respect to gd 0.9 V (+ 10%, – 7%) at the macro pins with respect to gd
High Voltage IO Supply	vph	High Voltage Power Supply	**When using the optional Voltage Regulator,** 3.3V (+ 10%, - 7%) at the macro pins with respect to gd **When not using the option Voltage Regulator,** 1.8V (+ 10%, – 7%) at the macro pins with respect to gd
PHY Ground	gd	Analog and Digital ground	0 V
CORE Ground	vss	CORE vss for Back-Back ESD diode connection	0 V

Mode		Current from vp 0.9V (in mA)	Current from vptx 0.9V (in mA)	Current from vph 1.8V (in mA)	Total Current (in mA)	Total Power Consumption (in mW)
Ready	6G Typ	27.4	12.0	23.0	62.4	76.8
	6G W.C.	62.1	19.0	30.2	111.3	127.4
	3G Typ	17.5	12.0	13.4	42.9	50.7
	3G W.C.	39.5	19.0	17.8	76.3	84.7
	1.5G Typ	12.2	12.0	8.6	32.8	37.3
	1.5G W.C.	26.9	19.0	15.0	60.9	68.4
Partial	Typical	8.9	2.4	11.6	22.9	31.0
	W.C.	38.1	5.9	15.0	58.9	66.5
Slumber	Typical	2.69	0.56	0.88	4.13	4.51
	W.C.	36.83	2.69	1.03	40.55	37.43
Power Down	Typical	1.814	0.555	0.046	2.415	2.216
	W.C.	35.107	2.691	0.111	37.909	34.217

5 USB 接口

5.1. USB 接口概述

USB 接口主要采用 Synopsys USB3.0 host 控制器 IP 和 3.0PHY IP 实现，对控制器 IP 进 行修改，实现 USB 接口的安全管理。USB 接口遵循 USB3.0 规范和 xHCI 设计规范，主要有 以下外接口：

- 128bit 数据宽度、32bit 地址宽度的 AXI 接口作为 Master 接口，用于 DMA 访问；
- 32bit 数据宽度、32bit 地址宽度的 AHB 接口作为 Slave 接口，用于 CSR 访问和 RAM Debug；
- RAM 接口，采用 3 个两端口 RAM，实现描述符 cache、TxFIFOs、RxFIFOs；
- 片内实现 6 个 3.0 端口。

5.2. 管脚说明

见本文档 1.3.13 节。

5.3. 初始化与测试

5.3.1. XHC 初始化

系统 boot 起来后，软件会先枚举到控制器，为 xHC 的寄存器分配一个基址，为 FLADJ 寄 存器设置一个系统规定的值。

然后进行以下一系列操作，实现控制器的初始化：

1、 初始化系统 IO 映射；

2、 芯片硬复位之后，等待 USBSTS 寄存器中 CNR（Controller Not Ready）置 1； 3、配 CONFIG 寄存器中 MaxSlotsEn 域，指明软件将可使用的最大 Slot 号；

4、 为 DCBAAP 寄存器配一个 64 位的地址，指向 Device Context 基址阵列的所在；

5、 为 Command Ring 控制寄存器配一个 64 位的地址，这是 Command Ring 中第一个 TRB 的起始地址；

6、 初始化中断：

a) 分配、初始化 MSI-X 消息表，设置消息地址和消息数据，使能向量。至少，表向 量条目 0 要被初始化、使能；

b) 分配、初始化 MSI-X Pending Bit Array；

c) 在 MSI-X 能力结构体中分别为消息控制表和 Pending Bit Array 指定 Offset； d) 初始化 MSI-X 能力结构体中的消息控制寄存器；

e) 初始化每一个活跃的中断源：

i. 定义 Event Ring：

1. 分配并初始化 Event Ring Segment(s)；
2. 分配 ERST（Event Ring Segment Table），初始化 ERST 表的各个条目， 指向并定义各个 Event Ring Segment 的大小；
3. 根据 ERST 中定义的大小，配置中断源 ERSTSZ（Event Ring Segment Table Size）寄存器；
4. 根据 ERST 中定义的起始地址，配置中断源 ERDP（Event Ring Dequeque Pointer）寄存器；
5. 用一个 64 位的地址指针指向 ERST，即配置中断源 ERSTBA（Event RingSegment Table Base Address）寄存器；写 ERSTBA 寄存器，则使能了 Event Ring；

ii. 定义中断：

1. MSI-X 能力结构体消息控制寄存器中的 MSI-X 使能标志设置为 1，使能 MSI-X 中断机制；
2. 根据目标中断调节率，初始化中断调节寄存器的 Interval 域；
3. 置 USBCMD 寄存器的 Interrupt 使能位 INTE 为 1，使能系统总线中断使 能；
4. 置中断管理寄存器的 Interrupt 使能位 IE 为 1，使能中断源；

7、 置 USBCMD 寄存器的 Run/Stop 位为 1，相当于打开控制器，表明控制器可以开始接收 按门铃了。

至此，控制器已经开始工作，Root Hub 端口开始报告设备的连接情况了，系统软件可以开 始枚举设备。

USB2.0 设备需要端口复位过程，使端口进入 Enabled 状态，一旦端口进入 Enabled 状态，端口上就会有有效的 SOF 了，但是 pipe 调度还没有使能；
SS 端口一旦检测到设备连接，可以直接进入 Enabled 状态；

5.3.2. 设备初始化

芯片硬件复位（HCRST, or commanded to the PLS = RxDetect state）之后，所有的 Root Hub 端口将处于 Disconnected 状态，端口处于上电状态（PP=1），等待设备连接。

当一个设备连接到处于 Disconnected 状态时：
3.0 端口将：
进入 Polling 状态
如果 Polling 成功，端口进入 Enable 状态，CCS 和 CSC 被置 1； 如果 Polling 不成功，端口进入 Disconnected 状态；
2.0 端口将：
进入 Disabled 状态，CCS 和 CSC 被置 1；

下述为一个典型的设备初始化过程：

1、 当控制器监测到一个设备连接时，它将置 CCS 和 CSC 为 1。如果 CSC 的置 1，导致 PSCEG(Port Status Change Event Generation)有一个从 0 到 1 的变化，则，控制器会生成 一个 Port Status Change Event；

2、 一旦收到 Port Status Change Event，系统软件分析 PORT ID，确定是哪个端口给出的这 个事件；

3、 然后软件来读这个端口的 PORTSC 寄存器，如果 CSC=1 且 CCS=1，表明这个 Event 的产生是由于设备连接；如果 CSC=1 且 CCS=0，表明这个 Event 的产生是由于设备移除； 现在讨论设备连接：

a) 3.0 端口将自动向 Enabled 状态转；
如果成功，端口进入 Enabled 状态，Port Enabled/Disabled（PED）位将置 1，Port Rest(PR)和 Port Link State(PLS)域将被清 0。设备将处于 Default 状态； 如果不成功，端口进入 Disconnected 状态，PED 和 PR 位将置 0，PLS 域将被置成 RxDetect (_5‘)。设备将仍处于 Powered 状态；

b) 2.0 端口需要软件来启动端口复位，促使端口向 Enabled 状态跳转、设备向 Default 状态跳转。
设备连接后，PED 和 PR 都是 0，PLS 域为 7(polling)。软件置 PR 为 1，启动端口 复位，等待 Port Status Change Event。
复位完成后，PRC 和 PED 将被置 1，PR 将被清 0，PLS 为 0(U0)。如果 PRC 的置 1，导致 PSCEG 有一个从 0 到 1 的变化，则，控制器会生成一个 Port Status Change Event。
复位操作使 2.0 设备进入 Default 状态，准备接收 SET_ADDRESS 请求。

4、 端口成功进入 Enabled 状态后，软件通过 Enable Slot 命令，为新连接上的设备得到一个 Device Slot；

5、 成功获得 Device Slot 之后，软件为其初始化各种数据结构；

6、 Solt 对应的各种数据结构都初始化好了之后，软件用一个 Address Device 命令为 Device 分配一个地址，使能设备的默认的控制端点；

7、 对于 LS、HS、SS 设备，对默认的控制端点的访问包有唯一允许的大小，分别为 8B、64B、512B，下面的步骤 a 可能跳过；
对于 FS 设备，软件通过发 GET_DESCRIPTOR 请求，先读设备描述符的前 8 个字节， 获得 bMaxPacketSize0 域，来确定对默认的控制端点的访问包的实际 Max Packet Size， 更新到 Default Control Endpoint Context

a) GET_DESCRIPTOR 请求需要一个 Setup Stage TD、一个 Data Stage TD、一个 Status Stage TD。软件将：

i. 分配一个 8 个字节的 buffer 准备接收 Device Descriptor；

ii. 在 Endpoint 0 的 Transfer Ring 上初始化 Setup Stage TD（单个的 Setup TRB）

1. TRB Type = Setup Stage TRB；
2. TRB Transfer Length = 8；8B 的 Setup 数据
3. IOC =0；
4. IDT =1；
5. bmRequestType = 0x80(Dir = Device-to-Host, Type = Standard, Recipient = Device)；

6. bRequest = 6 (GET_DESCRIPTOR);
7. wValue = 0100h. Low byte = 0 (Descriptor Index), High Byte = 1 (Descriptor type)
8. wIndex = 0
9. wLength = 8.
10. Cycle bit = Current Producer Cycle State

iii. 调整（Advance）Endpoint 0 的 Transfer Ring 的 Enqueue Pointer;

iv. 在 Endpoint 0 的 Transfer Ring 初始化 Data Stage TD（单个的 Data Stage TRB）

1. TRB Type = Data Stage TRB
2. Direction (DIR) = ‗1'.
3. TRB Transfer Length = 8，8B 的 Descriptor 数据
4. Chain bit (CH) = 0
5. Interrupt On Completion (IOC) = 0
6. Immediate Data (IDT) = 0
7. Data Buffer Pointer = The address of the Device Descriptor receive buffer.
8. Cycle bit = Current Producer Cycle State.

v. 调整（Advance）Endpoint 0 的 Transfer Ring 的 Enqueue Pointer;

vi. 在 Endpoint 0 的 Transfer Ring 初始化 Status Stage TD（单个的 Status Stage TRB）

1. TRB Type = Status Stage TRB
2. Direction (DIR) = ‗0'.
3. TRB Transfer Length = 0.
4. Chain bit (CH) = 0.
5. Interrupt On Completion (IOC) = 1.
6. Immediate Data (IDT) = 0.
7. Data Buffer Pointer = 0.
8. Cycle bit = Current Producer Cycle State.

vii. 调整（Advance）Endpoint 0 的 Transfer Ring 的 Enqueue Pointer;

viii. 摁该 Device Slot 对应的门铃，DB Target = Control EP 0 Enqueue Pointer Update.

ix. 当 GET_DESCRIPTOR Status Stage TRB 对应的传输完成 Event 返回时，软件将用 Device Descriptor buffer 中的 wMaxPacketSize 值来更新 Endpoint 0 Context 里的 Max Packet Size;

x. 软件接着会发出一个 Evaluate Context Command，V1 置 1，用来通知 xHC，Default Control endpoint 的 Max Packet Size 这个参数发生了改变。xHC 执行完 这个命令后，后续的经过 Default Control endpoint 的传输，将都使用这个新的 Max Packet Size。

8、 现在默认的控制端点（Default Control endpoint）已经完全可以进行所有的操作了。软件 可以读完整的设备描述符、配置描述符等，将设备交给对应的类驱动接管。通过 Default Control endpoint 发送 GET_DESCRIPTOR 请求即可。

9、 读完配置描述符，软件会发一个 Evaluate Context Command，V0 置 1，用来通知 xHC，Hub and Max Exit Latency 这些参数。Output Slot Context 的 Interrupter Target 域，也可以 通过这个命令来修改。

10、 现在，类驱动可以用 Configure Endpoint Command 来配置 Device Slot，以及通过 SET_CONFIGURATION 来配置设备本身（经 Default Control Endpoint）。这两个动作都 成功完成了，则设备状态将从 Address 跳到 Configured，而 xHC Device Slot 的状态将从 Addressed 跳到 Configured。

11、 如果需要的话，软件还要配置 Alternate Interface；

12、 现在个设备的 pipe interface 已经可以完成正常使用了。

注意，软件为了保证行为的正确性，从 Root hub 开始，会对 USB 拓扑中的每一层 hub 和 TTs（Transaction Translator）都做初始化之后，才开始做再下一层的设备初始化。

5.3.3. 边界扫描（BSR）测试

6 个 USB3.0 PHY 的边界扫描链穿在一起；

USB 3.0 PHY 内嵌的边界扫描基本单元下图所示。

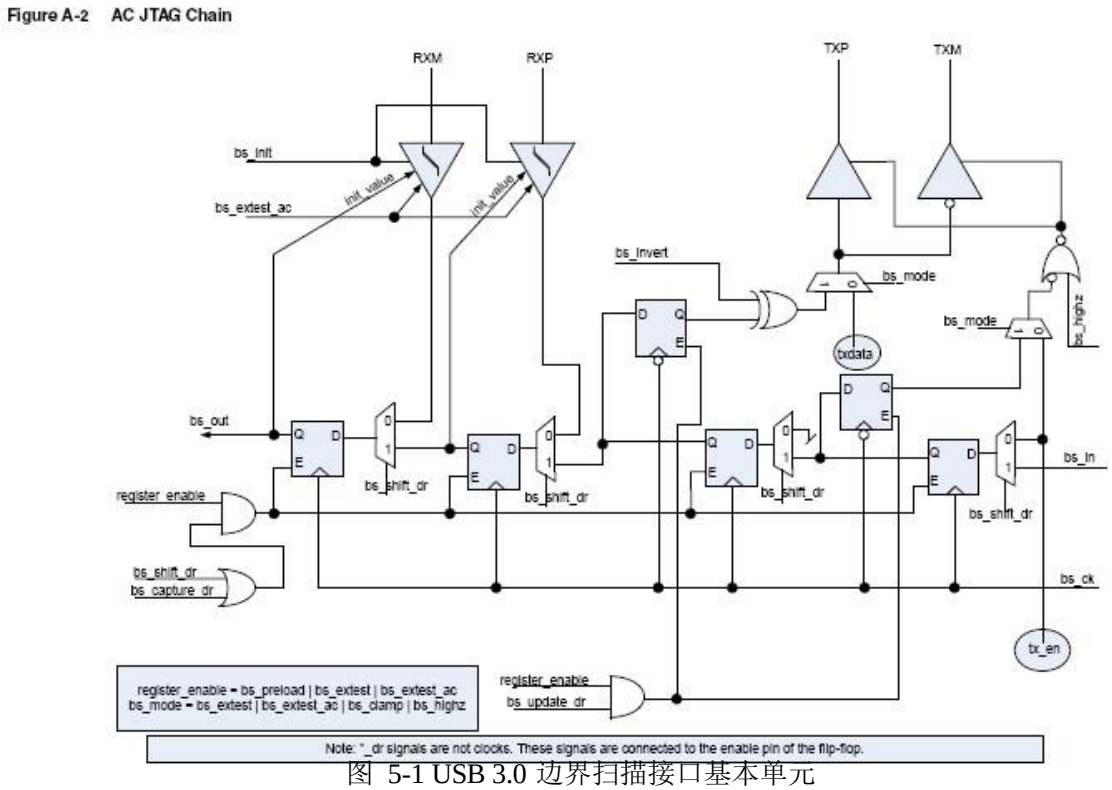


图 5-1 USB 3.0 边界扫描接口基本单元

TBOX 按照根据下列状态和指令（表 8-1）按照图 8-1 生成 USB 3.0 PHY 所需的接口信 号。此外，其它信号需要按表 8-2 进行设置（其中 scan_mode、scan_shift、ACJTLEVEL[4:0]由 TBOX 给出，其它信号要考虑实现方式。ACJTLEVEL 用于选择信号电平，详见表 8-3）。如果不支持边界扫描，则 TBOX 给 USB 的相关信号（表 3-3）设置为零。

表 5-1 JTAG TAP 控制器和 JTAG 指令

Table A-1 JTAG TAP Controller States and JTAG Instructions

JTAG TAP Controller States	JTAG Decoded Instructions	JTAG Input Pins
ShiftDR	sample	tms
CaptureDR	preload	tdi
UpdateDR	extest	tck
PauseDR	etest_train	
SelectDR	extest_pulse	
RunTestIdle	clamp	
	highz	
	intest	
	runbist	

表 5-2 边界扫描时其它信号设置

Table A-2 Additional Signal Settings Required to Control Boundary Scan

Signal Name	Value
cr_write	0
jtag_tms	0
phy_reset	Apply high pulse (POR).
test_burnin	0
TESTBYPASS	0
test_powerdown_ssp	0
test_powerdown_hsp	0
scan_mode	0
scan_shift	0
RX0RESET	0: Translates to top-level signal pipeP_reset_n = 1.
TX0BEACONEN	0: Translates to top-level signals pipeP_tx_elecidle = 1 and pipeP_tx_detectrx = 0.
lane0_tx2rx_loopback	0
RX0LOSEN	1
ACJTLEVEL[4:0]	The signal level for detecting the AC JTAG pulse must be selected based on Table A-3 on page 228.

表 5-3 ACJT_LEVEL 设置

Table A-3 acjt_level[4:0] Settings for Signal Levels to Detect AC JTAG Pulse

Differential Input Signal Swing (mVpp)		
Above	Below	acjt_level[4:0] to Center, 50% and 90% Trip Levels
235 mV	375 mV	00001
288 mV	463 mV	00010
344 mV	554 mV	00011
401 mV	649 mV	00100
460 mV	747 mV	00101
521 mV	849 mV	00110
432 mV	740 mV	10001
525 mV	> 967 mV	10010

具体步骤：

1） 写指令寄存器（CFGTEST 指令）；

2） 配置 CFG_REG 寄存器选择 USB 的 BSR 链；

3） 配置指令寄存器是 EXTEST 或者 SAMPLE/PRELOD 指令；

4） 启动数据移位。

5.3.4. ATPG（SCAN 链）测试

每个 USB3.0 PHY 有 1 条负沿链、40 条正沿链，把 6 个 PHY 的负沿链穿在一起，把每 个 PHY 自己的 40 条正沿链连在一起；最终 USB3 是 1 条负沿链、6 条正沿链；

Figure C-1 USB 3.0 PHY Scan Chain Implementation

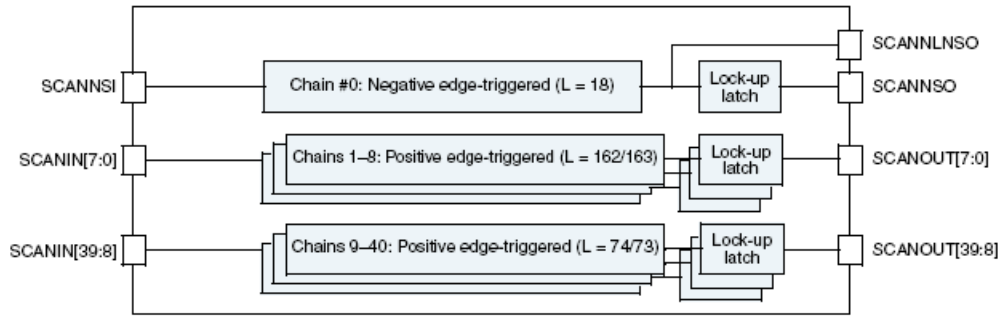


图 5-2 USB 3.0 PHY SCAN 链

Forty scan chains are triggered by the positive edge of SCANCLK, and one scan chain is triggered by the negative edge of SCANCLK:

- 3 positive-edge-triggered scan chains: length = 162; with terminal lock-up latch
- 5 positive-edge-triggered scan chains: length = 163; with terminal lock-up latch
- 2 positive-edge-triggered scan chains: length = 74; with terminal lock-up latch
- 30 positive-edge-triggered scan chains: length = 73; with terminal lock-up latch
- 1 negative-edge-triggered scan chain: length = 18; with 2 scan outputs, one with terminal lock-up, SCANNSO, one without terminal lock-up, SCANNLNSO

图 5-3 USB 3.0 PHY SCAN 链长

Figure C-3 Final Scan-Stitched Output of ASIC

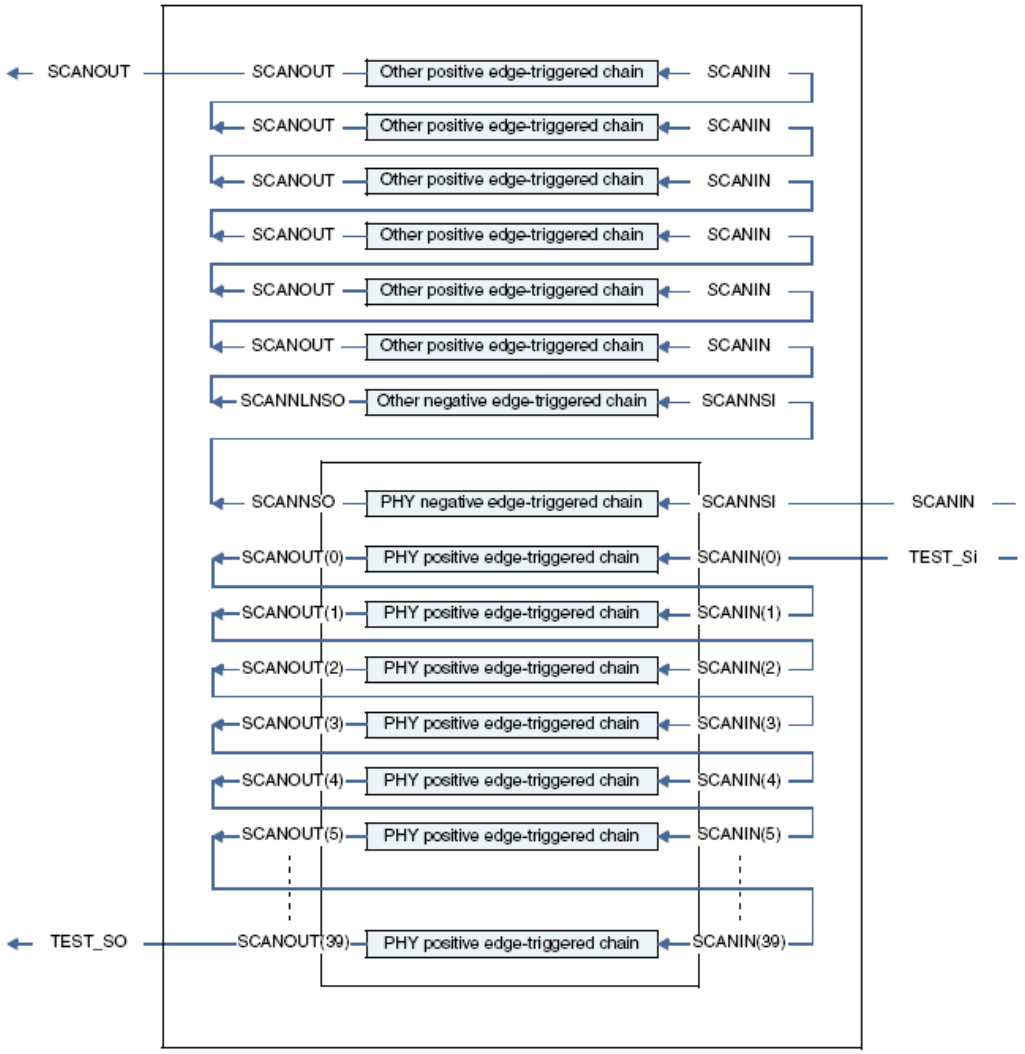


图 5-4 USB 3.0 PHY SCAN 链连接关系

5.3.5. JTAG 测试

1） 控制芯片引脚 JTAG_SEL，选择 USB 3.0 PHY 内嵌的 TAP 控制器。
2） 通过 JTAG 接口进行测试。

USB 3.0 PHY 的 TAP 控制器支持 18 位的 CRSEL 寄存器，通过该寄存器可以访问内部 控制寄存器。

表 5-4 CRSEL 寄存器的命令域

Table B-1 CRSEL Register Command

CMD	Encoding	Description
ADDR	2'b00	Captures ADDRESS field for use in a subsequent WRITE command. Shifts current value from most recently accessed register (via READ or WRITE) out on TDO.
WRITE	2'b01	Writes the value in the DATA field to the register selected by the address captured during the most recent ADDR or READ command. Shifts current value from most recently accessed register (via READ or WRITE) out on TDO.
DATA	2'b10	Shifts current value from most recently accessed register (via READ or WRITE) out on TDO. DATA value shifted in is not used.
READ	2'b11	Performs a read operation to the register selected by the ADDRESS field to make its current value available for subsequent CRSEL accesses. Shifts current value from most recently accessed register (via READ or WRITE) out on TDO.

通过对 18 位的 CRSEL 寄存器进行移位（18 位的移位数据中，16 位为数据或地址，2 位为操作码 CMD）可以完成四个操作：

1） 地址（CMD=2‘b00）：移入访问地址；
2） 写操作（CMD=2‘b01）：移入数据并进行写操作。
3） 数据（CMD=2‘b10）：移出读数据，移入数据不使用。
4） 读操作（CMD=2‘b11）：移入地址并进行读操作。

读寄存器过程：

1） 对 CRSEL 寄存器移入一个读操作命令（CMD=2‘b11），同时指定目标地址；
2） 对 CRSEL 寄存器的任何操作（CMD 码任意）都可以把读数据移出。

写寄存器过程：

1） 对 CRSEL 寄存器移入一个地址命令（CMD=2‘b00）指定目标地址，或者对 CRSEL 寄 存器移入一个读操作命令（CMD=2‘b11），指定目标地址；
2） 对 CRSEL 寄存器移入一个写操作命令（CMD=2‘b01），同时指定写数据。

5.4. 交直流电气特性

见本文档 12.1 节。

6 存储器接口

6.1. 存储器接口概述

ICH2 集成了一个存储控制器，用于访问显存，存控典型工作频率是 333MHz。该控制 器接收来自 4 个 AXI 接口和一个 AHB 接口的访存请求，并通过一个位宽可配（64/32 位） 的访存接口访问显存。该存控支持单 RANK 的应用。为了提升系统可靠性、缩短印制板布 线长度以及增加对高频高负载的适应性，存储器接口也支持地址镜像和 1T/2T 传输机制。

ICH2 的存储控制器有如下特色：

- 支持 DDR3-800/1067/1333 的各种 DDR3 SDRAM 类型
- 存储器容量为：单 Rank 支持 2GB/4GB
- 支持颗粒、UDIMM 单 Rank
- 对于颗粒和 UDIMM 支持 2T 模式
- 支持 DDR3 标准定义的各种延迟参数动态可调
- 支持带 ECC/不带 ECC 的 DDR3 SDRAM 商用存储器条
- 支持 DDR3 SDRAM 的浅睡眠（Precharge PowerDown）和深睡眠（Self Refresh），用于降低功耗
- 支持×8 和×16 的存储器颗粒类型
- 支持对 DDR3 SDRAM 中的模式寄存器 MR0～3 实时配置
- 仅支持突发长度为 8（BL8）的访存操作
- 仅支持交错的 Wrap 地址
- 支持 Write Leveling 功能
- 支持 MPR 访问功能
- 支持读写通路的 DQS/DQ 延迟调节功能
- 支持 LoopBack 测试
- 支持存储控制器的关闭功能

6.2. 详细说明

6.2.1. 名词解释

- **RANK：**在应用中为了提高访存容量而不大量增加芯片访存管脚数量，一般把多个存储 器共用数据和地址线，而独立使用片选(CS)、时钟使能(CKE)和 ODT 信号。具体如图 6-1 所示：

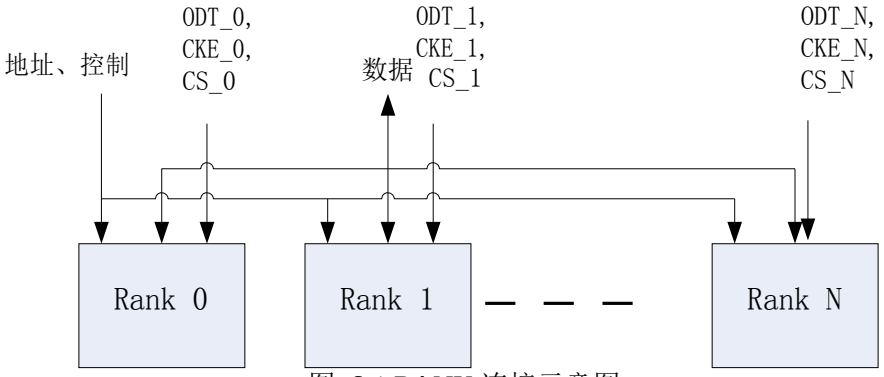


图 6-1 RANK 连接示意图

- **DIMM(Dual In-line Memory Module)：**目前的商用存储器条一般都使用 DIMM 方式来 实现，如图 6-2 和图 6-3 所示。一般而言，DIMM 可分为 UDIMM(Unbuffered DIMM) 和 RDIMM (Registered DIMM)。其中 RDIMM 相比 UDIMM 增加了一个寄存时钟驱动 和片选模块。

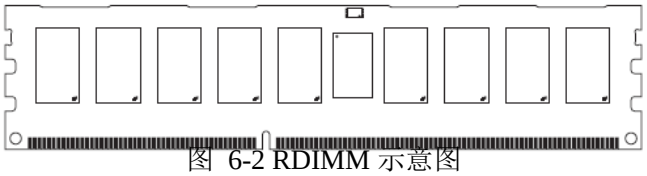


图 6-2 RDIMM 示意图

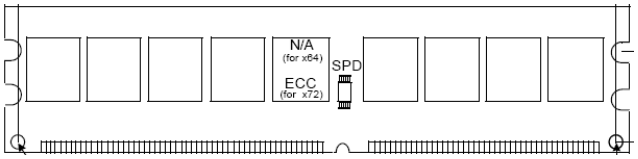


图 6-3 UDIMM 示意图 该模块锁存命令与地址输入信号，通过 PLL 恢复并重新驱动时钟信号、命令和地址信号以提高信号完整性。通常在 DIMM 上的双 RANK 是贴装在内存条的两面。

另外标准 DIMM 上集成 SPD(Serial Presence-detect) EEPROM，用来记录与 DIMM 相关 的各种硬件信息；而有些支持 ECC 存储的 DIMM 上还有温度传感器(Themal Sensor)。这些 功能为系统设计提供了更为直观可靠的信息。

- **1T/2T 传输：**在高负载以及高频工作时，为了提高工作的可靠性有时需要把存储器访问 的地址与命令加长为两个周期（片选信号维持一个时钟周期），以增加命令和地址的建 立时间。具体如图 6-4 和图 6-5 所示：

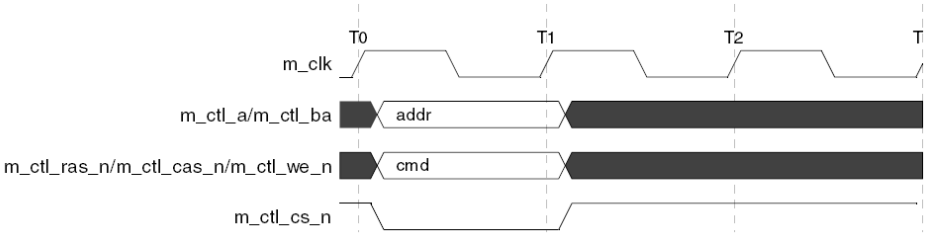


图 6-4 普通 1T 传输示意图

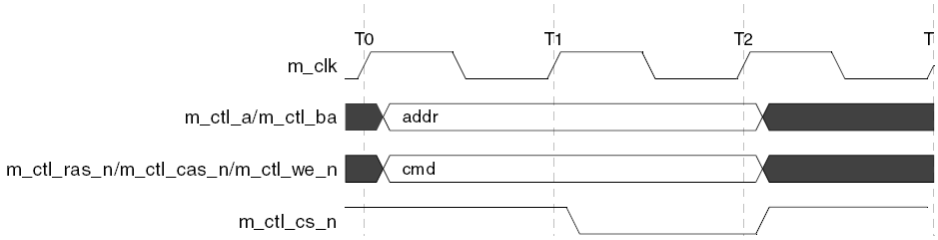


图 6-5 2T 传输示意图

2T 传输虽然提高了信号完整性，但是却降低了访存命令的发送频率，在一定程度上会 降低访存的性能。

6.2.2. 信号管脚

见本文档 1.3.10 节。

6.3. 使用方式

6.3.1. 地址映射

SDRAM 地址有行地址，列地址，Bank 地址。在进行地址换算时，需要将上层看到的 完整的访存地址映射到颗粒命令的地址。对于 DDR3，bank 地址固定为 3 位位宽。另外，可能用到的颗粒类型如下表 6-2 所示：

表 6-1 存储颗粒类型与地址映射关系

SDRAM Type			Address Width			
容量	位宽	模式	Bank 宽	行地址宽度	列地址宽度	总计
1Gb	x8	DDR2/3	3	14	10	27
2Gb	x8	DDR2/3	3	15	10	28
	x16	DDR2/3	3	14	10	27
4Gb	x8	DDR2/3	3	16	10	29
	x16	DDR2/3	3	15	10	28

显存位宽固定为 64 位，采用 1Gb x8 或者 2Gb x16 颗粒时，显存大小为 1GB；采用 2Gb x8 或者 4Gb x16 颗粒时，显存大小为 2GB；采用 4Gb x8 颗粒时，显存大小为 4GB。以 1Gb x8 颗粒来说明地址映射关系：

采用 1Gb x8 颗粒时，显存大小为 1GB，需要用到访存地址 30 位：addr[29:0]。颗粒的 行地址 14 位（row[13:0]），列地址 10 位（col[9:0]），bank 地址 3 位（bank[2:0]）。

uMCTL 只支持 BL8 的访存，一个 BL8 为 64 字节，因此，上层地址在 uMCTL 会转换 成 64 字节对界的地址，即地址的低 6 位被忽略。对于不对界的写请求，通过掩码来指明不 需要写入的数据字节；对于不对界的读请求，由颗粒读出数据后，再由存控接口逻辑进行处 理。

在本设计中，只支持单 rank 模式，有两种地址映射模式：{bank，row，col}，{row，bank，

col}，地址换算如下：

{bank，row，col}模式： col[9:0]={addr[12:5]，2'b00}
row[13:0]=addr[26:13]
bank[2:0]=addr[29:27]

{row，bank，col}模式： col[9:0]={addr[12:5]，
2'b00} bank[2:0]=addr[15:13]
row[13:0]=addr[29:16]

由于 col[2:0]的低两位为 0，对于读请求，当 col[2]为 0 时，颗粒出数顺序是 0，1，2，3，4，5，6，7；当 col[2]为 1 时，颗粒出数顺序是 4，5，6，7，0，1，2，3。针对这种出数顺序不同的问题，uMCTL 内部已经进行处理，用户不需要关心。

6.4. 初始化与测试

6.4.1. 初始化

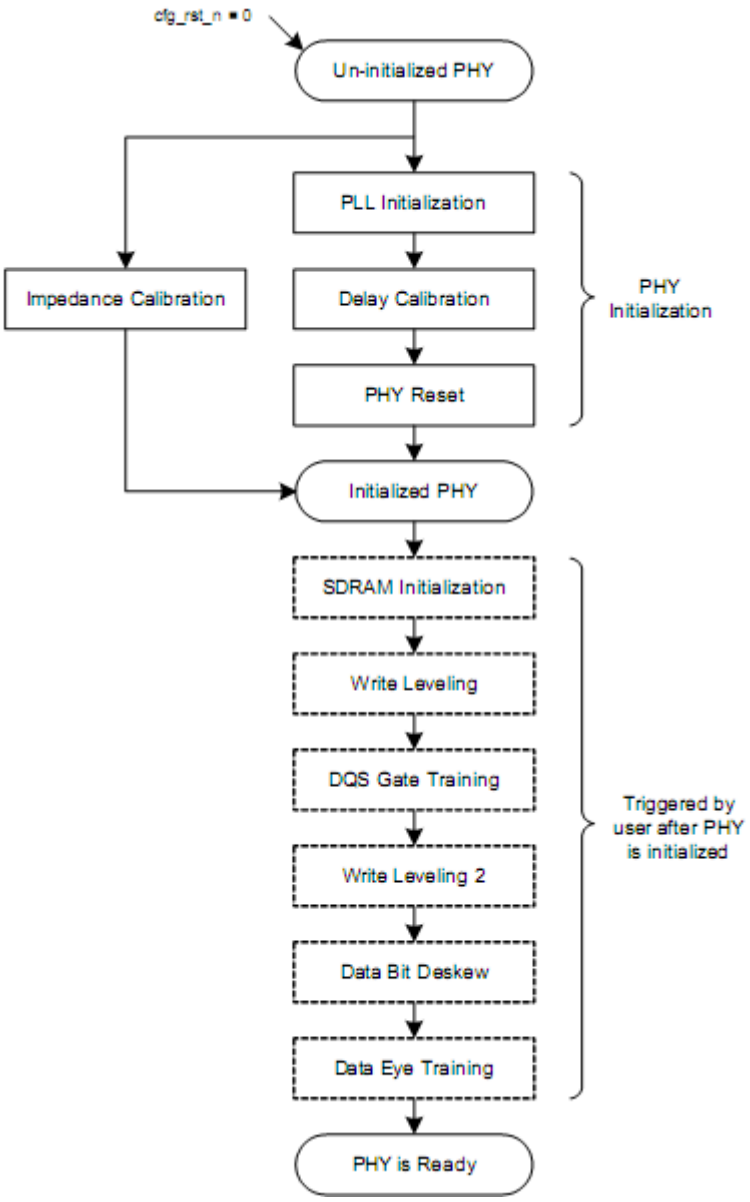


图 6-6 PHY 初始化流程

PHY 的初始化流程大致可以分为两个步骤，第一步是复位时自动完成的，包括：

1、 配置复位 cfg_rst_n 有效期间，PHY 处于非初始化状态；
2、 复位撤销，PHY 进入 PLL 初始化阶段，该阶段可以通过向 PIR[LOCKBYP]写 1 来旁路；
3、 PLL 初始化的同时，进行阻抗校准，该阶段可以通过向 PIR[ZCALBYP]写 1 来旁路；
4、 PLL 初始化完成后，开始延迟线校准，该阶段可以通过向 PIR[DCALBYP]写 1 来旁路；
5、 PLL 锁定且延迟线校准后，时钟不会立刻稳定，所以还要再等一些可配的时钟周期；
6、 当 PLL 初始化、阻抗校准、延迟线校准都完成了，表明 PHY 已经做完初始化了。

PHY Initialization Timing

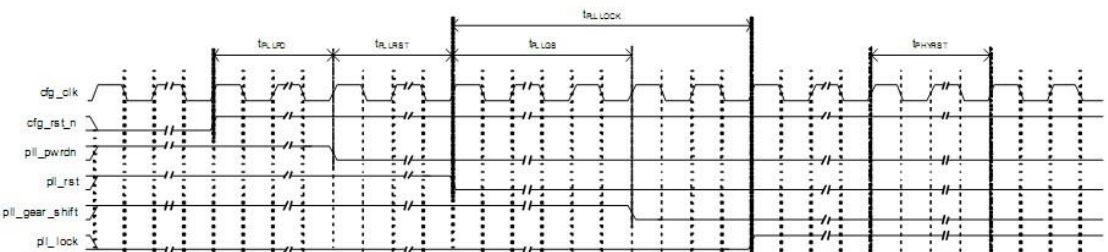


图 6-7 PHY 初始化时序 初始化流程的第二部分，需要用户或

控制器触发：

1、 启动 SDRAM 的初始化化。在触发 SDRAM 初始化之前，要配 PUB 里面的一些时序寄 存器和 DRAM 模式寄存器；

2、 SDRAM 初始化后，触发 PUB 做 write Leveling； 3、

触发 PUB 做 DQS gate training；

4、 触发 PUB 做读写数据比特 Deskew；

5、 至此可以通过 PHY 对 SDRAM 做读写访问了。

The initialization steps for DDR3 SDRAMs are as follows:

1. Optionally maintain RESET# low for a minimum of either 200 us (power-up initialization) or 100ns (power-on initialization). The PUB drives RESET# low from the beginning of reset assertion and therefore this step may be skipped when DRAM initialization is triggered if enough time may already have expired to satisfy the RESET# low time.
2. After RESET# is de-asserted, wait a minimum of 500 us with CKE low.
3. Apply NOP and drive CKE high.
4. Wait a minimum of tXPR.
5. Issue a load Mode Register 2 (MR2) command.
6. Issue a load Mode Register 3 (MR3) command.
7. Issue a load Mode Register 1 (MR1) command (to set parameters and enable DLL).
8. Issue a load Mode Register 0 (MR0) command to set parameters and reset DLL.
9. Issue ZQ calibration command.
10. Wait 512 SDRAM clock cycles for the DLL to lock (tDLLK) and ZQ calibration (tZQinit) to finish. This wait time is relative to Step 8, i.e. relative to when the DLL reset command was issued onto the SDRAM command bus.

DDR3 初始化步骤：

1、 如果是上电初始化，维持 RESET#为低至少 200us；如果是在电初始化，维持 RESET# 为低至少 100ns；从 reset 有效开始，PUB 就驱动 RESET#为低，所以可能不需要特别做 这一步；

2、 RESET#撤销后，维持 CKE 为低，持续 500us；

3、 驱动 CKE 为高，发 NOP；

4、 等待至少 tXPR；5、 发

load MR2 命令； 6、 发 load

MR3 命令；

7、 发 load MR1 命令（设置参数并使能 DLL）；

8、 发 load MR0 命令（设置参数并复位 DLL）；

9、 发 ZQ 校准命令；

10、 等待 512 个 SDRAM 时钟周期，等 DLL 锁定(tDLLK)和 ZQ 校准完成(tZQinit)。

6.4.2. 测试

存储控制器接口测试具体包括 Loopback 测试与 Data Training 相关的测试。

LoopBack 测试主要测试 Synopsys DDR3PHY 的内部读写数据通路。此时，存储控制器 发向 DDR3PHY 的写数据会自动绕回来，然后对绕回来的数据进行分析，可以检查读写通 路是否有问题。

Data Training 测试是对存储控制器的数据总线按照 8 位为粒度与 DQS 进行校准、字节 对齐的过程，这项操作是初始化的一部分。通过该测试可以提高输出传输的信号完整性，提 高抗干扰能力。

6.5. 交直流电气特性

见 12.1 节。

7 显示 DVO 接口

7.1. DVO 接口概述

ICH2 采用双 DVO（Digital Video Ouput）接口支持显示输出。其 DVO 接口规格为 24bit 位，采用 SDR 源同步传输，不支持 DDR 双边沿传输。通过与外部显示发送器芯片连接，DVO 接口可以转换成 LVDS、VGA、DVI 和 HDMI 等标准显示接口，进而与显示设备连接。

7.2. 信号管脚说明

ICH2 双 DVO 接口信号管脚列表如下：

表 7-1 DC 接口信号说明

管脚名称	位	方	含义	备注
DVO 接口 0				
DC0_Clk	1	输	通道 0 源同步像素时钟	最高频率
DC0_En	1	输	通道 0 使能	
DC0_Hsync	1	输	通道 0 水平同步	
DC0_Vsync	1	输	通道 0 场同步	
DC0PAB_IN	1	输	通道 0 外部中断	
DC0_Red_0	1	输	通道 0 像素红色分量第 0	
DC0_Red_1	1	输	通道 0 像素红色分量第 1	
DC0_Red_2	1	输	通道 0 像素红色分量第 2	
DC0_Red_3	1	输	通道 0 像素红色分量第 3	
DC0_Red_4	1	输	通道 0 像素红色分量第 4	
DC0_Red_5	1	输	通道 0 像素红色分量第 5	
DC0_Red_6	1	输	通道 0 像素红色分量第 6	
DC0_Red_7	1	输	通道 0 像素红色分量第 7	
DC0_Green_	1	输	通道 0 像素绿色分量第 0	
DC0_Green_	1	输	通道 0 像素绿色分量第 1	最高翻转率
DC0_Green_	1	输	通道 0 像素绿色分量第 2	90MHz
DC0_Green_	1	输	通道 0 像素绿色分量第 3	
DC0_Green_	1	输	通道 0 像素绿色分量第 4	
DC0_Green_	1	输	通道 0 像素绿色分量第 5	
DC0_Green_	1	输	通道 0 像素绿色分量第 6	
DC0_Green_	1	输	通道 0 像素绿色分量第 7	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 0	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 1	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 2	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 3	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 4	

DC0_Blue_	1	输	通道 0 像素蓝色分量第 5	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 6	
DC0_Blue_	1	输	通道 0 像素蓝色分量第 7	
DVO 接口 1				
DC1_Clk	1	输	通道 1 源同步像素时钟	最高频率
DC1_En	1	输	通道 1 使能	最高翻转率 90MHz
DC1_Hsync	1	输	通道 1 水平同步	
DC1_Vsync	1	输	通道 1 场同步	
DC1PAB_IN	1	输	通道 1 外部中断	
DC1_Red_0	1	输	通道 1 像素红色分量第 0	
DC1_Red_1	1	输	通道 1 像素红色分量第 1	
DC1_Red_2	1	输	通道 1 像素红色分量第 2	
DC1_Red_3	1	输	通道 1 像素红色分量第 3	
DC1_Red_4	1	输	通道 1 像素红色分量第 4	
DC1_Red_5	1	输	通道 1 像素红色分量第 5	
DC1_Red_6	1	输	通道 1 像素红色分量第 6	
DC1_Red_7	1	输	通道 1 像素红色分量第 7	
DC1_Green_	1	输	通道 1 像素绿色分量第 0	
DC1_Green_	1	输	通道 1 像素绿色分量第 1	
DC1_Green_	1	输	通道 1 像素绿色分量第 2	
DC1_Green_	1	输	通道 1 像素绿色分量第 3	
DC1_Green_	1	输	通道 1 像素绿色分量第 4	
DC1_Green_	1	输	通道 1 像素绿色分量第 5	
DC1_Green_	1	输	通道 1 像素绿色分量第 6	
DC1_Green_	1	输	通道 1 像素绿色分量第 7	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 0	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 1	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 2	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 3	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 4	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 5	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 6	
DC1_Blue_	1	输	通道 1 像素蓝色分量第 7	

每个 DVO 接口对应 29 根管脚信号，共 58 根。

7.3. 外接要求

ICH2 两 DVO 接口均采用独立的源同步传输，支持源同步时钟极限取反。

对于 DVO 接口 0，除源同步时钟外，其它源同步数据之间延迟最大偏斜为 290ps。以源 同步时钟 DC0_Clk 为参照点，其它源同步数据偏斜最小为-10ps（即源同步数据比源同步时 钟晚 10ps 到达），最大为 390ps（即源同步数据比源同步时钟早 390ps 到达）。

对于 DVO 接口 1，除源同步时钟外，其它源同步数据之间延迟最大偏斜为 310ps。以源 同步时钟 DC1_Clk 为参照点，其它源同步数据偏斜最小为-90ps（即源同步数据比源同步时 钟晚 90ps 到达），最大为 340ps（即源同步数据比源同步时钟早 340ps 到达）。

上述源同步数据与源同步时钟之间延迟偏斜列表如下（单位 ps）：

表 7-2 源同步时钟偏斜

偏斜类型	彼此偏斜	相对于源同步数据偏斜	
		最小偏斜	最大偏斜
DVO0 源同步数据	290	-10	390
DVO1 源同步数据	310	-90	340

根据上述管脚内部延迟偏斜，板级设计人员可根据所选显示发送器芯片目标器件对 DVO 接口时序要求进行具体的布局布线。要求源同步数据之间采用等长方式布线，建议源同步数据可根据需要适当增加布线延迟。需要强调的是：ICH2 内部支持对 DVO 源同步时 钟极限反转，可有效降低布线难度。

7.4. 交直流电气特性

显示 DVO 接口使用标准 LVCOMS 1.8V 供电的接口缓冲，其电气特性参见 0 节。DVO 接口为 SDR 源同步传输，DC 控制器支持通过配置寄存器使得源同步时钟 DISPLAY0CLK/DISPLAY0CLK 极性反转，默认配置下的时序特性如下所示：

表 7-3 DC 时钟时序特性

参数	Min	Typical	Maximum	Unit
DISPLAY0/1CLK 时钟周期	-	5.5	-	ns
T_SETUP	-	2.4	-	ns
T_HOLD	-	3.0	-	ns

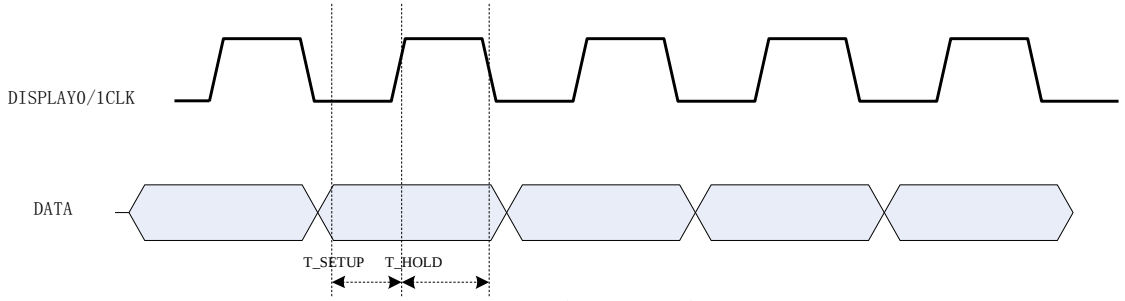


图 7-1 DVO 接口时序特性

8 声卡接口

8.1. 声卡接口概述

ICH2 片内部集成了 AC97 音频控制，通过 AC97 控制器，可以实现主机对 AC97 CODECs(音频数/模、模/数转换芯片)的控制和数据传输，从而实现数字音频的放音和录音过 程。声卡和 CODEC 芯片的接口采用标准 AC-link，支持 AC97 协议 2.3 版本。AC97 控制器 可通过 IO 读写或 DMA 方式进行音频数据的传输。主要特性如下：

- 支持 AC97 协议，2.3 版
- 支持主(DMA)、从操作方式
- 支持 48kHz 音频采样频率
- 硬件采样频率支持 8kHz 到 48kHz，Double Rate Audio 最高支持 96kHz
- 左、右、中声道支持 Double Rate Audio
- 支持 16, 18 和 20 bit 采样宽度
- 支持单声道、双声道、多声道 (1, 2, 4 或 6 声道) 输出
- 支持双声道输入或单声道话筒输入（总计 3 个声道）
- 支持多种电源管理模式（部分或全模块下电）

8.2. 详细说明

8.2.1. AC97 接口协议

AC'97（Audio Codec 1997）是以 Intel 为首的五个 PC 厂商 Intel、Creative Labs、NS、Analog Device 与 Yamaha 共同提出的规格标准。AC'97 不仅是一种数据格式，用于音频编码的内部架构规格，它还具有控制功能。

AC97 标准把音频设备中的数字部分和模拟部分分开,并规定数字信号处理由 CPU 来负 责或者采用专门的 DSP 芯片；而模拟部分,即 A/D、D/A 转换与 Mix 混音操作,则由 CODEC 芯片(Coder-Decoder:编码/解码器)完成。

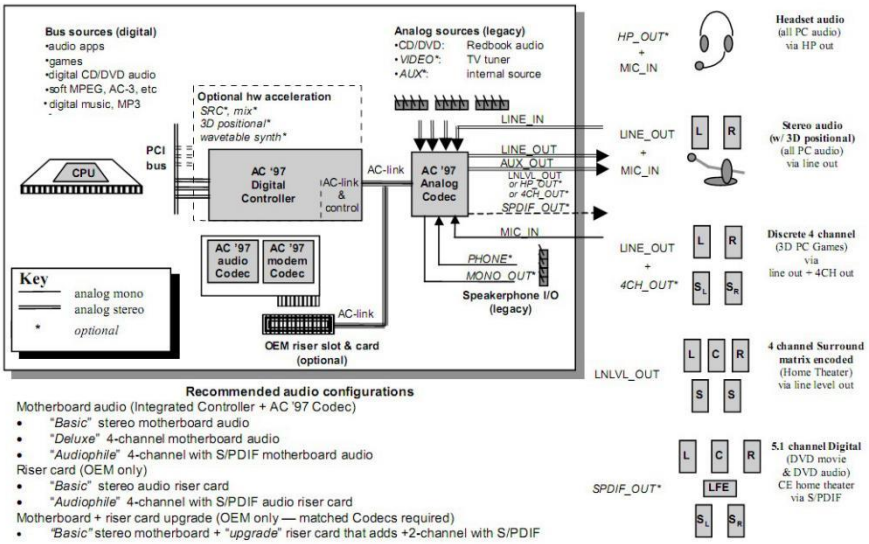


图 8-1 AC‘97 Codec 接口示意图

AC'97 数字串口协议(AC- Link)，是全双工、固定时钟、PCM 串口数据流协议。它采用 时分多路(TDM)复用机制来处理访问控制寄存器和多路输入、输出音频数据流。

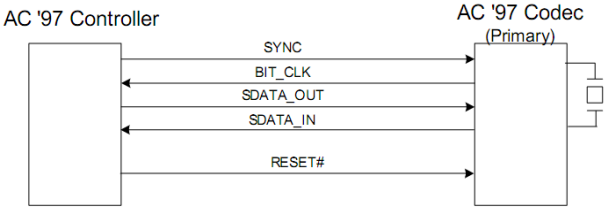


图 8-2 ACLink 连接关系图

AC-Link 接口包括位时钟（BITCLK）、同步信号校正（SYNC）和从编码到处理器及 从处理器中解码（SDATA_IN 与 SDATA_OUT）的数据队列。AC'97 数据帧以 SYNC 脉冲 开始，包括 12 个 20 位时间段（时间段为标准中定义的不同的目的服务）及 16 位–TAGⅡ段，
共计 256 个数据序列。例如，时间段–1Ⅱ和–2Ⅱ用于访问编码的控制寄存器，而时间段–3Ⅱ和–4Ⅱ 分别负载左、右两个音频通道，该负载是 PCM 编码的。–TAGⅡ段表示其他段中哪一个包含 有效数据。把帧分成时间段使传输控制信号和仅通过 4 根线到达 9 个音频通道或转换成其他 数据流成为可能。

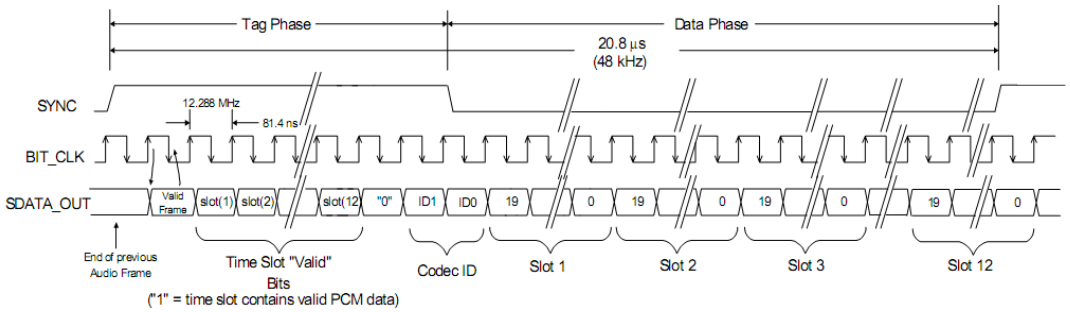


图 8-3 ACLink 时序要求

音频处理的流程如下:

- 输入：CODEC 采样音频数据,把模拟信号转换成数字信号通过 AC-link 和 AC 控制单元传送给 CPU 或 DSP 处理。
- 输出：CPU 对数字信号处理后交由 AC 控制单元通过 AC-link 传输给 CODEC 进行混音处理再转换成模拟信号输出。其中 AC 控制单元是 AC97 CODEC 的控制器。AC-link 是全双工的串行接口,负责传输音频数据、寄存器控制命令和状态信息。

8.2.2. 声卡接口信号

ICH2 内部 AC97 控制器对外接口采用标准 AC-link 接口形式。对应信号采用 1.8V LVCMOS 接口，驱动能力为 8mA。总共 5 根信号，其相关信号如下：

表 8-1 声卡接口信号定义

名称	位宽	IO	描述
AC97_BIT_CLK	1	输入	Codec 芯片产生的位传输时钟，
AC97_SDATA_IN	1	输入	串行传输输入数据
AC97_SYNC	1	输出	AC-link 位传输同频信号
AC97_SDATA_OUT	1	输出	串行传输输出数据
AC97_RESET_L	1	输出	Codec 芯片复位信号，低有效

8.3. 初始化与测试

8.3.1. 初始化

AC97 声卡控制器采用第三方 IP 实现，其初始化流程如下：

1) 每次使用前先软件检查，使用后软件关闭。检查 CORE_STATUS 寄存器，保证软件和 控制器的硬件间没有差别。
2) 软件通过使用 CORE_STATUS 寄存器释放 CODEC 复位：检测 AC97 控制器的活动状 态（ CORE_STATUS 31 位）； 设置 timing 寄存器（ COLD_CNT, WARM_CNT, SUSPEND_CNT）；释放 AC link 复位（CORE_STATUS 31 位）。
3) 上述复位完成后，将触发 AC97 CODEC 开始位时钟的生成，AC97 控制器生成 AClink 的同步信号（ac97_sync_pad_o）。所有的传输都将停止，直到 CODEC_STATUS 的 CODEC_READY 位有效。
4) 上层软件设置 CODEC 寄存器的合适值（音量、音调、gain、通用寄存器、采样频率）。 注：由于 AC97 2.3 标准不支持单声道音频数据流，因此左右声道是同时使能的。

8.3.2. 从模式传输

如果使用从模式，用户（软件）必须初始为所已使能的音频通道填充音频数据至嵌入式 存储器中。设置 INT_MASK 和 AUDIO_ENABLE 后，将启动音频数据传输。

如有中断请求，用户需要读 INT_SOURCE 寄存器，确定是哪个通道引起的中断，在何 地

址边界内（offset/wrap），（从偏移量开始/地址回环），并填充相应音频通道嵌入式存储 器的一半。

1) 嵌入式存储器填充
2) 更新 INT_MASK 寄存器
3) 更新 AUDIO_ENABLE 寄存器
4) 中断时，填嵌入式存储器的一半

8.3.3. 主模式传输

对每一个激活的 AClink 音频通道需要进行多个域/寄存器的设置，由 Master 的 DMA 模 块进行处理。

更新 STREAM_DESC 地址的 CHANNEL_DESC 为一个通道的 stream descriptor 数据。 这些动作只针对配置了 DMA 触发器（INT_DMA_TRIGGER）的音频通道。

为每个一个通道更新 STREAM_DESC 表的 buffer 基地址、长度、stream number 和 channel number、数据大小。对任何 stream，有多少个音频通道就有多少个 STREAM_DESC 条目。当收到启动 DMA 引擎的中断时，需要使用这个表来译码 stream 并正确的填写嵌入式存储器缓冲。

设置 INT_DMA_MASK 来使能相应的音频通道的 DMA 引擎，该引擎将处理数据传输。 这还将阻挡触发中断的信号 int_o（INT_SOURCE），并首次填充输出通道的嵌入式存储器 的缓冲。

设置 INT_DMA_TRIGGER 寄存器来将被中断请求使能 DMA 引擎，如果使用多通道音 频数据流，只置最后一个通道的中断位。

举例来说：发送给前置左右通道的单声道的数据流将置 INT_DMA_MASK 和 INT_DMA_TRIGGER 的[1:0]位为 1，而立体声会设置 INT_DMA_MASK 的[1:0]位和 INT_DMA_TRIGGER[1]。

AClink 的最后一个通道触发的 DMA 传输，使能 AC97 控制器进行单或多声道交叉的音 频数据流传输。

设置 AUDIO_ENABLE 启动音频的 playback/recording

1) 更新 CHANNEL_DESC
2) 更新 STREAM_DESC
3) 更新 INT_DMA_MASK
4) 更新 INT_DMA_TRIGGER
5) 更新 AUDIO_ENABLE

8.4. 交直流电气特性

声卡接口使用标准 LVCOMS 1.8V 供电的接口缓冲，其电气特性参见 12.1 节。

9 SIO 接口

9.1. LPC 接口

ICH2 内部集成了一个 LPC HOST 控制器，实现了 LPC HOST 功能，该功能下主要包含 LPC 控制器和 LEGACY-DMA 控制器，实现符合 LPC 总线标准规范的 LPC 总线接口。 特点：

- 基于 Intel LPC 总线规范（Intel Low Pin Count Interface Specification,Revision1.1）。
- 支持 I/O、Memory、Firmware memory 和 DMA 四种周期类型，不支持 Bus Master 周期类型。
- 内部实现 4 个通道的 Legacy DMA 控制器，同时支持 2 个带 DMA 功能的外设。
- 支持串行中断（Serial IRQ）。
- 支持在 Target（I/O、Memory 和 DMA）方式下设备超时等异常发生后由 Host 主动 终止总线周期，并返回全“E”数据（32‘hEEEE_EEEE）。
- Legacy IO 访问粒度大于 8-bit 时（如 16-bit、32-bit），自动拆分成 2 个或 4 个 IO 周 期，并按访问粒度返回数据。
- 支持 256MB 窗口的 Memory 直接寻址，64MB 窗口的 Firmware 段式空间寻址。

本设计中的 LPC HOST 所支持的周期类型见下表所示。

表 9-1 LPC HOST 支持的周期类型

周期类型	描 述	传输大小 /Byte	是否支持
I/O Read/Write	Target 模式，16 位地址寻址	1	支持
Memory Read/Write	Target 模式，32 位地址寻址	1	支持
Firmware Memory Read/Write	Target 模式，28 位地址寻址	1、2、4	支持
DMA Read/Write	Target 模式，通道	1、2、4	支持
Bus Master Read/Write	Master 模式，32 位地址	-	不支持

9.1.1. LPC 协议概述

LPC（Low Pin Count，低管脚数）总线，是 Intel 公司定义的一个数据地址命令多路复 用总线，工作频率为 33MHz。在工业计算机领域，LPC 总线已经逐渐取代了 ISA 总线而成 为新的接口。

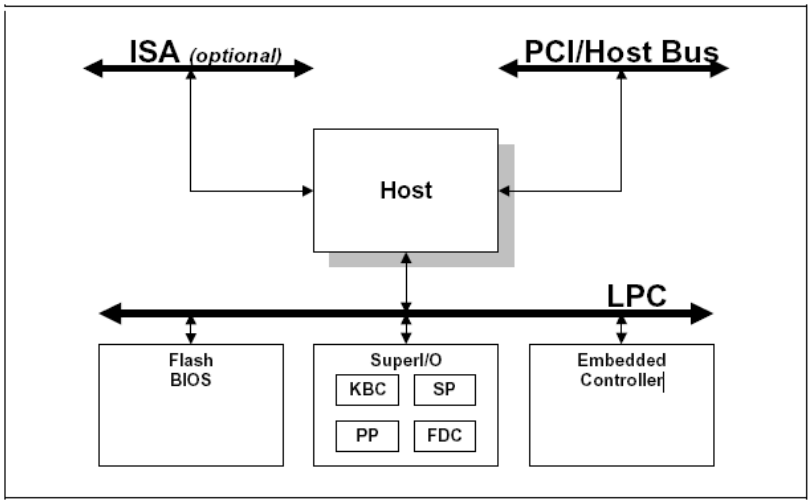


图 9-1　LPC 总线架构图

LPC 总线的优点：

- 相对与 ISA 总线简化了信号连接，降低了外接设备的引脚少（典型应用：从 ISA-xbus的 36 个减少到 7 个，从管脚的效果上来说，过去一个功能完全的 super I/O 设备可 能需要 160 左右的引脚，使用 LPC 接口后可以使用 88 甚至更少的引脚来实现。）
- 对软件的透明度：不需要针对该接口进行配置或使用专门的驱动程序。

LPC 周期概述

LPC 总线上的数据传输是在一个 4 位总线上的连续传输。总线特征如下：

- 一条控制线，叫做 LFRAME#，是 host 用来开始或停止传输的，Peripherals 不驱动 该信号。
- LAD[3:0]总线上传送的信息包括周期类型、周期方向（读/写）、片选、地址、数据 和空闲等待。
- 边频带信号可以选择实现，转达中断和电源管理。 周期的大致流

程如下：

1) 由 host 将 LFRAME#置有效来启动某个周期，并将合适的信息送到 LAD[3:0]上；
2) Host 继续发送该周期相关的信息，诸如地址、DMA 通道号等，对于 DMA 和目标 周期，host 也驱动周期类型（Memory 或 I/O）、读写方向和传输内容大小；
3) Host 要么驱动数据，要么将总线变为监听 Peripherals 等待完成该周期；
4) Peripherals 通过 LAD[3:0]发送相应的值来表明周期完成；
5) Peripherals 将总线交还给 Host，周期结束。

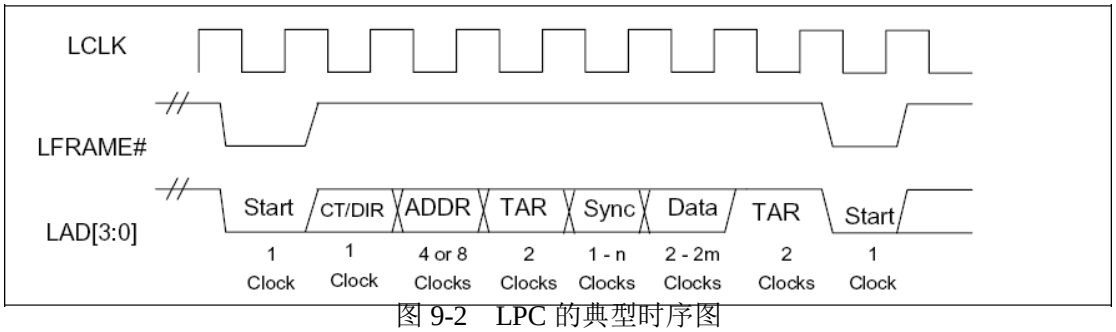


图 9-2　LPC 的典型时序图

9.1.2. 接口信号说明

表 9-2 LPC 接口信号说明

名称	位宽	IO	描述
LCLK	1	I	LPC 时钟，33MHz
CFG_LDRQ1N	1	I	LPC 设备 DMA 申请引脚 1，低有效
CFG_LDRQ2N	1	I	LPC 设备 DMA 申请引脚 2，低有效
LFRAMEN	1	O	LPC 帧起始信号：host 用此信号来开始一个传输 周期，或结束一个异常周期，peripheral 不得 驱 动此信号，低电平有效
LRSTN	1	O	LPC-Host 输出的复位信号，可用于 LPC 设备的 复位，低电平有效
SERIRQ	1	IO	LPC 串行中断信号，遵循 Serialized IRQ 协议
LAD[3:0]	4	IO	LPC 命令/地址/数据复用总线

9.1.3. 初始化与测试

1. 初始化流程

1) 若有 Memory 周期类型的设备，则配置寄存器 MEM_HADDR；若无，则采用默认值；2) 若有 Firmware 周期类型的设备，则根据 Firmware 芯片的 ID 号和规划的地址空间来配置寄存器 FWH_IDSEL_R#（#代表数字 1~5 中的某一个）；
3) 根据实际情况配置串行中断屏蔽寄存器 LPC_IRQ_MSK；
4) 若需要串行中断功能，则开启 SIRQ功能（LPC_CTRL[9]），并选择串行中断模式（LPC_CTRL[8]），然后配置 Start Frame 参数（LPC_CTRL[5:4]）和 Data Frame 参 数（建议 LPC_CTRL[4:0]=0xF）；
5) 根据实际需要配置 LPC 内部 IO/Memory/FWH 和 DMA 两条通路仲裁原则（LPC_CTRL[11]），以及进入长等待后多久报超时（LPC_CTRL[7:6]）；
6) 若 SIO 设备（如 NCT6776），则强烈建议配置 LPC_CTRL[12]=1；
7) 最后开启 LPC HOST 功能，即配置 LPC_CTRL[10]=1。

2. DMA 初始化流程

1) 配置所选通道对应的目的地址寄存器 CH#_ADDR（#代表数字 0~1 中的某一个）；
2) 配置所选通道对应的传输总长度寄存器 CH#_ADDR；
3) 配置所选通道对应的模式寄存器 CH#_MODE，确定工作模式、周期粒度（size）和 读写类型；
4) 配置通道屏蔽寄存器，开启要使用的通道；
5) 配置 DMA 控制寄存器 DMA_CTRL，配置通道仲裁原则并开启 DMA 功能。

3. 测试

1) 系统复位后，可先对所有读写（W/R）寄存器进行先写后读测试，以验证寄存器的 读写操作的正确性。

2) 启动一个简单的 IO/Memory/Firmware 周期（如读取设备的 Chip‘s ID），查看读取 结果和状态寄存器 LPC_STATE 和 CURT_INF 对应的状态标志位是否正确。

9.1.4. 布局布线要求

由于 LPC 总线时钟由控制器（ICH2，HOST）端提供，所以 LPC 外接器件尽量靠近 ICH2 放置，保证 HOST 端接收数据时相对时钟有足够的建立时间，建议 LAD 等长布线。

9.1.5. 电气特性

LPC 接口使用标准 1.8V 供电的接口缓冲，其电气特性参见 LPC 标准协议 1.1 中的第 10 节。

9.2. SPI 接口

9.2.1. SPI 接口概述

安全 IO 芯片集成了一个 SPI 控制器挂接在低速 IO 的 wishbone 交叉开关上，SPI 控制 器对 wishbone 交叉开关是一个 slave 端口。通过该 SPI 控制器可以实现对外部 SPI 接口芯片 的访问，SPI 控制器对外部的 flash 或 eeprom 芯片来说是一个 SPI master 端口。SPI 控制器支持对外部芯片的 16 位和 24 位地址模式访问，实现的是间接访问模式，无论操作码、还是地址和数据，都通过写入 SPI 控制器的数据寄存器实现读写外部芯片的操作。

9.2.2. 详细说明

9. 2. 2. 1. 传输模式

寄存器具体定义参考寄存器手册，通过寄存器 CTRLR0 的 TMOD 位的配置，可以选择 SPI 控制器进行如下四种模式的传输：

1） 发送和接收

TMOD=2‘b0，发送和接收逻辑都有效，发送数据从发送 FIFO 中取出送到 txd 输出 接口，并从 rxd 接口接收输入数据写入接收移位寄存器，然后写入接收 FIFO。数据 的接收需要发送 FIFO 不断的有数据输出到串行总线，以保证持续的串行时钟的产生。

2） 只发送

TMOD=2‘b01，只有发送逻辑有效，只往串行总线上上发送数据，不会从总线上接 收数据。

3） 只接收

TMOD=2‘b10，只有接收逻辑有效。 4）

EEPROM 读

TMOD=2‘b11，发送数据用于传输命令码和地址到 EEPROM 设备，一般需要 3 个数据 帧（8-bit 操作码，8-bit 高位地址，8-bit 低位地址）。在操作码和地址传输的过程中，控制 器不会

从串行总线上接收数据。当发送 FIFO 中的条目发送完毕，接收数据才开始被采样， 采样的数据帧数是 NDF+1，见寄存器 CTRLR1。

9.2.2.2. 操作模式

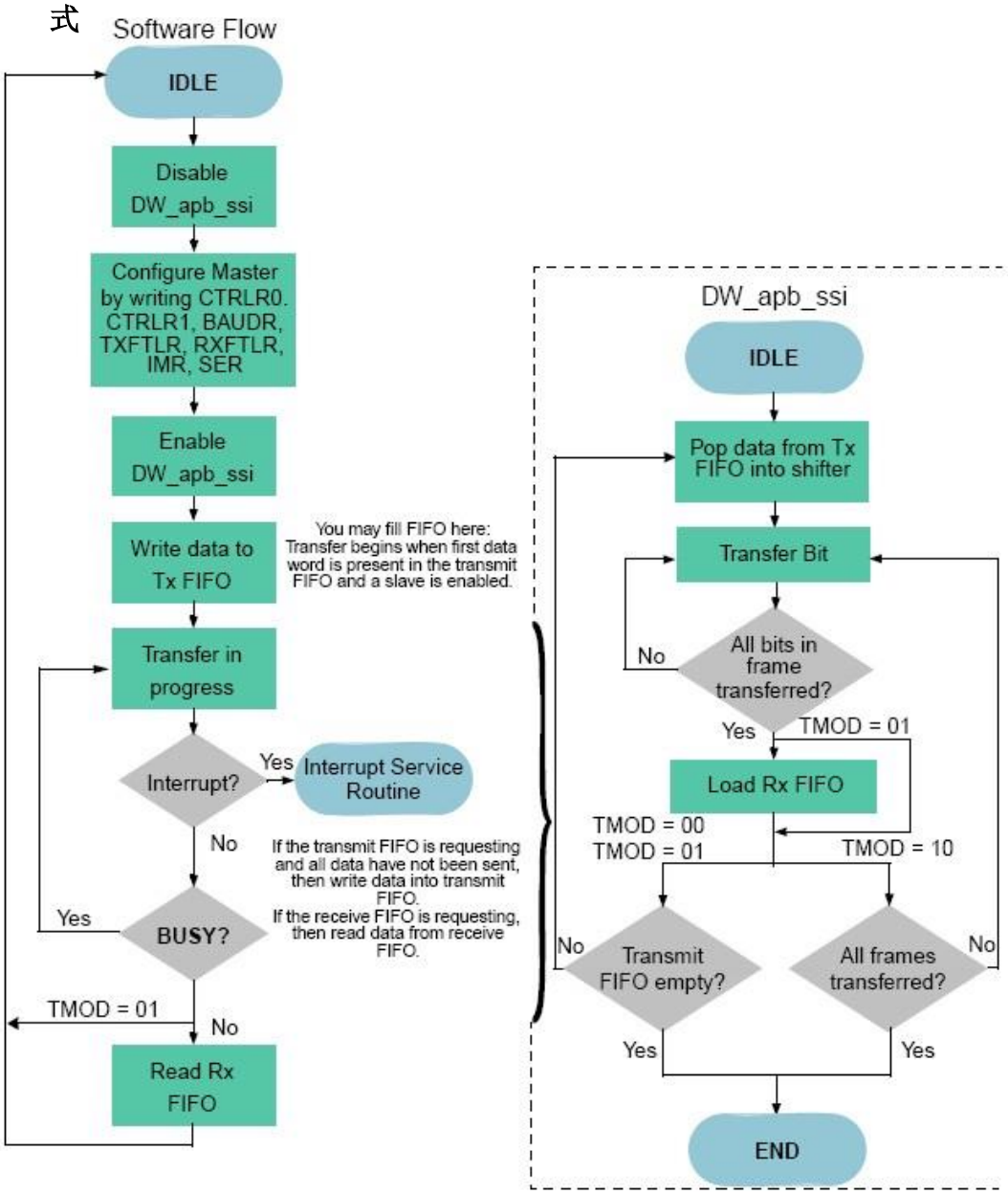


图 9-3 SPI 控制器操作流程

9.2.2.3. 工作流程

通过 SPI 控制器对 SPI 接口芯片进行编程流程如下：

表 9-3 SPI 控制器编程流程

序号	操作
1	disable SSI 控制器，写‘h8 地址 SSIENR 寄存器值为 0x0
2	配置 TMOD 为 transmit only 模式，写‘h0 地址 CTRLR0 寄存器值为 0x1c7，配置
3	配置波特率，写‘h14 地址 BAUD 寄存器值为 2
4	配置发送 FIFO 阈值，写‘h18 地址 TXFTLR 寄存器值为 0x4
5	配置接收 FIFO 阈值，写‘h1c 地址 RXFTLR 寄存器值为 0x4
6	使能 slave 设备，写‘h10 地址 SER 寄存器值为 0x1
7	enable SSI 控制器，写‘h8 地址 SSIENR 寄存器值为 0x1
8	发送写使能命令，写‘h60 地址 DR 寄存器值为 0x6
9	写‘h10 地址 SER 寄存器值为 0x0
10	发送擦除命令操作码，写‘h60 地址 DR 寄存器值为 0x20
11	发送擦除命令高段地址[23:16]，写‘h64 地址 DR 寄存器
12	发送擦除命令中段地址[15:8]，写‘h68 地址 DR 寄存器
13	发送擦除命令低段地址[7:0]，写‘h6c 地址 DR 寄存器
14	写‘h10 地址 SER 寄存器值为 0x1
等待擦除完毕	
15	发送写使能命令，写‘h60 地址 DR 寄存器值为 0x6
16	写‘h10 地址 SER 寄存器值为 0x0
17	发送编程命令操作码，写‘h60 地址 DR 寄存器值为 0x2
18	发送编程命令高段地址[23:16]，写‘h64 地址 DR 寄存器
19	发送编程命令中段地址[15:8]，写‘h68 地址 DR 寄存器
20	发送编程命令低段地址[7:0]，写‘h6c 地址 DR 寄存器
,,,,	从‘h70 地址寄存器开始到‘hec，再从‘h60 到‘hec 地址寄存器，依次写入希望编程
	写‘h10 地址 SER 寄存器值为 0x1
等待编程数据写入结束	

通过 SPI 控制器对 SPI 接口芯片进行 EEPROM 读工作流程如下：

表 9-4 SPI 控制器 EEPROM 读流程

序号	操作
1	disable SSI 控制器，写‘h8 地址 SSIENR 寄存器值为 0x0
2	配置 TMOD 为 eeprom read 模式，写‘h0 地址 CTRLR0 寄存器值为 0x3c7，配置
3	配置波特率，写‘h14 地址 BAUD 寄存器值为 2
4	配置发送 FIFO 阈值，写‘h18 地址 TXFTLR 寄存器值为 0x4
5	配置接收 FIFO 阈值，写‘h1c 地址 RXFTLR 寄存器值为 0x4
6	配置接收数据帧数目 NDF，写‘h4 地址 CTRLR1 寄存器值为 ndf，接收的数据帧 的数目从发送完 8-bit 操作码和 16-bit 地址后，间隔一个 8bit 串行数据收集时间

7	配置接收采样延迟，写‘hf0 地址 RSD 寄存器值为 0x0 或0x10，如果随命令后发 送 16-bit 地址，则写入 0x10，如果发送 24-bit 地址，则写入 0x0，原因见下面
8	不使能 slave 设备，写‘h10 地址 SER 寄存器值为 0x0
9	enable SSI 控制器，写‘h8 地址 SSIENR 寄存器值为 0x1
10	发送读数据命令操作码，写‘h60 地址 DR 寄存器值为 0x3
11	发送读数据命令高段地址[23:16]，写‘h64 地址 DR 寄存器
12	发送读数据命令中段地址[15:8]，写‘h68 地址 DR 寄存器
13	发送读数据命令低段地址[7:0]，写‘h6c 地址 DR 寄存器
14	使能 slave 设备，写‘h10 地址 SER 寄存器值为 0x1
等待读响应数据返回	
15	读地址‘h34 的原生中断状态寄存器的[4]位，判断是否为“1”，如果为 1，则代 表接收 FIFO 已经收到接收阈值数目以上的接收数据，可以启动系统总线读操作，连续的读‘h60~‘hec 地址数据寄存器的接收数据，直到该位始终保持为

9.2.3. 布局布线要求

SPI flash 尽量靠近 ICH2 芯片，时钟与数据等长布线。spi 协议规定下降沿发数据，上升 沿收数据。ICH2 的设计要求 SPI 串行时钟达到 50MHz，所以一个串行周期时间是 20ns。ICH2的 SPI 输出 pin_delay=2.45ns，输入的 pin_delay=2.12ns，对接收方向来说 ICH2 芯片用掉 4.6ns，片外线延时 +flash 出数延迟要求<5.4ns，板级设计需要充分考虑走线和 flash 的总延时不能大于 5.4ns。

9.3. PS2 接口

9.3.1. PS2 接口概述

PS2 接口实现了安全 IO 芯片的传统 PS2 键盘和鼠标设备的接入。PS2 鼠标和键盘履行 一种双向串行协议。每次数据线上发送一位数据并且每在时钟线上发一个脉冲就被读入。键 盘/鼠标可以发送数据到主机，而主机也可以发送数据到设备，但主机总是在总线上有优先 权，主机可以在任何时候抑制来自于键盘/鼠标的通讯，只要把时钟拉低即可。

键盘和鼠标如果通过 MSI 中断方式发给 CPU 时，MSI 中断信息中包含了区分键盘还是 鼠标的信息，软件只需要读取 60h 寄存器中的数据即可，完成 60h 寄存器的读操作后，PS2 接口自动清除对应的中断。

如果采用 INTx 中断方式时，软件首先需要读取 64h 寄存器中的值来区分是键盘还是鼠 标发出了中断，然后再读取 60h 寄存器中的值，之后 PS2 接口自动清除对应的中断。

PS2 接口设备时钟的最高频率是 33K，一般工作在 10KHz～20KHz。如果核心工作时钟 100MHz，分频系数为 500（devide_reg），能够得到 5 微秒的计时值（200KHz）。

9.4. UART 接口

9.4.1. UART 接口概述

UART（Universal Asynchronous Receiver/Transmitter）接口提供串行通信能力，可以和 modem 或其它外部设备，例如使用串口线实现 RS232 协议的另一台计算机通信。ICH2 芯片的 uart 接口实现的是全功能的串口，也可以通过该 uart 接口对 SPI 接口 Flash 进行编程。

9.5. GPIO 接口

9.5.1. GPIO 接口概述

GPIO 接口系统可见，实现 2 组，每组各 8 根输入/输出线，可以控制输出数据和 I/O pad 的方向，也可以使用寄存器读回 I/O pad 的数据。为了使用方便，GPIO 控制器实现了每组 的 8 根数据线的按位操作，而不是按字节操作。所以，增加对应 8 位数据的屏蔽码，和数据同拍写入数据寄存器的高位，即[7:0]位为写入的 8 位数据，[15:8]位为 8 位数据对应的屏蔽 码，GPIO 内部根据 8 位数据和 8 位屏蔽码进行对应位的修改。GPIO 控制器还可以作为中 断控制器使用，允许外部输入中断信号，并且可以将外部的多根中断信号或操作出一根中断 线输出。

9.5.2. 详细说明

GPIO 控制器的工作流程如下：

通过读写寄存器 gpio_swportx_dr 和 gpio_swportx_ddr 实现数据和方向的控制。

- 通过写寄存器 gpio_swportx_ddr 控制外部 I/O pad 的方向，写“ 1”到 gpio_swportx_ddr 对应位代表选择输出方向；
- 写数据到 gpio_swportx_dr 寄存器对应位代表输出数据，并且数据寄存器的低 8 位 是输出数据，第[15:8]位是对应低 8 位的写使能，写使能位写“1”代表允许对应位 写入，写“0”代表不进行对应位的写；
- 写“ 0 ”到 gpio_swportx_ddr 对应位代表选择输入方向， 软件通过读寄存器 gpio_ext_portx 获取输入数据。从输入信号 gpio_ext_portx 可以获得外部的输入数 据，但是前提是取决于 gpio_ext_portx 是被配置成输入还是输出：
  - 配置成输入——读取输入信号 gpio_ext_portx 的数据；
  - 配置成输出——读取 Portx 的数据寄存器。

  gpio_ext_portx 寄存器是只读的，驱动不能对该寄存器进行写操作。

GPIO 控制器的中断流程如下：

GPIO Port A 的 8 根线可以选择部分也可以选择全部配成中断信号，即可以根据板级实 现情况，按位配置 Port A 的信号为 GPIO 信号或中断信号。GPIO 的中断功能实现了中断屏 蔽寄存器，中断状态寄存器，中断清除寄存器。

- 配置寄存器 gpio_inttype_level，指定中断输入信号触发属性，是电平还是脉冲触发， 应该配置成电平触发；
- 可以配置中断屏蔽寄存器，选择中断源的屏蔽和不屏蔽；
- 配置寄存器 gpio_int_polarity，配置输入的每根中断信号是高电平触发还是低电平 触发；
- 配置方向寄存器，指定中断线对应位的方向为输入；
- 配置寄存器 gpio_inten，按位指定每一位是 GPIO 信号还是中断信号； 组合生成的一根中断线通过 PAB 桥以 MSI 包方式或 INTX 中断线方式提交到系统后，

软件查询 GPIO 的中断源寄存器并进行中断处理后，需要写 GPIO 的中断清除寄存器，清除 中断。虽然 GPIO 控制器提交了一个中断信号，但是可能是多个设备的中断源组合而成。

9.6. I2C-Master 接口

9.6.1. I2C Master 接口概述

I2C 是一个两线接口，包括串行数据总线 SDA 和串行时钟 SCL。通过 I2C，各设备之 间以及设备与系统的其它部分之间可以互相通信。I2C 为系统和电源管理相关的任务提供一 条控制总线。一个系统利用 I2C 可以和多个设备互传信息，而不需要使用独立的控制线路。 I2C 总线涉及三类设备：

- ✧ 主设备，用来发布命令，产生时钟和终止发送的设备；
- ✧ 从设备，接收或响应命令的设备；
- ✧ 主机，是一种专用的主设备，必须具有主-从机功能，一个系统里只有一个主机。

9.6.2. 详细说明

I2C 控制器工作流程如下：

1）写-0Ⅱ到 DW_apb_i2c 的 IC_ENABLE 寄存器，关闭控制器 IP 使能，可以进行相关模式配 置；

2）写 IC_CON 寄存器，可以设置最大的速度模式，7bit 还是 10bit 地址模式，IC_SLAVE_DISABLE 位设置成-1Ⅱ，MASTER_MODE 位设置成-1Ⅱ；

3）写 IC_TAR 寄存器，设置 I2C 目标设备地址；

4）写-1Ⅱ到 IC_ENABLE 寄存器，使能控制器 IP；

5）写 IC_DATA_CMD 寄存器，写入传输方向（读写类型）和写数据： i）如果是写请求，IC_DATA_CMD[8]位写-0Ⅱ，IC_DATA_CMD[7:0]为写数据； ii）如果是读请求，IC_DATA_CMD[8]位写-1Ⅱ。

6）如果传输的是写请求：当 I2C 控制器把写请求和数据从发送 FIFO 都发到总线上，控制 器就产生一个 STOP 中断，表示当前传输完成，即置寄存器 IC_RAW_INTR_STAT[9]位为-1Ⅱ， 并通知系统。驱动通过读中断源寄存器获取中断类型并处理，然后读寄存器 IC_CLR_STOP_DET 即可清除该中断。

7）如果传输的是读请求：读请求命令发到 I2C 总线，控制器按照 I2C 协议接收总线返回的 读响应数据，并写入接收 FIFO，控制器置寄存器 IC_RAW_INTR_STAT[2]位为-1Ⅱ，代表 RX_FULL，并发出中断通知系统。驱动读取中断源寄存器，确定中断类型，然后驱动读取 IC_DATA_CMD 寄存器获得读响应数据。RX_FULL 中断会被硬件自动清除当数据被逐个读 出接收 FIFO 后。

9.6.3. 布局布线要求

建议 I2C 的时钟和数据线等长。

10 时钟接口

10.1. 时钟接口概述

为了满足 ICH2 的时钟需求，内部集成了 TCI 公司的 Clock Generator 类型 PLL（Part No. TCITSMCN40GCGHPLLA1），该 PLL 为 ICH2 提供了显示加速部件工作时钟、PCIE 交叉开关工作时钟、内核主工作时钟和存储控制器接口参考时钟。该 PLL 支持 Deskew 功能，功耗低，适于系统时钟、DDR 以及要求较小面积、低功耗和低硬件开销的应用，具体参数如 下表：

表 10-1 PLL 性能指标详细说明

参数名称	描述
输入参考时钟范围	415KHz - 3.4GHz
输出时钟频率范围（/1）	680MHz - 3.4GHz
X 分频器的值（NR：参考分频）	1~64
Y 分频器的值（NF：反馈倍频）	1~4096
Z 分频器的值（OD：输出分频）	1, 2-16 (只能是偶数)
反馈信号延迟（最大）	n/a （内部反馈）
输出时钟占空比	50%，+/-5% (/1), +/-2% (/N)
输出峰-峰值周期抖动	+/-2.5% 输出时钟周期
功耗（标称）	5mA@1.7GHz（Z 分频器的值为 1）
最小复位时间	5us
复位期间输出频率（/1）	20MHz~200MHz
锁定时间（最小允许值）	500 个参考时钟周期
面积（包括隔离区）（最大）	0.063mm^2
PLL 供电管脚	1VDDA，1VSSA
允许的最大低频电源噪声(峰-峰值)	10%VDDA
允许的最大低频基板噪声(峰-峰值)	10%VDDA
参考输入时钟最大抖动(峰-峰值)	输入参考时钟的 2%
参考时钟最小脉冲宽度（高/低）	150ps
工艺	TSMC CLN40G 40nm
电源电压（VDD,VDDA）	0.9V，+/-10%
结温（标称值，最小值，最大值）	（70，-40C，125C）

10.1.1. 工作模式

PLL 的工作模式可以分为正常、旁路和测试三类，本节概要介绍这三种工作模式。

10. 1. 1. 1. 正常或旁路模式

正常工作模式为 TEST=0，CLKOUT 频率的计算公式为：CLKOUT=

RCLK*NF/NR/OD。

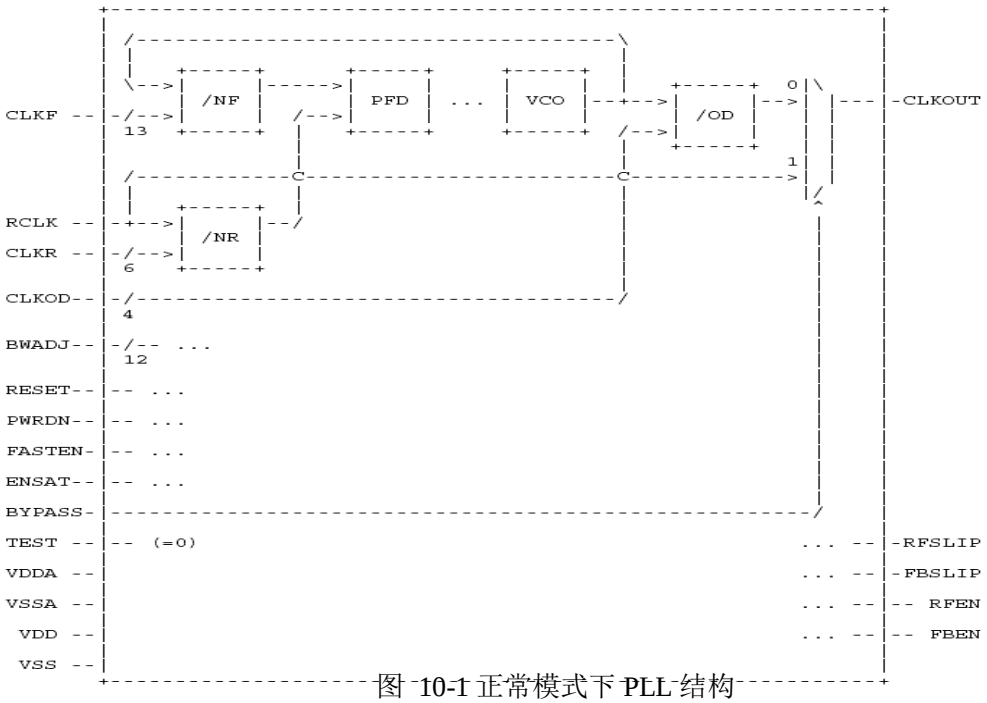


图 10-1 正常模式下 PLL 结构

10.1.1.2. 测试模式

如果 TEST 信号输入有效（高有效），则 PLL 工作在测试模式下。如果 NR 的值等于 1,NF 等于 1 或 OD 小于等于 2，输出将不翻转。输出频率的计算公式为：CLKOUT= RCLK/NF/NR/OD。

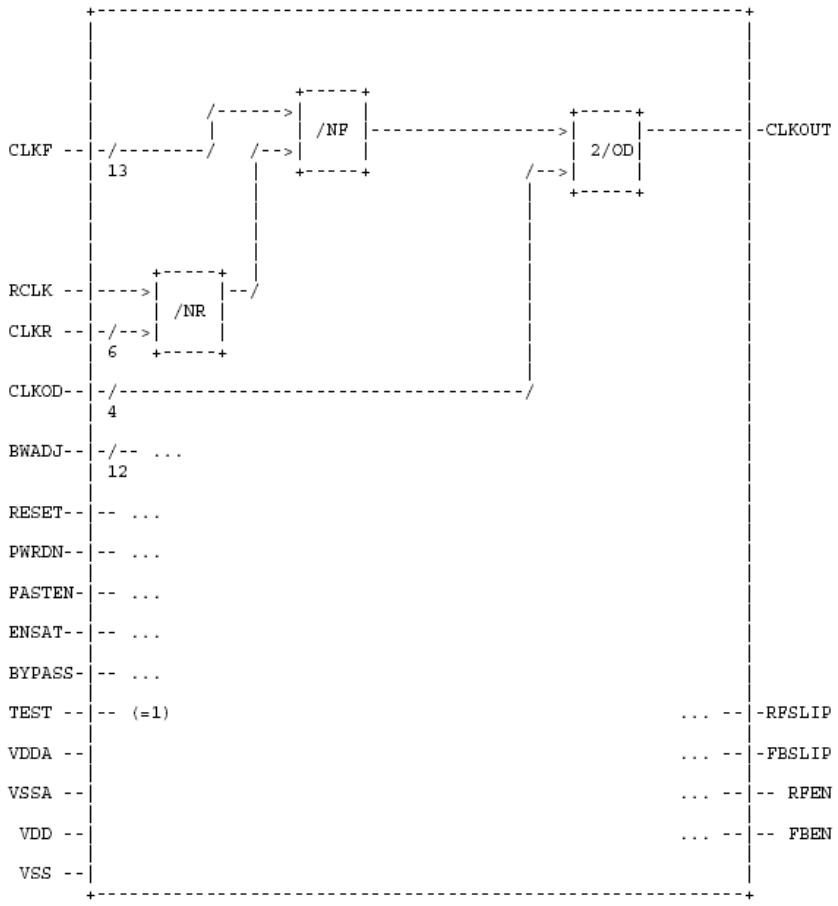


图 10-2 测试模式下 PLL 结构

10.1.2. 信号管脚

ICH2 时钟接口信号采用 1.8V LVDS 或 LVCMOS 电平，共 14 根输入信号。其中差分时 钟输入（100MHz）为一对 LVDS 电平信号，其它为 1.8V LVCMOS 信号。ICH2 内部集成 4个锁相环，通过管脚信号可以灵活配置各锁相环的分频参数，以调整锁相环输出时钟频率。 另外，设计中对芯片内各种锁相环的输出时钟进行分频输出进行监测，以便进行系统调试。

表 10-2 时钟接口管脚定义

信号名称	管脚方	管脚数量	工作电压	信号频率	信号说明	上下拉/驱动	缺省值
REFCLK_N	I	1	1.8V/2.5 V	100 MHz	参考时钟	输入片	
REFCLK_P	I	1	1.8V/2.5 V	100 MHz	参考时钟	输入片	
CFG_GCLKPLL_0	I	3	1.8v/2.5 V	电平信号	GCLK 锁相环参数配置引脚	输入片内上	
CFG_GCLKPLL_1							
CFG_GCLKPLL_2							

CFG_GPUPLL_0 CFG_GPUPLL_1 CFG_GPUPLL_2	I	3	1.8v/2.5 V	电平信号	GPUCLK 锁相环参数配置	输入片内上	
CFG_MCPLL_0 CFG_MCPLL_1 CFG_MCPLL_2	I	3	1.8v/2.5 V	电平信号	MCCLK 锁相环参数配置引脚	输入片内上	
CFG_SWCLKPLL_0 CFG_SWCLKPLL_1 CFG_SWCLKPLL_2	I	3	1.8v/2.5 V	电平信号	PSWCLK 锁相环参数配置	输入片内上	
WATCH_OUT	O	1	1.8V	50MHz	片内信号观测	输出 8mA 驱动	

10.2. 使用方式

10.2.1. 显示部件时钟配置

ICH2 内部集成的显示部件（GPU 与 VPU）采用一个锁相环产生所需工作时钟。该锁相 环参考时钟采用 100MHz 的 LVDS 电平输入，通过 PLL 配置产生所需工作频率。

配置的前提：VCO=180MHz～900MHz；为了减少输出时钟抖动，VCO 的输出尽量高，如果需要分频，尽量在输出分频器 OD 处（OD 尽量大于 1）进行分频。参考时钟分频器 NR尽可能小，反馈时钟分频器 NF 尽可能大。

ICH2 显示部件时钟 GPUPLL 配置信号 CFG_GPUPLL [2:0]有 3 根，可设置该 PLL 输出频率为 100～850MHz，当配置管脚接为全高 3‘b111 时，还可以通过维护寄存器配置任意输出频率。具体定义如下：（红色字体为典型工作频率）

表 10-3 显示部件工作时钟配置表

配置引脚 CFG_GPUPLL	CLKR	CLKF	CLKOD	BWADJ	F_{VCO}	PLLOUT
0	1	16	16	8	1600	BYPAS
1	1	30	6	15	3000	500
2	1	24	4	12	2400	600
3	1	28	4	14	2800	700
4	1	30	4	15	3000	750
5	1	32	4	16	3200	800
6	1	34	4	17	3400	850
7	选用寄存器配置方式控制时钟工作频率					

当 CFG_GPUPLL[2:0] = 3‘b111 时，可通过寄存器方式对锁相环的输出时钟进行控制， 相应寄存器如下表所示。

表 10-4 GPU 时钟锁相环寄存器控制

寄存器名称	描述
GPUPLL_Bypass	配置 PLL0 的 Bypass。上电复位置 0。维护时钟域，通过

GPUPLL_CLKR[5:0] NR-1	PLL0 内部参考时钟的分频器控制信号，配置 CLKR。自
GPUPLL_CLKF[12:0] NF-1	PLL0 内部反馈时钟的分频器控制信号，配置 CLKF。自
GPUPLL_CLKOD [3:0] OD-1	PLL0 内部输出时钟的分频器控制信号，配置 CLKOD。

NF/2-1	PLL0 内部输出时钟的滤波带宽控制信号，配置

芯片内显示部件包含三个部分：GPU、VPU 与双端口显示控制器 DC。其中 VPU 使用时钟全局工作时钟。DC 的时钟通过有 2 种模式，当 DCSEL 为 1 时，双端口显示控制器使用管脚输入时钟 DC0CLK 与 DC1CLK；当 DCSEL 为 0 时，显示接口 0 采用 GPUPLL 输出时钟的 1/8 分频时钟（500MHz 时 62.5MHz 对应 VGA 显示模式），显示接口 1 采用 GPUPLL 输出时钟的 16 分频（500MHz 时 31.25MHz 对应 XGA 显示模式）。

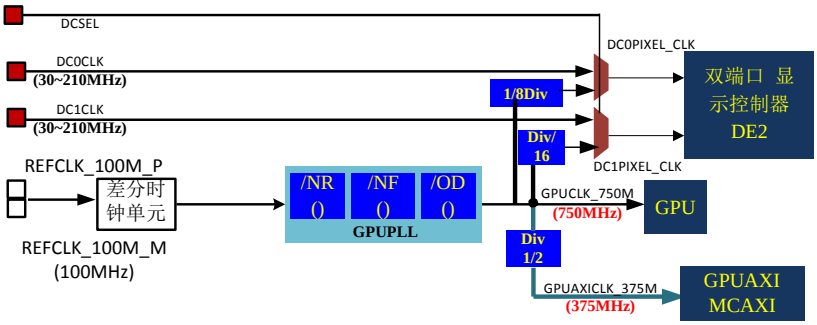


图 10-3 GPU 时钟结构示意图

10.2.2. 存储控制器时钟配置

通过芯片管脚信号 CFG_MCPLL[2:0]，可实现存储器接口 DDR3 PHY 的 PLL 输入参考时钟频率 100MHz~400MHz 的调整，当配置管脚接为全高 3'b111 时，还可以通过维护寄存器配置任意输出频率。时钟输入为 LVDS 电平，输入频率 100MHz，存储器接口 DDR3 PHY 的 PLL 输入参考时钟采用第三方的锁相环实现，其配置如下：

表 10-5 存储控制器输入时钟配置表

配置引脚 CFG_MCPLL	CLKR	CLKF	CLKOD	BWADJ	F_{VCO}	PLLOUT
0	1	16	16	8	1600	BYPAS
1	3	80	16	40	2666.7	166.66
2	1	32	16	16	3200	200
3	3	96	12	48	3200	266.
4	3	100	10	50	3333.3	333.33
5	3	88	8	44	2933.3	366.66
6	1	32	8	16	3200	400
7	选用寄存器配置方式控制时钟工作频率					

注意：配置 0 为旁路有效，PLL1 输出给 DDR3 PHY 的时钟频率等于 RCLK。

当 CFG_MCPLL[2:0] = 3‘b111 时，可通过寄存器方式对锁相环的输出时钟进行控制，相 应寄存器如下表所示。

表 10-6 MC 时钟锁相环寄存器控制

寄存器名称	描述
MCPLL_Bypass	配置 PLL1 的 Bypass。上电复位置 0。维护时钟域，通过
MCPLL_CLKR[5:0] NR-1	PLL1 内部参考时钟的分频器控制信号，配置 CLKR。自
MCPLL_CLKF[12:0] NF-1	PLL1 内部反馈时钟的分频器控制信号，配置 CLKF。自
MCPLL_CLKOD[3:0] OD-1	PLL1 内部输出时钟的分频器控制信号，配置 CLKOD。
NF/2-1	PLL1 内部输出时钟的滤波带宽控制信号，配置

10.2.3. PCIE 交叉开关时钟配置

通过芯片管脚信号 CFG_SWCLKPLL[2:0]，可实现芯片内 PCIE 交叉开关工作时钟频率 100MHz~320MHz 的调整，当配置管脚接为全高 3‘b111 时，还可以通过维护寄存器配置任 意输出频率。锁相环输入参考时钟为 LVDS 电平，输入频率 100MHz。当 PCIE 工作在 1.0 模式时，配置 1,2,3 可用（保证工作时钟频率大于 125MHz），当 PCIE 工作在 2.0 模式时， 配置 4,5,6 可用（保证工作时钟频率大于 125MHz），其配置如下：

表 10-7 PCIE 交叉开关输入时钟配置

配置引脚 CFG_PSWPLL	CLKR	CLKF	CLKOD	BWADJ	F_{VCO}	PLLOUT
0	1	16	16	8	1600	BYPAS
1	1	24	16	12	2400	150
2	1	32	16	16	3200	200
3	5	156	12	78	3120	260
4	5	168	12	84	3360	280
5	1	30	10	15	3000	300
6	5	160	10	80	3200	320
7	选用寄存器配置方式控制时钟工作频率					

当 CFG_PSWPLL[2:0] = 3‘b111 时，可通过寄存器方式对锁相环的输出时钟进行控制，相应寄存器如下表所示。

表 10-8 PSW 时钟锁相环寄存器控制

寄存器名称	描述
PSWPLL_Bypass	配置 PLL2 的 Bypass。上电复位置 0。维护时钟域，通过
PSWPLL_CLKR[5:0] NR-1	PLL2 内部参考时钟的分频器控制信号，配置 CLKR。自
PSWPLL_CLKF[12:0] NF-1	PLL2 内部反馈时钟的分频器控制信号，配置 CLKF。自

PSWPLL_CLKOD [3:0] OD-1	PLL2 内部输出时钟的分频器控制信号，配置 CLKOD。
NF/2-1	PLL2 内部输出时钟的滤波带宽控制信号，配置

注意：PCIE_SWITCH IP 每个端口有两个异步 FIFO；一是发送 TX FIFO（SWITCH->PIPE），WCLK 是 CoreCLK，RCLK 是 PIPECLK，SWITCH 设计时，只对该 FIFO 满进行处理，不处理 FIFO 空的情况，因此 CoreCLK≥PipeCLK；二是 RX FIFO（PIPE -> SWITCH），WCLK 是 PIPECLK，RCLK 是 CoreCLK，SWITCH 设计时，只对该 FIFO 空进行处理，不处理 FIFO 满的情况，因此 CoreCLK≥PipeCLK。这种设计的原因是为保证 SWITCHBANDWIDTH≥LINKBANDWIDTH，从而保证 SWITCH 不会限制各端口交叉连接的带宽瓶颈。

当系统 PCIE 只工作在 GEN1 模式时，PCIE SWITCH 的核心工作时钟必须大于 125MHz。当系统 PCIE 要求工作在 GEN2 时，PCIE SWITCH 核心工作频率必须大于 250MHz。

10.2.4. 全局工作时钟配置

通过芯片管脚信号 CFG_GCLKPLL[2:0]，可实现芯片内 PCIE 交叉开关工作时钟频率 100MHz~320MHz 的调整，当配置管脚接为全高 3‘b111 时，还可以通过维护寄存器配置任 意输出频率。锁相环输入参考时钟为 LVDS 电平，输入频率 100MHz。其配置如下：

表 10-9 全局工作时钟配置

配置引脚 CFG_GCLKPLL	CLKR	CLKF	CLKOD	BWADJ	F_{VCO}	PLLOUT
0	1	16	16	8	1600	BYPAS
1	1	24	16	12	2400	150
2	1	32	16	16	3200	200
3	1	30	12	15	3000	250
4	5	168	12	84	3360	280
5	1	30	10	15	3000	300
6	5	160	10	80	3200	320
7	选用寄存器配置方式控制时钟工作频率					

当 CFG_ GCLKPLL [2:0] = 3‘b111 时，可通过寄存器方式对锁相环的输出时钟进行控制，相应寄存器如下表所示。

表 10-10 核心时钟锁相环寄存器控制

寄存器名称	描述
GCLKPLL_Bypass	配置 PLL3 的 Bypass。上电复位置 0。维护时钟域，通
GCLKPLL_CLKR[5:0] NR-1	PLL3 内部参考时钟的分频器控制信号，配置 CLKR。
GCLKPLL_CLKF[12:0] NF-1	PLL3 内部反馈时钟的分频器控制信号，配置 CLKF。
GCLKPLL_CLKOD [3:0] OD-1	PLL3 内部输出时钟的分频器控制信号，配置 CLKOD。
NF/2-1	PLL3 内部输出时钟的滤波带宽控制信号，配置 BWADJ。

注意：

为了支持 PSW 旁路设计，要求 PAB 中的 USP 使用 PSW 给的 pipe_clk，其他逻辑使用 的是 GCLK，而 USP 代码的异步 FIFO 要求 GCLK>pipe_clk，所以保险起见需要将内部的GCLKPLL 配置为 255MHz（通过寄存器配置方式）。

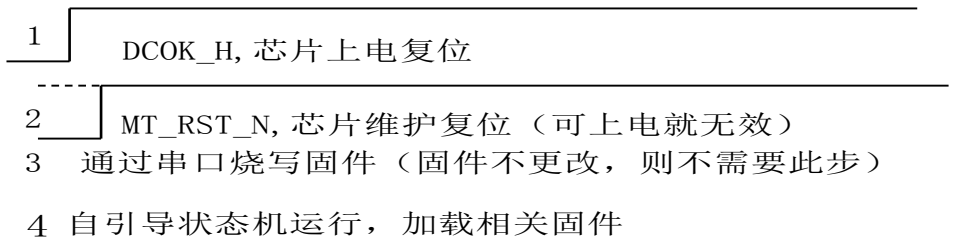
10.3. 使用说明

1) 对于第三方的锁相环，其测试模式优先级最高，旁路优先级次之； 在没有旁路的情况下，PLL 复位无效期间，输出给内部各部件的时钟为 3.125MHz；
2) 正常工作情况下，将 PLL 输出作为电路的工作时钟；
3) 维护时钟是由外部直接输入 100MHz 时钟的 4 分频；
4) PLL 复位时间通过自引导固件进行控制。

10.4. 初始化与测试

10.4.1. 初始化

ICH2 内锁相环的上电、复位初始化流程受芯片自引导固件控制，在初始化上电时需提 前设定 CFG_GPUPLL [2:0]、CFG_MCPLL[2:0]、CFG_SWCLKPLL[2:0]和 CFG_GCLKPLL[2:0]配置值。芯片上电后先对芯片锁相环上电，电源稳定至少 10us 后，将 PLL_PWRDN 置无效（由 1 变 0），对 PLL 进行上电，PLL 上电至少稳定 5us 以上，置 PLL 复位信号 PLL_RESET 信号无效至少 5us，则各 PLL VCO 将输出给定配置下的时钟信号。



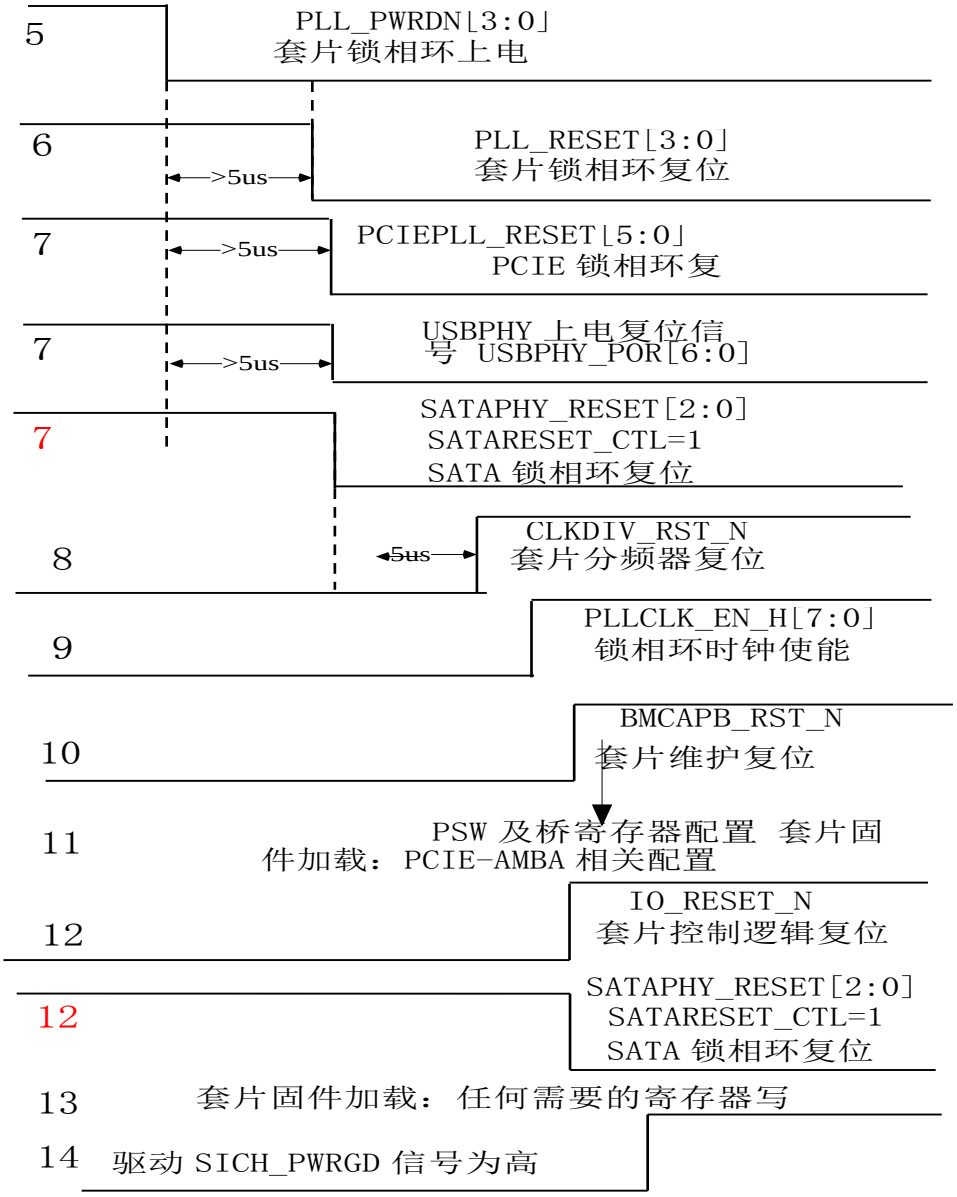


图 10-4 芯片时钟系统初始流程图

芯片内除所采用的 4 个商用 PLL 外，还有 6 个 PCIE 的 PHY、3 个 SATAPHY 及 6 个 USBPHY，这些硬 PHY 内含有锁相环模块用于产生所需时钟信号。芯片的上电流程也必须 地这些硬 PHY 的上电与复位进行控制。具体如下，维持 PCIEPLL_RESET_N[2:0]信号有效 至少 5us 后撤销，维持 SATAPHY_RESET_N[2:0]信号有效至少 5us 后撤销，维持 USB 上电 复位信号 USBPHY_POR[5:0]至少有效 5us 后撤销。10us 后芯片内所有时钟信号稳定，撤销 各分频器复位信号 CLKDIV_RST_N。在这之后，置 PLLCLK_EN_N[4:0]有效，将各功能模 块工作时钟由维护时钟切换至各 PLL 输出时钟。

10.4.2. 测试

ICH2 芯片内部包括多种类型的时钟产生电路，PLL0：GPUCLKPLL； PLL1：MCPLL；PLL2：SWCLKPLL，PLL3：GCLKPLL。ICH2 内集成 8 个 X4 PCIEPHY，有PCIE0~7PHYPLL；

集成 3 个 SATAPHY，有 SATA0~2CLKPLL；集成 6 个 USB nanoPHY，有 USB0~5PHYCLK；集成 DDR3 PHY，有 DDRPHYPLL；包含 PHY 的 PLL，ICH2 内总共有 22 个 PLL。为了保证这些电路能够为系统提供正确、稳定的时钟信号，需要对其工作状态进行监测，监测的内 容包括各锁相环状态 Lock 和输出时钟频率。

根据寄存器 Sel_Watchout[31:0]内容选择输出哪个 PLL 的 LOCK 信号或时钟观测信号。Sel_Watchout[31:0]ICH2 固件内容。具体含义如下：

Sel_Watchout[1:0]选择 SATAOOBCLK 中的某一个输出监测，具体含义如下：

表 10-11 时钟观测选择 1

时钟状态监测寄存器 Sel_Watchout[1:0]	Clk_OUT
0x0	SATA0PHY 50MHz oob 时钟输出
0x1	SATA1PHY 50MHz oob 时钟输出
0x2	SATA2PHY 50MHz oob 时钟输出
0x3	utmi_phy[5]_clock 60MHz 时钟输出

Sel_Watchout[7:4]选择 USBUTMIPHYCLK 中的某一个输出监测，具体含义如下：

表 10-12 时钟观测选择 2

时钟状态监测寄存器 Sel_Watchout[6:4]	Clk_OUT
0x0	usb0_pipe3_pclk 125MHz 监测时钟 4 分频输出
0x1	usb1_pipe3_pclk 125MHz 监测时钟 4 分频输出
0x2	usb2_pipe3_pclk 125MHz 监测时钟 4 分频输出
0x3	usb3_pipe3_pclk 125MHz 监测时钟 4 分频输出
0x4	usb4_pipe3_pclk 125MHz 监测时钟 4 分频输出
0x5	usb5_pipe3_pclk 125MHz 监测时钟 4 分频输出
0x6	utmi_phy[0]_clock 60MHz 监测时钟输出
0x7	utmi_phy[3]_clock 60MHz 监测时钟输出

Clk_Watchout [22:20]选择 PCIEPHYCLK 中的某一个输出监测，具体含义如下：

表 10-13 时钟观测选择 3

时钟状态监测寄存器 Sel_Watchout[10:8]	Clk_OUT
0x0	PCIE_US_PHY0 16 分频输出
0x1	PCIE_US_PHY1 16 分频输出
0x2	PCIE_DS13_PHY0 16 分频输出
0x3	PCIE_DS13_PHY1 16 分频输出
0x4	PCIE_DS24_PHY0 16 分频输出
0x5	PCIE_DS24_PHY1 16 分频输出
0x6	PCIE_DS5_PHY0 16 分频输出
0x7	PCIE_DS5_PHY1 16 分频输出

Sel_Watchout[14:12]选择各锁相环输出时钟分频后的监测时钟，具体含义如下：

表 10-14 时钟观测选择 4

时钟状态监测寄存器 Sel_Watchout[14:12]	OBS_CLK
0x0	GPU 的 PLL 输出 32 分频观测时钟
0x1	MC 的 PLL 输出 16 分频观测时钟
0x2	PCIE SWITCH 的 PLL 输出 16 分频观测时钟

0x3	GCLK 的 PLL 输出 16 分频观测时钟
0x4	MC AC_DTO[1] 32 分频输出
0x5	SATA PHY 50MHz oob 监测时钟输出
0x6	USBPHY 60MHz 监测时钟输出
0x7	PCIEPHY 16 分频输出

对片内 PCIE 及 PAB 相关状态信号，采用 Sel_Watchout[18:16]选择输出

表 10-15 状态观测选择

PCIE 与 PAB 状态信号 Sel_Watchout [18:16]	OBS_PCI
0x0	US DL_UP
0x1	DS1 DL_UP
0x2	DS2 DL_UP
0x3	DS3 DL_UP
0x4	DS4 DL_UP
0x5	DS5 DL_UP
0x6	PAB_HEART_JUMP
0x7	PAB_OUT_ORDER

对片内主锁相环及 PCIE 锁相环，采用 Sel_Watchout[29:20]选择输出

表 10-16 时钟锁定观测选择

时钟状态监测寄存器 Sel_Watchout[29:20]	OBS_LOCK
0	GPUPLL_Lock
1	MCPLL_Lock
2	SWPLL_Lock
3	GCPLL_Lock
4	PCIE_US_PLL0_LOCK
5	PCIE_DS13_PLL0_LOCK
6	PCIE_DS13_PLL1_LOCK
7	PCIE_DS24_PLL0_LOCK
8	PCIE_DS24_PLL1_LOCK
9	PCIE_DS5_PLL0_LOCK

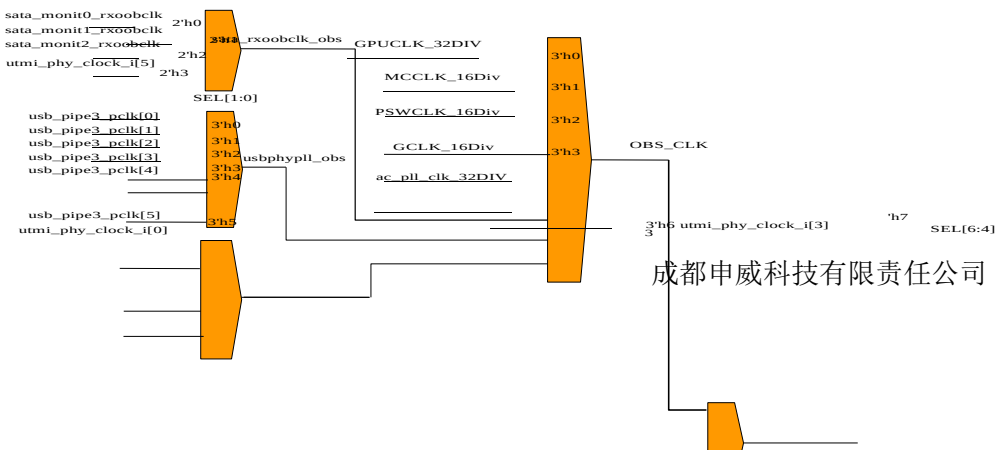
第三级选择器，采用 Sel_Watchout[31:30]选择输出 OBS_CLK、OBS_PCI 或 OBS_LOCK，输出到芯片管脚，供观测使用

表 10-17 观测信号选择

监测选择信号 Sel_Watchout[31:30]	Watch_Out_Pin
0x0	OBS_CLK，选择分频监测时钟输出
0x1	OBS_PCIE，选择 PCIE 监测时钟输出
0x2、0x3	OBS_LOCK，选择 LOCK 信号输出

时钟信号监测分多级选择输出。

第一级分为三个选择器，分别选出各分频时钟给第二级选择器，第三级选择器输出给管脚。



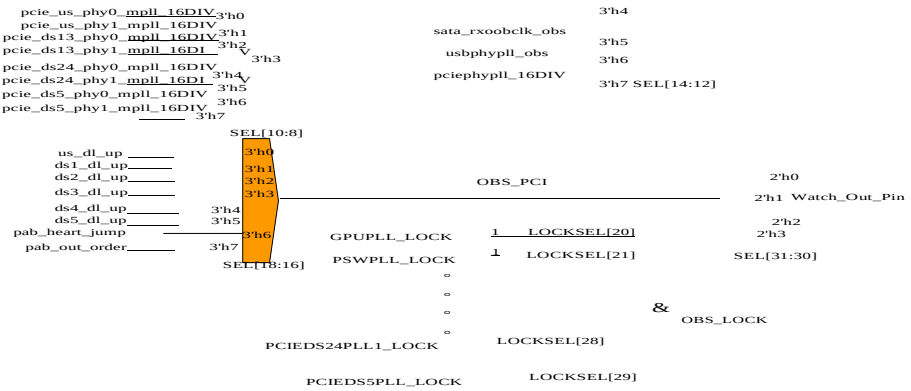


图 10-5 时钟监测信号选择输出

10.5. 交直流电气特性

时钟接口的差分输入时钟（100MHz）为一对 LVDS 电平信号，其它信号均为 1.8V LVCMOS，具体细节参见 12.1 节。

11 测试接口

11.1. 测试接口概述

ICH2 的芯片级测试接口由符合 IEEE1149.1 和 IEEE1149.6 标准规范的边界扫描测试总 线组成，可以实现边界扫描测试和部分 IP 内部的 ATPG 测试和 BIST 测试。整个测试接口 由 ICH2 测试控制部件（简称 TBOX）管理和分配，测试接口信号如表 10-1 所示。整个测试结构包含 8 个符合 IEEE1149.1 标准的 TAP 控制器，其中 7 个集成在不同的部件内，1 个由TBOX 实现，这些 TAP 控制器由 TBOX 统一管理和使用。TBOX 通过 3 位的 JTAG_SEL 输入信号来选择使用哪一个 TAP 控制器，所有部件内的测试功能由部件内的 TAP 控制器实现，这部分的测试原理在本章节不做说明。TBOX 内部 TAP 控制器实现了 PCIEPHY、USBPHY、DDR3PHY 和 SATAPHY 的测试接口管理，以及 GPIO 的边界功能。

表 11-1 ICH2 测试管脚列表

引脚名	类型	数量	含义
TCK	输入	1	JTAG 测试时钟
TMS	输入	1	JTAG 测试方式选择
TRST	输入	1	JTAG 测试复位
TDI	输入	1	JTAG 测试数据输入
TDO	输出	1	JTAG 测试数据输出
JTAG_SEL	输入	3	使用 JTAG 引脚的 TAP 控制器选择： 000： JTAG 端口为 TBOX 内 TAP 控制器使用 001：芯片 JTAG 端口为 WBSYS 使用 010：芯片 JTAG 端口为 PCIEport0TAP 使用 011：芯片 JTAG 端口为 PCIEport1 TAP 使用 100：芯片 JTAG 端口为 PCIE port2 TAP 使用 101：芯片 JTAG 端口为 PCIE port3 TAP 使用 110：芯片 JTAG 端口为 SATA TAP 使用 111：芯片 JTAG 端口为 USB TAP 使用

11.2. 详细说明

ICH2 测试控制部件（简称 TBOX）实现了一个兼容 IEEE1149.1 标准的芯片测试接口。测试控制部件的总体结构如图 11-1 所示：

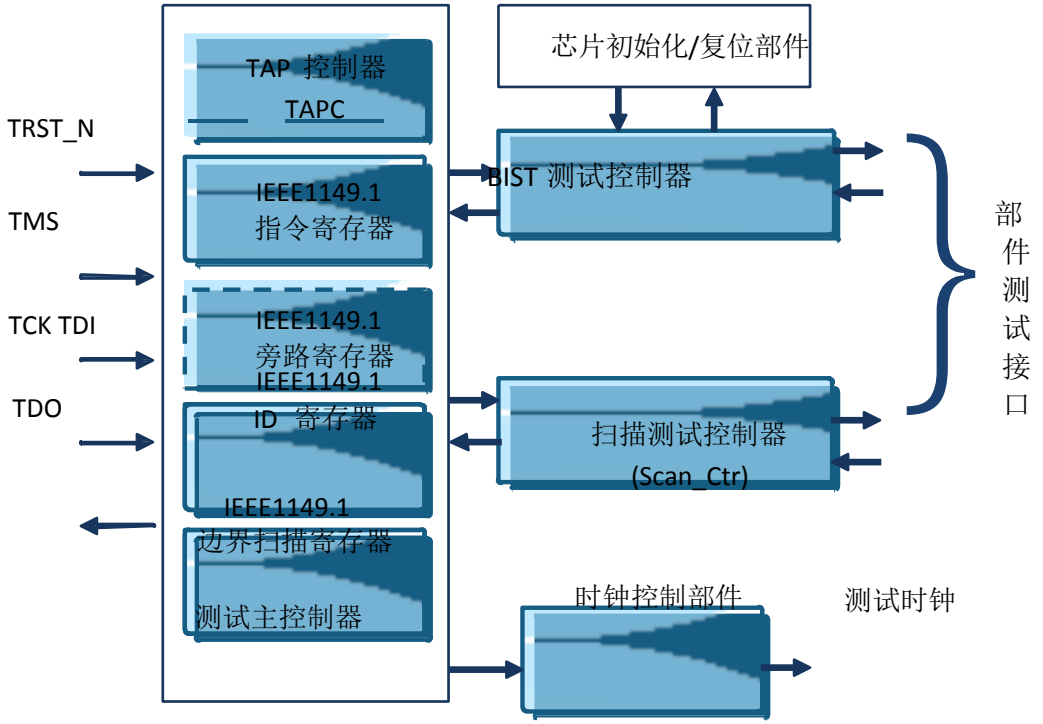


图 11-1 测试控制部件（TBOX）总体结构

ICH2 的测试控制部件包括主控制器、指令寄存器、旁路寄存器、ID（芯片标识号）寄 存器、边界扫描寄存器、BIST 测试控制器、内部扫描测试控制器、时钟控制部件几个部分。其中 TAP 控制器、指令寄存器、旁路寄存器、ID（芯片标识号）寄存器、边界扫描寄存器组成测试主控制器。

BIST 测试控制器实现芯片的内建自测试功能。扫描测试控制器在测试主控制器和部件 测试接口的配合下，实现芯片的内部扫描测试。时钟控制部件与测试主控制器协同配合，生 成内部扫描测试的时钟。

11.2.1. TAP 控制器

TAPC 是一个 16 状态的状态机，与 IEEE1149.1 标准兼容。状态由每个测试时钟上升沿 的 TMS 决定或由异步的低有效复位 TRST_N 决定。如图 11-2 所示。

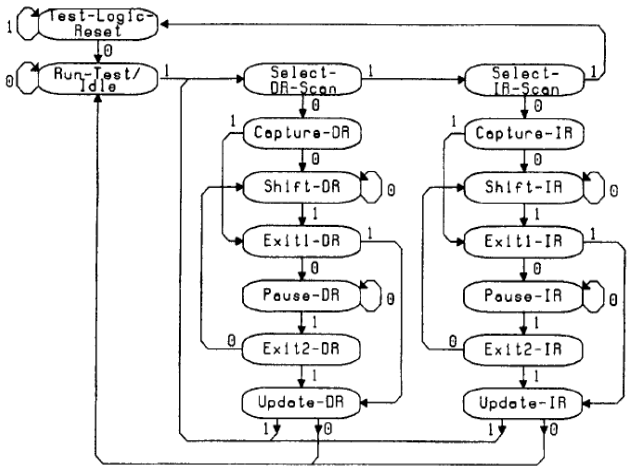


图 11-2 TAPC控制器状态机

TAPC 控制测试控制部件内部的指令寄存器以及所有的测试数据寄存器，包括旁路寄存器、ID 寄存器、边界扫描寄存器、所有的内部扫描链等数据寄存器。

11.2.2. 指令寄存器

TBOX 的指令寄存器是一个具有输出锁存和输入锁存功能的 8 位移位寄存器，除了支持标准强制需要的指令外，还提供用户自定义指令，为 ICH2 提供调试与测试支持，指令类型如表 11-2 所示。

表 11-2 ICH2 测试指令

类	指令名称	指令编码	含义
标准兼容指令	EXTEST	000000	实现芯片在系统中的互连测试
	SAMPLE/PRELO	000001	实现对芯片引脚的采样与预加载
	IDCODE	000010	访问标准中的标志寄存器
	EXTEST_PULSE	000011	AC boundary Scan 脉冲测试指令，用于 SATA、 PCIE AC 测试
	EXTEST_TRAIN	000100	AC boundary Scan 脉冲串测试指令，用于 SATA、 PCIE AC 测试
	INTEST	000101	FOR DDR3 PHY
	RUNBIST	000110	FOR DDR3 PHY
	HIGHZ	000111	将 PCIE、DDR3 端口设置为高阻状态
	CLAMP	001000	将 PCIE、DDR3 端口设置为钳制状态
	BYPASS	111111	芯片处于旁路
自定义	CFGTEST	100000	设置测试配置寄存器，选择相应的测试模式以标志、相应的测试目标以及测试目标内的测试链。
	RUNSCAN	100001	启动扫描测试
	保留	其他编码	

11.2.3. 数据寄存器

11.2.3.1. Bypass 寄存器

Bypass 寄存器是一个单数据位（1bit）的只写寄存器。它为芯片提供从 TDI 到 TDO 之

间最短的串行移位数据通路。在默认情况下，TAP 控制器选择 Bypass 寄存器作为从 TDI 到 TDO 的串行移位数据通路。

11.2.3.2. 器件标志寄存器

器件标志寄存器必须由并行输入和串行输出的 32 位移位寄存器组成，但不需要并行输出 锁存器。标志寄存器的如图：

表 11-3 标志寄存器

31 28	27 12	11 1	0
版本号	器件类型	制造厂标志	1

器件标志寄存器的最低有效位总是装入逻辑_1‘。标志寄存器的第 1 位到第 11 位为制造厂商代码，第 12 到第 27 位为器件类型代码，而第 28 位到 31 位为同一器件不同的版本号。

表 11-4 IDCODE 定义

IDCODE[0]	固定为–1‖
IDCODE[11：1]	制造厂标志固定为–000 0000 0000‖
IDCODE[27：12]	器件类型：ICH2–0000 0000 0000 1001‖
IDCODE[31：28]	版本号：V0.0 –0000‖

11.2.3.3. 边界扫描寄存器

在 ICH2 中，部分 IP 的硬核包括针对某差分信号的特殊。

Boundary-Scan 寄存器就是构成边界扫描链的基本单元。通过边界扫描链，可以进行部件间的连通性测试。当然，更重要的是可以对测试器件的输入输出进行观测和控制，以达到测试器件的内部逻辑的目的。

边界扫描寄存器由一串边界扫描单元组成。这些单元排列成围绕以测试芯片为核心的一条扫描通路上。

11.2.3.4. 测试配置寄存器

该寄存器宽度为 10 位，可读写，功能为设置本次测试的相关配置参数。这些配置参数主要包括：扫描链类型、扫描链子链号等。其格式如下表所示：

表 11-5 测试配置寄存器格式

14	13	12 8	7 4	3 0
INVERT	DC/AC	acjt_level	Chain_Type 链类型	Sub_Chain_Type 子链号

CFG_REG[13] == 1‘b1　AC 模式（FOR PCIE & SATA）
CFG_REG[13] == 1‘b0　DC 模式（FOR PCIE & SATA）

acjt_level 用于设置 SATA 和 USB PHY 的 AC JTAG 的灵敏度级别，默认值为 0
当不进行边界扫描时，acjt_level 设置为 0.

表 11-6 测试配置寄存器定义

Chain_Type	Chain_Type 编码说明	Sub_Chain_Type	Sub_Chain_Type 编码说明
4‘b0000	保留		4‘b0000：复位初值
4‘b0001	边界扫描链	0	PCIE BS 接口
		1	DDR3 BS 接口
		2	SATA BS 接口
		3	USB BS 接口
		4	显示接口/BMC/GMAC/GPIO 接口
		其他	保留
4‘b0010	DDR3 PHY ATPG 扫描链	0	AC 的 Positive 链（211）
		1	AC 的 Negative 链（112）
		2	DATX0-2 的 Positive 链（999）
		3	DATX3-5 的 Positive 链（999）
		4	DATX6-8 的 Positive 链（999）
		5	DATX0-8 的 Negative 链（783）
4‘b0100	USB PHY ATPG 扫描链	5~0	Port[x] 的上升沿扫描链
		6	所有 Port 的下降沿扫描链
		其他	保留
4‘b0101	SATA PHY ATPG 扫描链	0~2	PHY[x]的扫描链
4‘b0111	DDR3 PHY JTAG 链	0	DDR3 PHY JTAG 链
4‘b1000	各个部件内部扫描链	0	PCIE Switch 内部扫描链
		1	PCIE AMBA 桥内部扫描链
		2	DMA 控制器内部扫描链
		3	DDR3 存控内部扫描链
		4	VPU 内部扫描链
		5	GPU 内部扫描链
		6	DC 内部扫描链
		7	USB 主机控制器内部扫描链
		8	SATA 主机控制器内部扫描链
		其他	保留

其他编码		无效，将以 Chain_Type 的最低位为 1，作为子链选择 信号。如果 Chain_Type=4‘b0111，则只选择边界扫描

以上保留的编码，都将选择 gpio 的边界扫描链。

11.2.4. 结构和测试流程

TBOX 的 TAP 控制器通过以上指令管理了各种测试链如图 11-3 所示，实现了以下测试 功能：

- 支持 PCI-E PHY 的 JTAG 测试；（IP 自带的 TAP 控制器）
- 支持 PCI-E PHY 引脚的 BSR 测试；（IP 自带的边界扫描单元，由 TBOX 进行控制）
- 支持 USB PHY 的 JTAG 测试；（IP 自带的 TAP 控制器）
- 支持 USB PHY 引脚的 BSR 测试；（IP 自带的边界扫描单元，由 TBOX 进行控制）
- 支持 USB PHY 的 ATPG 测试；（IP 自带的 ATPG 扫描链，由 TBOX 进行控制）
- 支持 DDR3 PHY 引脚的 BSR 测试；（IP 自带的边界扫描单元，由 TBOX 进行控制）
- 支持 DDR3 PHY 的 ATPG 测试；（IP 自带的 ATPG 扫描链，由 TBOX 进行控制）
- 支持 SATA PHY 的 JTAG 测试；（IP 自带的 TAP 控制器）
- 支持 SATA PHY 引脚的 BSR 测试;（IP 自带的边界扫描单元，由 TBOX 进行控制）
- 支持 SATA PHY 的 ATPG 测试；（IP 自带的 ATPG 扫描链，由 TBOX 进行控制）
- 支持芯片的 GPIO 引脚的 BSR 测试；

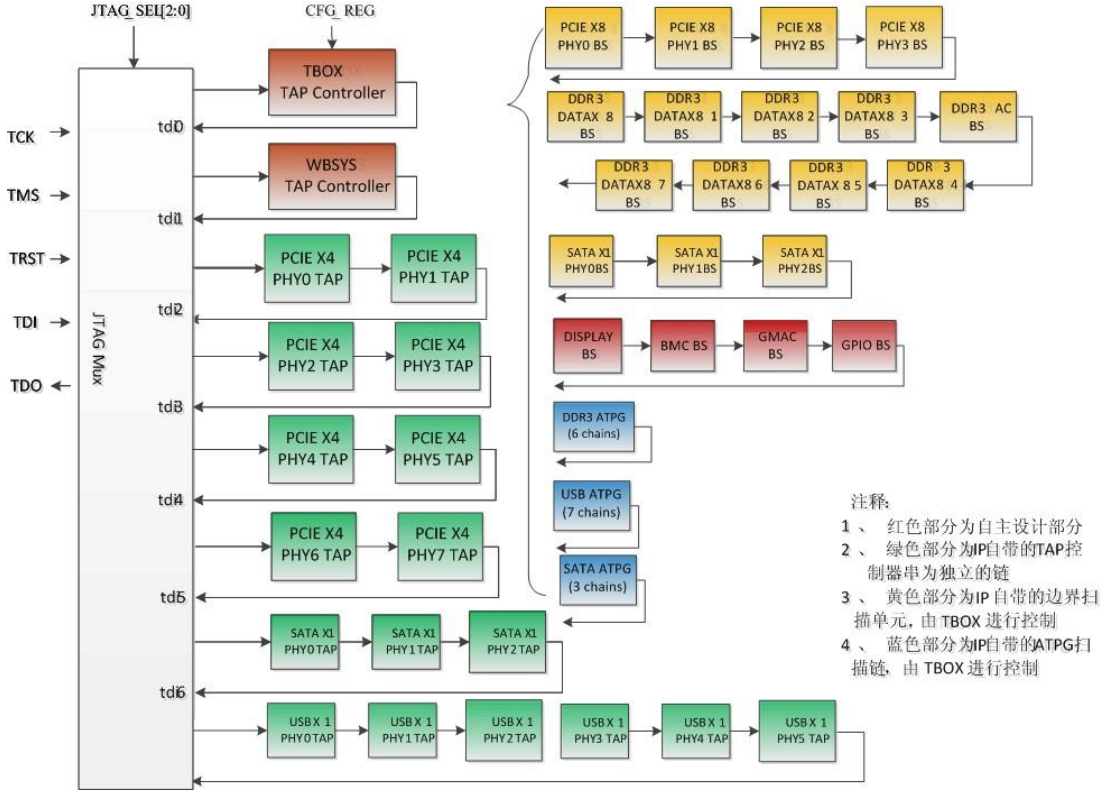


图 11-3 测试链和 TAP 控制器

对于 TBOX 实现的基于 JTAG 接口的测试，不同的测试目标其详细的测试流程各有不 同，

但从总体上来说，具有一个基本的通用测试流程，如下图所示：

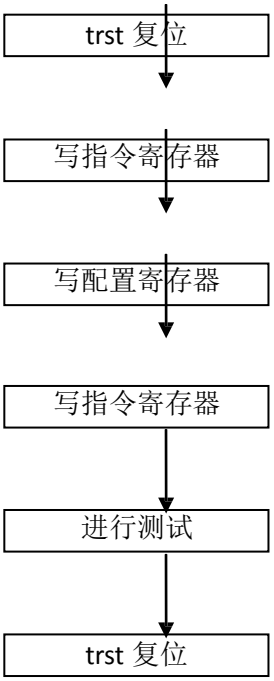


图 11-4 TBOX JTAG 接口测试流程

流程说明：

1、 在进行测试前，先对 JTAG 相关逻辑进行复位，该复位应该包括被测试逻辑。
2、 写指令寄存器，将“设置配置寄存器”指令写入 TBOX 的指令寄存器内。
3、 写配置寄存器。将预定的相关配置内容按一定顺序经 TDI 送入。包括测试标志、 扫描链号等内容。
4、 写指令寄存器，将不同的测试命令写入指令寄存器。一般包括边界扫描命令、ATPG扫描命令等。
5、 通过 TDI 输入相应数据，同时根据 JTAG 的协议由 TDO 获取相应的输出数据。 其详细测试流程根据选择的链或者命令的不同，有不同的处理过程。
6、 完成当前测试后，进行复位（trst）。

11.2.4.1. PCI-E PHY 的边界扫描测试流程

PCI-E PHY 的边界扫描由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。

ICH2 有 4 个 PCI-E 接口，每个 PCI-E 接口包含 2 个 PHY。PCI-E 接口将 8 个 PHY 的 BSR 串为一条链，从 TBOX 的角度看，这个 PCI-E 接口只有一条 BSR 链。 具体步骤：

1) 写指令寄存器为 6‘b100000（CFGTEST 指令）；
2) 配置 CFG_REG 寄存器为 8‘h10，选择 PCI-E 的 BSR 链；
3) 配置指令寄存器是 EXTEST 或者 SAMPLE/PRELOD 指令；
4) 启动数据移位；

11.2.4.2. PCI-E PHY 的 JTAG 访问流程

PCI-E PHY 内嵌的 TAP 控制器实现了与 IEEE1149.1-2001 标准兼容的指令 IDCODE 和 一些自定义的指令。通过 TAP 控制器实现的指令 CRSEL 可以读/写 PCIE2PHY 内部的寄存 器。每个 X4 一套 JTAG 接口；每个 PCI-E 的两个 X4 串在一起；每个 JTAG 接口内嵌 TAP 控制器，TBOX 通过 JTAG_SEL 进行选择 PCIE 的 TAP 控制器。

PCI-E PHY 内部寄存器的 JTAG 访问是通过其自己内部的 TAP 控制器来进行控制的， 所以应设置芯片引脚 JTAG_SEL 信号，选择所对应的 TAP 控制器。对应关系如下：

010: PCI-E port0 TAP；

011: PCI-E port1 TAP；

100: PCI-E port2 TAP；

101: PCI-E port3 TAP。

PCI-E 的 PHY 支持的 JTAG 指令如下：

表 11-7 PCIE PHY 支持的 JTAG 指令

Table 6-1 Supported JTAG Instructions

Instruction	IR	DR	Operation
IDCODE	8'h01	32 bits	Shift out ID code
CRSEL	8'h31	18 bits	Control register access; for information about control register operations, "Control Register Operations," on page 55.
DSCAN	8'h51	3210 bits	MUX-D scan testing
ATEOVRD	8'h5D	5 bits	ATE override
BYPASS	8'hFF	1 bit	TDI to TDO bypass (bit is preloaded to 1)

对于指令 CRSEL，JTAG 接口内实现了两个影子寄存器：Addr_reg、Data_reg，通过这 两个影子寄存器来访问 PCIE PHY 内部的寄存器。Addr_reg 用于存放访问的寄存器地址，Data_reg 用于存放读/写寄存器的数据。通过 18 位的 DR 移位操作可以完成四个操作：Addr_reg 和 Data_reg 访问、寄存器读或写，18 位的移位数据中，其中 16 位为数据或地址， 2 位为操作码 OP。具体的操作如下：

1） 地址（OP=2‘b00）：移位进入的 16 位数据被装入到 Addr_reg 和 Data_reg；

2） 写操作（OP=2‘b01）：移入的 16 位数据被装入到 Data_reg。Addr_reg 对应的寄存器被写入该值；

3） 读操作（OP=2‘b11）：串行移入的 16 位数据被装入到 Addr_reg，CAPTURE-DR 时读出的寄存器数据被锁存到 Data_reg；

4） 空操作（OP=2‘b10）： NOP 操作可以被用于移出 Data_reg 的当前内容。

11.2.4.3. USB PHY 的边界扫描测试流程

USB PHY 的边界扫描由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。 具体步骤：

1) 写指令寄存器为 6‘b100000（CFGTEST 指令）；

2) 配置 CFG_REG 寄存器为 8‘h13，选择 USB 的 BSR 链；

3) 配置指令寄存器为 EXTEST 或者 SAMPLE/PRELOD 指令；

4) 启动数据移位。

11.2.4.4. USB PHY 的 ATPG 测试流程

USB PHY 的 ATPG 测试由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。

USB 3.0 PHY 包含 40 个上升沿触发的链和 1 个下降沿触发的链。

Figure C-1 USB 3.0 PHY Scan Chain Implementation

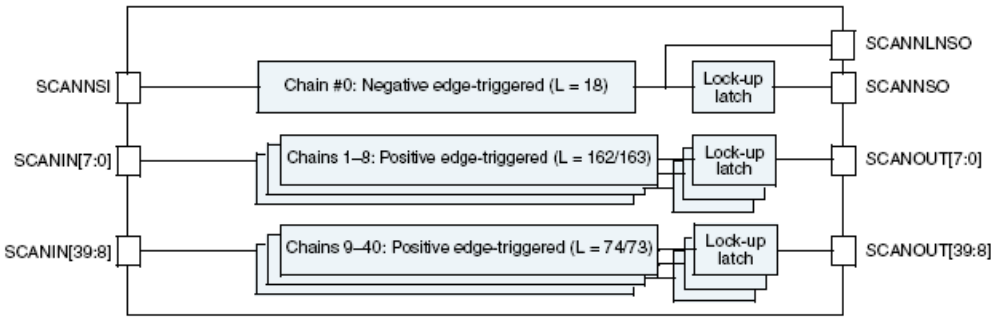


图 11-5 USB 3.0 PHY SCAN链

Forty scan chains are triggered by the positive edge of SCANCLK, and one scan chain is triggered by the negative edge of SCANCLK:

- 3 positive-edge-triggered scan chains: length = 162; with terminal lock-up latch
- 5 positive-edge-triggered scan chains: length = 163; with terminal lock-up latch
- 2 positive-edge-triggered scan chains: length = 74; with terminal lock-up latch
- 30 positive-edge-triggered scan chains: length = 73; with terminal lock-up latch
- 1 negative-edge-triggered scan chain: length = 18; with 2 scan outputs, one with terminal lock-up, SCANNSO, one without terminal lock-up, SCANNLNSO

图 11-6 USB 3.0 PHY SCAN链长

Figure C-3 Final Scan-Stitched Output of ASIC

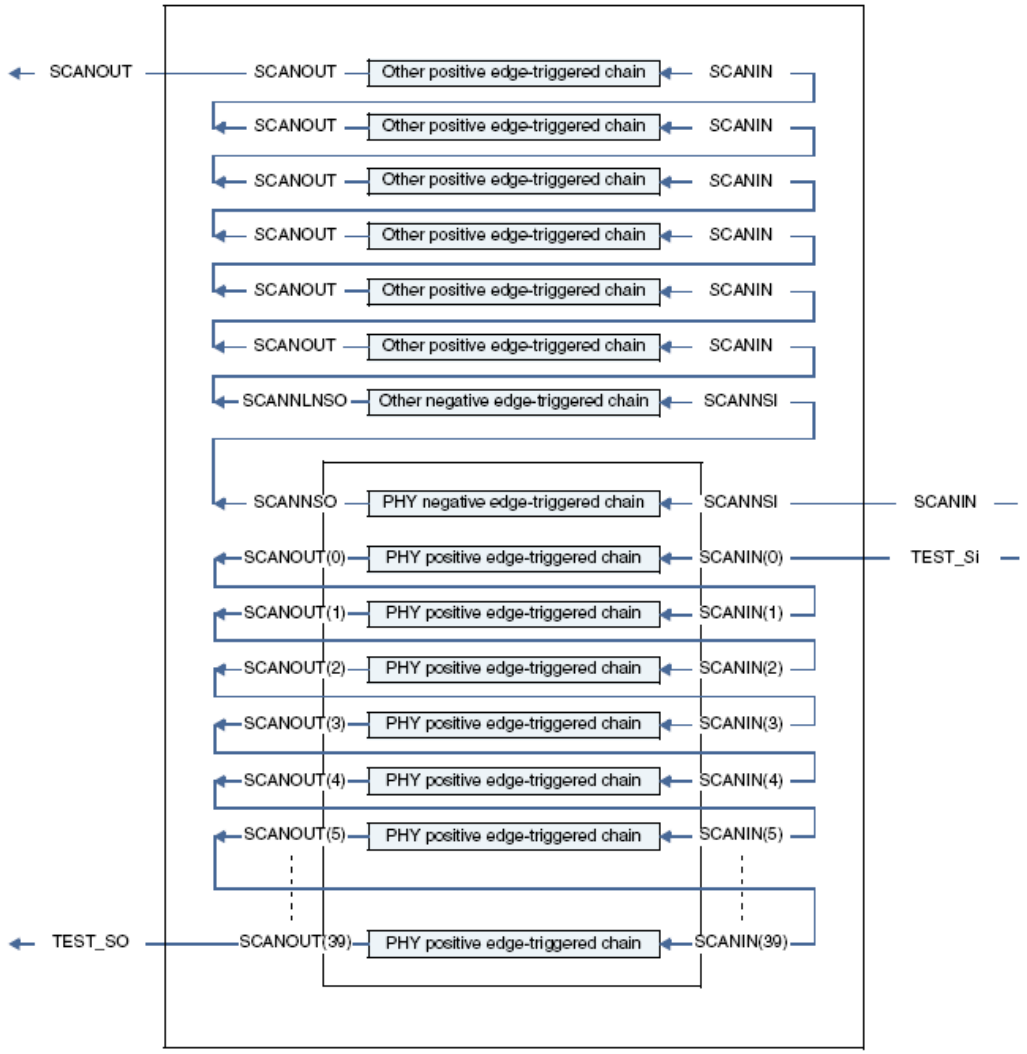


图 11-7 USB 3.0 PHY SCAN链连接关系

具体步骤：

1)写指令寄存器为 6‘b100000（CFGTEST 指令）；；

2)配置 CFG_REG 寄存器，选择 USB 的 ATPG 链；

3)配置指令寄存器为 RUNSCAN 指令；

4)启动数据移位。

11.2.4.5. USB PHY 的 JTAG 访问测试流程

USB PHY 内部寄存器的 JTAG 访问是通过其自己内部的 TAP 控制器来进行控制的，所 以应设置芯片引脚 JTAG_SEL 信号为 3‘b111。

USB 3.0 PHY 内嵌的 TAP 控制器实现了与 IEEE1149.1-2001 标准兼容的指令 IDCODE 和一些自定义的指令。通过 TAP 控制器实现的指令 CRSEL 可以读/写 USB 3.0 PHY 内部的 寄存器。

USB 3.0 PHY 的 TAP 控制器支持 18 位的 CRSEL 寄存器，通过该寄存器可以访问内部 控制寄存器。

表 11-8 CRSEL 寄存器的命令域

Table B-1 CRSEL Register Command

CMD	Encoding	Description
ADDR	2'b00	Captures ADDRESS field for use in a subsequent WRITE command. Shifts current value from most recently accessed register (via READ or WRITE) out on TDO.
WRITE	2'b01	Writes the value in the DATA field to the register selected by the address captured during the most recent ADDR or READ command. Shifts current value from most recently accessed register (via READ or WRITE) out on TDO.
DATA	2'b10	Shifts current value from most recently accessed register (via READ or WRITE) out on TDO. DATA value shifted in is not used.
READ	2'b11	Performs a read operation to the register selected by the ADDRESS field to make its current value available for subsequent CRSEL accesses. Shifts current value from most recently accessed register (via READ or WRITE) out on TDO.

通过对 18 位的 CRSEL 寄存器进行移位（18 位的移位数据中，16 位为数据或地址，2 位为操作码 CMD）可以完成四个操作：

1） 地址（CMD=2‘b00）：移入访问地址；

2） 写操作（CMD=2‘b01）：移入数据并进行写操作。

3） 数据（CMD=2‘b10）：移出读数据，移入数据不使用。

4） 读操作（CMD=2‘b11）：移入地址并进行读操作。 读寄存器过程：

1) 对 CRSEL 寄存器移入一个读操作命令（CMD=2‘b11），同时指定目标地址；

2) 对 CRSEL 寄存器的任何操作（CMD 码任意）都可以把读数据移出。 写寄存器过程：

1) 对 CRSEL 寄存器移入一个地址命令（CMD=2‘b00）指定目标地址，或者对 CRSEL 寄存器移入一个读操作命令（CMD=2‘b11），指定目标地址；

2) 对 CRSEL 寄存器移入一个写操作命令（CMD=2‘b01），同时指定写数据。

11.2.4.6. SATA PHY 的边界扫描测试流程

SATA PHY 的边界扫描由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。 具体步骤：

1)写指令寄存器为 6‘b100000（CFGTEST 指令）；；

2)配置 CFG_REG 寄存器为 8‘h12，选择 SATA 的 BSR 链；

3)配置指令寄存器为 EXTEST 或者 SAMPLE/PRELOD 指令；

4)启动数据移位。

11.2.4.7. SATA PHY 的 ATPG 测试流程

SATA PHY 的 ATPG 测试由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。 具体步骤：

1)写指令寄存器为 6‘b100000（CFGTEST 指令）；；

2)配置 CFG_REG 寄存器，选择 SATA 的 ATPG 链；

3)配置指令寄存器为 RUNSCAN 指令；

4)启动数据移位。

11.2.4.8. SATA PHY 的 JTAG 访问流程

SATA PHY 内部寄存器的 JTAG 访问是通过其自己内部的 TAP 控制器来进行控制的， 所以应设置芯片引脚 JTAG_SEL 信号为 3‘b110。

SATA PHY 核心的 JTAG 接口可访问 PHY 内的 TAP 控制器，该控制器符合 IEEE Std 1149.1-2001 标准，支持 IDCODE, USERCODE 的指令。CRSEL 指令也可以读写 IP 核内的 寄存器，为 ATE 测试等提供手段，不建议在正常操作过程中读写寄存器

三个 SATA PHY 的 JTAG 接口单端串成一条链， SATA 的链通过芯片外引脚上的 JTAG_SEL 选择，单独接一个链。SATA PHY 的 JTAG 访问与 USB PHY 的访问一致，可参考 USB PHY 的 JTAG 访问流程。

11.2.4.9. DDR3 PHY 的边界扫描测试流程

SATA PHY 的边界扫描由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。 具体步骤：

1) 写指令寄存器（CFGTEST 指令）；
2) 配置 CFG_REG 寄存器选择 DDR3PHY 的 BSR 链；
3) 配置指令寄存器是 EXTEST 或者 SAMPLE/PRELOD 指令；
4) 启动数据移位。

11.2.4.10. DDR3 PHY 的 ATPG 测试流程

DDR3 PHY 的 ATPG 测试由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。

DDR3 PHY 的 ATPG 链接说明如表所示。

表 11-9 DDR3PHY IO ATPG 链接说明

TBOX 链号	连接关系（按顺序）	链长	备注
0	AC 的 Positive 链： 链 1->链 2->链 5->链 6	211	
1	AC 的 Negative 链： 链 3->链 4	112	
2	DATX0-2 的 Positive 链： DATX0:链 1->链 2->链 4->链 5->链 6 DATX1:链 1->链 2->链 4->链 5->链 6 DATX2:链 1->链 2->链 4->链 5->链 6	333*3	
3	DATX3-5 的 Positive 链： DATX3:链 1->链 2->链 4->链 5->链 6	333*3	

	DATX4:链 1->链 2->链 4->链 5->链 6 DATX5:链 1->链 2->链 4->链 5->链 6		
4	DATX6-8 的 Positive 链： DATX6:链 1->链 2->链 4->链 5->链 6 DATX7:链 1->链 2->链 4->链	333*2	
5	DATX0-8 的 Negative 链(链 3)： DATX0-> DATX1->……->DATX7	87*8	
6~7	保留		

说明：

1) 连接顺序中的-链 xⅡ是指 AC/DATX8 链说明（图 11-3）中 Scan Chain；
2) DDR3PHY_ALL 层对底层的链进行如上连接；
3) TBOX 看来 DDR3PHY_ALL 共有 6 个链； 具体步骤：
1) 写指令寄存器（CFGTEST 指令）；
2) 配置 CFG_REG 寄存器选择 DDR3PHY 的 ATPG 链；
3) 配置指令寄存器是 RUNSCAN 指令；
4) 启动数据移位。

11.2.4.11. DDR3 PHY 的 JTAG 访问流程

DDR3 PHY 内部寄存器的 JTAG 访问是通过 TBOX 的 TAP 控制器来进行控制的，所以 应设置芯片引脚 JTAG_SEL 信号为 3‘b000。

JTAG 接口数据寄存器格式如表 11-10 所示。

表 11-10 JTAG 接口数据寄存器格式

Bits	Name	Description
[1:0]	INSTR	Instruction: Selects the instruction to be performed on the configuration port. Valid values are: 00 = Reserved. No action on the configuration port 01 = Load configuration register address and execute a register read operation 10 = Load configuration register address and data and execute a register write operation 11 = Reserved. No action on the configuration port
[9:2]	ADDR	Address: Address for the register to be read or written.
[41:10]	DATA	Data: Data to be written to a register or read from a register.

具体步骤：

1) 写指令寄存器（CFGTEST 指令）；
2) 配置 CFG_REG 寄存器选择 DDR3PHY 的 JTAG 链；
3) 执行写动作，则对 JTAG 接口数据寄存器移入相应数据（INSTR 为 10，ADDR 和 DATA 相应的地址和写数据；执行读动作，则对 JTAG 接口数据寄存器移入相应数据（INSTR 为 01，ADDR 相应的地址）；

11.2.4.12. GPIO PHY 的边界扫描测试流程

GPIO 的边界扫描由 TBOX 里的 TAP 控制器来控制，所以应设置芯片引脚 JTAG_SEL 信号为 3‘b000。

GPIO 的扫描链顺序如表 11-11 所示：

表 11-11 GPIO 扫描链

引脚名	IOBUF	属性	iobuf 顺序号	每个单元	
GBUF_I_DS1_SENS_PRES_DET_STATE	PDDDG	in10	21	1	
GBUF_I_DS1_SET_PWR_FAULT_DETECTE D	PDDDGZ	in9	218	1	
GBUF_I_DS1_SET_MRL_SENS_CHANGE	PDDDG	in8	21	1	
GBUF_I_DS1_ATTN_BTN_PRESD	PDDDG	in7	21	1	
GBUF_I_DS1_SENS_MRL_SENS_STATE	PDDDG	in6	21	1	
GBUF_O_DS1_PWR_INDICATOR_CNTRL[0]	PDO08CD	out10	214	1	
GBUF_O_DS1_PRES_DET_CLEAR	PDO08CD	out9	213	1	
GBUF_O_DS1_POWER_CONTROLLER	PDO08CD	out8	212	1	
GBUF_O_DS1_PWR_INDICATOR_CNTRL[1]	PDO08CD	out7	211	1	
GBUF_O_DS1_ATTN_INDICATOR_CNTRL[1]	PDO08CD	out6	210	1	
GBUF_O_DS1_ATTN_INDICATOR_CNTRL[0]	PDO08CD	out5	209	1	
GBUF_I_UART_HDPLX	PDUDG	in5	20	1	
GBUF_I_UART_RX	PDUDG	in4	20	1	
GBUF_I_UART_RI	PDDDG	in3	20	1	
GBUF_I_UART_DSR	PDDDG	in2	20	1	
GBUF_I_UART_CTS	PDDDG	in1	20	1	
GBUF_I_UART_DCD	PDDDG	in0	20	1	
GBUF_O_WATCH_OUT	PDO08CD	out4	202	1	
GBUF_O_UART_TX	PDO08CD	out3	201	1	
GBUF_O_CPU_LINKUP	PDO08CD	out2	200	1	
GBUF_O_UART_RTS	PDO08CD	out1	199	1	
GBUF_O_UART_DTR	PDO08CD	out0	198	1	22
GBUF_O_USB_VBUS_F_CTRL	PDO08CD	out84	197	1	
GBUF_I_USB_OVRCUR_F_N	PDUDG	in37	19	1	

GBUF_I_GMAC1_MII_TX_CLK	PDUDG	in36	19	1
GBUF_O_USB_VBUS_CTRL	PDO08C	out8	19	1

	G			
GBUF_I_GMAC0_MII_TX_CLK	PDUDG	in35	193	1
GBUF_IO_KDAT	PDU08DG	inout	192	2
GBUF_IO_MDAT	PDU08DG	inout	191	2
GBUF_IO_MCLK	PDU08DG	inout	190	2
GBUF_IO_KCLK	PDU08DG	inout9	189	2
GBUF_O_SPI_CLK	PDO08CD	out82	188	1
GBUF_O_SPI_CS_N	PDO08CD	out81	187	1
GBUF_I_FLASH_ADDRSEL	PDUDG	in34	186	1
GBUF_I_SPI_RXD	PDUDG	in33	185	1
GBUF_O_SPI_TXD	PDO08CD	out80	184	1
GBUF_IO_SDA	PDU08DG	inout8	183	2
GBUF_IO_IIC2_CLK	PDU12DG	inout7	182	2
GBUF_IO_IIC2_DATA	PDU12DG	inout6	181	2
GBUF_IO_IIC1_CLK	PDU12DG	inout5	180	2
GBUF_I_SCLK	PDUDG	in32	179	1
GBUF_IO_IIC0_CLK	PDU12DG	inout4	178	2
GBUF_IO_IIC0_DATA	PDU12DG	inout3	177	2
GBUF_IO_IIC1_DATA	PDU12DG	inout2	176	2
GBUF_O_DC2RGB_DISPLAY1BLUE[7]	PDO12CD	out79	175	1
GBUF_O_DC2RGB_DISPLAY1BLUE[3]	PDO12CD	out78	174	1
GBUF_O_DC2RGB_DISPLAY1BLUE[4]	PDO12CD	out77	173	1
GBUF_O_DC2RGB_DISPLAY1BLUE[5]	PDO12CD	out76	172	1
GBUF_O_DC2RGB_DISPLAY1BLUE[6]	PDO12CD	out75	171	1
GBUF_O_DC2RGB_DISPLAY1BLUE[2]	PDO12CD	out74	170	1
GBUF_O_DC2RGB_DISPLAY1BLUE[1]	PDO12CD	out73	169	1
GBUF_O_DC2RGB_DISPLAY1GREEN[7]	PDO12CD	out72	168	1
GBUF_O_DC2RGB_DISPLAY1BLUE[0]	PDO12CD	out71	167	1

GBUF_O_DC2RGB_DISPLAY1GREEN[6]	PDO12CD	out70	166	1
GBUF_O_DC2RGB_DISPLAY1GREEN[2]	PDO12CD	out69	165	1

GBUF_O_DC2RGB_DISPLAY1GREEN[3]	PDO12CD	out68	164	1
GBUF_O_DC2RGB_DISPLAY1GREEN[4]	PDO12CD	out67	163	1
GBUF_O_DC2RGB_DISPLAY1GREEN[5]	PDO12CD	out66	162	1
GBUF_O_DC2RGB_DISPLAY1GREEN[1]	PDO12CD	out65	161	1
GBUF_O_DC2RGB_DISPLAY1GREEN[0]	PDO12CD	out64	160	1
GBUF_O_DC2RGB_DISPLAY1RED[5]	PDO12CD	out63	159	1
GBUF_O_DC2RGB_DISPLAY1RED[6]	PDO12CD	out62	158	1
GBUF_O_DC2RGB_DISPLAY1RED[7]	PDO12CD	out61	157	1
GBUF_O_DC2RGB_DISPLAY1RED[4]	PDO12CD	out60	156	1
GBUF_O_DC2RGB_DISPLAY1RED[0]	PDO12CD	out59	155	1
GBUF_O_DC2RGB_DISPLAY1RED[1]	PDO12CD	out58	154	1
GBUF_O_DC2RGB_DISPLAY1RED[2]	PDO12CD	out57	153	1
GBUF_O_DC2RGB_DISPLAY1RED[3]	PDO12CD	out56	152	1
GBUF_O_DC2RGB_DISPLAY1VSYNC	PDO12CD	out55	151	1
GBUF_O_DC2RGB_DISPLAY0BLUE[7]	PDO12CD	out54	150	1
GBUF_O_DC2RGB_DISPLAY1HSYNC	PDO12CD	out53	149	1
GBUF_O_DC2RGB_DISPLAY0BLUE[6]	PDO12CD	out52	148	1
GBUF_O_DC2RGB_DISPLAY0BLUE[5]	PDO12CD	out51	147	1
GBUF_O_DC2RGB_DISPLAY1CLK	PDO24CD	out50	146	1
GBUF_O_DC2RGB_DISPLAY1EN	PDO12CD	out49	145	1
GBUF_O_DC2RGB_DISPLAY0BLUE[4]	PDO12CD	out48	144	1
GBUF_O_DC2RGB_DISPLAY0BLUE[3]	PDO12CD	out47	143	1
GBUF_O_DC2RGB_DISPLAY0CLK	PDO24CD	out46	142	1
GBUF_O_DC2RGB_DISPLAY0BLUE[2]	PDO12CD	out45	141	1
GBUF_O_DC2RGB_DISPLAY0EN	PDO12CD	out44	140	1
GBUF_O_DC2RGB_DISPLAY0GREEN[7]	PDO12CD	out43	139	1
GBUF_O_DC2RGB_DISPLAY0BLUE[0]	PDO12CD	out42	138	1
GBUF_O_DC2RGB_DISPLAY0BLUE[1]	PDO12CD	out41		

GBUF_O_DC2RGB_DISPLAY0GREEN[6]	PDO12CD	out40	136	1
GBUF_O_DC2RGB_DISPLAY0GREEN[2]	PDO12CD	out39	135	1
GBUF_O_DC2RGB_DISPLAY0GREEN[3]	PDO12CD	out38	134	1
GBUF_O_DC2RGB_DISPLAY0GREEN[4]	PDO12CD	out37	133	1
GBUF_O_DC2RGB_DISPLAY0GREEN[5]	PDO12CD	out36	132	1
GBUF_O_DC2RGB_DISPLAY0GREEN[1]	PDO12CD	out35	131	1
GBUF_O_DC2RGB_DISPLAY0GREEN[0]	PDO12CD	out34	130	1
GBUF_O_DC2RGB_DISPLAY0RED[6]	PDO12CD	out33	129	1
GBUF_O_DC2RGB_DISPLAY0RED[7]	PDO12CD	out32	128	1
GBUF_O_DC2RGB_DISPLAY0RED[5]	PDO12CD	out31	127	1
GBUF_O_DC2RGB_DISPLAY0RED[1]	PDO12CD	out30	126	1
GBUF_O_DC2RGB_DISPLAY0RED[2]	PDO12CD	out29	125	1
GBUF_O_DC2RGB_DISPLAY0RED[3]	PDO12CD	out28	124	1
GBUF_O_DC2RGB_DISPLAY0RED[4]	PDO12CD	out27	123	1
GBUF_O_DC2RGB_DISPLAY0RED[0]	PDO12CD	out26	122	1
GBUF_O_DC2RGB_DISPLAY0VSYNC	PDO12CD	out25	121	1
GBUF_I_DISPLAY02PAB_INT	PDDDG	in31	12	1
GBUF_I_DISPLAY12PAB_INT	PDDDG	in30	110	1
GBUF_O_DC2RGB_DISPLAY0HSYNC	PDO12CD	out24	118	1
GBUF_O_GMAC1_PHY_TXD[7]	PDO12CD	out23	117	1
GBUF_O_GMAC1_PHY_TXD[5]	PDO12CD	out22	116	1
GBUF_O_GMAC1_PHY_TXD[6]	PDO12CD	out21	115	1
GBUF_O_GMAC1_PHY_TXD[4]	PDO12CD	out20	114	1
GBUF_O_GMAC1_PHY_TXD[0]	PDO12CD	out19	113	1
GBUF_O_GMAC1_PHY_TXD[1]	PDO12CD	out18	112	1
GBUF_O_GMAC1_PHY_TXD[2]	PDO12CD	out17	111	1
GBUF_O_GMAC1_PHY_TXD[3]	PDO12CD	out16	110	1
GBUF_I_GMAC1_PHY_RXD[7]	PDUDGZ	in29	10	1
GBUF_I_GMAC1_PHY_RXD[6]	PDUDGZ	in28	10	1
GBUF_I_GMAC1_PHY_RXD[4]	PDUDGZ	in27	108	

GBUF_I_GMAC1_PHY_RXD[5]	PDUDG	in26	106	1
GBUF_I_GMAC1_PHY_RXD[3]	PDUDG	in25	105	1
GBUF_I_GMAC1_PHY_RXD[2]	PDUDG	in24	104	1
GBUF_O_GMAC1_GMII_MDC	PDO12CD	out15	103	1
GBUF_IO_GMAC1_GMII_MDIO	PDU12DG	inout1	102	2
GBUF_I_GMAC1_PHY_RXD[0]	PDUDG	in23	101	1
GBUF_I_GMAC1_PHY_RXD[1]	PDUDG	in22	100	1
GBUF_I_GMAC1_PHY_COL	PDDDG	in21	99	1
GBUF_I_GMAC1_PHY_CRS	PDDDG	in20	98	1
GBUF_I_GMAC1_PHY_RXER	PDDDG	in19	97	1
GBUF_I_GMAC1_PHY_RXDV	PDDDG	in18	96	1
GBUF_O_GMAC1_PHY_TXEN	PDO12CD	out14	95	1
GBUF_I_GMAC1_CLK_RX	PDUDG	in17	94	1
GBUF_O_GMAC1_PHY_TXER	PDO12CD	out13	93	1
GBUF_I_GMAC1_CLK_OSC	PDUDG	in16	92	1
GBUF_I_GMAC1_PHY_INTF_SEL	PDUDG	in15	91	1
GBUF_O_GMAC1_CLK_TX	PDO12CD	out12	90	1
GBUF_O_GMAC0_PHY_TXD[6]	PDO12CD	out11	89	1
GBUF_O_GMAC0_PHY_TXD[7]	PDO12CD	out10	88	1
GBUF_I_GMAC0_CLK_RX	PDUDG	in14	87	1
GBUF_O_GMAC0_PHY_TXD[5]	PDO12CD	out9	86	1
GBUF_O_GMAC0_CLK_TX	PDO12CD	out8	85	1
GBUF_O_GMAC0_PHY_TXD[4]	PDO12CD	out7	84	1
GBUF_O_GMAC0_PHY_TXD[1]	PDO12CD	out6	83	1
GBUF_O_GMAC0_PHY_TXD[2]	PDO12CD	out5	82	1
GBUF_O_GMAC0_PHY_TXD[3]	PDO12CD	out4	81	1
GBUF_O_GMAC0_PHY_TXD[0]	PDO12CD	out3	80	1
GBUF_I_GMAC0_PHY_RXD[4]	PDUDG	in13	79	1
GBUF_I_GMAC0_PHY_RXD[5]	PDUDG	in12	78	1
GBUF_I_GMAC0_PHY_RXD[6]	PDUDG	in11	77	1
GBUF_I_GMAC0_PHY_RXD[7]	PDUDG	in10	76	1
GBUF_I_GMAC0_PHY_RXD[3]	PDUDG	in9	75	1
GBUF_O_GMAC0_PHY_TXER	PDO12CD	out2	74	1
GBUF_I_GMAC0_PHY_RXD[1]	PDUDG	in8	73	1
GBUF_I_GMAC0_PHY_RXD[2]	PDUDG	in7	72	1
GBUF_I_GMAC0_PHY_RXD[0]	PDUDG	in6	71	1

GBUF_IO_GMAC0_GMII_MDIO	PDU12 DG	inout 0	70	2

GBUF_I_GMAC0_PHY_RXER	PDDDG	in5	69	1	
GBUF_I_GMAC0_PHY_RXDV	PDDDG	in4	68	1	
GBUF_O_GMAC0_GMII_MDC	PDO12CD	out1	67	1	
GBUF_O_GMAC0_PHY_TXEN	PDO12CD	out0	66	1	
GBUF_I_GMAC0_PHY_COL	PDDDG	in3	65	1	
GBUF_I_GMAC0_CLK_OSC	PDUDGZ	in2	64	1	
GBUF_I_GMAC0_PHY_INTF_SEL	PDUDGZ	in1	63	1	
GBUF_I_GMAC0_PHY_CRS	PDDDGZ	in0	62	1	
GBUF_O_AC97_SDATA_O_PAD	PDO08CD	out11	61	1	147
GBUF_O_AC97_SYNC_PAD	PDO08CD	out10	60	1	
GBUF_I_AC97_SDATA_I_PAD	PDUDGZ	in27	59	1	
GBUF_O_LFRAMEN	PDO12CD	out9	58	1	
GBUF_O_AC97_RESET_PAD_ON	PDO08CD	out8	57	1	
GBUF_IO_LAD[2]	PDU12DG	inou	56	2	
GBUF_IO_LAD[3]	PDU12DG	inou	55	2	
GBUF_IO_LAD[1]	PDU12DG	inou	54	2	
GBUF_IO_SERIRQ	PDU12DG	inou	53	2	
GBUF_IO_LAD[0]	PDU12DG	inou	52	2	
GBUF_I_AC97_BIT_CLK_PAD	PDUDGZ	in26	51	1	
GBUF_I_LDRQ1N	PDUDGZ	in25	50	1	
GBUF_I_LDRQ2N	PDUDGZ	in24	49	1	
GBUF_I_CFG_SWCLKPLL[2]	PDUDGZ	in23	48	1	
GBUF_O_LRSTN	PDO12CD	out7	47	1	
GBUF_I_CFG_SWCLKPLL[0]	PDUDGZ	in22	46	1	
GBUF_I_CFG_SWCLKPLL[1]	PDUDGZ	in21	45	1	
GBUF_I_DC0CLK	PDUDGZ	in20	44	1	
GBUF_I_CFG_MCPLL[2]	PDUDGZ	in19	43	1	
GBUF_I_DC1CLK	PDUDGZ	in18	42	1	
GBUF_I_CFG_MCPLL[0]	PDUDGZ	in17	41	1	
GBUF_I_CFG_MCPLL[1]	PDUDGZ	in16	40	1	
GBUF_I_LCLK	PDUDGZ	in15	39	1	
GBUF_O_PWRGD	PDO08CD	out6	38	1	
GBUF_I_CFG_GCLKPLL[2]	PDUDGZ	in14	37	1	
GBUF_I_CFG_GCLKPLL[0]	PDUDGZ	in13	36	1	
GBUF_I_CFG_GCLKPLL[1]	PDUDGZ	in12	35	1	
GBUF_I_CFG_GPUPLL[2]	PDUDGZ	in11	34	1	
GBUF_I_CFG_GPUPLL[0]	PDUDGZ	in10	33	1	
GBUF_I_CFG_GPUPLL[1]	PDUDGZ	in9	32	1	
GBUF_I_MTRST_N	PDUDGZ	in8	31	1	

GBUF_I_DCOK_H	PDDDG	in7	30	1
GBUF_O_SATA_AHCI2_ACT_LED_N	PDO08CD	out5	29	1
GBUF_O_SATA_AHCI0_CP_POD	PDO08CD	out4	28	1
GBUF_O_SATA_AHCI1_ACT_LED_N	PDO08CD	out3	27	1
GBUF_O_SATA_AHCI1_CP_POD	PDO08CD	out2	26	1
GBUF_O_SATA_AHCI0_ACT_LED_N	PDO08CD	out1	25	1
GBUF_I_SATA_AHCI1_MP_SWITCH	PDDDG	in6	2	1
GBUF_I_SATA_AHCI1_CP_DET	PDDDG	in5	2	1
GBUF_I_SATA_AHCI2_MP_SWITCH	PDDDG	in4	2	1
GBUF_I_SATA_AHCI2_CP_DET	PDDDG	in3	2	1
GBUF_I_SATA_AHCI0_CP_DET	PDDDG	in2	2	1
GBUF_I_SATA_AHCI0_MP_SWITCH	PDDDG	in1	10	1
GBUF_O_SATA_AHCI2_CP_POD	PDO08CD	out0	18	1
GBUF_IO_PORTA[6]	PDU12DG	inou	17	2
GBUF_IO_PORTA[7]	PDU12DG	inou	16	2
GBUF_I_USB_OVRCUR_N	PDUDGZ	in0	15	1
GBUF_IO_PORTA[5]	PDU12DG	inou	14	2
GBUF_IO_PORTA[4]	PDU12DG	inou	13	2
GBUF_IO_PORTA[1]	PDU12DG	inou	12	2
GBUF_IO_PORTA[2]	PDU12DG	inou	11	2
GBUF_IO_PORTA[3]	PDU12DG	inout9	10	2
GBUF_IO_PORTB[5]	PDU12DG	inout8	9	2
GBUF_IO_PORTB[6]	PDU12DG	inout7	8	2
GBUF_IO_PORTB[7]	PDU12DG	inout6	7	2
GBUF_IO_PORTA[0]	PDU12DG	inout5	6	2
GBUF_IO_PORTB[4]	PDU12DG	inout4	5	2
GBUF_IO_PORTB[3]	PDU12DG	inout3	4	2
GBUF_IO_PORTB[1]	PDU12DG	inout2	3	2
GBUF_IO_PORTB[2]	PDU12DG	inout1	2	2
GBUF_IO_PORTB[0]	PDU12DG	inout0	1	2

总长	283

11.3. 交直流电气特性

测试接口使用标准 LVCMOS 1.8V 供电的接口缓冲，其电气特性参见 12.1 节。

12 电气特性

12.1. 电气特性概述

在 ICH2 芯片中， DDR3 PHY 接口使用 1.5V SSTL 电平，USB PHY 接口、SATA PHY 接口和 PCI-E 接口使用 LVDS 电平，其他的管脚均采用了 1.8V LVCMOS，其中 1.8V 与 2.5V LVCMOS IO 缓冲是同一种器件，通过配置管脚实现不同 IO 接口电平支持。其工作范围及 工作条件如下：

表 12-1 1.8V/2.5V/3.3V LVCOMS 工作范围

参数	描述	工作范围	单位
V_{VDD}	核心供电电源电压范围	-0.5~1.32	V
V_{DVDD}	I/O 缓冲电源电压范围	-0.5～3.6	V
V_{PAD}	I/O PAD 上的电压范围	-0.5～V_{DVDD}+0.5	V
T_J	节温范围	-40～125	℃

表 12-2 1.8V/2.5V/3.3V LVCOMS 推荐的工作条件

参数	描述	最小值	正常值	最大值	单位
V_{VDD}	核心供电电压	0.9	1.0	1.1	V
V_{DVDD}	I/O 缓冲电源电压	1.62	1.8	1.98	V
		2.25	2.5	2.75	V
		2.97	3.3	3.63	V
T_J	节温范围	-40	25	+125	℃
V_{PAD}	I/O PAD 上的电压	-0.3		V_{DVDD}+0.3	V
V_{IH}	输入高电平	0.7* V_{DVDD}		V_{DVDD}+0.3	V
V_{IL}	输入地电平	V_{DVDD}-0.3		0.3 *V_{DVDD}	V
V_{HYS}	输入滞后电压	0.4			V

IOBUF 的上下拉和输出驱动能力见 1.3 节说明。

12.2. 电地说明

12.2.1. DDR3 接口

DDR3PHY 的模拟电地管脚如下图所示：

表 12-3 DDR3PHY 模拟电/地列表

名称	位	IO	描述

DDR3_VDD10A	1	双向	PLL core 电压 1.0V
DDR3_VSS10A	1	双向	PLL core 电压 1.0V 对应的地
DDR3_VDD25A	1	双向	PLL2.5V 电压
DDR3_VSS25A	1	双向	PLL VSS 对应 2.5V 电压（模拟地）
DDR3_PVT0~2	3	双向	外部端接高精度电阻 240 欧姆，用于 PVT 补偿。在布局时，要求每 5mm 加一个补
DDR3_ATO	1	输出	测试模拟信号，用于观察存储器 PHY 内部锁相环的测试输出

说明：

DDR3_VDD10A 如果要与芯片的内核电压 VDD 共用，则需要通过电感隔离。在芯片的 封装设计中，DDR3_VDD10A 出芯片管脚，板级设计中通过电感隔离与内核电压 VDD 合用。

此外，为了保证 Synopsys DDR3PHY 中的 PLL 稳定的工作，建议在连接 PLL_VDD25 时增加一个滤波电路。

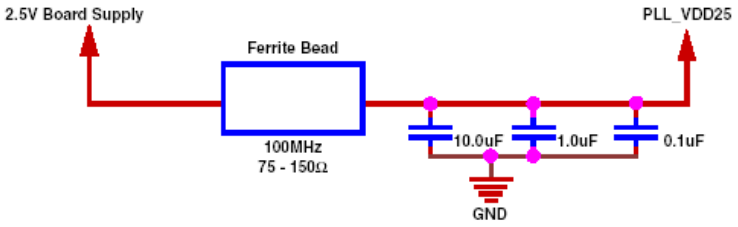


图 12-1 电源滤波电路图

DDR3PHY 的数字电地管脚如下图所示：

表 12-4 DDR3PHY 数字电源/地列表

名称	位	IO	描述
PHY_VDD	1	双向	PHY core 电压 1.0V
VDDQ15	29	双向	PHY SSTL 供电电压 1.5V
DDR3_VREF0~4	5	双向	PHY SSTL 参考电压,为 VDDQ15 一半

12.2.2. PCI-E 接口

PCI-E 接口内部包含 SerDes 内置 PLL 等模拟电路，因此在模拟电源和地的排布上需要仔细考虑。PCI-E 接口模拟电地列表如下：

表 12-5 PCI-E 接口模拟电源/地列表

名称	位宽	IO	描述
PCI_VSSA	1	双向	PCI-E PHY 的模拟地
PCI_VDDA25	12	双向	PCI-E PHY 2.5V I/O 模拟电压，作为其内部 PLL 的电源(2.25-2.75V)
PCI_VDDA10	6	双向	PCI PHY 1.0V 内核模拟电压(0.9-1.1V)
PCI_VTT	12	双向	传输终端供电电压 1.4—2.75V（1.5V，1.8V，2.5V），每个 2X PHY 4 个

12.2.3. USB 接口

USB 接口内部包含内置 PLL 等模拟电路，因此在模拟电源和地的排布上需要仔细考虑。USB 接口模拟电地列表如下：

表 12-6 USB 接口模拟电源/地列表

名称	位宽	IO	描述
USB_VDD33	2	双向	USB PHY 的 3.3V 模拟电源，标称值 3.3V(+10%,-7%)
USB_VDD25	2	双向	USB PHY 2.5V I/O 模拟电压，作为其内部 PLL 的电源，标称值 2.5V(+10%,-7%)
USB_VSSA	2	双向	USB PHY 接收发送器的模拟地
USB_VSSAC	2	双向	USB PHY 其它模块的模拟地
USB_ANALOGTEST	1	输出	USB PHY 测试模拟信号，用于观察存储器 PHY 内部锁相环的测试输出

12.2.4. SATA 接口

SATA 接口内部包含内置 PLL 等模拟电路，SATA PHY 采用商用 IP，电地已经排布厂商已经考虑其完整性等方面需求。SATA 接口模拟电地列表如下：

表 12-7 SATA 接口模拟电源/地列表

名称	位宽	IO	描述
SATA_VP	2	双向	SATA PHY 模拟和数字逻辑供电 0.9V(+10%,-7%)
SATA_VPH	2	双向	SATA PHY 高电压供电 1.8V(+10%,-7%)
SATA_VPTX	8	双向	SATA PHY 发送逻辑供电 0.9V(+10%,-7%)

12.3. 电流和功耗

表 12-8 ICH2 电流和功耗

信号功能	信号名称	管脚数	工作电压	信号说明	管脚电流	功耗
核心电	VDD	128	0.95V	核心数字 0.95V	5A	4.5W
数字地	VSS	466	GND	数字地		

PLL 模拟 电源	PLL_VDD09A_0	1	0.9V	GPUPLL 的模拟 0.9V 供电，建议 与其他电源之间 用磁珠	2.5mA	0.00225W
	PLL_VDD09A_1	1	0.9V	MCPLL 的模拟 0.9V 供电，建议 与其他电源之间 用磁珠	2.5mA	0.00225W
	PLL_VDD09A_2	1	0.9V	PSWPLL 的模拟 0.9V 供电，建议 与其他电源之间 用磁珠	2.5mA	0.00225W
	PLL_VDD09A_3	1	0.9V	GCLKPLL 的模拟 0.9V 供电，建议 与其他电	2.5mA	0.00225W
	PLL_VSS09A_0	1	GND	连接数字地 VSS	-	-
	PLL_VSS09A_1	1	GND	连接数字地 VSS	-	-
	PLL_VSS09A_2	1	GND	连接数字地 VSS	-	-
	PLL_VSS09A_3	1	GND	连接数字地 VSS	-	-
IO 电源	LVDS_VDDQ18	1	1.8V	LVDS(差分转单端 IOBUF)类型 IO 的 1.8V 供电，建 议与其他电	-	-

	LVDS_VSSQ18	1	GND	连接数字地 VSS。	-	-
	VDDQ15	19	1.5V	DDR3 接口 IO 数	2.5A	3.8W
	VDDQ	35	1.8V	其他 GPIO 接口 IO 数字 1.8V	500mA	0.9W
PSW 电源	PCIe_US_VDDA	4	0.9V	US 端口模拟电压 0.9V，建议与其他电源之间用磁珠隔开	203mA	0.18W
	PCIe_US_VDDQA	4	1.8V	US 端口模拟电压 1.8V，建议与其他电源之间用磁珠隔开	125mA	0.225W
	PCIe_US_VTT	4	1.5V	US 端口模拟电压 1.5V，建议与其他电源之间用磁珠隔开	240mA	0.36W
	PCIe_DS13_VDDA	4	0.9V	DS13 端口模拟电压 0.9V，建议与其他电源之间用磁珠隔开	203mA	0.18W
	PCIe_ DS13_VDDQA	4	1.8V	DS13 端口模拟电压 1.8V，建议与其他电源之间用磁珠隔开	125mA	0.225W
	PCIe_ DS13_VTT	4	1.5V	DS13 端口模拟电压 1.5V，建议与其他电源之间用磁珠隔开	240mA	0.36W
	PCIe_DS24_VDDA	4	0.9V	DS24 端口模拟电压 0.9V，建议与其他电源之间用磁珠隔开	203mA	0.18W
	PCIe_ DS24_VDDQA	4	1.8V	DS24 端口模拟电压 1.8V，建议与其他电源之间用磁珠隔开	125mA	0.225W
	PCIe_ DS24_VTT	4	1.5V	DS24 端口模拟电压 1.5V，建议与其他电源之间用磁珠隔开	240mA	0.36W
	PCIe_DS5_VDDA	4	0.9V	DS5 端口模拟电压 0.9V，建议与其他电源之间用 磁珠隔开	203mA	0.18W

	PCIe_ DS5_VDD QA	4	1.8V	DS5 端口模拟电压 1.8V，建议与 其他电源之间用 磁珠隔开	125mA	0.225W

	PCIe_ DS5_VTT	4	1.5V	DS5 端口模拟电压 1.5V，建议与 其他电源之间用 磁珠隔	240mA	0.36W
SATA	SATA_VP	2	0.9V（+10%，	3 个 SATA 端口的 模拟 0.9V，建议 与其他电	138mA	0.124W
	SATA_VPH	2	1.8V（+10%，	3 个 SATA 端口的 模拟 1.8V，建议 与其他电	24mA	0.043W
	SATA_VPTX	2	0.9V（+10%，-7%	3 个 SATA 端口的 发送模拟 0.9V， 建议与其他电源之间用磁珠隔开	138mA	0.124W
USB	USB_VP_0	2	0.9V（+10%，	USB_0/1 端口的模拟 0.9V，建议 与其他电源之间 用磁珠	90mA	0.081W
	USB_VP_1	2	0.9V（+10%，	USB_2/3 端口的模拟 0.9V，建议 与其他电源之间 用磁珠	90mA	0.081W
	USB_VP_2	2	0.9V（+10%，	USB_4/5 端口的模拟 0.9V，建议 与其他电源之间 用磁珠	90mA	0.081W
	USB_VDD180_0	1	1.8V（+10%，-	USB_0/1 端口的（VPH/VDD180） 模拟 IO 1.8V，建 议与其他电	32mA	0.057W
	USB_VDD180_1	1	1.8V（+10%，-	USB_2/3 端口的（VPH/VDD180） 模拟 IO 1.8V，建 议与其他电	32mA	0.057W
	USB_VDD180_2	1	1.8V（+10%，-	USB_4/5 端口的（VPH/VDD180） 模拟 IO 1.8V，建 议与其他电	32mA	0.057W

	USB_VPTX_0	1	0.9V（+10%，-7%	USB_0/1 端口的 tx 模拟 0.9V，建 议与其他电源之间用磁珠隔开（建议与	90mA	0.081W
				隔开）		
	USB_VPTX_1	1	0.9V（+10%，-7%	USB_2/3 端口的 tx 模拟 0.9V，建 议与其他电源之间用磁珠隔开（建议与 VP 也用磁珠	90mA	0.081W
	USB_VPTX_2	1	0.9V（+10%，-7%	USB_4/5 端口的 tx 模拟 0.9V，建 议与其他电源之间用磁珠隔开（建议与 VP 也用磁珠	90mA	0.081W
	USB_VDD330_0	1	3.3V（+10%，	USB_0/1 端口的模拟 3.3V，建议 与其他电源之间 用磁珠	32mA	0.105W
	USB_VDD330_1	1	3.3V（+10%，	USB_2/3 端口的模拟 3.3V，建议 与其他电源之间 用磁珠	32mA	0.105W
	USB_VDD330_2	1	3.3V（+10%，	USB_4/5 端口的模拟 3.3V，建议 与其他电源之间 用磁珠	32mA	0.105W
MC（DDR3）	DDR3_VDDA18_0	1	1.8V（+10%，-	DATAX8_0/1/2 的 PLL 模拟 1.8V， 建议与其他电源之间用磁珠隔开	168mA	0.302W
	DDR3_VDDA18_1	1	1.8V（+10%，-	DATAX8_3/8（AC）/4 的 PLL 模拟 1.8V，建议与其他电源之间	168mA	0.302W
	DDR3_VDDA18_2	1	1.8V（+10%，-	DATAX8_5/6/7 的 PLL 模拟 1.8V， 建议与其他电源之间用磁珠隔开	168mA	0.302W
	DDR3_VSSA18_0	1	GN	连接数字地 VSS	-	-
	DDR3_VSSA18_1	1	GN	连接数字地 VSS	-	-
	DDR3_VSSA18_2	1	GN	连接数字地 VSS	-	-
总计		733				~14W

13 封装与管脚

13.1. 封装与管脚概述

ICH2 片内集成了 1 路 64 位/32 位存储控制器、PCI-E 开关、SATA 控制器、USB 控制 器、声卡控制器、千兆以太网以及维护测试等接口，芯片的信号管脚数量为 648 根，整体管 脚数达到 1296 个。结合多方面考虑选用 FC-BGA1296 (Flip-chip BGA)封装。

13.2. 封装说明

ICH2 使用 FC-BGA 封装，如图 13-1，有如下特点：

- 封装尺寸（L*W*H）：37.5±0.05 mm×37.5±0.05 mm×3.22±0.28 mm
- 封装焊球：
  - 数量：1296 个
  - 直径：0.60mm
  - 间距：1.0mm
  - 高度：0.50mm
  - 成分：Sn63Pb37
  - 焊接：回流焊（reflow）顶视

  焊球分布如图 13-2。
- 散热与热特性：
  - VDD 功耗：约 4W（典型）
  - VDDQ15 功耗：约 2W（典型）
  - VDDQ18 功耗：约 2W（典型）
  - 散热方式：风冷

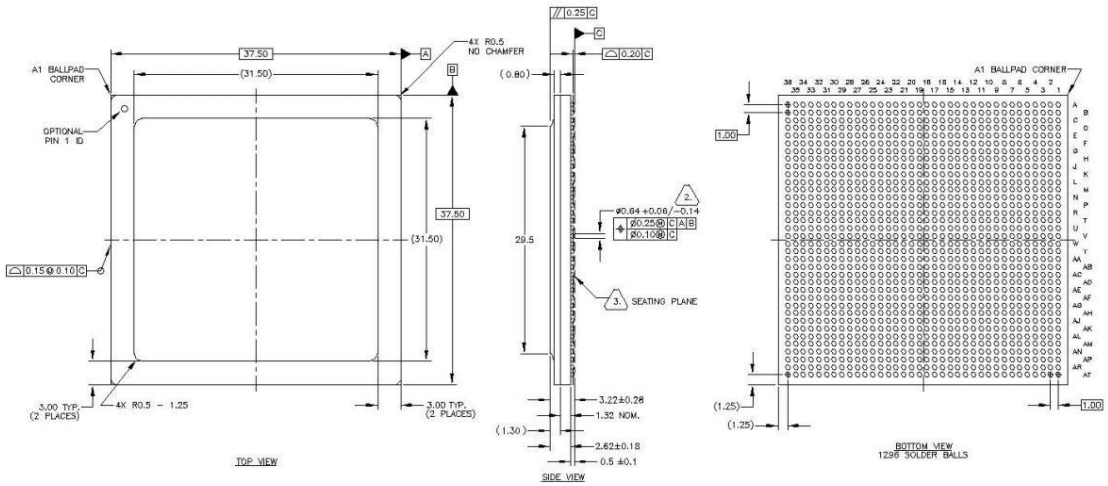


图 13-1 FC-BGA 封装尺寸图

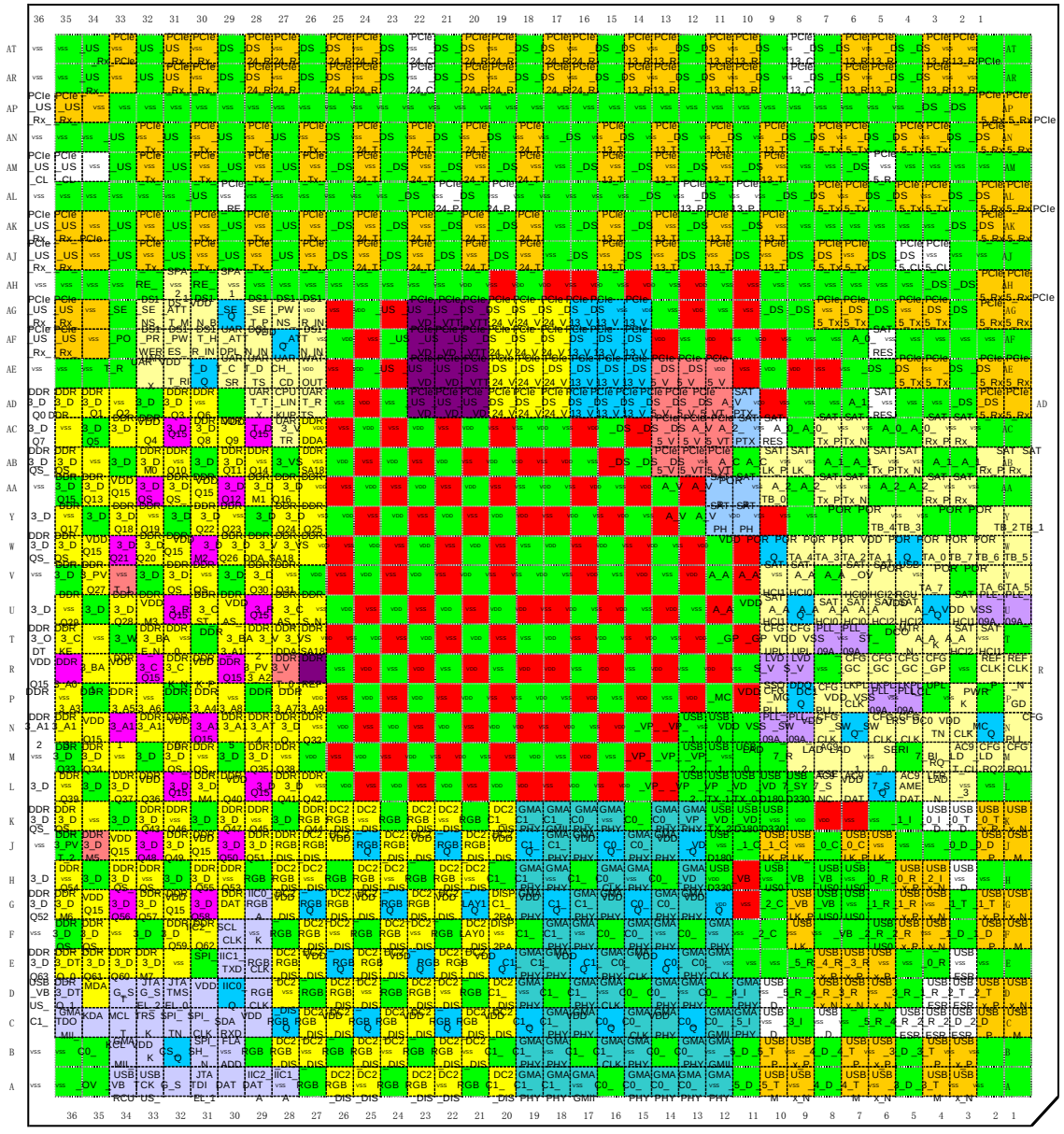


图 13-2 FC-BGA 顶视焊球分布图

封装基板特性如下：

- 基板尺寸(L*W*H)：37.5mm×37.5mm×1.32mm
- 基板布局（参见下图）
  - Die 信息：
    - 尺寸 17.13mm×15.25mm
    - 厚度约为 750um
  - 去耦电容（参见表 13-1）
    - 数量：4
    - 用于 VDDQ15

表 13-1 基板去耦电容说明

电容类型	电容容值	电容尺寸（L*W*H）

TDK CLLC1AX7S0G225M	2.2uf@4V	1.60mm * 0.80mm * 0.50mm

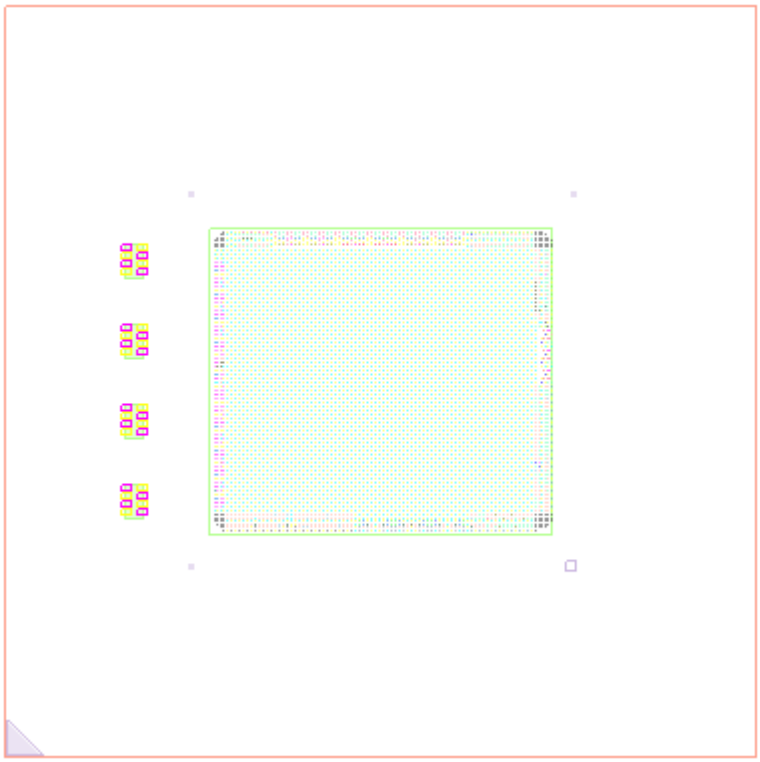


图 0-3 基板布局

- 基板采用 4+4+4 结构
  - 上面 4 层：PWR/SIGNAL/VSS/SIGNAL
  - 中间 4 层：VSS/PWR/PWR/VSS
  - 下面 4 层：SIGNAL / VSS / SIGNAL / VSS
- Bump 情况：
  - 直径：90um
  - 高度：70um
  - 间距：最小 150um
  - 材料：Eutectic
- 信号、电源和地
  - 单端信号阻抗控制 50ohm+/-10%
  - USB 差分信号阻抗控制 90ohm+/-10%
  - 其它差分信号阻抗控制 100ohm+/-10%

13.3. 引脚编号

ICH2 的引脚分配如下表所示：

表 13-2 ICH2 引脚分配

管脚	名称	管脚	名称	管脚	名称
A36	VSS	N36	DDR3_A12	AE36	VSS
A35	VSS	N35	DDR3_A13	AE35	VSS
A34	USB_OVRCUR_F_N	N34	VDDQ15	AE34	VSS
A33	USB_VBUS_CTRL	N33	DDR3_A11	AE33	UART_RX
A32	TCK	N32	DDR3_A10	AE32	UART_RI

A3	JTAG_SEL_1	N31	VDDQ15	AE3	VDDQ
A3	TDI	N30	DDR3_A15	AE3	UART_DSR
A2	IIC2_DATA	N29	DDR3_A14	AE2	UART_CTS
A2	IIC1_DATA	N28	DDR3_ATO	AE2	UART_DCD
A2	VSS	N27	DDR3_DQ32	AE2	WATCH_OUT
A26	DC2RGB_DISPLAY1GREEN_2	N26	VSS	AE26	VDD
A25	DC2RGB_DISPLAY1GREEN_0	N25	VDD	AE25	VSS
A2	VSS	N24	VSS	AE2	VDD
A23	DC2RGB_DISPLAY0BLUE_5	N23	VDD	AE23	PCIe_US_VDDQ_3A
A22	DC2RGB_DISPLAY0BLUE_1	N22	VSS	AE22	PCIe_US_VDDA_0
A2	VSS	N21	VDD	AE2	PCIe_US_VTT_3
A2	DC2RGB_DISPLAY0RED_	N20	VSS	AE2	PCIe_DS24_VDDQA_3
A1	GMAC1_PHY_TXD_4	N19	VDD	AE1	PCIe_DS24_VDDA_0
A1	GMAC1_PHY_TXD_2	N18	VDD	AE1	PCIe_DS24_VTT_4
A1	GMAC1_GMII_MDIO	N17	VSS	AE1	PCIe_DS13_VDDQ_3
A1	VSS	N16	VDD	AE1	PCIe_DS13_VDDA_0
A1	GMAC0_PHY_TXD_7	N15	VSS	AE1	PCIe_DS13_VTT_3
A1	GMAC0_PHY_TXD_3	N14	VDD	AE1	PCIe_DS5_VDDQA_0
A1	GMAC0_PHY_TXD_0	N13	USB_VP_1_1	AE1	PCIe_DS5_VDDQA_1
A1	GMAC0_PHY_RXDV	N12	USB_VP_0_1	AE1	PCIe_DS5_VDDQA_2
A1	VSS	N11	VSS	AE1	VDD
A1	USB_5_DM	N10	PLL_VDD09A_3	AE1	VSS
A	USB_5_Tx_N	N9	PLL_VSS09A_3	AE	VDD
A	VSS	N8	CFG_SWCLKPLL_2	AE	VDD
A	USB_4_DM	N7	VDDQ	AE	VSS
A	USB_4_Tx_N	N6	CFG_SWCLKPLL_1	AE	VSS
A	VSS	N5	CFG_SWCLKPLL_0	AE	PCIe_DS5_Tx_0_P
A	USB_3_DM	N4	LRSTN	AE	PCIe_DS5_Tx_0_N
A	USB_3_Tx_N	N3	DC0CLK	AE	VSS
A	VSS	N2	VDDQ	AE	PCIe_DS5_Rx_1_P
A	VSS	N1	CFG_MCPLL_2	AE	PCIe_DS5_Rx_1_N
B3	VSS	P36	DDR3_A3	AF3	PCIe_US_Rx_7_N
B3	VSS	P35	VSS	AF3	PCIe_US_Rx_7_P
B3	GMAC0_MII_TX_CLK	P34	DDR3_A5	AF3	VSS
B3	KCLK	P33	DDR3_A6	AF3	DS1_POWER_CONTROL
B3	VDDQ	P32	VSS	AF3	DS1_PRES_DET_CLEA
B3	SPI_CS_N	P31	DDR3_A4	AF3	DS1_PWR_INDICATOR_CN
B3	FLASH_ADDRSEL	P30	DDR3_A8	AF3	UART_HDPLX
B2	VSS	P29	VSS	AF2	DS1_ATTN_INDICATOR_CN
B28	DC2RGB_DISPLAY1BLUE_2	P28	DDR3_A7	AF28	VDDQ
B27	DC2RGB_DISPLAY1BLUE_7	P27	DDR3_A9	AF27	DS1_ATTN_INDICATOR_CNTR
B2	VSS	P26	VDD	AF2	VSS
B2	DC2RGB_DISPLAY1VSYN	P25	VSS	AF2	VDD
B24	DC2RGB_DISPLAY0BLUE_6	P24	VDD	AF24	VSS
B2	VSS	P23	VSS	AF2	PCIe_US_VDDQA_1
B2	DC2RGB_DISPLAY0GREE	P22	VDD	AF2	PCIe_US_VDDQA_2

	_1				
B2	DC2RGB_DISPLAY0RED_	P21	VSS	AF2	PCIe_US_VTT_2
B2	VSS	P20	VDD	AF2	PCIe_DS24_VDDQA_1
B1	GMAC1_PHY_TXD_5	P19	VSS	AF1	PCIe_DS24_VDDQA_2
B1	GMAC1_PHY_TXD_3	P18	VSS	AF1	PCIe_DS24_VTT_3
B1	VSS	P17	VDD	AF1	PCIe_DS13_VDDQA_1
B1	GMAC1_CLK_OSC	P16	VSS	AF1	PCIe_DS13_VDDQA_2
B1	GMAC1_PHY_INTF_SEL	P15	VDD	AF1	PCIe_DS13_VTT_2
B1	VSS	P14	VSS	AF1	VDD
B1	GMAC0_PHY_RXD_0	P13	VDD	AF1	VSS
B1	GMAC0_GMII_MDC	P12	VSS	AF1	VDD
B1	VSS	P11	VDD	AF1	VSS
B1	USB_5_DP	P10	CFG_MCPLL_1	AF1	VDD
B	USB_5_Tx_P	P9	VDDQ	AF	VSS
B	VSS	P8	CFG_MCPLL_0	AF	VSS
B	USB_4_DP	P7	DC1CLK	AF	VSS
B	USB_4_Tx_P	P6	PLL_VDD09A_2	AF	SATA_0_RESREF
B	VSS	P5	PLL_VSS09A_2	AF	VSS
B	USB_3_DP	P4	VSS	AF	VSS
B	USB_3_Tx_P	P3	LCLK	AF	VSS
B	VSS	P2	VSS	AF	VSS
B	VSS	P1	PWRGD	AF	VSS
C3	GMAC1_MII_TX_CLK	R36	VDDQ15	AG3	PCIe_US_Rx_6_N
C3	TDO	R35	DDR3_A0	AG3	PCIe_US_Rx_6_P
C3	KDAT	R34	DDR3_BA1	AG3	VSS
C3	MCLK	R33	VDDQ15	AG3	DS1_SENS_PRES_DET_ST
C3	TRSTN	R32	DDR3_CK_N	AG3	DS1_SET_MRL_SENS_CHANGED
C3	SPI_CLK	R31	DDR3_CK_P	AG3	DS1_ATTN_BTN_PRES
C3	SPI_RXD	R30	VDDQ15	AG3	VDDQ
C2	Reserve	R29	DDR3_A2	AG2	DS1_SET_PWR_FAULT_DETEC
C2	VDDQ	R28	DDR3_PVT_0	AG2	DS1_SENS_MRL_SENS_S
C2	DC2RGB_DISPLAY1GREEN_6	R27	DDR3_VREF	AG2	DS1_PWR_INDICATOR_CNTR
C2	DC2RGB_DISPLAY1GREEN_3	R26	VSS	AG2	VDD
C2	VDDQ	R25	VDD	AG2	VSS
C2	DC2RGB_DISPLAY0BLUE_ 7	R24	VSS	AG2	VDD
C2	DC2RGB_DISPLAY0EN	R23	VDD	AG2	PCIe_US_VDDA_0
C2	VDDQ	R22	VSS	AG2	PCIe_US_VTT_0
C2	DC2RGB_DISPLAY0RED_	R21	VDD	AG2	PCIe_US_VTT_1
C2	DC2RGB_DISPLAY0RED_	R20	VSS	AG2	PCIe_DS24_VDDA_0
C1	VDDQ	R19	VDD	AG1	PCIe_DS24_VTT_1
C1	GMAC1_PHY_RXD_3	R18	VDD	AG1	PCIe_DS24_VTT_2
C1	GMAC1_PHY_RXD_0	R17	VSS	AG1	PCIe_DS13_VDDQA_0
C1	VDDQ	R16	VDD	AG1	PCIe_DS13_VTT_0
C1	GMAC0_CLK_TX	R15	VSS	AG1	PCIe_DS13_VTT_1
C1	GMAC0_PHY_TXD_4	R14	VDD	AG1	VSS
C1	VDDQ	R13	VSS	AG1	VDD
C1	GMAC0_GMII_MDIO	R12	VDD	AG1	VSS

C1	GMAC0_PHY_CRS	R11	VSS	AG1	VDD
C1	USB_5_ID	R10	LVDS_VSSQ18	AG1	VSS
C	VSS	R9	LVDS_VDDQ18	AG	VSS
C	USB_3_ID	R8	VSS	AG	PCIe_DS5_Tx_1_P
C	VSS	R7	CFG_GCLKPLL_2	AG	PCIe_DS5_Tx_1_N
C	VSS	R6	CFG_GCLKPLL_1	AG	VSS
C	USB_5_RESREF	R5	CFG_GCLKPLL_0	AG	PCIe_DS5_Tx_2_P
C	USB_4_RESREF	R4	CFG_GPUPLL_2	AG	PCIe_DS5_Tx_2_N
C	USB_2_RESREF	R3	VSS	AG	VSS
C	USB_2_DP	R2	REFCLK_P	AG	PCIe_DS5_Rx_2_P
C	USB_2_DM	R1	REFCLK_N	AG	PCIe_DS5_Rx_2_N
D3	USB_VBUS_F_CTRL	T36	DDR3_ODT	AH3	VSS
D3	DDR3_DTO_1	T35	DDR3_CKE	AH3	VSS
D3	MDAT	T34	VSS	AH3	VSS
D3	JTAG_SEL_2	T33	DDR3_WE_N	AH3	VSS
D3	JTAG_SEL_0	T32	DDR3_BA0	AH3	SPARE_2
D3	TMS	T31	VSS	AH3	VSS
D3	VDDQ	T30	DDR3_A1	AH3	SPARE_1
D2	IIC0_CLK	T29	DDR3_BA2	AH2	VSS
D28	DC2RGB_DISPLAY1BLUE_3	T28	DDR3_VDDA18_2	AH28	VSS
D2	VSS	T27	DDR3_VSSA18_2	AH2	VSS
D26	DC2RGB_DISPLAY1GREEN_4	T26	VDD	AH26	VSS
D2	DC2RGB_DISPLAY1RED_	T25	VSS	AH2	VSS
D2	VSS	T24	VDD	AH2	VSS
D2	DC2RGB_DISPLAY0CLK	T23	VSS	AH2	VSS
D22	DC2RGB_DISPLAY0GREEN_2	T22	VDD	AH22	VSS
D2	VSS	T21	VSS	AH2	VSS
D2	DC2RGB_DISPLAY0RED_	T20	VDD	AH2	VDD
D1	GMAC1_PHY_TXD_6	T19	VSS	AH1	VSS
D1	VSS	T18	VSS	AH1	VDD
D1	GMAC1_PHY_RXD_1	T17	VDD	AH1	VDD
D1	GMAC1_CLK_RX	T16	VSS	AH1	VSS
D1	VSS	T15	VDD	AH1	VDD
D1	GMAC0_PHY_RXD_3	T14	VSS	AH1	VSS
D1	GMAC0_PHY_TXER	T13	VDD	AH1	VDD
D1	VSS	T12	VSS	AH1	VSS
D1	GMAC0_PHY_COL	T11	VDD	AH1	VDD
D1	USB_4_ID	T10	CFG_GPUPLL_1	AH1	VSS
D	VSS	T9	CFG_GPUPLL_0	AH	VSS
D	USB_5_Rx_N	T8	PLL_VDD09A_1	AH	VSS
D	USB_4_Rx_N	T7	PLL_VSS09A_1	AH	VSS
D	USB_3_Rx_N	T6	VSS	AH	VSS
D	VSS	T5	MTRST_N	AH	VSS
D	USB_3_RESREF	T4	DCOK	AH	VSS
D	USB_1_RESREF	T3	SATA_AHCI2_ACT_LED_	AH	VSS
D	USB_2_Tx_P	T2	SATA_AHCI1_CP_POD	AH	PCIe_DS5_Rx_3_P
D	USB_2_Tx_N	T1	VSS	AH	PCIe_DS5_Rx_3_N
E3	DDR3_DQ63	U36	DDR3_DQ29	AJ3	PCIe_US_Rx_5_N

E3	DDR3_DTO_0	U35	VSS	AJ3	PCIe_US_Rx_5_P
E3	DDR3_DQ61	U34	DDR3_DQ28	AJ3	VSS
E3	DDR3_DQ60	U33	DDR3_DM3	AJ3	PCIe_US_Tx_7_P
E3	DDR3_DM7	U32	VDDQ15	AJ3	VSS
E3	VSS	U31	DDR3_RST_N	AJ3	PCIe_US_Tx_5_P
E3	SPI_TXD	U30	DDR3_CAS_N	AJ3	VSS
E2	IIC1_CLK	U29	VDDQ15	AJ2	PCIe_US_Tx_3_P
E28	DC2RGB_DISPLAY1BLUE_4	U28	DDR3_RAS_N	AJ28	VSS
E27	DC2RGB_DISPLAY1GREEN_7	U27	DDR3_CS_N	AJ27	PCIe_US_Tx_1_P
E2	VDDQ	U26	VSS	AJ2	VSS
E2	DC2RGB_DISPLAY1RED_	U25	VDD	AJ2	PCIe_DS24_Tx_7_P
E2	DC2RGB_DISPLAY1HSYN	U24	VSS	AJ2	VSS
E2	VDDQ	U23	VDD	AJ2	PCIe_DS24_Tx_5_P
E22	DC2RGB_DISPLAY0GREEN_3	U22	VSS	AJ22	VSS
E2	DC2RGB_DISPLAY0RED_	U21	VDD	AJ2	PCIe_DS24_Tx_3_P
E2	VDDQ	U20	VSS	AJ2	VSS
E1	GMAC1_PHY_TXD_7	U19	VDD	AJ1	PCIe_DS24_Tx_1_P
E1	GMAC1_PHY_RXD_4	U18	VDD	AJ1	VSS
E1	VDDQ	U17	VSS	AJ1	VSS
E1	GMAC1_PHY_TXER	U16	VDD	AJ1	PCIe_DS13_Tx_7_P
E1	GMAC0_CLK_RX	U15	VSS	AJ1	VSS
E1	VDDQ	U14	VDD	AJ1	PCIe_DS13_Tx_5_P
E1	GMAC0_PHY_RXD_1	U13	VSS	AJ1	VSS
E1	GMAC0_CLK_OSC	U12	VDD	AJ1	PCIe_DS13_Tx_3_P
E1	VSS	U11	VSS	AJ1	VSS
E1	VSS	U10	SATA_AHCI1_ACT_LED_	AJ1	PCIe_DS13_Tx_1_P
E	VSS	U9	VDDQ	AJ	VSS
E	USB_5_Rx_P	U8	SATA_AHCI0_CP_POD	AJ	PCIe_DS5_Tx_3_P
E	USB_4_Rx_P	U7	SATA_AHCI0_ACT_LED_	AJ	PCIe_DS5_Tx_3_N
E	USB_3_Rx_P	U6	SATA_AHCI2_CP_DET	AJ	VSS
E	VSS	U5	SATA_AHCI2_MP_SWITC	AJ	PCIe_DS5_CLK_P
E	VSS	U4	VDDQ	AJ	PCIe_DS5_CLK_N
E	USB_0_RESREF	U3	SATA_AHCI1_CP_DET	AJ	VSS
E	VSS	U2	PLL_VDD09A_0	AJ	VSS
E	VSS	U1	PLL_VSS09A_0	AJ	VSS
F3	VSS	V36	VSS	AK3	PCIe_US_Rx_4_N
F3	DDR3_DQS_7_N	V35	DDR3_DQ27	AK3	PCIe_US_Rx_4_P
F3	DDR3_DQS_7_P	V34	DDR3_PVT_1	AK3	VSS
F3	VSS	V33	VSS	AK3	PCIe_US_Tx_7_N
F3	DDR3_DQ59	V32	DDR3_DQS_3_N	AK3	VSS
F3	DDR3_DQ62	V31	DDR3_DQS_3_P	AK3	PCIe_US_Tx_5_N
F3	IIC2_CLK	V30	VSS	AK3	VSS
F2	Reserve	V29	DDR3_DQ30	AK2	PCIe_US_Tx_3_N
F2	VSS	V28	DDR3_DQ31	AK2	VSS
F27	DC2RGB_DISPLAY1BLUE_0	V27	VSS	AK27	PCIe_US_Tx_1_N
F26	DC2RGB_DISPLAY1GREEN_5	V26	VDD	AK26	VSS
F2	VSS	V25	VSS	AK2	PCIe_DS24_Tx_7_N

F2	DC2RGB_DISPLAY1CLK	V24	VDD	AK2	VSS
F23	DC2RGB_DISPLAY0BLUE_2	V23	VSS	AK23	PCIe_DS24_Tx_5_N
F2	VSS	V22	VDD	AK2	VSS
F21	DC2RGB_DISPLAY0GREEN_0	V21	VSS	AK21	PCIe_DS24_Tx_3_N
F2	DISPLAY02PAB_INT	V20	VDD	AK2	VSS
F1	VSS	V19	VSS	AK1	PCIe_DS24_Tx_1_N
F1	GMAC1_PHY_RXD_5	V18	VSS	AK1	VSS
F1	GMAC1_PHY_RXD_2	V17	VDD	AK1	VSS
F1	VSS	V16	VSS	AK1	PCIe_DS13_Tx_7_N
F1	GMAC0_PHY_TXD_5	V15	VDD	AK1	VSS
F1	GMAC0_PHY_RXD_4	V14	VSS	AK1	PCIe_DS13_Tx_5_N
F1	VSS	V13	VDD	AK1	VSS
F1	GMAC0_PHY_INIF_SEL	V12	VSS	AK1	PCIe_DS13_Tx_3_N
F1	VSS	V11	VDD	AK1	VSS
F1	VSS	V10	SATA_AHCI1_MP_SWITC	AK1	PCIe_DS13_Tx_1_N
F	USB_2_CLK_N	V9	SATA_AHCI0_CP_DET	AK	VSS
F	VSS	V8	VSS	AK	VSS
F	VSS	V7	SATA_AHCI0_MP_SWITC	AK	VSS
F	USB_VBUS0_0	V6	SATA_AHCI2_CP_POD	AK	VSS
F	USB_2_Rx_P	V5	USB_OVRCUR_N	AK	VSS
F	USB_2_Rx_N	V4	PORTA_7	AK	VSS
F	VSS	V3	VSS	AK	VSS
F	USB_1_DP	V2	PORTA_6	AK	PCIe_DS5_Rx_4_P
F	USB_1_DM	V1	PORTA_5	AK	PCIe_DS5_Rx_4_N
G3	DDR3_DQ52	W36	DDR3_DQS_2_N	AL3	VSS
G3	DDR3_DM6	W35	DDR3_DQS_2_P	AL3	VSS
G3	VDDQ15	W34	VDDQ15	AL3	VSS
G3	DDR3_DQ56	W33	DDR3_DQ21	AL3	VSS
G3	DDR3_DQ57	W32	DDR3_DQ20	AL3	VSS
G3	VDDQ15	W31	VDDQ15	AL3	VSS
G3	DDR3_DQ58	W30	DDR3_DM2	AL3	PCIe_US_RESREF
G2	IIC0_DATA	W29	DDR3_DQ26	AL2	VSS
G28	DC2RGB_DISPLAY1BLUE_5	W28	DDR3_VDDA18_1	AL28	VSS
G2	VDDQ	W27	DDR3_VSSA18_1	AL2	VSS
G2	DC2RGB_DISPLAY1RED_	W26	VDD	AL2	VSS
G2	DC2RGB_DISPLAY1RED_	W25	VSS	AL2	VSS
G2	VDDQ	W24	VDD	AL2	VSS
G23	DC2RGB_DISPLAY0BLUE_3	W23	VSS	AL23	VSS
G22	DC2RGB_DISPLAY0GREEN_4	W22	VDD	AL22	PCIe_DS24_PHY1_RESREF
G2	VDDQ	W21	VSS	AL2	VSS
G2	DISPLAY12PAB_INT	W20	VDD	AL2	PCIe_DS24_PHY0_RESR
G1	GMAC1_PHY_RXD_7	W19	VSS	AL1	VSS
G1	VDDQ	W18	VSS	AL1	VSS
G1	GMAC1_PHY_CRS	W17	VDD	AL1	VSS
G1	GMAC1_PHY_TXEN	W16	VSS	AL1	VSS
G1	VDDQ	W15	VDD	AL1	VSS
G1	GMAC0_PHY_RXD_5	W14	VSS	AL1	VSS

G1	GMAC0_PHY_RXD_2	W13	VDD	AL1	PCIe_DS13_PHY1_RESR
G1	VDDQ	W12	VSS	AL1	VSS
G1	VDD	W11	VDD	AL1	PCIe_DS13_PHY0_RESR
G1	VSS	W10	VDDQ	AL1	VSS
G	USB_2_CLK_P	W9	PORTA_4	AL	VSS
G	USB_VBUS0_4	W8	PORTA_3	AL	PCIe_DS5_Tx_5_P
G	USB_VBUS0_2	W7	PORTA_2	AL	PCIe_DS5_Tx_5_N
G	VSS	W6	PORTA_1	AL	VSS
G	USB_1_Rx_P	W5	VDDQ	AL	PCIe_DS5_Tx_4_P
G	USB_1_Rx_N	W4	PORTA_0	AL	PCIe_DS5_Tx_4_N
G	VSS	W3	PORTB_7	AL	VSS
G	USB_1_Tx_P	W2	PORTB_6	AL	PCIe_DS5_Rx_5_P
G	USB_1_Tx_N	W1	PORTB_5	AL	PCIe_DS5_Rx_5_N
H3	DDR3_DQ54	Y36	DDR3_DQ17	AM	PCIe_US_CLK_N
H3	VSS	Y35	VSS	AM	PCIe_US_CLK_P
H3	DDR3_DQS_6_N	Y34	DDR3_DQ18	AM	VSS
H3	DDR3_DQS_6_P	Y33	DDR3_DQ19	AM	PCIe_US_Tx_6_P
H3	VSS	Y32	VSS	AM	VSS
H3	DDR3_DQ55	Y31	DDR3_DQ22	AM	PCIe_US_Tx_4_P
H3	DDR3_DQ53	Y30	DDR3_DQ23	AM	VSS
H2	VSS	Y29	VSS	AM	PCIe_US_Tx_2_P
H2 8	DC2RGB_DISPLAY1BLUE_ 6	Y28	DDR3_DQ24	AM 28	VSS
H2 7	DC2RGB_DISPLAY1BLUE_ 1	Y27	DDR3_DQ25	AM 27	PCIe_US_Tx_0_P
H2	VSS	Y26	VSS	AM	VSS
H2	DC2RGB_DISPLAY1RED_	Y25	VDD	AM	PCIe_DS24_Tx_6_P
H2	DC2RGB_DISPLAY1EN	Y24	VSS	AM	VSS
H2	VSS	Y23	VDD	AM	PCIe_DS24_Tx_4_P
H2 2	DC2RGB_DISPLAY0GREEN_5	Y22	VSS	AM 22	VSS
H2	DC2RGB_DISPLAY0RED_	Y21	VDD	AM	PCIe_DS24_Tx_2_P
H2	VSS	Y20	VSS	AM	VSS
H1	GMAC1_PHY_TXD_0	Y19	VDD	AM	PCIe_DS24_Tx_0_P
H1	GMAC1_PHY_RXD_6	Y18	VDD	AM	VSS
H1	VSS	Y17	VSS	AM	VSS
H1	GMAC1_CLK_TX	Y16	VDD	AM	PCIe_DS13_Tx_6_P
H1	GMAC0_PHY_TXD_1	Y15	VSS	AM	VSS
H1	VSS	Y14	VDD	AM	PCIe_DS13_Tx_4_P
H1	GMAC0_PHY_TXEN	Y13	VSS	AM	VSS
H1	USB_VDD330_2	Y12	SATA_VPH_1	AM	PCIe_DS13_Tx_2_P
H1	VDD	Y11	SATA_VPH_2	AM	VSS
H1	USB_VBUS0_5	Y10	VDD	AM	PCIe_DS13_Tx_0_P
H	VSS	Y9	VSS	AM	VSS
H	USB_VBUS0_3	Y8	VSS	AM	VSS
H	USB_VBUS0_1	Y7	VSS	AM	VSS
H	VSS	Y6	PORTB_4	AM	PCIe_DS5_RESREF
H	USB_0_Rx_P	Y5	PORTB_3	AM	VSS
H	USB_0_Rx_N	Y4	VSS	AM	VSS
H	USB_2_ID	Y3	VSS	AM	VSS
H	VSS	Y2	PORTB_2	AM	VSS

H	VSS	Y1	PORTB_1	AM	VSS
J3	VSS	AA36	VSS	AN3	VSS
J3	DDR3_PVT_2	AA35	DDR3_DQ15	AN3	VSS
J3	DDR3_DM5	AA34	DDR3_DQ13	AN3	VSS
J3	VDDQ15	AA33	VDDQ15	AN3	PCIe_US_Tx_6_N
J3	DDR3_DQ48	AA32	DDR3_DQS_1_N	AN3	VSS
J3	DDR3_DQ49	AA31	DDR3_DQS_1_P	AN3	PCIe_US_Tx_4_N
J3	VDDQ15	AA30	VDDQ15	AN3	VSS
J2	DDR3_DQ50	AA29	DDR3_DQ12	AN2	PCIe_US_Tx_2_N
J2	DDR3_DQ51	AA28	DDR3_DM1	AN2	VSS
J2 7	DC2RGB_DISPLAY1GREEN _1	AA27	DDR3_DQ16	AN2 7	PCIe_US_Tx_0_N
J2	DC2RGB_DISPLAY1RED_	AA26	VDD	AN2	VSS
J2	VDDQ	AA25	VSS	AN2	PCIe_DS24_Tx_6_N
J2 4	DC2RGB_DISPLAY0BLU E_ 4	AA24	VDD	AN2 4	VSS
J2 3	DC2RGB_DISPLAY0GREEN _7	AA23	VSS	AN2 3	PCIe_DS24_Tx_4_N
J2	VDDQ	AA22	VDD	AN2	VSS
J2	DC2RGB_DISPLAY0RED_	AA21	VSS	AN2	PCIe_DS24_Tx_2_N
J2	DC2RGB_DISPLAY0HSYN	AA20	VDD	AN2	VSS
J1	VDDQ	AA19	VSS	AN1	PCIe_DS24_Tx_0_N
J1	GMAC1_PHY_COL	AA18	VSS	AN1	VSS
J1	GMAC1_PHY_RXER	AA17	VDD	AN1	VSS
J1	VDDQ	AA16	VSS	AN1	PCIe_DS13_Tx_6_N
J1	GMAC0_PHY_TXD_2	AA15	VDD	AN1	VSS
J1	GMAC0_PHY_RXD_6	AA14	VSS	AN1	PCIe_DS13_Tx_4_N
J1	VDDQ	AA13	VDD	AN1	VSS
J1	USB_VDD180_2	AA12	SATA_VP_1	AN1	PCIe_DS13_Tx_2_N
J1	VSS	AA11	SATA_VP_2	AN1	VSS
J1	USB_1_CLK_P	AA10	PORTB_0	AN1	PCIe_DS13_Tx_0_N
J9	USB_1_CLK_N	AA9	VSS	AN	VSS
J8	VSS	AA8	SATA_2_Tx_P	AN	PCIe_DS5_Tx_7_P
J7	USB_0_CLK_P	AA7	SATA_2_Tx_N	AN	PCIe_DS5_Tx_7_N
J6	USB_0_CLK_N	AA6	VSS	AN	VSS
J5	VSS	AA5	VSS	AN	PCIe_DS5_Tx_6_P
J4	VSS	AA4	SATA_2_Rx_P	AN	PCIe_DS5_Tx_6_N
J3	VSS	AA3	SATA_2_Rx_N	AN	VSS
J2	USB_0_DP	AA2	VSS	AN	PCIe_DS5_Rx_6_P
J1	USB_0_DM	AA1	VSS	AN	PCIe_DS5_Rx_6_N
K3	DDR3_DQS_5_N	AB36	DDR3_DQS_0_N	AP3	PCIe_US_Rx_3_N
K3	DDR3_DQS_5_P	AB35	DDR3_DQS_0_P	AP3	PCIe_US_Rx_3_P
K3	VSS	AB34	VSS	AP3	VSS
K3	DDR3_DQ43	AB33	DDR3_DM0	AP3	VSS
K3	DDR3_DQ46	AB32	DDR3_DQ10	AP3	VSS
K3	VSS	AB31	VSS	AP3	VSS
K3	DDR3_DQ47	AB30	DDR3_DQ11	AP3	VSS
K2	DDR3_DQ45	AB29	DDR3_DQ14	AP2	VSS
K2	VSS	AB28	VSS	AP2	VSS
K2	DDR3_DQ44	AB27	DDR3_VSSA18_0	AP2	VSS
K2	DC2RGB_DISPLAY1RED_	AB26	VSS	AP2	VSS

K2	DC2RGB_DISPLAY1RED_	AB25	VDD	AP2	VSS
K2	VSS	AB24	VSS	AP2	VSS
K2 3	DC2RGB_DISPLAY0BLUE_0	AB23	VDD	AP2 3	VSS
K2 2	DC2RGB_DISPLAY0GREEN_6	AB22	VSS	AP2 2	VSS
K2	VSS	AB21	VDD	AP2	VSS
K2	DC2RGB_DISPLAY0VSYN	AB20	VSS	AP2	VSS
K1	GMAC1_PHY_TXD_1	AB19	VDD	AP1	VSS
K1	GMAC1_GMII_MDC	AB18	VDD	AP1	VSS
K1	GMAC1_PHY_RXDV	AB17	VSS	AP1	VSS
K1	GMAC0_PHY_TXD_6	AB16	VDD	AP1	VSS
K1	VSS	AB15	VSS	AP1	VSS
K1	GMAC0_PHY_RXD_7	AB14	PCIe_DS5_VDDA_3	AP1	VSS
K1	GMAC0_PHY_RXER	AB13	PCIe_DS5_VTT_2	AP1	VSS
K1	USB_VPTX_2	AB12	PCIe_DS5_VTT_3	AP1	VSS
K1	USB_VDD180_1	AB11	VSS	AP1	VSS
K1	USB_VDD330_1	AB10	SATA_CLK_P	AP1	VSS
K	VSS	AB9	SATA_CLK_N	AP	VSS
K	VDD	AB8	VSS	AP	VSS
K	VDD	AB7	VSS	AP	VSS
K	VSS	AB6	SATA_1_Tx_P	AP	VSS
K	VSS	AB5	SATA_1_Tx_N	AP	VSS
K	USB_1_ID	AB4	VSS	AP	VSS
K	USB_0_ID	AB3	VSS	AP	VSS
K	USB_0_Tx_P	AB2	SATA_1_Rx_P	AP	PCIe_DS5_Rx_7_P
K	USB_0_Tx_N	AB1	SATA_1_Rx_N	AP	PCIe_DS5_Rx_7_N
L3	DDR3_DQ39	AC36	DDR3_DQ7	AR3	VSS
L3	VSS	AC35	VSS	AR3	VSS
L3	DDR3_DQ37	AC34	DDR3_DQ5	AR3	PCIe_US_Rx_2_P
L3	DDR3_DQ36	AC33	DDR3_DQ4	AR3	VSS
L3	VDDQ15	AC32	VDDQ15	AR3	PCIe_US_Rx_1_P
L3	DDR3_DM4	AC31	DDR3_DQ8	AR3	PCIe_US_Rx_0_P
L3	DDR3_DQ40	AC30	DDR3_DQ9	AR3	VSS
L2	VDDQ15	AC29	VDDQ15	AR2	PCIe_DS24_Rx_7_P
L2	DDR3_DQ41	AC28	UART_DTR	AR2	PCIe_DS24_Rx_6_P
L2	DDR3_DQ42	AC27	DDR3_VDDA18_0	AR2	VSS
L2	VSS	AC26	VDD	AR2	PCIe_DS24_Rx_5_P
L2	VDD	AC25	VSS	AR2	PCIe_DS24_Rx_4_P
L2	VSS	AC24	VDD	AR2	VSS
L2	VDD	AC23	VSS	AR2	PCIe_DS24_CLK_P
L2	VSS	AC22	VDD	AR2	VSS
L2	VDD	AC21	VSS	AR2	PCIe_DS24_Rx_3_P
L2	VSS	AC20	VDD	AR2	PCIe_DS24_Rx_2_P
L1	VDD	AC19	VSS	AR1	VSS
L1	VDD	AC18	VSS	AR1	PCIe_DS24_Rx_1_P
L1	VSS	AC17	VDD	AR1	PCIe_DS24_Rx_0_P
L1	VDD	AC16	VSS	AR1	VSS
L1	VSS	AC15	VDD	AR1	PCIe_DS13_Rx_7_P
L1	VDD	AC14	PCIe_DS5_VDDA_1	AR1	PCIe_DS13_Rx_6_P

L1	USB_VP_2_2	AC13	PCIe_DS5_VDDA_2	AR1	VSS
L1	USB_VPTX_1	AC12	PCIe_DS5_VTT_1	AR1	PCIe_DS13_Rx_5_P
L1	USB_VPTX_0	AC11	SATA_VPTX2	AR1	PCIe_DS13_Rx_4_P
L1	USB_VDD180_0	AC10	SATA_2_RESREF	AR1	VSS
L	USB_VDD330_0	AC9	VSS	AR	PCIe_DS13_CLK_P
L	AC97_SYNC	AC8	SATA_0_Tx_P	AR	VSS
L	AC97_SDATA_O	AC7	SATA_0_Tx_N	AR	PCIe_DS13_Rx_3_P
L	VDDQ	AC6	VSS	AR	PCIe_DS13_Rx_2_P
L	AC97_SDATA_I	AC5	VSS	AR	VSS
L	LFRAMEN	AC4	SATA_0_Rx_P	AR	PCIe_DS13_Rx_1_P
L	LAD_3	AC3	SATA_0_Rx_N	AR	PCIe_DS13_Rx_0_P
L	VSS	AC2	VSS	AR	VSS
L	VSS	AC1	VSS	AR	VSS
M	VSS	AD36	DDR3_DQ0	AT3	VSS
M	DDR3_DQ33	AD35	DDR3_DQ1	AT3	VSS
M	DDR3_DQ34	AD34	DDR3_DQ2	AT3	PCIe_US_Rx_2_N
M	VSS	AD33	VSS	AT3	VSS
M	DDR3_DQS_4_N	AD32	DDR3_DQ3	AT3	PCIe_US_Rx_1_N
M	DDR3_DQS_4_P	AD31	DDR3_DQ6	AT3	PCIe_US_Rx_0_N
M	VSS	AD30	VSS	AT3	VSS
M	DDR3_DQ35	AD29	UART_TX	AT2	PCIe_DS24_Rx_7_N
M	DDR3_DQ38	AD28	CPU_LINKUP	AT2	PCIe_DS24_Rx_6_N
M	VSS	AD27	UART_RTS	AT2	VSS
M	VDD	AD26	VSS	AT2	PCIe_DS24_Rx_5_N
M	VSS	AD25	VDD	AT2	PCIe_DS24_Rx_4_N
M	VDD	AD24	VSS	AT2	VSS
M	VSS	AD23	PCIe_US_VDDA_1	AT2	PCIe_DS24_CLK_N
M	VDD	AD22	PCIe_US_VDDA_2	AT2	VSS
M	VSS	AD21	PCIe_US_VDDA_3	AT2	PCIe_DS24_Rx_3_N
M	VDD	AD20	PCIe_DS24_VDDA_1	AT2	PCIe_DS24_Rx_2_N
M	VSS	AD19	PCIe_DS24_VDDA_2	AT1	VSS
M	VSS	AD18	PCIe_DS24_VDDA_3	AT1	PCIe_DS24_Rx_1_N
M	VDD	AD17	PCIe_DS13_VDDA_1	AT1	PCIe_DS24_Rx_0_N
M	VSS	AD16	PCIe_DS13_VDDA_2	AT1	VSS
M	VDD	AD15	PCIe_DS13_VDDA_3	AT1	PCIe_DS13_Rx_7_N
M	VSS	AD14	PCIe_DS5_VDDA_0	AT1	PCIe_DS13_Rx_6_N
M	USB_VP_2_1	AD13	PCIe_DS5_VDDQA_3	AT1	VSS
M	USB_VP_1_2	AD12	PCIe_DS5_VTT_0	AT1	PCIe_DS13_Rx_5_N
M	USB_VP_0_2	AD11	SATA_VPTX1	AT1	PCIe_DS13_Rx_4_N
M	VSS	AD10	VDD	AT1	VSS
M	LAD_2	AD9	VSS	AT	PCIe_DS13_CLK_N
M	AC97_RESET	AD8	VSS	AT	VSS
M	LAD_1	AD7	VSS	AT	PCIe_DS13_Rx_3_N
M	LAD_0	AD6	SATA_1_RESREF	AT	PCIe_DS13_Rx_2_N
M	VSS	AD5	VSS	AT	VSS
M	SERIRQ	AD4	VSS	AT	PCIe_DS13_Rx_1_N
M	AC97_BIT_CLK	AD3	VSS	AT	PCIe_DS13_Rx_0_N
M	CFG_LDRQ2N	AD2	PCIe_DS5_Rx_0_P	AT	VSS
M	CFG_LDRQ1N	AD1	PCIe_DS5_Rx_0_N	AT	VSS

13.4. 焊接要求

MSD(器件湿敏)等级：4 级；

密封袋内有效期（小于 40°C 并且小于 90％相对湿度）：12 个月； 峰值封装温度：225°C；打开封装袋后，芯片必须在如下条件进行回流焊或其它高温处理：

a) 安装后的 72 小时内必须在小于 30°C 及不超过 60％相对湿度条件下，或者 b) 保存在小于 10％相对湿度条件下；

在如下条件下，芯片安装后需要进行烘烤：

a) 在 23±5°C 范围内且相对湿度大于 10%； b) a)或 b)不满足；

需要烘烤时，芯片需要在 125±5°C 范围内烘烤 48 小时。

附录 A（存控训练）

A.1 显存存控数据训练

在 PHY 初始化完成以及 SDRAM 初始化完成以后，且在正常读写 SDRAM 颗粒之前， 需要通过数据训练来调整 PHY 内部的延迟线，保证信号的正确性。

数据训练分为以下几个步骤：

1、 Loopback 测试：简单的回路测试，测试 PHY 的内部读写数据通路，LOOPBACK 需要在 uMCTL 内状态机为 ACCESS 时进行；
2、 写平衡调整：将 DQS 和 CK 对齐，该功能集成在 PHY 中，通过 IO 写置开始标志， 然后通过 IO 读判断结束标志；
3、 读平衡训练：通过 MPR 寄存器读来使读 DQS 对齐到读数据眼图的中间；
4、 差拍写平衡调整：调整各 DATX8 之间的差拍；
5、 写数据以及掩码的位偏斜调整：调整一个字节通道内 8 位读数据之间的偏斜，并将 写 DQS 对齐到写数据眼图的中间；
6、 读数据位偏斜调整：调整一个字节通道内 8 位读数据之间的偏斜，并将读 DQS 对 齐到读数据眼图的中间；
7、 DQS gating 训练：调整 DQS 门控的延迟以及宽度。

由于数据训练会影响到内建 PVT 漂移补偿逻辑，所以在数据训练开始前，需要写 GCR[4:0]以及 CCR[6:5]为 0 来禁止 PVT 漂移补偿，数据训练完成后再打开。

A.1.1 关闭更新使能和延迟线漂移补偿

1、 等待寄存器 STAT[2:0]==0x3
2、 写 CCR[6:5]为 0，关闭更新使能；
3、 写 8 个 DATX8 的 GCR[4:0]为 0，关闭延迟线漂移补偿；
4、 读出 TREFI 的值并存储；
5、 写 TREFI 为全 0，关闭刷新功能；
6、 读出 TZQCSI 的值并存储；
7、 写 TZQCSI 为全 0，关闭 ZQCS；

A.1.2 Loopback 测试

1、 配置 DSCR 寄存器：DSHOLD=0；DVEXT=0；DVWIDTH=4；DVLAT=WL；DSWIDTH=5；DSLAT=WL-1；
2、 配置 QSCR 寄存器：QSHOLD=0；QSEXT=0；QSWIDTH=4
3、 配置 QSLR0，QSLR1 寄存器：QSLAT0~7=WL
4、 写 PHYCR[4]为 1，打开 Loopback 使能；
5、 读出 8 个 DATX8 的 WDSDR 以及 RDSDR 的值并存储；

6、 配置 8 个 DATX8 的 WDSDR 为 0x1f
7、 配置 8 个 DATX8 的 RDSDR 为 0；
8、 写 8 个 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑；
9、 写 8 个 DTAX8 的 ACR[9]为 1，清复位；
10、 写 SFRR[0]为 0，复位对齐缓冲；
11、 写 SFRR[0]为 1，清复位；
12、 配置 DTUCFG：dtu_enable=1；dtu_generate_random=0；dtu_target_lane=0； 13、 写 DTUWACTL 寄存器为全 0；
14、 写 DTURACTL 寄存器为全 0；
15、 写 DTUWD0~DTUWD3，写入测试数据；
16、 写 DTUWDM 为 0x0；
17、 写 DTUECTL[0]为 1，开始测试；
18、 轮询 DTUECTL[0]，直到其为 0；
19、 查询 DTUPDES 寄存器，看测试是否出错；
20、 写 DTUCFG[13:10]为 1，变更测试字节通道为 Byte1；
21、 写 DTUECTL[0]为 1，开始测试；
22、 轮询 DTUECTL[0]，直到其为 0；
23、 查询 DTUPDES 寄存器，看测试是否出错；
24、 写 DTUCFG[13:10]为 2，变更测试字节通道为 Byte2；
25、 写 DTUECTL[0]为 1，开始测试；
26、 轮询 DTUECTL[0]，直到其为 0；
27、 查询 DTUPDES 寄存器，看测试是否出错；
28、 写 DTUCFG[13:10]为 3，变更测试字节通道为 Byte3；
29、 写 DTUECTL[0]为 1，开始测试；
30、 轮询 DTUECTL[0]，直到其为 0；
31、 查询 DTUPDES 寄存器，看测试是否出错；
32、 写 DTUCFG[13:10]为 4，变更测试字节通道为 Byte4；
33、 写 DTUECTL[0]为 1，开始测试；
34、 轮询 DTUECTL[0]，直到其为 0；
35、 查询 DTUPDES 寄存器，看测试是否出错；
36、 写 DTUCFG[13:10]为 5，变更测试字节通道为 Byte5；
37、 写 DTUECTL[0]为 1，开始测试；
38、 轮询 DTUECTL[0]，直到其为 0；
39、 查询 DTUPDES 寄存器，看测试是否出错；
40、 写 DTUCFG[13:10]为 6，变更测试字节通道为 Byte6；
41、 写 DTUECTL[0]为 1，开始测试；
42、 轮询 DTUECTL[0]，直到其为 0；
43、 查询 DTUPDES 寄存器，看测试是否出错；
44、 写 DTUCFG[13:10]为 7，变更测试字节通道为 Byte7；
45、 写 DTUECTL[0]为 1，开始测试；
46、 轮询 DTUECTL[0]，直到其为 0；
47、 查询 DTUPDES 寄存器，看测试是否出错；

48、 配置 DTUCFG，写 dtu_enable=0，关闭 DTU；

49、 配置 DSCR 寄存器：DVWIDTH=5；DVLAT=WL-1；
50、 配置 QSCR 寄存器：QSWIDTH=7；
51、 配置 QSLR0~1 寄存器：QSLAT0~7=RL-1
52、 写 PHYCR[4]=0，关闭 Loopback；
53、 恢复 8 个 DATX8 的 WDSDR 以及 RDSDR 值为之前的读出值；
54、 写 8 个 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑；
55、 写 8 个 DTAX8 的 ACR[9]为 1，清复位； 56、 Loopback 测试结束。

A.1.3 写平衡

该步骤通过增加写 DS 的延迟，使写的 DS 与 CK 对齐；使用的方法是，在 SDRAM 处， 使用 DQS 采样 CK，并将值返回给 PHY，由 PHY 进行处理。该逻辑集成在 PHY 中，只需 要写相关寄存器进行触发即可。在该过程中，PHY 调整了各 DATX8 的 WLDR0 寄存器(影 响 DS 和 D 的延迟)的值。

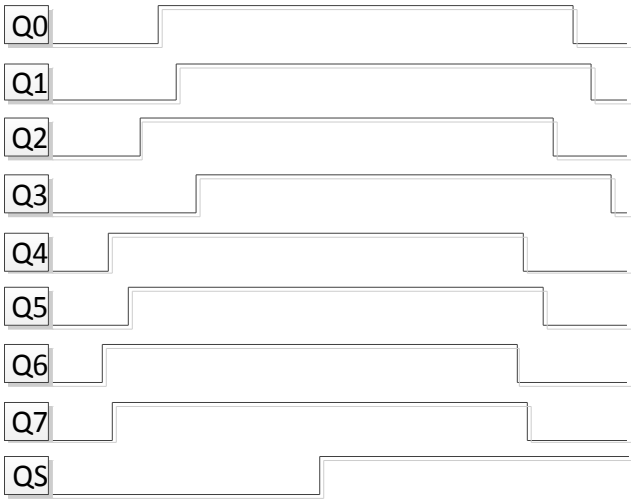
1、 写 SCTL 为 1，使 uPCTL 状态机进入 Config 状态；
2、 轮询 STAT 寄存器直到其为 1；
3、 写 DFIODTCFG[4]为 1，使 ODT 常开；
4、 配置 MCMD 寄存器，发送 PREA 命令；
5、 轮询 MCMD 直到命令完成；
6、 配置 MCMD 寄存器，发送 MR2 命令，写 MR2[10:9]为 0，关闭动态 ODT；
7、 轮询 MCMD 直到命令完成；
8、 配置 MCMD 寄存器，发送 MR1 命令，写 MR1[12]为 1，MR1[7]为[1]，关闭 DQ/DQS 输出，同时打开写平衡；
9、 轮询 MCMD 直到命令完成；
10、 配置 MCMD 寄存器，发送 MR1 命令，写 MR1[12]为 0，打开 DQ/DQS 输出；
11、 轮询 MCMD 直到命令完成；
12、 配置 ACGCR[2]为 1，进行 PHY 写平衡；
13、 轮询 ACGSR，判断 WL 是否完成以及是否正确；
14、 配置 ACGCR[2]为 0，退出 WL 状态；
15、 配置 MCMD 寄存器，发送 MR1 命令，写 MR1[12]为 1，MR1[7]为 0，使存储 条退出写平衡状态；
16、 轮询 MCMD 直到命令完成；
17、 配置 MCMD 寄存器，发送 MR1 命令，写 MR1[12]为 0，打开 DQ/DQS 输出；
18、 轮询 MCMD 直到命令完成；
19、 写 DFIODTCFG[4]，恢复 ODT 默认值；
20、 写 SCTL 为 2，使 uMCTL 状态机进入 Access 状态；
21、 轮询 STAT 寄存器直到其为 3；
22、 写 8 个 DATX8 的 ACR[9]为 0，复位 PHY 的高速逻辑；
23、 写 8 个 DATX8 的 ACR[9]为 1，清复位；

A.1.4 使用 MPR 读进行读数据眼图训练

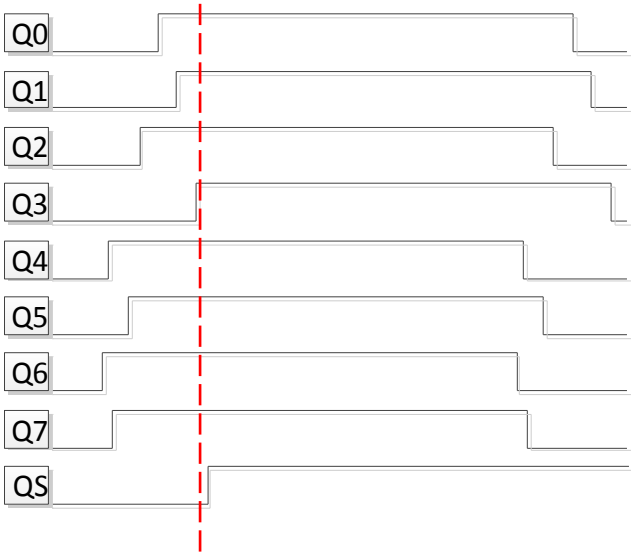
该步骤通过读取 SDRAM 内特定寄存器的固定值（该值与 DIMM 条相关）来进行读平 衡以及眼图调整。在该过程中，调整了各 DATX8 的 RDSDR、RDBDMR 以及 RDBDR0~7 的值。

各个步骤的示意图如下：

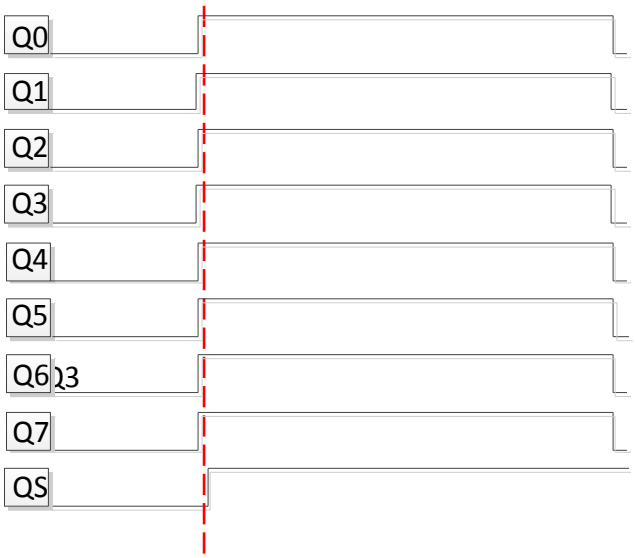
PHY 初始化后，RDSDR 会被写入等效半个 DDR 周期的延迟值，初始状态下的 Q[7:0] 和 QS 的示意图：



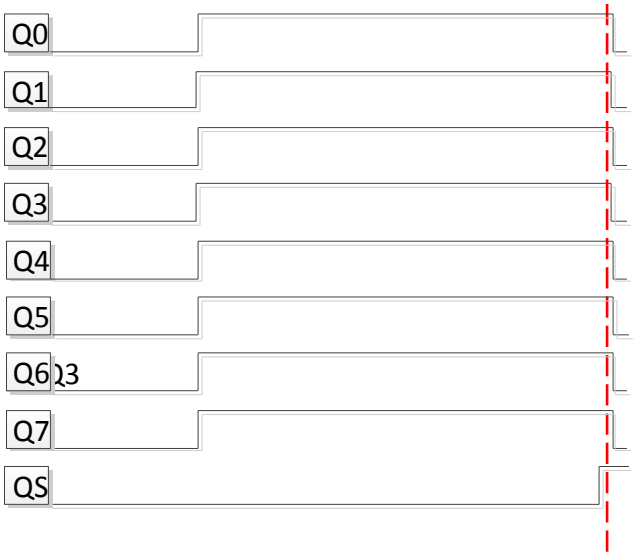
减小 RDSDR，找到左边沿：



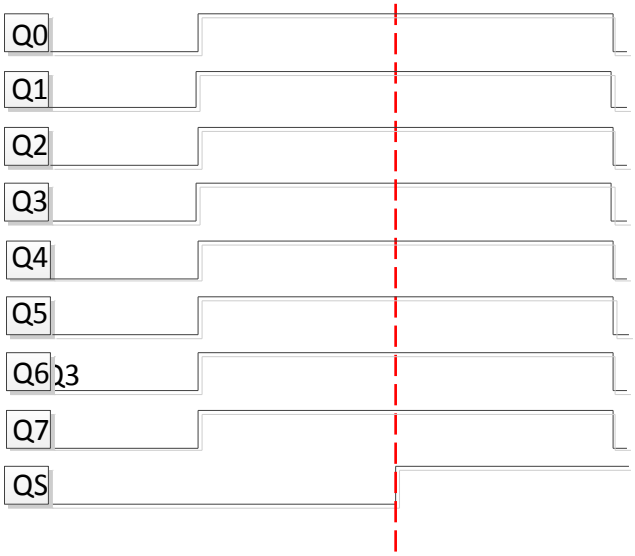
增加各 RDBDR 的值，使所有 Q 的左边沿都对齐到 QS：



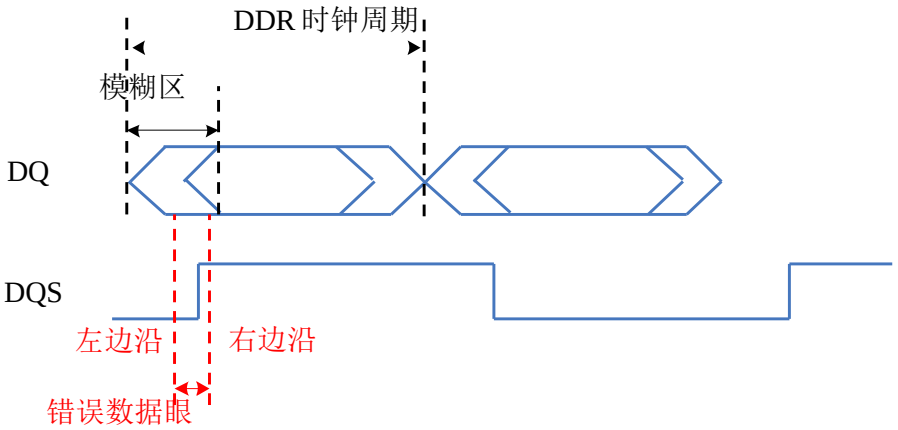
增大 RDSDR 的值，找到右边沿：



将左边沿与右边沿取平均，最佳化眼图：



注意：在上述过程中，QS 的上升/下降沿可能要经过数据 Q 的模糊区，在模糊区内，采 样结果是不确定的，这导致找到的数据眼可能存在于模糊区内，而这种结果是不正确的，致 使训练失败，如下图所示：



为了解决上述情况，有必要对所有的 RDSDR 值进行遍历，找到最大的数据眼，然后选 取合适的 RDSDR 值，使 QS 置于数据眼的中间。但是，同时考虑到 RDSDR 的取值有 1024种，全部搜寻会使数据训练的时间较长。而在找到一个能使读数据正确的 QS 延迟值以后（假设此时的 RDSDR 值为 rdsdr_a），最多再将 QS 推迟一个 DDR 时钟周期，即可结束搜寻， 所以，可将 RDSDR 的搜寻上限设为：rdsdr_max = rdsdr_a + tddr_clk /tdpr（其中，tddr_clk 是 DDR_CLK 的时钟周期，单位 ps； tdpr 是一个单位的 RDSDR 所代表的延迟，在软环境 下为 12ps，实际环境可能会变化。）。

即，改进后的算法是，在 0~rdsdr_max 之间搜寻可采样数据 Q 正确的 QS 的最大数据眼， 选取该数据眼的左边沿 RDSDR 值作为 QS 的延迟值；调整 8 个 bit 的 RDBDR 值，使 Q 均 对其到 QS；增加 RDSDR，查找 QS 的右边沿；选取左边沿与右边沿 RDSDR 的平均值作为最终结果。

需要注意的是，如果最大数据眼的大小小于一定的宽度，即可认定数据训练失败，该宽 度需要在工程调试中获得。另外，在查找 QS 左边沿以及将 Q 对齐到 QS 时，需要对同一延 迟值多次测试，这样才能获得一个较好的结果。

least_eye_width=0x20 //根据工程经验选择
读 MDTPR 寄存器值：lat_eq_ddr_clk=MDTPR；
bit_pattern[31:0]=0xff00ff00；(与具体颗粒 MRR 的数据模式有关) 写
QSCR[6:3]=7；
写 SCTL 寄存器为 1，使 uMCTL 进入 Config 状态； 轮询
STAT 寄存器直到其为 1；
配置 MCMD 寄存器，发送 PREA 命令； 轮询
MCMD 直到命令完成；
通过 MCMD 发送 MR3 命令，写 MR3[2]=1，使 SDRAM 进入 MPR 状态； 轮询
MCMD 直到命令完成；
写 8 个 DATX8 的 RDSDR 值为 0；
写 8 个 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑； 写 8
个 DTAX8 的 ACR[9]为 1，清复位；
for(DATX8=0;DATX8<8;DATX8++)
{
写 MRRCFG0.mrr_byte_sel 选择读平衡的字节通道；
biggest_left_edge=0；
biggest_right_edge=0;
biggest_width=0;
current_left_edge=0;
current_right_edge=0;
current_width=0;
for(rdsdr=0;rdsdr< 0x400;rdsdr++) //0x400 为 RDSDR 可以写入的最大值
{
配置 RDSDR 寄存器，RDSDR=rdsdr；
写 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑； 写
DTAX8 的 ACR[9]为 1，清复位；
配置 MCMD 寄存器，发送 MRR 命令； 轮询
MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if(mrr0==bit_pattern
&&mrr1==bit_pattern)
{
rdsdr_max=rdadr+lat_eq_ddr_clk； break；

```
}
}
find_left_edge=1；
while(rdsdr<rdsdr_max)       //查找最大数据眼
{
    if(find_left_edge)
    {
    配置 RDSDR 寄存器，RDSDR=rdsdr；
写 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑； 写
DTAX8 的 ACR[9]为 1，清复位；
配置 MCMD 寄存器，发送 MRR 命令； 轮询
MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if(mrr0==bit_pattern
&&mrr1==bit_pattern)
{
        current_left_edge=rdsdr；
        find_left_edge=0；
    }
}
else            //find_right_edge
{
    配置 RDSDR 寄存器，RDSDR=rdsdr；
写 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑； 写
DTAX8 的 ACR[9]为 1，清复位；
配置 MCMD 寄存器，发送 MRR 命令； 轮询
MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if(mrr0!=bit_pattern ||
mrr1!=bit_pattern)
{
    current_right_edge=rdsdr；
    current_width = current_right_edge – current_left_edge；
    if(current_width>biggest_width)
    {
        biggest_left_edge=current_left_edge；
```

```
            biggest_right_edge=current_right_edge；
            biggest_width=current_width；
}
      find_right_edge=1；
}
rdsdr++；
}
}
if(biggest_width<least_eye_width) report
      error；

for(i = biggest_left_edge;i < biggest_left_edge+ biggest_width; i ++)
{
test_result=1；
配置 RDSDR 寄存器，RDSDR=i；
写 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑； 写
DTAX8 的 ACR[9]为 1，清复位；
for(j=0;j<8;j++)     //为了保证算法的健壮性，对一个值测试 8 次
{
配置 MCMD 寄存器，发送 MRR 命令； 轮询
MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if(mrr0!=bit_pattern ||
mrr1!=bit_pattern)
{
      test_result=0； break；
}
}
if(test_result==1)
{
biggest_left_edge=i；break；
}
}

//将 Q 对齐到 QS
pass[7:0]=0xff while(|pass)
{
```

```
for(BIT=0;BIT<8;BIT++)
{
if(pass[BIT]==1)
将 RDBDR[BIT]寄存器值加 1；
}
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
for(i=0;i<8;i++)        //为了保证算法的健壮性，对一个值测试 8 次
{
    配置 MCMD 寄存器，发送 MRR 命令； 轮询
MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据；
test_result0 = mrr0 xor bit_pattern；
test_result1 = mrr1 xor bit_pattern；
test_result=test_result0 | test_result1；
err_bit[7:0]=test_result[7:0] | test_result[15:8] | test_result[23:16] |test_result[31:24]
for(BIT=0;BIT<8;BIT++)
{
    if(err_bit[BIT]==1 && pass[BIT]==0) report
        error；
    else if(err_bit[BIT]==1)
    {
        将 RDBDR[BIT]寄存器值减 1；
        pass[BIT]=0；
    }
}
}

}
查找最大 RDBDR 值，写入 RDBDMR 寄存器中；

//查找 QS 右边沿
写 RDSDR 寄存器值 biggest_right_edge；
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位； 配置
MCMD 寄存器，发送 MRR 命令；
轮询 MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
```

MRRSTAT1：mrr1=MRRSTAT1；

```
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if
(mrr0!=bit_pattern ||mrr1!=bit_pattern)
{
    while(RDSDR>biggest_left_edge+ least_eye_width)
    {
        将 RDSDR 寄存器值减 1；
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位； 配置
MCMD 寄存器，发送 MRR 命令；
轮询 MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if
(mrr0==bit_pattern &&mrr1==bit_pattern)
{
    biggest_right_edge=RDSDR； break；
}
if(RDSDR==biggest_left_edge+least_eye_width) report
    error；
}
}
else
{
      while(1)
      {
        将 RDSDR 寄存器值加 1；
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位； 配置
MCMD 寄存器，发送 MRR 命令；
轮询 MCMD 直到命令完成；
写 SFRR[0]=0，复位对齐缓冲； 写
SFRR[0]=1，清复位；
读出 MRRSTAT0：mrr0=MRRSTAT0； 读出
MRRSTAT1：mrr1=MRRSTAT1；
写 MRRRSTAT0 以及 MRRSTAT1 为全 0，清除之前的数据； if
(mrr0!=bit_pattern ||mrr1!=bit_pattern)
{
biggest_right_edge=RDSDR-1； break；
}
}
}
```

RDSDR=(biggest_left_edge + biggest_right_edge)； 通过
DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
}
配置 MCMD 寄存器，发送 MR3 命令，写 MR3[2]=0，使 SDRAM 退出 MPR 状态； 轮询
MCMD 直到命令完成；
配置 MCMD 寄存器，发送 PREA 命令； 轮询
MCMD 直到命令完成；
写 SCTL=2，使 uMCTL 进入 ACCESS 状态； 轮询
STAT 寄存器直到其为 3。

A.1.5 差拍写平衡调整

由于布线的原因，DQS 线可能比 CK 线的延迟大，也可能比 CK 线的延迟小，而在进行 第一步写平衡的时候，由于是增大 DS 的延迟，使 DS 对齐 CK，所以当某个 DATX8 的 DQS 的延迟大于 CK 延迟，而另一些 DATX8 的 DQS 的延迟小于 CK 延迟时，各 DATX8 之间会 产生差拍。

差拍写平衡调整是为了调整第一步写平衡造成的 DATX8 之间的差拍，调节的寄存器是 S2WLR(存控内的写数据延迟调整)。

写各 DATX8 的 ACR[9]为 0，复位 PHY 的高速逻辑； 写各
DTAX8 的 ACR[9]为 1，清复位；
写 S2WLR 为全 0；
配置 DTUCFG：dtu_enable=1；dtu_generate_random=0；dtu_target_lan=0； 写
DTUWACTL、DTURACTL 为全 0；
写 DTUWD0~DTUWD3，写入测试数据； 写
DTUWDM 为全 0；
写 DTUECTL 为 1，开始测试；轮询
DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
写 DTUCFG[13:10]为 1，变更测试字节通道为 Byte1； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
写 DTUCFG[13:10]为 2，变更测试字节通道为 Byte2； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
写 DTUCFG[13:10]为 3，变更测试字节通道为 Byte3； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；

写 DTUCFG[13:10]为 4，变更测试字节通道为 Byte4； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
写 DTUCFG[13:10]为 5，变更测试字节通道为 Byte5； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
写 DTUCFG[13:10]为 6，变更测试字节通道为 Byte6； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
写 DTUCFG[13:10]为 7，变更测试字节通道为 Byte7； 写
DTUECTL[0]为 1，开始测试；
轮询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看测试是否出错；
更改 S2WLR 的值，将上述步骤中出错的字节通道的差拍调整数加 1； 更改
DTUCFG[13:10]为出错的字节通道，对其进行测试；
更改 S2WLR 的值，将仍然出错的字节通道的差拍调整值加 1； 对未完成的字节通
道进行测试；

当某个字节通道的差拍调整值为 3，且仍然出错时，训练失败，此时需要调整 WL 的值 来找到合适的窗口位置，并重新进行差拍调整测试；

所有 DATX8 的差拍调整完成后，写 DTUCFG.dtu_enable=0，关闭 DTU； 写 8
个 DTAX8 的 ACR[9]为 0，复位 PHY 的高速逻辑；
写 8 个 DTAX8 的 ACR[9]为 1，清复位； 写
SCTL=1，使 uMCTL 进入 Config 状态； 轮询
STAT 寄存器直到其为 1；
配置 MCMD 寄存器，发送 PREA 命令； 轮询
MCMD 直到命令完成；
写 SCTL=2，使 uMCTL 进入 Access 状态； 轮询
STAT 寄存器直到其为 3。

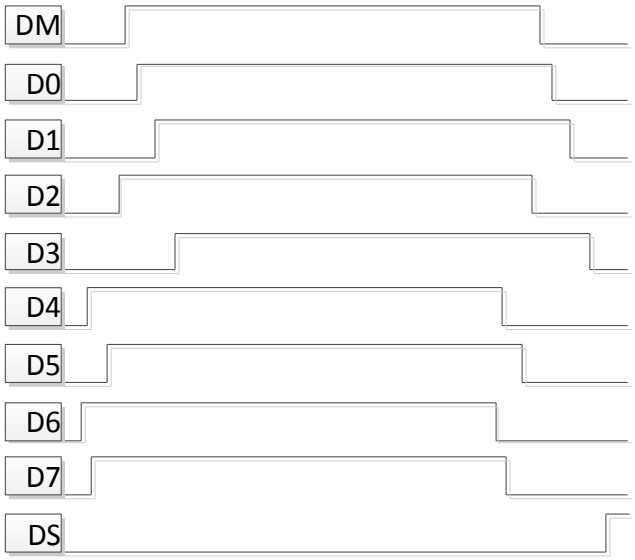
A.1.6 写数据以及掩码的位偏斜调整

A.1.6.1 写数据偏斜调整基本原理

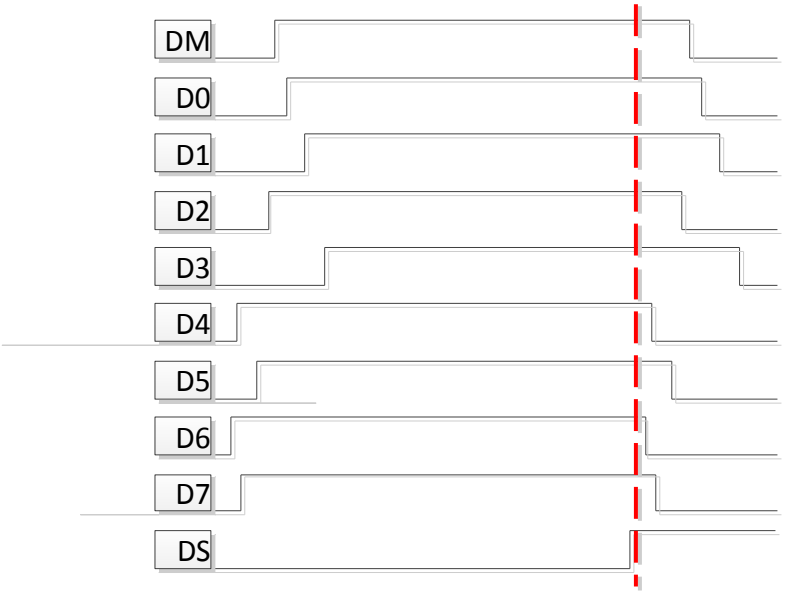
该步骤主要调节写数据一个 DATX8 内各 bit 之间的延迟差异，最优化信号眼图，使 DS 置于眼图的中间，使信号质量达到最佳。各个步骤的

示意图如下：

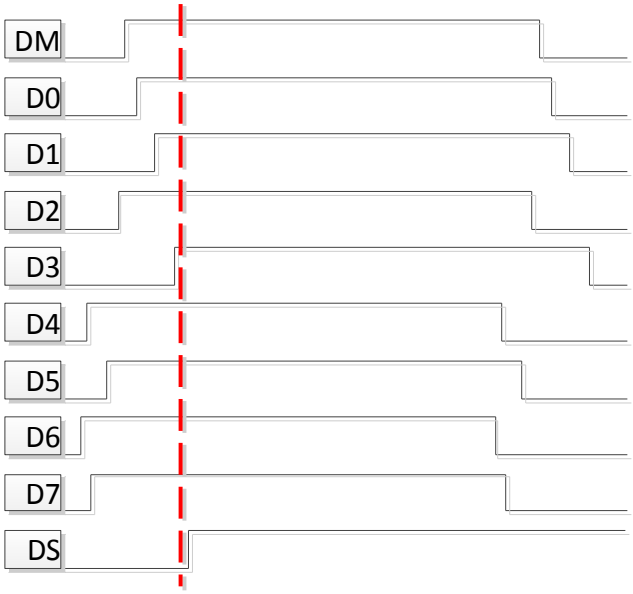
将 WDSDR 清零后，初始状态下的 DM，D[7:0]和 DS 的示意图：



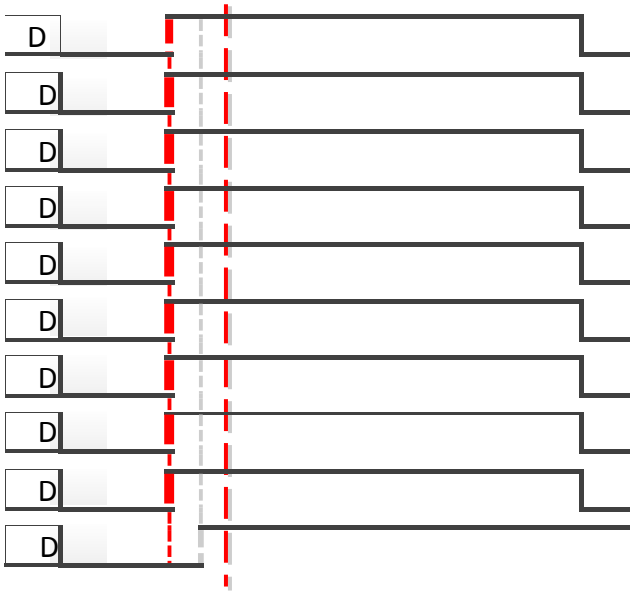
增大 WDSDR 的值，DS 相对于 D 以及 DM 信号的位置往左移，先找到右边沿：



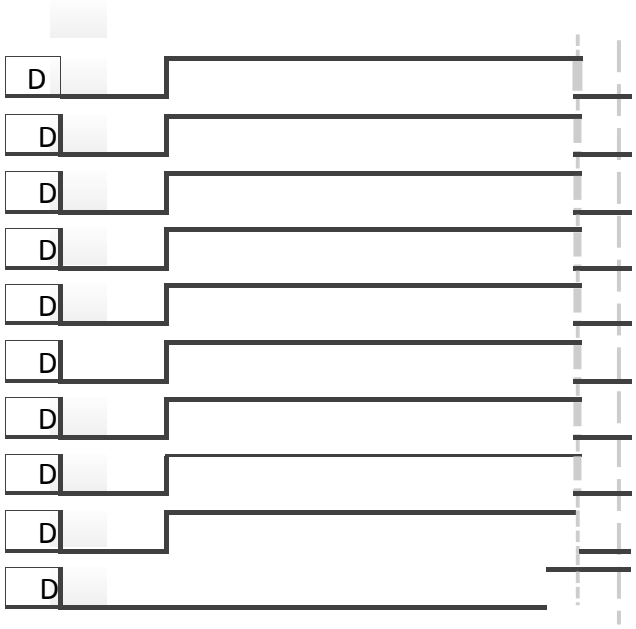
继续增加 WDSDR 的值，找到左边沿后，示意图如下：



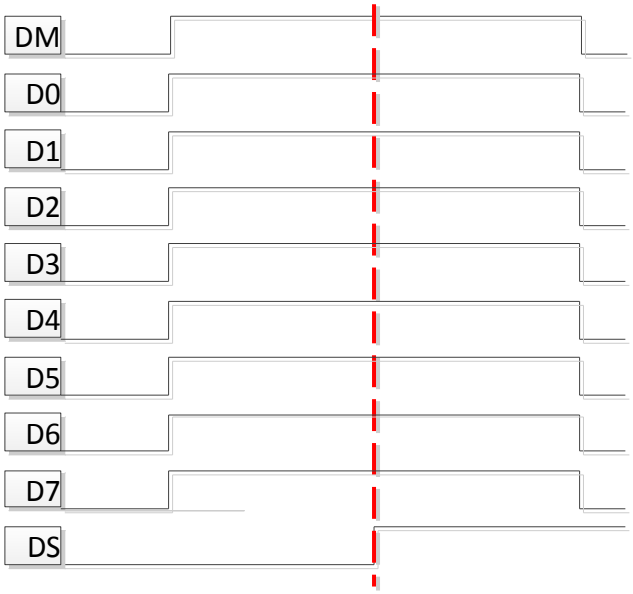
针对每个比特位，增加 WDBDR 寄存器的值，即该比特位的延迟，直到所有比特 都与 DS 对齐，示意图如下：



减小 WDSDR 的值，找到信号的右边沿，示意图如下：



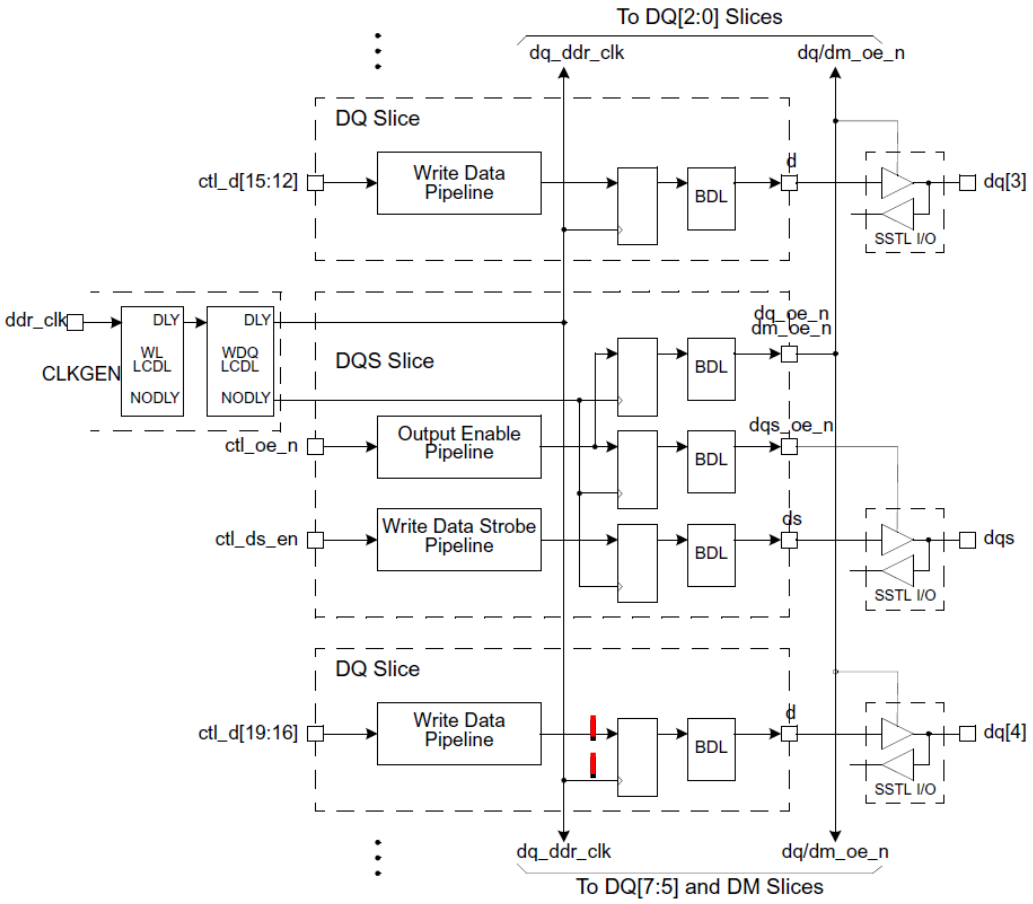
将 WDSDR 的左边沿取值与右边沿取值取平均，将 DS 置于信号的中央，示意图如 下：



A.1.6.2 涉及的延迟线

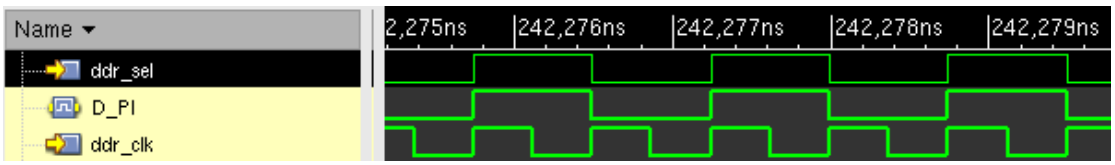
WDQ LCDL 是用于调整一个 datx8 内所有 Q 的延迟的延迟线，其在 PHY 中的位置如下 图所示：

Figure 3-11 DATX8 Write Path



可以发现，WDQ LCDL 延迟线在 dq_ddr_clk 的路径上，其延迟值直接作用于 dq_ddr_clk，而对于时钟来说，推迟一个周期后的时钟与原时钟在波形上没有什么变化。 从上图还可以看出，WDQ LCDL 调整数据 D 的延迟的原理是，通过调整触发器的输入时钟 dq_ddr_clk 来使数据线上的数据产生延迟，所以，在使用 WDQ LCDL 调整数据延迟时，不会超过一个 DDR 时钟周期。

实际上，在 DQ Slice 的 Write Data Pipeline 内部，还通过 dq_ddr_clk 对 SDR 时钟 域的数据进行采样，将数据由 SDR 时钟域转换到 DDR 时钟域，所以在图中的红色虚线处，即触发器的输入端，时钟和数据是同步的，如下图：

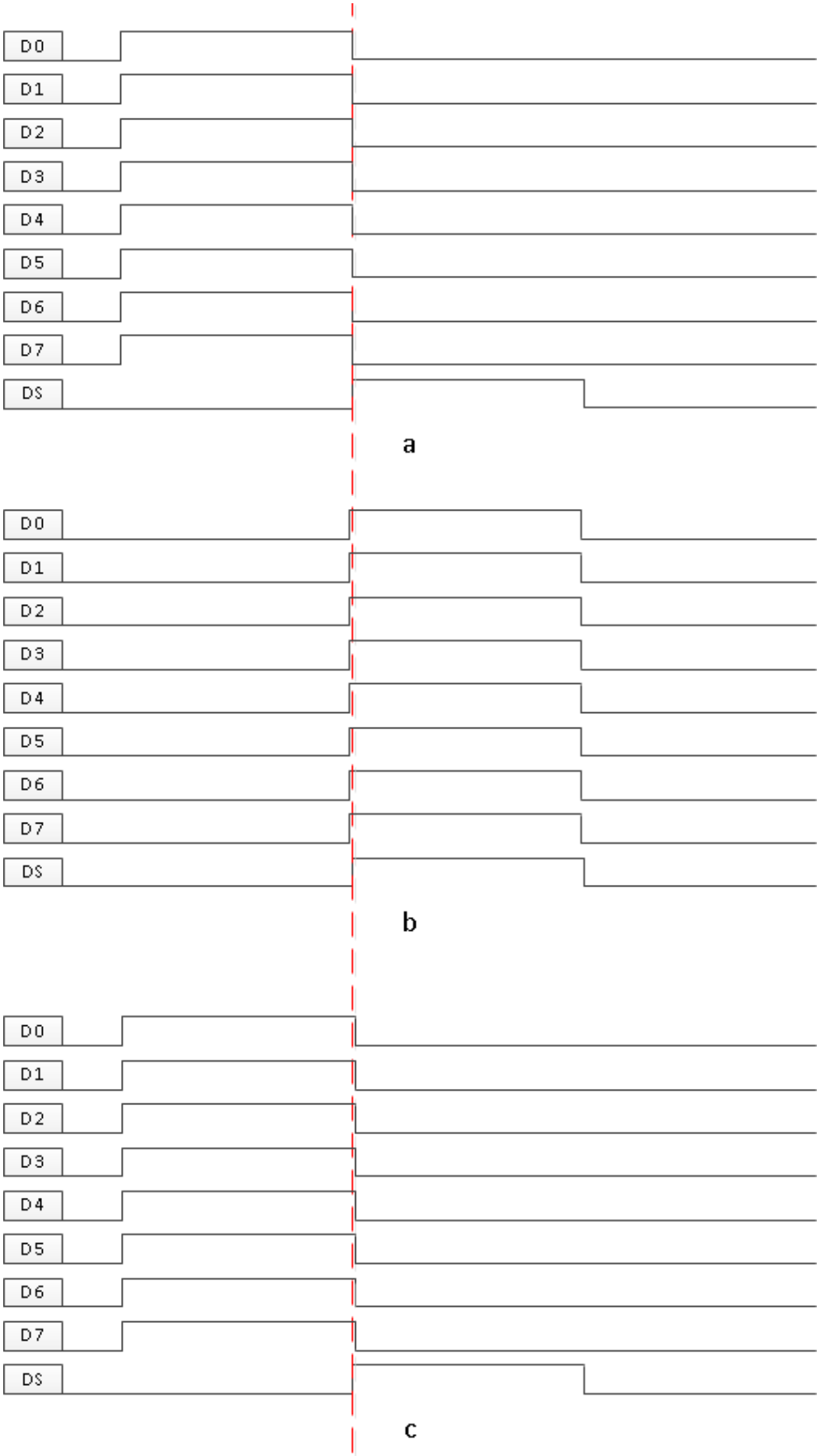


WBDL 用于调整一个 datx8 内某条数据线上的延迟，其在 PHY 中的位置如上图所示。 可以看出，WBDL 位于数据线 D[7:0]上，直接影响了数据线的延迟，所以，在增大 WBDL 延迟线取值的同时，会一直增大该条数据线的延迟，其调整能力是 27 个延迟单元，每个单元的延迟在软环境下是 12ps。

A.1.6.3 软环境下碰到的问题

在软环境下进行写数据偏斜调整训练时，发现存在下边的情况：

下图中，图 a 是 LCDL 为 0 时的波形，图 b 是 LCDL 延迟值即将超过一个 DDR 周期的 波形，图 c 是 LCDL 延迟值为图 b 中的 LCDL 延迟值+1 后，正好超过一个 DDR 周期的波形：

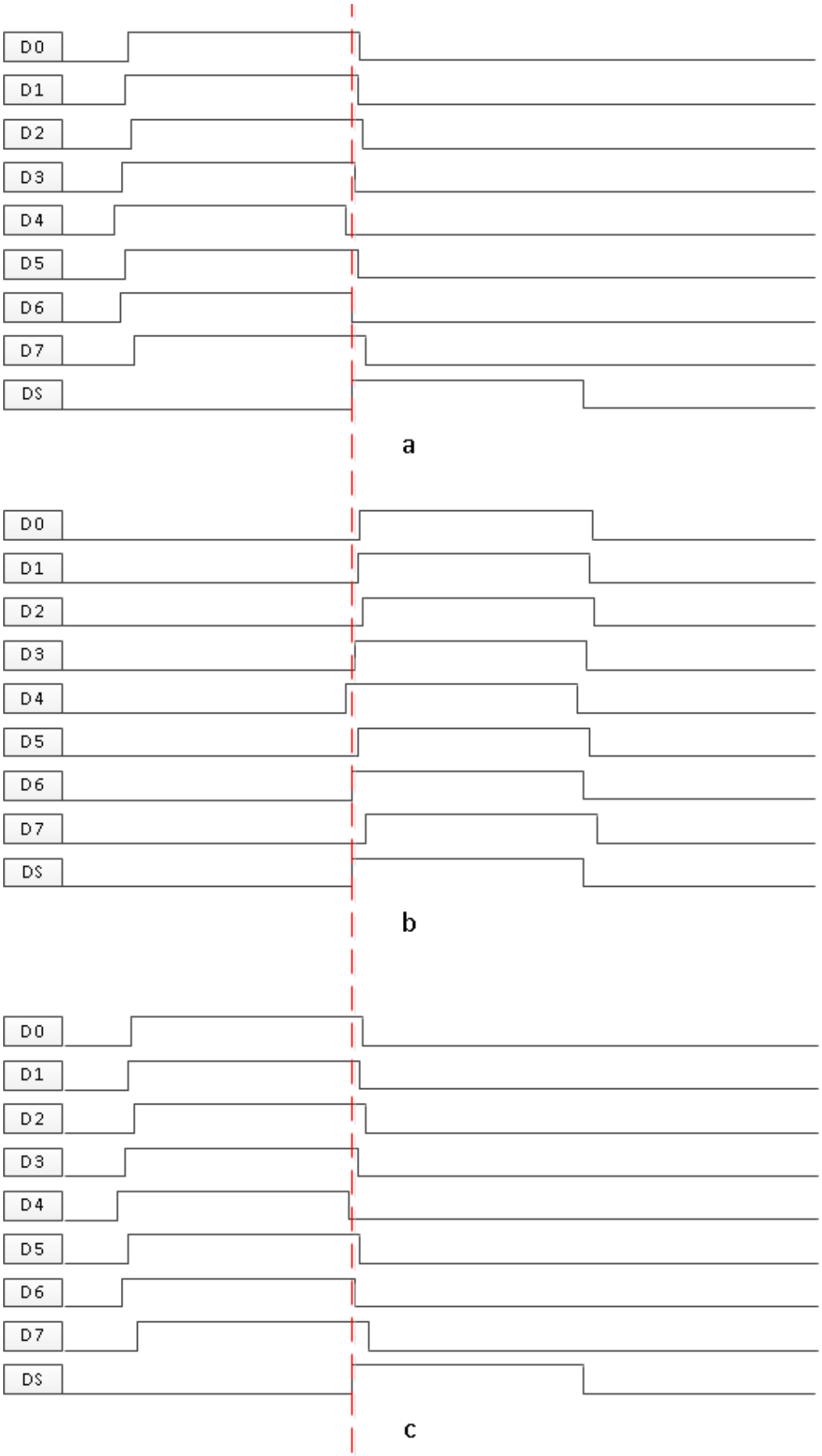

D0
D1
D2
D3
D4
D5
D6
D7
DS
a
D0
D1
D2
D3
D4
D5
D6
D7
DS
b
D0
D1
D2
D3
D4
D5
D6
D7
DS
c

上图中可以看出，无论 LCDL（即 WDSDR）延迟值怎么取，结果都是正确的，于是导 致仿真的失败。

A.1.6.4 真实环境下的情况

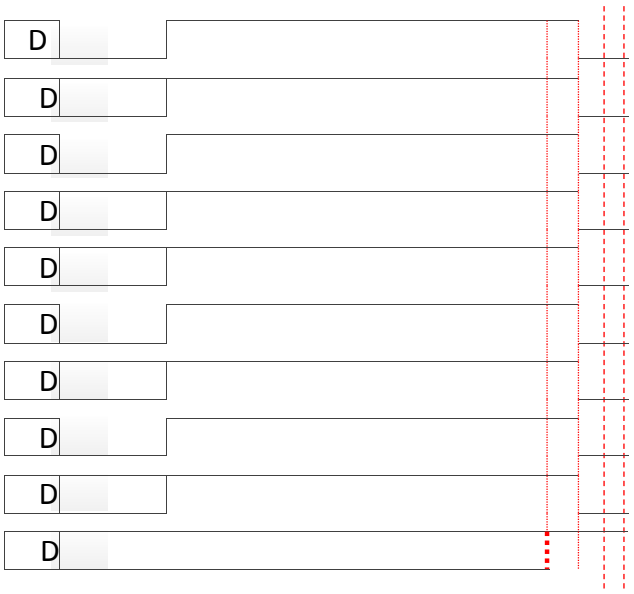
考虑到真实情况下，由于布线差异，DS 相对于 D 的延迟有可能有以下三种情况：

1、 DS 延迟值在 D0~D7 之间 此时，可能会出现的情况如下图所示：


D0
D1
D2
D3
D4
D5
D6
D7
DS
a
D0
D1
D2
D3
D4
D5
D6
D7
DS
b
D0
D1
D2
D3
D4
D5
D6
D7
DS
c

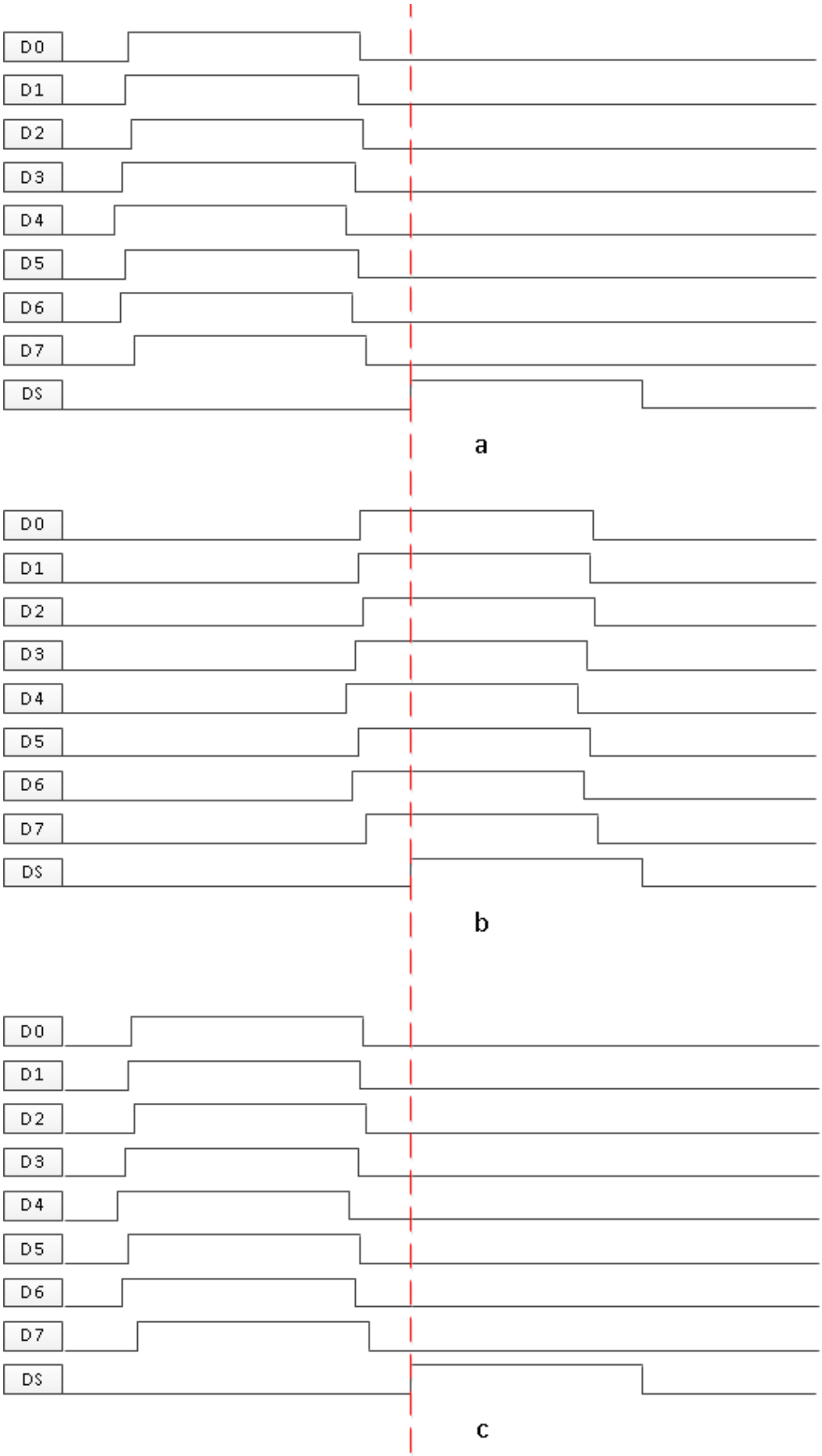
上图中，a)是初始情况，b)是 WDSDR 延迟值即将超过一个 DDR 周期的情形，c)是 WDSDR 延迟值正好超过一个 DDR 周期的情形。

由于 DS 的固有延迟比某些数据线 D 小，所以在查找 WDSDR 右边沿时，即使 WDSDR 减小到零，DS 距离数据眼的右边沿也可能有一小段距离，此时，训练结果会导致 WDSDR 稍稍偏左，如下图所示：



2、 DS 延迟值大于 D0~D7

此时，可能会出现的情况如下图所示：

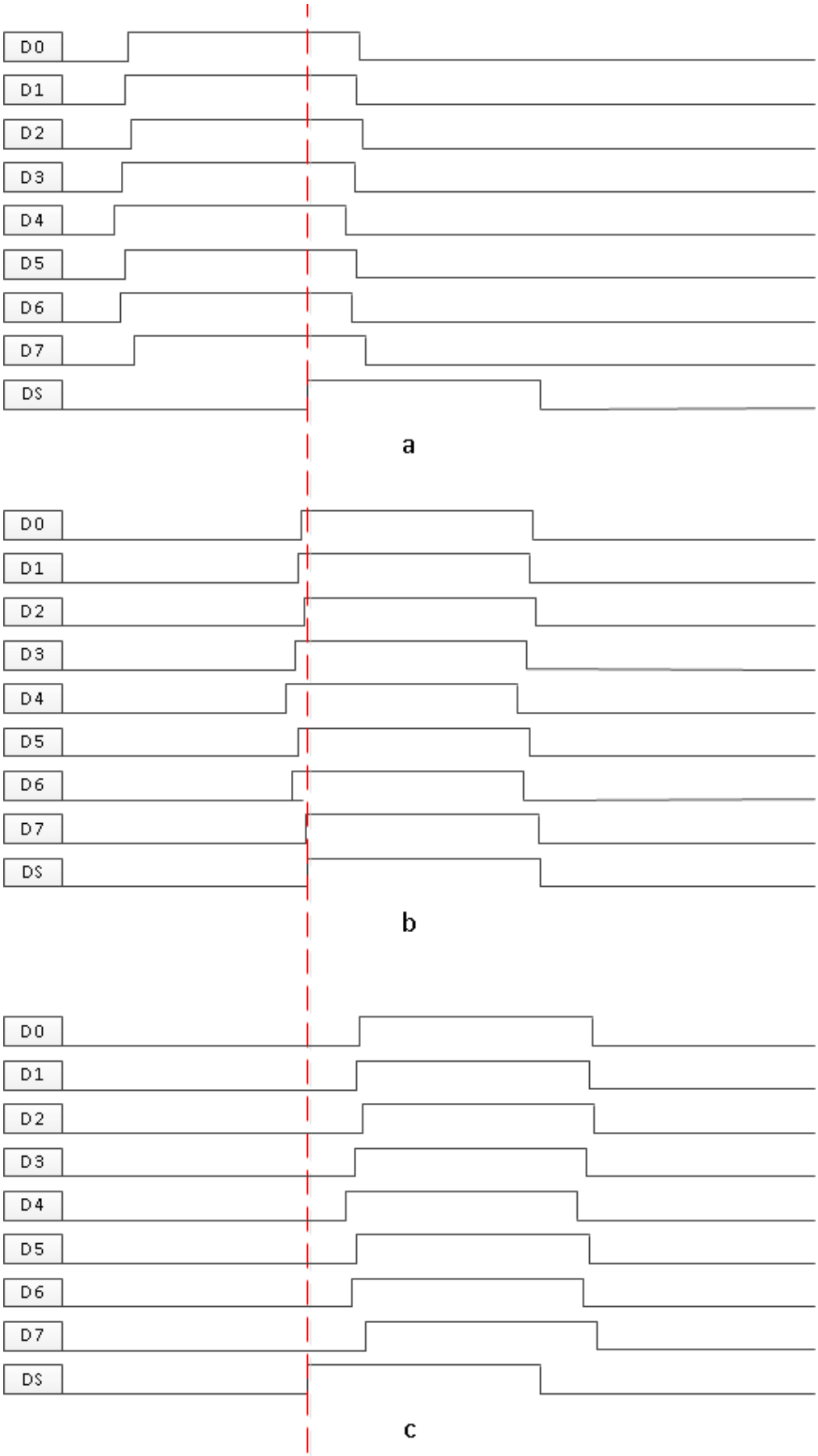

D0
D1
D2
D3
D4
D5
D6
D7
DS
a
D0
D1
D2
D3
D4
D5
D6
D7
DS
b
D0
D1
D2
D3
D4
D5
D6
D7
DS
c

上图中，a)是初始情况，b)是 WDSDR 延迟值即将超过一个 DDR 周期的情形，c)是 WDSDR 延迟值正好超过一个 DDR 周期的情形。

可以看出，初始情况下，DS 延迟大于 D 的延迟时，增大 WDSDR 去寻找数据眼的左边 沿，会出现 b)所示的情况，此时 DS 距离 D 的左边沿还有一段距离，但是，可以通过增加 D 的 BDL 延迟，使数据全部对齐到 DS，此时，再查找 D 的右边沿时，就能找到真正的右边 沿，这种情况下，训练出来的结果最佳。

3、 DS 延迟值小于 D0~D7

此时，可能会出现的情况如下图所示：


D0
D1
D2
D3
D4
D5
D6
D7
DS
a
D0
D1
D2
D3
D4
D5
D6
D7
DS
b
D0
D1
D2
D3
D4
D5
D6
D7
DS
c

a)是 DS 为 0 时的情况，DS 距离 D 的右边沿还有一定的裕量，这就导致了找到的眼睛 的宽度比真实的宽度小。在将 DS 延迟取平均值时，必然会造成偏差。b)是临界情况，此时 WDSDR 的延迟值小于一个 DDR 周期；再增大 WDSDR 时，数据开始出错，但是到了 c)所 示的位置时，数据正好错位一个字节。

在这种情况下，训练结果也会出现 1、中的情形，即找到的数据眼右边沿偏左，并且偏 的比 1、中多。

当然，由于工程中采用的是等长布线，产生延迟差异的地方主要在芯片内部，可能有几 十个 ps，在正常情况下，这个延迟差异不会对使用造成太大影响。

在实际调试时，如果觉得训练结果不理想，也可以通过增大 DS 的 BDL 延迟值来使 1、 以及 3、中的两种情况转变为 2、。但是，改变 DS 的 BDL 延迟值后，需要重新进行写平衡调整。所以，如果要改变 DS 的 BDL 延迟的话，需要将该步骤加在写平衡之前。DS 的 BDL 延迟可以通过寄存器 PMBDR0 调整，在调整 DS 的 BDL 时，需要同时调整 DS 输出使能信号的 BDL，即 PMBDR1。

A.1.6.5 需要注意的问题

注意 1：由于 DM 信号的特殊性，若 DM 信号为 1 表示该一拍数据不写入显存，所以， 在该拍的某些数据位发生的错误不易被检测到。因而，在查找数据眼图的左/右边沿时，首先查找数据的左/右边沿，再查找数据与掩码的左/右边沿。

注意 2：在该过程中，DS 信号需要经过各数据位的模糊区，此时的结果有可能是错的； 考虑到算法的健壮性，在查找数据眼的左右边沿时，需要找到一个数据眼，该数据眼的宽度 大于一个经验值，此时，方可认定该数据眼的左/右边沿为正确的左/右边沿。

A.1.6.6 算法流程

该过程主要调整了 WDSDR、WDBDMR 以及 WDBDR0~8 的值。 算法的伪代码描述如下：

//查找右边沿for(DATX8=0;DATX8<8;DATX8++)

{

读出 WDSDR 寄存器的值，并存为 Rhd； Ra=Rhd；

left_edge_wdsdr=Ra curr_wdsdr_right=1；

ADDR = 随机地

BIT_PATTERN0 = 0xff00ff00

BIT_PATTERN1 = 0x00ff00ff

配置 DTUCFG：dtu_enable=1；dtu_generate_random=0；dtu_target_lane=DATX8； 写 DTUWACTL 寄存器值为 ADDR；

写 DTURACTL 寄存器值为 ADDR； 写
DTUWD0 值为 BIT_PATTERN0； 写
DTUWD1 值为 BIT_PATTERN0； 写
DTUWD2 值为 BIT_PATTERN1； 写
DTUWD3 值为 BIT_PATTERN1；
写 DTUWDM 值为 0x0；
//不带掩码，查找数据眼的左边沿
while(Ra 不等于 0)
{
if(curr_wdsdr_right ==0)
WDSDR=WDSDR-Ra
else
WDSDR=WDSDR+Ra
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
for(i=0;i<8;i++) //为了保证算法的健壮性，对一个值测试 8 次
{
写 DTUECTL[0]为 1，开始测试； 轮询
DTUECTL[0]，直到其为 0；
写 DTUWACTL 寄存器值为 DTUWACTL+2； 写
DTURACTL 寄存器值为 DTURACTL+2；
if(查询 DTUPDES 寄存器有错)
{
curr_wdsdr_right =0； break；
}
else if(i==7)
curr_wdsdr_right =1
}
if(curr_wdsdr_right ==1) left_edge_wdsdr
=WDSDR
写 DTUWACTL 寄存器值为 DTUWACTL+2；
写 DTURACTL 寄存器值为 DTURACTL+2；
Ra=Ra>>1；
} //找到了数据眼左边沿

//查找带 DM 信号的左边沿

写 DTUWDM 值为 0x5555。 while(curr_wdsdr_right
==0 && WDSDR>Rhd)
{
WDSDR=WDSDR-1；
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
for(i=0;i<8;i++) //为了保证算法的健壮性，对一个值测试 8 次
{
写 DTUECTL[0]为 1，开始测试； 轮询
DTUECTL[0]，直到其为 0；
写 DTUWACTL 寄存器值为 DTUWACTL+2； 写
DTURACTL 寄存器值为 DTURACTL+2； 写
DTUWDM 值为~DTUWDM；
if(查询 DTUPDES 寄存器有错)
{
curr_wdsdr_right =0； break；
}
else if(i==7)
curr_wdsdr_right =1
}
}//找到带 DM 信号的数据眼左边沿；
left_edge_wdsdr=WDSDR； if(WDSDR==Rhd)
训练失败；

//不带 DM，将所有数据比特的延迟对齐到左边沿 写
DTUWDM 为 0；
pass[7:0]=0xff； while(|pass)
{ for(BIT=0;BIT<8;BIT++)
{
if(pass[BIT]==1)
将 WDBDR[BIT]寄存器值加 1；
}

for(i=0;i<8;i++)　　//为了保证算法的健壮性，对一个值测试 8次

{

通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；　写

DTUECTL[0]为 1，开始测试；

轮询 DTUECTL[0]，直到其为 0；

写 DTUWACTL 寄存器值为 DTUWACTL+2；　写

DTURACTL 寄存器值为 DTURACTL+2；

if(查询 DTUPDES 寄存器有错)

{ for(BIT=0;BIT<8;BIT++)

{

if(DTUPDES[BIT]==1 && pass[BIT]==0)

　　报告错误；

else if(DTUPDES[BIT]==1)

{

将 WDBDR[BIT]寄存器值减 1；　pass[BIT]=0；

}

}

}

}

}

//将 DM 对齐到左边沿

写 DTUWDM 值为 0x5555；

写 DTUWACTL 寄存器值为 DTUWACTL+2；　写

DTURACTL 寄存器值为 DTURACTL+2；

curr_wdsdr_right=1；　while(curr_wdsdr_right==1)

{

将 WDBDR8 寄存器值加 1；

通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；

for(i=0;i<8;i++)　　//为了保证算法的健壮性，对一个值测试 8次

{

写 DTUECTL[0]为 1，开始测试；　轮

询 DTUECTL[0]，直到其为 0；

写 DTUWDM 值为~DTUWDM；

写 DTUWACTL 寄存器值为 DTUWACTL+2； 写

DTURACTL 寄存器值为 DTURACTL+2；

if(查询 DTUPDES 寄存器有错)

{

curr_wdsdr_right =0；

break；

}

else if(i==7)

curr_wdsdr_right =1

}

}

将 WDBDR8 寄存器值减 1；

将 WDBDR0~8 中的最大者写入 WDBDMR；

//查找不带 DM 的 WDSDR 的右边沿 写

DTUWDM 值为 0；

Ra=Rhd；

right_edge_wdsdr=WDSDR

while(Ra 不等于 0)

{

if(curr_wdsdr_right ==0) WDSDR=WDSDR+Ra

else

WDSDR= max(WDSDR-Ra，0)

通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；

for(i=0;i<8;i++) //为了保证算法的健壮性，对一个值测试 8 次

{

写 DTUECTL[0]为 1，开始测试； 轮

询 DTUECTL[0]，直到其为 0；

写 DTUWACTL 寄存器值为 DTUWACTL+2； 写

DTURACTL 寄存器值为 DTURACTL+2；

if(查询 DTUPDES 寄存器有错)

{

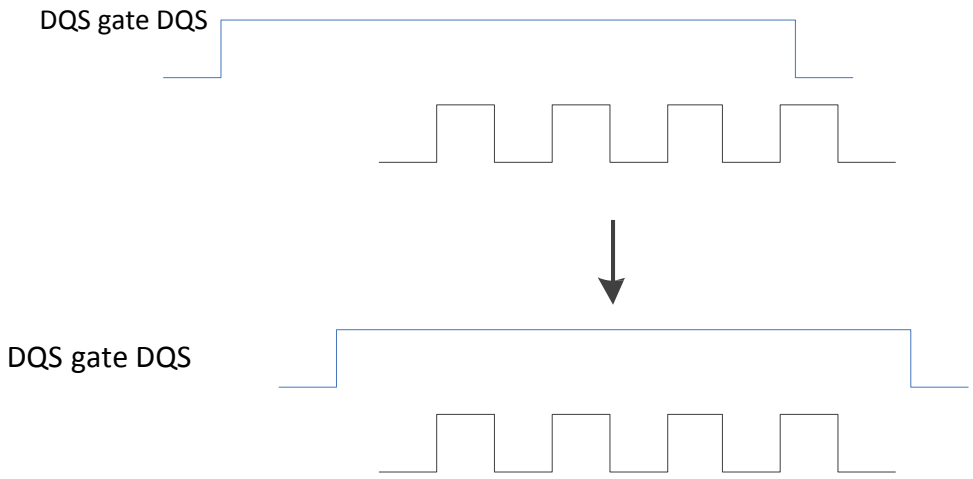
curr_wdsdr_right =0； break；

```
}
else if(i==7)
    curr_wdsdr_right =1
}
if(curr_wdsdr_right ==1)
    right_edge_wdsdr=WDSDR
Ra=Ra>>1；
} //找到了数据眼右边沿
WDSDR=right_edge_wdsdr；
//查找带 DM 的 WDSDR 右边沿  写
DTUWDM 值为 0x5555；
while(curr_wdsdr_right==0)
{
for(i=0;i<8;i++)              //为了保证算法的健壮性，对一个值测试 8次
{
    写 DTUECTL[0]为 1，开始测试；  轮询
DTUECTL[0]，直到其为 0；
DTUWDM=~DTUWDM
写 DTUWACTL 寄存器值为 DTUWACTL+2；  写
DTURACTL 寄存器值为 DTURACTL+2；
if(查询 DTUPDES 寄存器有错)
{
    curr_wdsdr_right  =0；    break；
}
else if(i==7)
    curr_wdsdr_right =1
}
if(curr_wdsdr_right==0)
{
WDSDR=WDSDR+1；
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
}
}//找到右边沿
right_edge_wdsdr=WDSDR
```

将 left_edge_wdsdr 与 right_edge_wdsdr 取平均，并写入 WDSDR； 通过
DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
}
写 DTUCFG.dtu_enable=0，关闭 DTU；
写 SCTL=1，使 uMCTL 进入 Config 状态； 轮
询 STAT 寄存器直到其为 1；
配置 MCMD 寄存器，发送 PREA 命令； 轮询
MCMD 直到命令完成；
写 SCTL=2，使 uMCTL 进入 Access 状态； 轮询
STAT 寄存器直到其为 3；

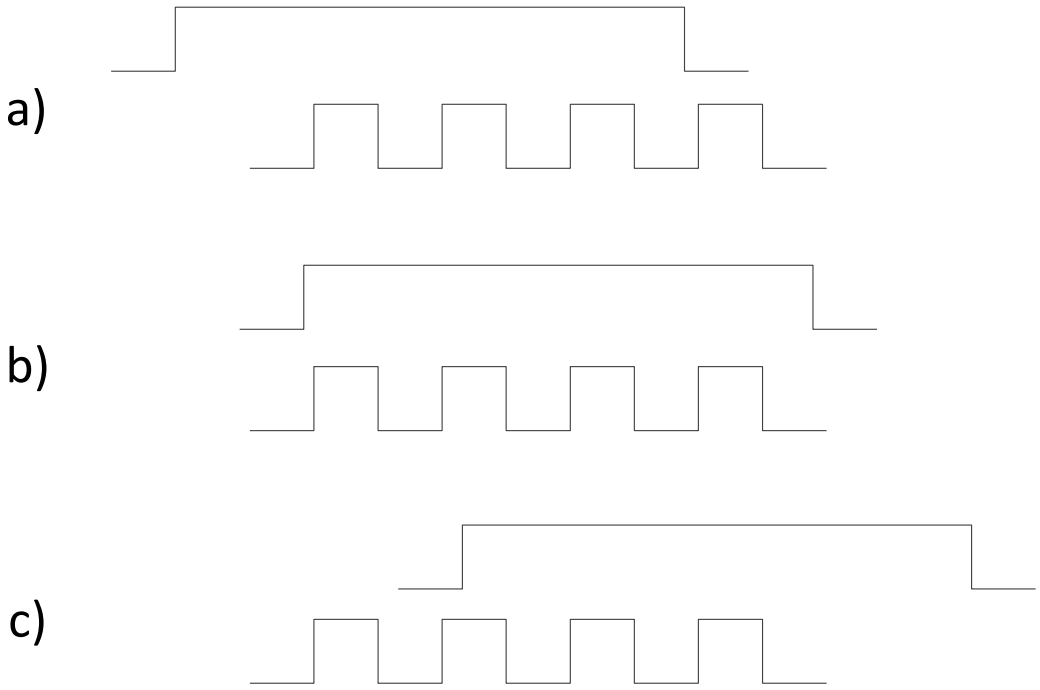
A.1.7 DQS gating 训练

该步骤主要调整 ctl_qs_enable 信号的延迟以及宽度，以使对读数据信号的干扰减到最 小。算法示意图如下：



本过程需要调节的寄存器是 QSCR、QSLR0 以及 QSLR1。由于 65nm 的 DDR3PHY 内 部，没有针对 DQS gate 信号的延迟线，对该信号调整延迟时，只能通过存控调整 ctl_qs_enable 的方法进行，因而，调整的单位是 CK 时钟。所以，查找 DQS 门控的合适延迟时，只能采 用模糊查找的方法。

在这种情况下，若 DQS gate 的宽度为 4 个 CK 时钟时，有可能出现下述的情况：



其中，a）中由于最后一个 DQS 高电平被门控信号截除了一大部分，导致数据采样错误； b）中的第一个 DQS 高电平虽然被门控信号截除了一部分，但是剩下的 DQS 信号采样数据仍然正确；c）中的 DQS 电平经过门控后丢失了一个 DQS 电平导致采样数据错误。

在上述情况下，训练结果为 b），但是，由于第一个 DQS 电平变窄，很可能导致在正 常使用情况下，由于数据或者 DQS 的漂移导致数据采样错误。还有一种可能就是，由于噪声以及 setup/hold 时间的影响，在 b）情况下，也不能得到正确的结果，那么 DQS gate 训练就会失败；

所以，在进行 DQS gate 训练时，一般采用门控宽度为 5 进行训练，此时，可能出现下 述的情况：

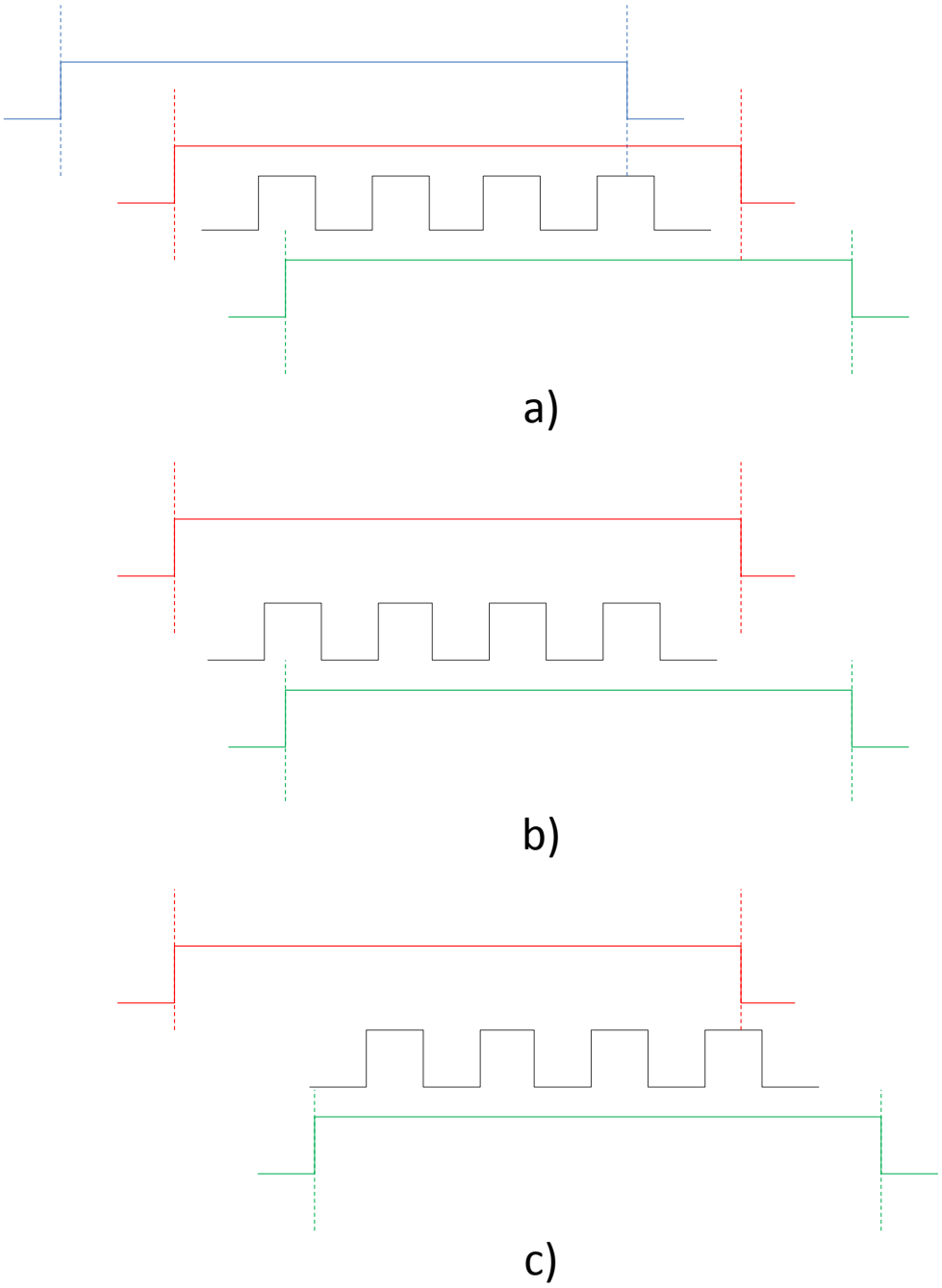


图 a 中的情况，由于蓝色门控与绿色门控截取了部分 DQS 电平，导致采样数据失败， 即只有红色门控的结果正确，此时，DQS 的有效电平基本上处于红色门控的中央，此时， 可以采用红色门控作为训练结果。

图 b 中的情况，红色门控以及绿色门控的结果均正确，此时，若将读 DQS 以及读 DQ 均向后再延迟半个 DDR 周期，红色门控的结果仍然正确。此时可以使用红色门控作为训练 结果。

图 c 中的情况，红色门控以及绿色门控的结果均正确，此时，若将读 DQS 以及读 DQ 再延迟半个 DDR 周期，红色门控的结果就会出现错误，但绿色门控的结果仍然正确，这说

明 DQS 信号比较靠近绿色门控的中心，此时可以使用绿色门控，并将读 DQ 以及 DQS 延迟半个 DDR 周期作为训练结果。

上述的算法，能够保证 DQS 处于门控之中，并且，门控的边沿不会截取到 DQS 的高电 平，即保证了其 8 个 DQS 上下沿能够较为精确的采样时钟。

该算法的伪代码如下：

ADDR = 随机地

BIT_PATTER = 0xff00ff00

BIT_PATTER = 0x00ff00ff

配置 QSCR 寄存器，写 QSHOLD 为 0，QSWIDTH 为 5；GATSEL0~7 为 0；

写 DTUWACTL 寄存器值为 ADDR； 写
DTURACTL 寄存器值为 ADDR； 写
DTUWD0 值为 BIT_PATTERN0； 写
DTUWD1 值为 BIT_PATTERN1； 写
DTUWD2 值为 BIT_PATTERN2； 写
DTUWD3 值为 BIT_PATTERN3；
写 DTUWDM 值为 0；
写 SCTL=1，使 uMCTL 进入 Config 状态； 轮
询 STAT 寄存器直到其为 1；
for(DATX8=0;DATX8<8;DATX8++)
{
将 PPCFG[7:1]（对应 datx8 7~1）中对应于当前 datx8 的位清零，其他位置一，为 0 号 datx8 时，全置 1；
写 DFICFG0[2]为 1，打开 dfi_data_byte_disable_en;
配置 DTUCFG：dtu_enable=1；dtu_target_lane=DATX8；其他位为 0；
for(LAT=RL-1;LAT<19;LAT++)
{
配置 QSLR0 或者 QSLR1 的 QSLAT[DATX8]值为 LAT；
写 DTUECTL[0]为 1，开始测试； 轮
询 DTUECTL[0]，直到其为 0；
查询 DTUPDES 寄存器，看结果是否出错；
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位； 通过
SFRR[0]进行对齐缓冲的复位；
}

通过上述 for 循环找出使测试无错的 QSLAT 值的集合，若该集合不存在，则增大 QSCR 寄存器中，该 DATX8 对应的 GATSEL（置为 7）。然后从 LAT=19-14 开始，重新进行上述 for 循环测试；

若集合中只有一个正确的值，则使用该值作为训练结果。 若集合中有两个正确的值：

写 QSLR0/1 中对应当前 datx8 的 QSLAT[DATX8]值为较小者；

写 RDBDR0~7 以及 PMBDR3，将其延迟值增加半个 DDR 周期； 通过
DATX8 的 ACR[9]进行 PHY 高速逻辑复位；
通过 SFRR[0]进行对齐缓冲的复位； 写
DTUECTL[0]为 1，开始测试； 轮询
DTUECTL[0]，直到其为 0； if(查询
DTUPDES 寄存器出错)
写 QSLR0/1，将 QSLAT[DATX8]加 1；
else
写 RDBDR0~7 以及 PMBDR3，将其延迟值减少半个 DDR 周期；
}
通过 DATX8 的 ACR[9]进行 PHY 高速逻辑复位； 通
过 SFRR[0]进行对齐缓冲的复位；
写 DTUCFG.dtu_enable=0，关闭 DTU； 配置
MCMD 寄存器，发送 PREA 命令； 轮询
MCMD 直到命令完成；

//打开刷新，ZQCS 以及更新等功能
写各 DATX8 的 GCR[4:0]为 0x1f，打开延迟线漂移补偿； 写
TREFI，恢复刷新功能；
写 TZQCS，恢复 ZQCS；
写 SCTL=2，使 uMCTL 进入 Access 状态； 轮询
STAT 寄存器直到其为 3；

A.1.8 其他

uPCTL 内有一个状态机，该状态机会控制 uPCTL 内部 IO 寄存器的访问权限，见下表：

Register	Init_mem	Config	Access	Low Power
SCFG	Yes	Yes	Yes	Yes
SCTL	Yes	Yes	Yes	Yes
MCMD	Yes	Yes	No	No
POWCTL	Yes	No	No	No
ECCCLR	Yes	Yes	Yes	Yes
DTU* (page 165 - 169) (page 192 - 199)	Yes	Yes	Yes	Yes
All remaining R/W or R/WSC registers	Yes	Yes	No	Yes

该状态机可以通过写 SCTL 来控制其进入某个状态，通过 SCFG 可以获知当前所处状态。所以，在访问 uPCTL 的 IO 寄存器时，首先要写 SCTL 使其进入 Config 状态。

另外，虽然可以在 Config 状态下访问 DTU 寄存器，但是 Config 状态下，发送命令的 权限被 uPCTL 内的 DCU 模块控制，DTU 模块的命令发不出去，所以要进行 DTU 测试时， 需要通过 SCTL 使状态机进入 Access 状态方可。

附录 B（固件内容）

ICH2 的固件放置在 SPI 接口的 Flash 或 E2PROM 内前 304 字节（0~303），固件内容由 串口或 JTAG 维护接口写入，上电复位时由自引导模块读入。维护复位或次级总线复位等不 会重置自引导流程。

固件内容如下：

地址	内容	说明	全芯片使能
0x00[1:0]	BYPASS_MODE[1:0]	旁路模式配置 2‘b00: 全芯片模式，不 Bypass 任 何部件 2‘b01: 旁路 PSW，不使能 PSWPLL 2‘b1x: 旁路 PAB，仅需要使能	2‘b00
0x0[	DCSEL	显示控制器时钟选择信号，为 1 时使用片外时钟，为 0 时使用	1‘b1
0x0[	SATARESET_CTL	SATA 模块上电控制，为 1 时，在 撤维护复位前进行上电；为 0	1‘b1
0x	PIPE13SEL[1:0]	PSW 下游端口配置模式 2‘b00: 低功耗模式 2‘b01: 一个 8X 端口 2‘b10: 两个 4X 端口 2‘b11: 保留	2‘b10
0x0[7	PIPE24SEL[1:0]		2‘b10
0x1	PCIEPHY_ENABLE[5:0]	为 1 时使能，进行上电操作、为 0 时不使能 5: ds5_phy 4: ds3_phy 3: ds4_phy 2: ds2_phy 1: ds1_phy 0: usp_phy 上游端口使能，进	6‘b111111 若旁路 PSW， 则为 6‘h100001，仅 使能 0:usp_phy 与

0x2	INSIDE_CLKEN[7:0]	内部时钟使能控制位，针对全局 时钟 GCLK_250MHz 进行选择， 为 1 时表示选择 250MHz 时钟网 络，为 0 时选择 3.125MHz 休眠时 钟 0：SATA 控制器 1：USB 控制器 2：GMAC0 控制器 3：GMAC1 控制器 4：DMAC 5：VPU	8‘hff 若旁路 PAB，8‘h0。 内部时钟网络全不使能，使 PAB 与 AMBA 部分 工作在低功耗状态
		DC0 时钟门控，当 0x0[2]位 DCSEL 为 0 选择片内时钟时，此 位为 1 时表示选择 GPU 锁相环时 钟 GPUPLLCLK 的 8 分频，为 0 时选择 3.125MHz 休眠时钟 6：DC0 DC1 时钟门控，当当 0x0[2]位 DCSEL 为 0 选择片内时钟时，此 位为 1 时表示选择 GPU 锁相环时 钟 GPUPLLCLK 的 16 分频，为 0 时选择 3.125MHz 休眠时钟 7：DC1	
0x3~0x4	PLL_RST_CNT[15:0]	PLL 复位保持时间，主锁相环最小 5us，PCIPHY 锁相环最小 41us 100MHz 时钟域，实际工作最少需 要 16‘h2900 在模拟环境中设为 16‘h190，加	16‘h190
0x5[1:0]	LegacyIOSEL[1:0]	0x4018805C LegacyIO 路由选择寄存器，缺省 0x0 Bit0：为 0 表示 60h/64h 路由给 PS2， 为 1 表示路由给 LPC Bit1：为 0 表示 3f8h~3ffh 路由给 uart，为 1 表示路由给 LPC	2‘b0
0x5[7:4]	PLL_ENABLE[3:0]	//0:GPU;1:MC;2:PSW;3:GCLK 固件内含义：1,使能对应 PLL, 对 PLL 进行上电和复位；0, 不使能PLL 当 PSW 旁路模式有效时，[2]必须 为 0；其他根据需要配置，不用时为 0。 当 PAB 旁路模式有效时，除	全芯片模式， 设为 4‘b1111 旁路 PSW 时， 设为 4‘h1011 旁路 PAB 时，

0x6~0x9	GPUPLL_CFG[29:0]	29 TEST 锁相环测试模式控制 28 BYPASS 锁相环旁路模式控 制 27:24 CLKOD 输出时钟的分频系数 23:22 冗余 21:16 CLKR[5:0] 参考时钟的分 频系数 15:13 冗余 12:0 CLKF[12:0] 反馈时钟的分 频系数	默认值 30‘h50001D (500MHz)
10~13	MCPLL_CFG[29:0]	29 TEST 锁相环测试模式控制 28 BYPASS 锁相环旁路模式控 制 27:24 CLKOD 输出时钟的分频 系数	默认值 30‘h90001D (300MHz)
		23:22 冗余 21:16 CLKR[5:0] 参考时钟的分 频系数 15:13 冗余 12:0 CLKF[12:0] 反馈时钟的分 频系数	
14~17	PSWPLL_CFG[29:0]	29 TEST 锁相环测试模式控制 28 BYPASS 锁相环旁路模式控 制 27:24 CLKOD 输出时钟的分频 系数 23:22 冗余 21:16 CLKR[5:0] 参考时钟的分 频系数 15:13 冗余 12:0 CLKF[12:0] 反馈时钟的分 频系数	默认值 30‘hB04009B (260MHz)
18~21	GCLKPLL_CFG[29:0]	29 TEST 锁相环测试模式控制 28 BYPASS 锁相环旁路模式控 制 27:24 CLKOD 输出时钟的分频 系数 23:22 冗余 21:16 CLKR[5:0] 参考时钟的分 频系数 15:13 冗余 12:0 CLKF[12:0] 反馈时钟的分 频系数	默认值 30‘hB00001D (250MHz)
22[2:0]	SATAPHY_RESET[2:0]	SATAPHY 上电使能，为 1 时上电，	3‘b111
22[6:4]	SATAPHY_SSP_EN[2:0]	SATAPHY 扩频使能，为 1 时使能	3‘b111
23	USBPHY_RESET[5:0]	USBPHY 上电使能，为 1 时上电，	6‘b111111

24	USBPHY_SSP_EN[5:0]	USBPHY 扩频使能，为 1 时使能扩	6‘b111111
25~28	SEL_WATCHOUT[31:0]	WATCH_OUT_PIN 监测信号选择 默认值 32‘h0，选择 GPUCLK_32DIV 输出到管脚 31:30 最后一级选择输出到管脚 2‘b00: OBS_CLK，分频监测时钟 2‘b01:OBS_PCI，PCI 监测信号 2‘b10:OBS_LOCK，锁相环锁定 信号 2‘b11:RING_SELOUT，环振输出信号 29:20 锁相环锁定信号 OBS_LOCK 选择	32‘b0

		22: SWPLL_Lock 23: GCPLL_Lock 24: PCIE_US_PLL_LOCK 25: PCIE_DS13_PLL0_LOCK 26: PCIE_DS13_PLL1_LOCK 27: PCIE_DS24_PLL0_LOCK 28: PCIE_DS24_PLL1_LOCK 29: PCIE_DS5_PLL_LOCK 18:16 PCIE 状态信号 OBS_PCI 选 择 0x0 us_dl_up 0x1 ds1_dl_up 0x2 ds2_dl_up 0x3 ds3_dl_up 0x4 ds4_dl_up 0x5 ds5_dl_up 0x6 pab_heart_jump 0x7 pab_out_order 14:12 分频时钟 OBS_CLK 选择 0x0:GPU 的 PLL 输出 32 分频观测 时钟 0x1:MC 的 PLL 输出 16 分频观测 时钟 0x2:PSW 的 PLL 输出 16 分频观测 时钟 0x3:GCLK 的 PLL 输出 16 分频观 测时钟 0x4:MC PHY 32 分频输出 0x5:SATA PHY 分频输出 0x6:USBPHY 分频输出 0x7:PCIEPHY 16 分频输出 11:8 PCIE16DIVSEL 0x0 PCIE_US_PHY0 16 分频输出 0x1 PCIE_US_PHY1 16 分频输出 0x2 PCIE_DS13_PHY0 16 分频输 出 0x3 PCIE_DS13_PHY1 16 分频输 出 0x4 PCIE_DS24_PHY0 16 分频输 出	

		0x3 USB3PHY 60MHz 监测时钟 输出 0x4 USB4PHY 60MHz 监测时钟 输出 0x5 USB5PHY 60MHz 监测时钟 输出 0x6 ohci_0_pll_clock 监测时钟输 出 0x7 ohci_1_pll_clock 监测时钟输 出 1:0 SATAPHY 监测时钟输出 0x0 SATA0PHY 50MHz oob 时钟 输出 0x1 SATA1PHY 50MHz oob 时钟 输出 0x2 SATA2PHY 50MHz oob 时 钟输出	
WISHBONE 的 APB 及 AHB 维护接口			
29[3:0]	SEL_VRAM128M [3:0]	维护访存地址编移 将显存划分为 16 个 128MB 空间，可通过维护进行访问显存 2GB 空间	4‘b0
30~32	MAX_COUNT_VALUE[23:0]	默认值 0x100000 AHB 维护和 APB 维护响应超时阀值 软环境设 24'h00_0100，256cycle 实际运行中 24'h10_0000，1Mcycle	24‘h100
与 PAB 桥及 AXI 总线接口部件			

3~36	CFG_AXIARB[31:0]	AXI 总线上的 8 个 master，采用自 行设计的加权轮转仲裁器，轮转 仲裁权重初值寄存器： CFG_AXIARB[31:0] 复位默认值为 32'h11100000； w_mst_priority_gar, --gar 初值权重，复位默认值 4'h1 w_mst_priority_ac97, --ac97 初值 权重，复位默认值 4'h1 w_mst_priority_dmac, --dmac 初值 权重，复位默认值 4'h1 w_mst_priority_xhci,--xhci 初值权 重，复位默认值 4'h0 w_mst_priority_gmac1,--gmac1 初 值权重，复位默认值 4'h0 w_mst_priority_gmac0--gmac0 初 值权重，复位默认值 4'h0 w_mst_priority_sata --sata 初值权 重，复位默认值 4'h0 w_mst_priority_LPC --LPC 初值权	32‘h11100000
		重，复位默认值 4'h0	
37~40	PAB_IO_ACK_CNT[31:0]	IO 响应超时值，初始值	32‘hfff_ffff
41~44	PAB_DMA_ACK_CNT[31:0]	DMA 响应超时值，复位值32'hffffffff	32‘hfff_ffff
45~46	PABDVEN[15:0]	0x40188000 设备配置空间开放使能寄存器， 高表示使能，缺省 0xFFFFFFFFh 14： Bit[31]：PS2 13：Bit[30]：Uart0 12：Bit[25]：SPI 11：Bit[24]：I2C0 10：Bit[23]：LPC 9：Bit[21]：GPIO 8：Bit[13]：GMAC1 7：Bit[12]：GMAC0 6：Bit[11]：SATA 5：Bit[10]：GP 4：Bit[9]：XHCI 3：Bit[6]：AC97 2：Bit[2]：I2C2 1：Bit[1]：I2C1	16‘h7fff
[49:47]	保留		
50[3:0]	PRE_CFG_CNT[3:0]	维护复位后、系统复位前配置寄 存器数量（最多 15	4‘b10
50[7:4]	AFT_CFG_CNT[3:0]	系统复位后配置寄存器数量（最 多 15 个）。为 0	4‘b0

51~52	PS2_DIV_LIMIT[11:0]	{spi52_dat [3:0], spi51_dat}	12‘h1f4
53~55	UART_BAUD_LIMIT[19:0]	{spi55_dat [3:0], spi54_dat, spi54_dat}	20‘h3889
56~63	预留		
64~67	PRE_CFG_ADDR1[31:0]	IO 复位前，第 1 个配置地址	0x401004b0
68~71	PRE_CFG_Value1[31:0]	IO 复位前，第 1 个配置内容	0x1
72~75			0x401804b0
76~79			0x1
80~183	……...	……...	
184~18	AFT_CFG_ADDR1[31:0]	IO 复位后，第 1 个配置地址	
188~19	AFT_CFG_Value61[31:0]	IO 复位后，第 1 个配置内容	
192~30	……...	……...	

附录 C（DS5 端口训练问题板级解决方案）

ICH2 芯片内部 PCIe 链路的 DS5 端口会有偶发训练不上的问题，在核心电压由 0.9V 升 到 0.95V 的情况下，大部分芯片不存在训练不上的问题了，但是仍然有少数芯片问题没有解决。为了避免 DS5 端口训练不上影响系统运行，对板级解决方案提出要求。板级解决按照 PSW（PCIe 交叉开关）是否旁路分为两种方案。

PSW 旁路模式

1） ICH2 芯片的 AD28 管脚 CPU_LINKUP，一般会在板级接出来点灯，用于指示CPU 和 ICH2 之间的 PCIe 链路训练是否成功。在 PSW 旁路模式下， CPU_LINKUP 信号就用于指示 CPU 和 DS5 端口之间的 PCIe 链路训练是否成 功。CPU_LINKUP 信号引到 CPLD 等板级控制管理的可编程逻辑器件。

2） 在 CPLD 增加一个定时器，用于判断是否在约定的时间内 DS5 端口 PCIe 链路训练失败，类似如下的逻辑：

```
reg[16:0]   r_wait_link_cnt;wire[16:0]
c_wait_link_cnt; always @(clk32K or
negedge rstn) begin
   if(!rstn)
      r_wait_link_cnt <= 17‘h0; else
      r_wait_link_cnt <=c_wait_link_cnt;
end  always @*
begin
   if(CPU_LINKUP)
      c_wait_link_cnt = 17‘h0; else
   if(ich2_dcok)
      c_wait_link_cnt = r_wait_link_cnt + 17‘h1; else
      c_wait_link_cnt = r_wait_link_cnt;
end
```

3） 增加重新开工的条件，c_link_time_out 作为触发软复位的条件之一：

```
wire   c_link_time_out =  &r_wait_link_cnt;
```

从 CPLD 给 ICH2 芯片的上电 OK 信号 ich2_dcok 有效开始计时，在大约 4s 时 间内如果 DS5 端口的 PCIe 链路还是没有训练上，CPLD 逻辑就重新触发整个板级 的软复位，DS5 端口会重新进行训练，直到训练成功。

PSW 非旁路模式

ICH2 芯片的 AE27 管脚 WATCH_OUT 是对芯片内部部分信号的一个观测引脚，其中实 现了一个功能就是对 DS5 端口是否训练上的观测。参考附录二固件内容的定义，固件的 [28:25]字节需要配置成 0x40050000 就可以实现对 DS5 端口 PCIe 链路训练情况的观测。所以在固件按照上面要求配置的情况下，把 WATCH_OUT 信号引入 CPLD，WATCH_OUT（ds5_dl_up）有效作为

计时器清零的条件，用 PSW 旁路模式下类似的解决方法实现对 DS5 端口问题的规避。

所以板级实现的时候需要把 ICH2 的 AE27 和 AD28 管脚信号都接入板级控制管理可编 程逻辑器件，以便后续实现逻辑规避 DS5 端口训练不上的问题。但是为了区分实际使用中 是否使用的是旁路模式，还是全芯片模式，便于 CPLD 统一一份代码，CPLD 最好再增加一 个输入跳线管脚，用于区分是否为旁路模式。