Rockchip Linux PCIe 开发指南

文件标识：RK-KF-YF-141

发布版本：V6.18.0

日期：2024-12-03

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

产品版本

芯片名称	内核版本
RK1808	4.4， 4.19
RK3528	4.19，5.10，6.1
RK3562	5.10，6.1
RK3566/RK3568	4.19，5.10，6.1
RK3576	6.1
RK3588	5.4，5.10，6.1

RK3399使用不同的PCIe控制器IP，不在本文档覆盖范围，请参考《Rockchip_RK3399_Developer_Guide_PCIe_CN》。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	林涛	2021-01-15	初始版本
V1.1.0	林涛	2021-01-22	增加PCIe 3.0控制器异常情况的检查信息
V1.2.0	林涛	2021-01-26	增加PCIe 2.0 Combo phy异常排除信息
V1.3.0	林涛	2021-02-04	增加MSI和MSI-X支持数量的问题描述
V1.4.0	林涛	2021-02-05	增加地址分配异常信息
V1.5.0	林涛	2021-02-06	增加PCIe2x1的PHY支持SSC说明
V1.6.0	林涛	2021-02-23	增加MSI/MSI-X调试支持和运行态设备异常说明
V1.7.0	林涛	2021-02-26	增加Legacy INT的说明
V1.8.0	林涛	2021-02-27	增加标注EP功能件开发说明
V1.9.0	林涛	2021-03-16	增加FW存在异常设备的说明
V2.0.0	林涛	2021-04-12	增加用户态访问异常说明
V2.1.0	林涛	2021-04-21	增加PCIe转XHCI芯片异常说明
V2.2.0	林涛	2021-04-23	增加lane拆分复位IO说明以及休眠唤醒异常说明
V2.3.0	林涛	2021-05-12	增加lane拆分下电源配置说明
V2.4.0	林涛	2021-06-04	增加LTSSM附录以及新增debug项说明
V2.5.0	林涛	2021-06-08	增加PCIe 3.0驱动因PHY异常而退出的说明
V2.6.0	林涛	2021-07-07	增加PCIe 地址空间配置说明
V2.7.0	林涛	2021-07-14	增加PCIe设备重扫描说明

版本号	作者	修改日期	修改说明
V2.7.1	林涛	2021-07-15	常见应用问题章节的格式修订
V2.8.0	林涛	2021-07-23	修订开发资源获取地址
V2.9.0	林涛	2021-07-29	增加legacy中断异常以及IO端口访问需求说明
V3.0.0	林涛	2021-08-02	增加PCIe 2.0 combo phy的充电检测异常处理方法
V3.1.0	林涛	2021-08-23	增加PCIe 3.0电源配置对时钟芯片的影响说明
V3.2.0	林涛	2021-09-01	增修PCIe BAR地址空间在64位和32位模式的说明
V3.3.0	林涛	2021-09-09	增加FW报错log以及可能的修复手段
V3.4.0	林鼎强	2021-09-18	增加设备IO BAR地址空间配置异常的说明
V3.5.0	林涛	2021-09-23	增加PCIe内存访问性能的说明
V3.6.0	林涛	2021-11-03	增加DMA支持说明
V3.7.0	林涛	2021-11-25	增加标准EP的bar空间异常说明
V4.0.0	杨凯	2021-12-20	增加RK3588说明
V4.1.0	林涛	2022-01-28	增加present信号， 线程初始化说明以及去除combophy补丁说明
V4.2.0	林鼎强	2022-03-21	更新“芯片互联功能”描述
V4.3.0	林涛	2022-03-30	增加可配置#PERST复位时间说明
V4.4.0	林鼎强	2022-05-24	增加 PM L1 Substates 支持说明
V4.5.0	林涛	2022-07-24	增加debugfs的操作说明
V4.6.0	林涛	2022-07-26	增加平台关于cache一致性的说明

版本号	作者	修改日期	修改说明
V4.7.0	林涛	2022-08-08	增加设备quirks修复说明
V4.8.0	林涛	2022-08-17	增加RK1808说明， 追加debugfs信息
V4.9.0	林涛	2022-08-23	明晰DMA通道情况， 增加3588的5.4内核支持说明
V5.0.0	林涛	2022-08-25	调整MEM与IO BAR异常说明， 增加M.2接口说明， 增加link失败情况
V5.1.0	林涛	2022-08-28	增加Wi-Fi DTS配置范例和休眠唤醒设备状态切换异常
V5.2.0	林涛	2022-11-18	增加RK3528芯片说明以及模块掉电不充分的排查
V5.3.0	林涛	2022-12-14	调整标准EP的说明, 新增PHY时钟模式和各芯片PCIe控制器访问优先级
V5.4.0	林涛	2023-01-03	增调内核版本， 调整部分RK356X说明， 新增资源过滤方法， 明确Lane极性和线序交换
V5.5.0	林涛	2023-02-06	修复RK3528 combo phy时钟调试选项
V5.6.0	薛小明	2023-02-24	修正补充BAR地址访问异常的修复措施
V5.7.0	杨凯	2023-03-14	增加进入物理信号compliance测试模式说明
V5.8.0	杨凯	2023-04-18	删除芯片互联及标准EP的指南， EP模式由 <Rockchip_Developer_Guide_PCIE_EP_Stardard_Card_CN> 进行说明
V5.9.0	林涛	2023-05-29	增加NVMe设备温控说明
V6.0.0	林涛	2023-06-10	增加Link正常却没有正常枚举出设备的说明
V6.1.0	林涛	2023-06-25	增加PCIe设备唤醒的说明
V6.2.0	林鼎强	2023-07-05	增加 DMATEST、增加几个常见应用说明

版本号	作者	修改日期	修改说明
V6.3.0	肖垚	2023-07-31	修正PCIe WiFi的REG ON脚的配置说明
V6.4.0	林涛	2023-08-09	统一规范compliance测试的配置
V6.5.0	林鼎强	2023-08-15	修改使用 DMATEST 新版本命令
V6.6.0	林涛	2023-12-19	修正msi-map说明
V6.7.0	杨凯	2024-01-12	增加低速IO信号说明
V6.8.0	林涛	2024-02-26	更新内核版本和新芯片支持
V6.9.0	林鼎强	2024-05-25	更新域地址分配实例
V6.10.0	林鼎强	2024-07-25	优化Lane reversal/Lane polarity说明
V6.11.0	林涛	2024-08-01	添加基于GPIO模式拔插的说明， 进一步明确Lane reversal使用
V6.12.0	林涛	2024-08-22	添加关于下游设备绑定ID的问题说明
V6.13.0	林涛	2024-09-04	添加PCIe PMU perf支持说明
V6.14.0	林涛	2024-09-07	独立出稳定性统计章节，并添加错误注入说明
V6.15.0	林涛	2024-10-18	补充板级可配置的时序
V6.16.0	林涛	2024-11-05	补充NVMe异常处理
V6.17.0	林涛	2024-11-19	添加设备休眠不断电的配置；其他配置修复
V6.18.0	林涛	2024-12-03	增加PC模型卡死的问题说明

目录

1. 芯片资源介绍

1.1 RK1808

资源	模式	支持lane拆分	支持DMA	支持MMU	支持ASPM	备注
PCIe Gen2 x 2 lane	RC /EP	否	2个读Channel + 2个写Channel	否	L0s/L1	内部时钟

1.2 RK3528

资源	模式	支持lane拆分	支持DMA	支持MMU	支持ASPM	备注
PCIe Gen2 x 1 lane	RC only	否	否	否	ALL	内部时钟

1.3 RK3562

资源	模式	支持lane拆分	支持DMA	支持MMU	支持ASPM	备注
PCIe Gen2 x 1 lane	RC only	否	否	否	ALL	内部时钟

1.4 RK3566

资源	模式	支持lane拆分	支持DMA	支持MMU	支持ASPM	备注
PCIe Gen2 x 1 lane	RC only	否	否	否	L0s/L1	内部时钟

1.5 RK3568

1.5.1 控制器

资源	模式	支持lane拆分	支持DMA	支持MMU	支持ASPM	备注
PCIe Gen2 x 1 lane	RC only	否	否	否	L0s/L1	内部时钟
PCIe Gen3 x 2 lane	RC/EP	1 lane RC+ 1 lane RC	2个读Channel + 2个写Channel	否	ALL	外置晶振，仅支持pcie30phy
PCIe Gen3 x 1 lane	RC	1 lane RC	否	否	ALL	外置晶振，仅支持pcie30phy

1.5.2 PHY

资源	dts节点	参考时钟	拆分	是否combo
pcie30phy	phy@fe8c0000	外部	2Lane：默认 1Lane + 1Lane：rockchip,bifurcation	pcie专用
combphy2_psq	phy@fe840000	内部/外部	-	与SATA/RGMII combo

说明：

- pcie30phy 2Lane默认配置为PCIe Gen3 x 2 lane，拆分后“PCIe Gen3 x 2 lane”和“PCIe Gen3 x 1 lane” 控制器各1Lane

1.6 RK3576

资源	模式	支持lane拆分	支持DMA	支持MMU	支持ASPM	备注
PCIe Gen2 x 1 lane	RC only	否	否	是	ALL	内部时钟
PCIe Gen2 x 1 lane	RC only	否	否	是	ALL	内部时钟

1.7 RK3588

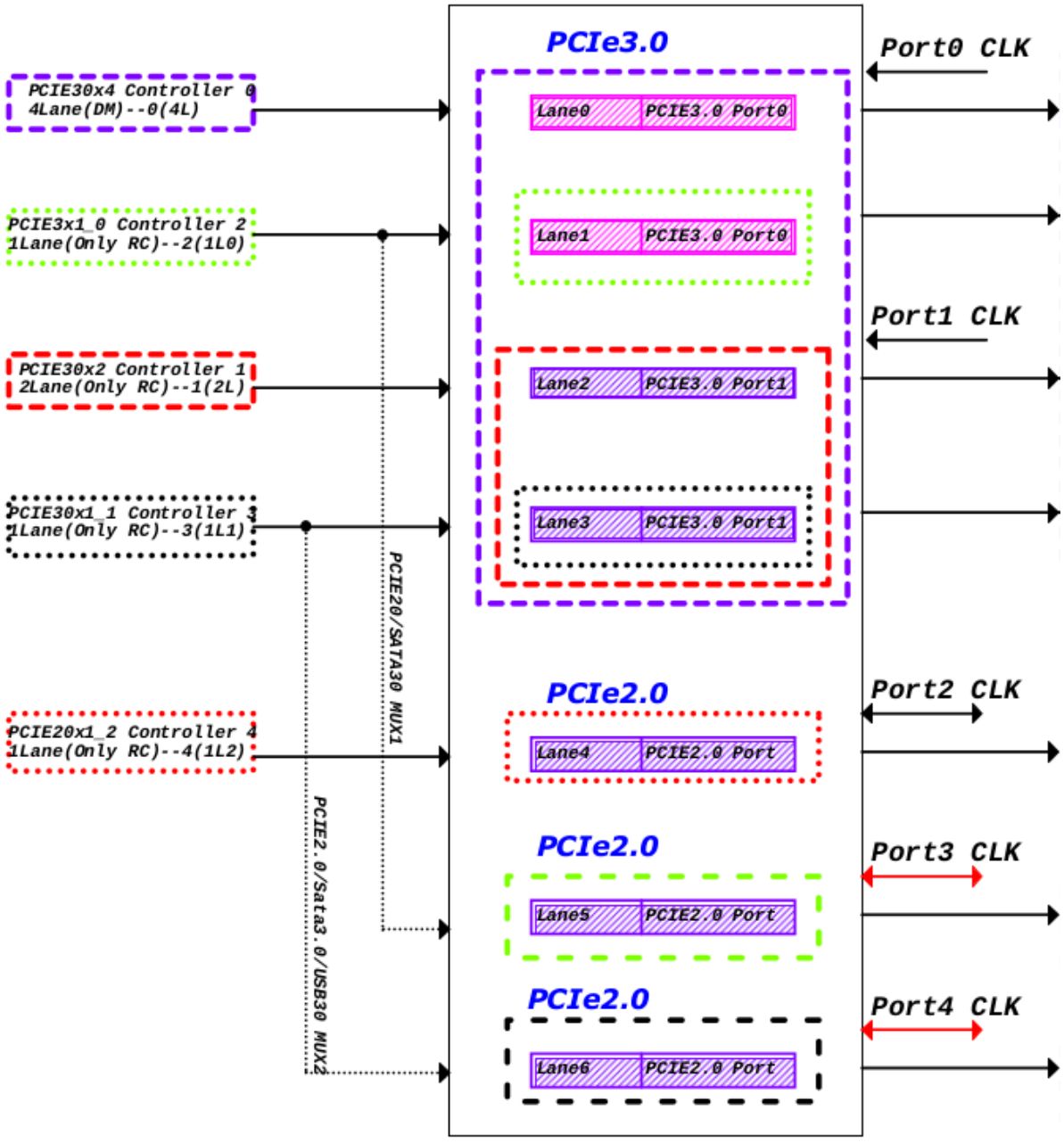
RK3588共有5个PCIe的控制器，硬件IP是一样的，配置不一样，其中一个4Lane DM模式可以支持作为EP使用，另外一个2Lane和3个1Lane控制器均只能作为RC使用。

RK3588有两种PCIe PHY，其中一种为pcie3.0PHY，含2个Port共4个Lane，另一种是pcie2.0的PHY有3个，每个都是2.0 1Lane，跟SATA和USB combo使用。

pcie3.0 PHY的4Lane可以根据实际需求拆分使用，拆分后需要合理配置对应的控制器，所有配置在DTS中完成，无需修改驱动。

使用限制：

1. pcie30phy拆分后，pcie30x4控制器，工作于2Lane模式时只能固定配合pcie30phy的port0，工作于1Lane模式时，只能固定配合pcie30phy的port0lane0；
2. pcie30phy拆分后，pcie30x2控制器，工作于2Lane模式时只能固定配合pcie30phy的port1，工作于1Lane模式时，只能固定配合pcie30phy的port1lane0；
3. pcie30phy拆分为4个1Lane，pcie3phy的port0lane1只能固定配合pcie2x1l0控制器，pcie3phy的port1lane1只能固定配合pcie2x1l1控制器;
4. pcie30x4控制器工作于EP模式，可以使用4Lane模式，或者2Lane模式使用pcie30phy的port0，pcie30phy的port1中2lane可以作为RC配合其他控制器使用。默认使用common clock作为reference clock时，无法实现pcie30phy port0的lane0工作于EP模式，lane1工作于RC模式配合其他控制器使用，因为Port0的两个lane是共用一个输入的reference clock，RC和EP同时使用clock可能会有冲突。
5. RK3588 pcie30phy 如果只使用其中一个port，另一个port也需要供电，refclk等其他信号可接地。


PCIe3.0
PCIE30x4 Controller 0
4Lane(DM)--0(4L)
Lane0
PCIE3.0 Port0
Port0 CLK
PCIE3x1_0 Controller 2
1Lane(Only RC)--2(1L0)
Lane1
PCIE3.0 Port0
PCIE30x2 Controller 1
2Lane(Only RC)--1(2L)
Lane2
PCIE3.0 Port1
Port1 CLK
PCIE30x1_1 Controller 3
1Lane(Only RC)--3(1L1)
Lane3
PCIE3.0 Port1
PCIE20/SATA30 MUX1
PCIe2.0
PCIE20x1_2 Controller 4
1Lane(Only RC)--4(1L2)
Lane4
PCIE2.0 Port
Port2 CLK
PCIE2.0/Sata3.0/USB30 MUX2
PCIe2.0
Lane5
PCIE2.0 Port
Port3 CLK
PCIe2.0
Lane6
PCIE2.0 Port
Port4 CLK

1.7.1 控制器

资源	模式	dts 节点	可用phy	内部DMA	支持ASPM	支持MMU
PCIe Gen3 x 4 lane	RC/EP	pcie3x4: pcie@fe150000	pcie30phy	2个读Channel + 2个写Channel	ALL	是
PCIe Gen3 x 2 lane	RC only	pcie3x2: pcie@fe160000	pcie30phy	否	ALL	是
PCIe Gen3 x 1 lane	RC only	pcie2x1l0: pcie@fe170000	pcie30phy, combphy1_ps	否	ALL	是
PCIe Gen3 x 1 lane	RC only	pcie2x1l1: pcie@fe180000	pcie30phy, combphy2_psu	否	ALL	是
PCIe Gen3 x 1 lane	RC only	pcie2x1l2: pcie@fe190000	combphy0_ps	否	ALL	是

1.7.2 PHY

资源	dts节点	参考时钟	拆分	是否combo
pcie30phy	phy@fee80000	外部	4Lane*1: PHY_MODE_PCIE_AGGREGATION 2Lane+2Lane: PHY_MODE_PCIE_NANBNB 2Lane+1Lane+1Lane:PHY_MODE_PCIE_NANBBI 1Lane*4: PHY_MODE_PCIE_NABIBI	pcie专用
combphy0_ps	phy@fee00000	内部/外部	-	与SATA combo
combphy1_ps	phy@fee10000	内部/外部	-	与SATA combo
combphy2_psu	phy@fee20000	内部/外部	-	与SATA/USB3 combo

1.8 RK3588S

RK3588S的PCIe较为简单，有2个1Lane控制器和2个可用于pcie 2.0的1Lane comboPHY，是一一对应关系。

1.8.1 控制器

资源	模式	dts 节点	可用phy	内部 DMA	支持 ASPM	支持 MMU
PCIe Gen3 x 1 lane	RC only	pcie2x1l1: pcie@fe180000	combphy2_psu	否	ALL	是
PCIe Gen3 x 1 lane	RC only	pcie2x1l2: pcie@fe190000	combphy0_ps	否	ALL	是

1.8.2 PHY

资源	dts节点	参考时钟	拆分	是否combo
combphy0_ps	phy@fee00000	内部/外部	-	与SATA combo
combphy2_psu	phy@fee20000	内部/外部	-	与SATA/USB3 combo

2. DTS 配置

2.1 配置要点

pcie的配置大部分是固定的，需要在板级dts配置的变量并不多，参考以下要点进行配置即可：

1. 控制器/PHY使能：确定方案后，根据原理图选择使能正确的控制器和PHY，注意控制器的index和phy的index不一定是顺序匹配的，如RK3588的pcie2x1l0不是对应combphy0_ps；
2. 控制器：部分控制器(如RK3588的pcie2x1l0和pcie2x1l1)有不止一个phy可选，按方案设计正确配置“phys”；
3. 控制器：作为RC通常需要配置"reset-gpios", 该项对应原理图PCIE的"PERSTn"信号；
4. 控制器：作为RC可能需要配置"vpcie3v3-supply"，该项对应的是PCIE的"PWREN"gpio信号控制的fixed regulator；
5. 控制器：作为EP使用时，需要修改"compatible"为EP模式对应字串;
6. PHY：pcie30phy共4个lane，可拆分使用，需要根据方案正确配置"rockchip,pcie30-phymode"模式；

2.2 RK1808 DTS配置

RK1808的dts配置，所有的实现模式都在SDK的evb代码中有范例可以参考，可以依照下面表中的模式选择匹配的内容拷贝到产品板级dts中使用。

资源	模式	参考配置	控制器节点	PHY节点	备注
PCIe Gen2 x 2 lane	RC	rk1808-evb.dtsi	pcie0	combphy	需要关闭usbdrd_dwc3 和 usbdrd3
PCIe Gen2 x 2 lane	EP	rk1808-evb.dtsi的pcie0节点添加compatible = "rockchip,rk1808-pcie-ep", "snps,dw-pcie";	pcie0	combphy	需要关闭usbdrd_dwc3和usbdrd3

2.3 RK3528 DTS配置

RK3528的dts配置，所有的实现模式都在SDK的evb代码中有范例可以参考，可以依照下面表中的模式选择匹配的内容拷贝到产品板级dts中使用。

资源	模式	参考配置	控制器节点	PHY节点	备注
PCIe Gen2 x 1 lane	RC	rk3528-evb2-ddr3-v10.dtsi	pcie2x1	combphy_pu	-

2.4 RK356X DTS配置

RK356x的dts配置，所有的实现模式都在SDK的evb代码中有范例可以参考，可以依照下面表中的模式选择匹配的内容拷贝到产品板级dts中使用。

2.4.1 RK3562 dts

资源	模式	参考配置	控制器节点	PHY节点
PCIe Gen2 x 1 lane	RC	rk3562-evb1-lp4x-v10.dtsi	pcie2x1	combphy_pu

2.4.2 RK3566 dts

资源	模式	参考配置	控制器节点	PHY节点
PCIe Gen2 x 1 lane	RC	rk3566-evb1-ddr4-v10.dtsi	pcie2x1	combphy2_psq

2.4.3 RK3568 dts

资源	模式	参考配置	控制器节点	PHY节点
PCIe Gen2 x 1 lane	RC	rk3568-evb2-lp4x-v10.dtsi	pcie2x1	combphy2_psq
PCIe Gen3 x 2 lane	RC	rk3568-evb1-ddr4-v10.dtsi	pcie3x2	pcie30phy
PCIe Gen3 拆分1 lane + 1 lane	RC	rk3568-evb6-ddr3-v10.dtsi	pcie3x2 pcie3x1	pcie30phy
PCIe Gen3 x 2 lane	EP	rk3568-iotest-ddr3-v10.dts	pcie3x2	pcie30phy

2.4.4 RK3576 dts

资源	模式	参考配置	控制器节点	PHY节点
PCIe Gen2 x 1 lane	RC	rk3576-test1.dtsi	pcie0	combphy0_ps
PCIe Gen2 x 1 lane	RC	rk3576-test1.dtsi	pcie1	combphy1_psu

2.5 RK3588 DTS配置

RK3588的控制器和PHY较多，无法在SDK的evb代码中进行穷举所有组合，按配置要点进行配置即可，这里给出几个典型范例供参考。

2.5.1 示例1 pcie3.0 4Lane RC + 2个pcie 2.0(comboPHY) (RK3588 evb1)

```
/ {
    vcc3v3_pcie30: vcc3v3-pcie30 {
        compatible = "regulator-fixed";
        regulator-name = "vcc3v3_pcie30";
        regulator-min-microvolt = <3300000>;
        regulator-max-microvolt = <3300000>;
        enable-active-high;
        gpios = <&gpio3 RK_PC3 GPIO_ACTIVE_HIGH>;
```

```
        startup-delay-us = <5000>;
        vin-supply = <&vcc12v_dcin>;
    };
};

&combphy1_ps {
    status = "okay";
};

&combphy2_psu {
    status = "okay";
};

&pcie2x1l0 {
    phys = <&combphy1_ps PHY_TYPE_PCIE>;
    reset-gpios = <&gpio4 RK_PA5 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

&pcie2x1l1 {
    phys = <&combphy2_psu PHY_TYPE_PCIE>;
    reset-gpios = <&gpio4 RK_PA2 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&rtl8111_isolate>;
    status = "okay";
};

&pcie30phy {
    rockchip,pcie30-phymode = <PHY_MODE_PCIE_AGGREGATION>;
    status = "okay";
};

&pcie3x4 {
    reset-gpios = <&gpio4 RK_PB6 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};
```

2.5.2 示例2 pcie3.0phy拆分2个2Lane RC, 3个PCIe 2.0 1Lane(comboPHY)

```
/ {
    vcc3v3_pcie30: vcc3v3-pcie30 {
        compatible = "regulator-fixed";
        regulator-name = "vcc3v3_pcie30";
        regulator-min-microvolt = <3300000>;
        regulator-max-microvolt = <3300000>;
        enable-active-high;
        gpios = <&gpio3 RK_PC3 GPIO_ACTIVE_HIGH>;
        startup-delay-us = <5000>;
        vin-supply = <&vcc12v_dcin>;
    };
};
```

```
&combphy0_ps {
    status = "okay";
};

&combphy1_ps {
    status = "okay";
};

&combphy2_psu {
    status = "okay";
};

&pcie2x1l0 {
    phys = <&combphy1_ps PHY_TYPE_PCIE>;
    reset-gpios = <&gpio4 RK_PA5 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie2x1l1 {
    phys = <&combphy2_psu PHY_TYPE_PCIE>;
    reset-gpios = <&gpio4 RK_PA2 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie2x1l2 {
    reset-gpios = <&gpio4 RK_PC1 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie30phy {
    rockchip,pcie30-phymode = <PHY_MODE_PCIE_NANBNB>;
    status = "okay";
};

&pcie3x2 {
    reset-gpios = <&gpio4 RK_PB0 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie3x4 {
    num-lanes = <2>;
    reset-gpios = <&gpio4 RK_PB6 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};
```

2.5.3 示例3 pcie3.0phy拆分为4个1Lane, 1个使用PCIe 2.0 1 Lane(comboPHY)

须知：

- pcie2x1l0/pcie2x1l1硬件上接入pcie3.0phy对应TX/RX信号，dts关闭combphy1_ps/combphy2_psu节点

参考代码：

```
/ {
    vcc3v3_pcie30: vcc3v3-pcie30 {
        compatible = "regulator-fixed";
        regulator-name = "vcc3v3_pcie30";
        regulator-min-microvolt = <3300000>;
        regulator-max-microvolt = <3300000>;
        enable-active-high;
        gpios = <&gpio3 RK_PC3 GPIO_ACTIVE_HIGH>;
        startup-delay-us = <5000>;
        vin-supply = <&vcc12v_dcin>;
    };
};

&combphy0_ps {
    status = "okay";
};

&pcie2x1l0 {
    phys = <&pcie30phy>;
    reset-gpios = <&gpio4 RK_PA5 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie2x1l1 {
    phys = <&pcie30phy>;
    reset-gpios = <&gpio4 RK_PA2 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie2x1l2 {
    reset-gpios = <&gpio4 RK_PC1 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};

&pcie30phy {
    rockchip,pcie30-phymode = <PHY_MODE_PCIE_NABIBI>;
    status = "okay";
};

&pcie3x2 {
    num-lanes = <1>;
    reset-gpios = <&gpio4 RK_PB0 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
```

```
    status = "okay";
};

&pcie3x4 {
    num-lanes = <1>;
    reset-gpios = <&gpio4 RK_PB6 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie30>;
    status = "okay";
};
```

2.6 DTS property说明

2.6.1 控制器dts常用配置

1. compatible = <?>;

可选配置项：此项目设置PCIe接口使用的是RC模式还是EP模式。

针对RK3568作为RC功能时，需要配置成compatible = "rockchip,rk3568-pcie", "snps,dw-pcie"; 而如果需要修改成EP模式，则需要修改为compatible = "rockchip,rk3568-pcie-ep", "snps,dw-pcie"; 同理针对RK1808、RK3588，仅需将rk3568字段分别替换为rk1808和rk3588。

2. reset-gpios = <&gpio3 13 GPIO_ACTIVE_HIGH>;`

必须配置项：此项是设置 PCIe 接口的 PERST#复位信号；不论是插槽还是焊贴的设备，请在原理图上找到该引脚，并正确配置。否则很有可能将无法稳定完成链路建立。另需特别提醒，如果将多个lane的PCIe接口拆分，那么所拆分出来的每个节点均需配置不同的PERST#信号线。

3. `num-lanes = <4>;`

可选配置项：此配置设置 PCIe 设备所使用的 lane 数量，已在芯片级的dtsi中配置，默认不需要调整，跟可拆分的phy组合使用时，建议按实际硬件配置。

4. `max-link-speed = <2>;`

可选配置项：此配置设置 PCIe 的带宽版本，1 表示 Gen1，2 表示 Gen2，3表示Gen3。需要注意，此配置与芯片相关，原则上不需要每个板子配置，因此我们在芯片级的dtsi中已配置，仅仅是做为一个测试手段，或者客户板子设计异常后的降级手段。

5. `status = <okay>;`

必须配置项：此配置需要在 pcie控制器节点和对应的 phy 节点同时使能。

6. `vpcie3v3-supply = <&vdd_pcie3v3>;`

可选配置项：用于配置 PCIe 外设的 3V3 供电(原则上我司的硬件参考原理图上将PCIe插槽的12V电源和3V3电源合并控制，所以配置3v3的电源之后，12V电源一并控制)。如果板级针对 PCIe 外设的 3V3 需要控制使能，则如范例所示定义一组对应的 regulator，regulator 的配置请参考 Documentation/devicetree/bindings/regulator/。

另需要特别注意，如果是PCIe3.0的控制器，一般需要外接100M晶振芯片，那么该晶振芯片的供电原则上硬件设计与PCIe外设的3V3共用。所以配置了该项之后，除了确认外设3V3供电之外，还需要确认外置晶振芯片的时钟是否输出正常。一般而言，外置晶振芯片需要一个稳定周期来输出时钟。因此请严格参考时钟芯片的手册中规定的最小数值，并在留有测试余量的基础上，在电源节点中指定startup-delay-us属性的数值。另外针对关闭电源后放电较慢的硬件设计，在电源节点中指定 off-on-delay-us属性的数

值，保证上下电操作充分。以rk3568为例，详细范例可参考rk3568-evb1-ddr4-v10.dtsi文件中的vcc3v3_pcie节点。

7. `phys`

可选配置项：用于配置控制器使用的phy的phandle引用，部分控制器可以路由到多个phy(如RK3588的pcie2x1l0和pcie2x1l1)，需要注意的是不同的phy引用方式可能有差异，comboPHY需要同时指定phy的工作模式，具体如下：

```
phys = <&pcie30phy>;
phys = <&combphy1_ps PHY_TYPE_PCIE>;
```

8. `rockchip,bifurcation;`

可选配置项：此为**RK3568芯片**特有配置。可以将pcie3x2的2个lane 拆成两个1个lane的控制器来使用。具体的配置方法就是dts中pcie3x1和pcie3x2控制器节点和pcie30phy都使能，并且pcie3x2和pcie3x1节点中都添加rockchip,bifurcation属性。可参考rk3568-evb6-ddr3-v10.dtsi。否则默认情况下，pcie3x1控制器无法使用。

此时lane0是由pcie3x2控制器使用，lane1是由pcie3x1控制器使用，硬件布板上严格按照我司原理图。另注意，此模式下两个1-lane的控制器必须同时工作在RC模式下。

另外需要特别注意，PCIe 3.0拆分成2个单lane后接两个不同外设，由于晶振及其电源是同一路控制。此时请不要将vpcie3v3-supply配置给其中某一个控制器，否则会造成获取了3v3电压操作权限的这路控制器干扰另一控制器所接的外设的正常初始化。此时应该将vpcie3v3-supply所对应的regulator配置成regulator-boot-on和regulator-always-on。

9. `prsnt-gpios = <&gpio4 15 GPIO_ACTIVE_LOW>;`

可选配置项：用于驱动识别是否存在外设以及相关外围电路，若检测到有效电平则跳过设备检测流程 。根据PCIe电气化特性协议文档，此gpio为低电平时候表示有设备接入。若板子设计与此相反，可修改为GPIO_ACTIVE_HIGH来标识高电平为有设备接入。该信号用于同套软件支持相同板型带PCIe3的产品和不带PCIe3的产品，避免pcie控制器初始化时发生rcu stall的系统异常；

10. `rockchip,perst-inactive-ms = <500>;`

可选配置项：用于配置设备#PERST复位信号的复位时间，单位为毫秒。根据PCIe Express Card Electromechanical Spec要求，下游设备电源稳定到释放#PERST最小需求为100ms，不配置此项则RK驱动默认配置了200ms。若仍不满足外设工作要求，可以酌情调整，以实测为准。

11. `rockchip,s2r-perst-inactive-ms = <1>;`

可选配置项：用于配置休眠唤醒时设备#PERST复位信号的复位时间，单位为毫秒。若不配置，它的数值等同于`rockchip,perst-inactive-ms`的配置。如果外设在休眠期间不断电，以Wi-Fi为例，则休眠过程中#PERST一直处于复位状态，因此唤醒过程中可以缩短#PERST复位信号的时间，甚至可以配置成0。

12. `rockchip,wait-for-link-ms = <1>;`

可选配置项：用于配置设备#PERST复位信号释放后的等待时间，单位为毫秒。此配置用于部分需要较长时间进行内部初始化的外设，防止因其内部初始化较久而使得系统等待链接超时的情况发生。目前常见的需要此配置的是FPGA和部分AI算力卡。

13. `supports-clkreq;`

可选配置项：仅在 RC mode 下有效，请确认已配置 CLKREQ# pinctrl iomux 为 function io 后，如果存在此属性，则指定存在从root port到下游设备的CLKREQ#信号路由，并且主机网桥驱动程序可以根据CLKREQ#信号的存在进行编程，例如，如果没有CLKREQ#信号，则将root port设置为不支持PM L1 Substates。

14. `rockchip,lpbk-master`

特殊调试配置：此配置是针对loopback信号测试，使用PCIe控制器构造模拟loopback master环境，让待测试对端设备进入slave模型，非模拟验证实验室的RX环路需求请勿配置。另注意，Gen3控制器可能需要配置compliance模式，才可以loopback slave模式。如果阅读者不理解什么是loopback测试，说明这不是你要找的配置，请勿针对此配置提问。

15. `rockchip,compliance-mode`

特殊调试配置：此配置是针对compliance信号测试，使PCIe控制器强制进入compliance测试模式或者当使用SMA夹具进入测试模式后不断电。这是一个包含两个配置的数组，数组的第一个配置表示测试模式，第二个配置表示在前述配置下的preset数值。如果使用SMA夹具测试，建议配置 `rockchip,compliance-mode=<0 0>;`；如果测试焊接设备，需要固定配置模式和preset数值，`rockchip,compliance-mode=<mode preset>;` 。mode根据需要配置成1、2或者3，分别代表2.5GT、5.0GT和8GT信号。仅在5GT和8GT模式下，preset的有效数值为0到10，分别代表P0到P10不同的协议预加重等配置，可参考附录中"关于PCIe TX加重预设值对照表"的部分。

16. `rockchip,keep-power-in-suspend`

可选配置项：仅在 RC mode 下有效，用于实现在休眠状态下不关闭外设的电源和对其进行复位。允许外设在系统休眠后脱机工作。此模式生效还需要pcie节点引用 `vpcie3v3-supply`。

2.6.2 comboPHY dts配置

注意：

- 以下配置不适用于RK1808的combphy节点。
- combphy节点数字表示Mux关系，后缀表示复用关系，p、s、u、q分别表示PCIe、SATA、USB、QSGMII 。

1. `rockchip,ext-refclk`

特殊调试配置：首先请注意此配置仅仅针对combophy。默认combphy使用SoC内部时钟方案，以RK356X为例，可参阅rk3568.dtsi节点，默认使用24MHz时钟源。除了24MHz时钟源，还支持25M和100M，仅需要调整assigned-clock-rates = <24000000>数值为所需频率即可。内部时钟源方案成本最优，所以作为SDK默认方案，但combphy仍然预留了外部晶振芯片的时钟源输入选择。如果确实需要使用外部时钟晶振芯片提供时钟的方案，请在板级dts的PCIe控制器用的combphy节点中加入rockchip,ext-refclk，且需要注意在节点中加入assigned-clock-rates = <时钟频率> 来指定外部时钟芯片输入的频率，仍然只支持24M,25M,100M三档。

2. `rockchip,enable-ssc`

特殊调试配置：首先请注意此配置仅仅针对PCIe所使用的combphy结点。默认情况下，combophy输出时钟不开启展频。如果用户需要规避一些EMI问题，可尝试在对应的combphy节点加入此配置项，开启SSC。

2.6.3 pcie30phy dts配置

1. `rockchip,pcie30-phymode`

可选配置项：该配置为pcie30phy的组合使用模式，需要合理配置，默认为4Lane共用。详细的可选内容参考`include/dt-bindings/phy/phy-snps-pcie3.h`：

```
/*
 * pcie30_phy_mode[2:0]
 * bit2: aggregation
 * bit1: bifurcation for port 1
 * bit0: bifurcation for port 0
 */
#define PHY_MODE_PCIE_AGGREGATION 4 /* PCIe3x4 */
#define PHY_MODE_PCIE_NANBNB    0   /* P1:PCIe3x2  +  P0:PCIe3x2 */
#define PHY_MODE_PCIE_NANBBI    1   /* P1:PCIe3x2  +  P0:PCIe3x1*2 */
#define PHY_MODE_PCIE_NABINB    2   /* P1:PCIe3x1*2 + P0:PCIe3x2 */
#define PHY_MODE_PCIE_NABIBI    3   /* P1:PCIe3x1*2 + P0:PCIe3x1*2 */
```

2.7 根据原理图填写DTS

2.7.1 低速IO说明

PCIe模块的芯片信号连接，除了数据线和参考时钟差分对，可能还有以下这些低速IO：

低速IO名称	RC模式	EP模式	说明
PERSTn	GPIO输出	接nPOR	必选，配置dts "reset-gpios"项
WAKE	GPIO(PMU域)输入	GPIO输出	可选，function驱动注册对应GPIO中断及唤醒源，非PCIe控制器驱动处理
PWREN	GPIO输出	无	可选，配置dts "vpcie3v3-supply"项
CLKREQ	FUNCTION	FUNCTION	可选，支持L1SS时使用，配置dts "supports-clkreq"项
PRSNT	GPIO输入	无	可选，配置dts "prsnt-gpios"项

其中只有CLKREQ使用PCIe的function功能，该信号仅L1SS功能需要使用，否则可以不接，使用时必须添加dts对应属性；

其他信号都是使用GPIO功能，请勿在pinctrl里面配置为pcie function，具体用法请参考dts对应配置说明；

PERST信号是协议要求的必选信号，其他信号根据实际项目需求进行配置；

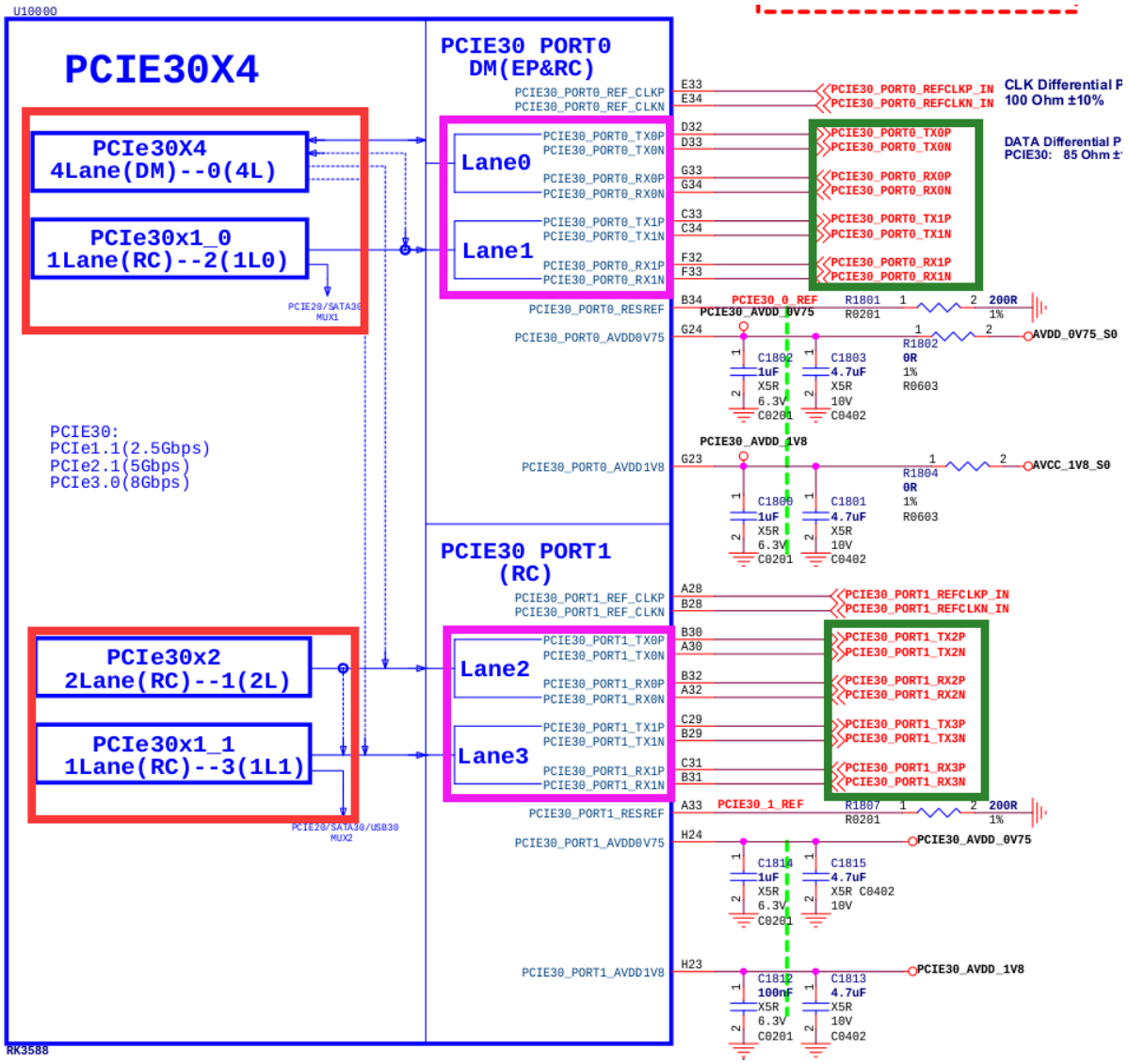
2.7.2 dts配置方法

原理图是基于IO信号的视角来描述硬件，IO信号是跟PHY的index强相关的，前面提到RK3588的controller和PHY的index可能不一致，所以看原理图的时候需要特别注意这一点。这里给出一些填写建议，并通过示例说明如何将原理图中的PHY和控制器对应到dts的节点。

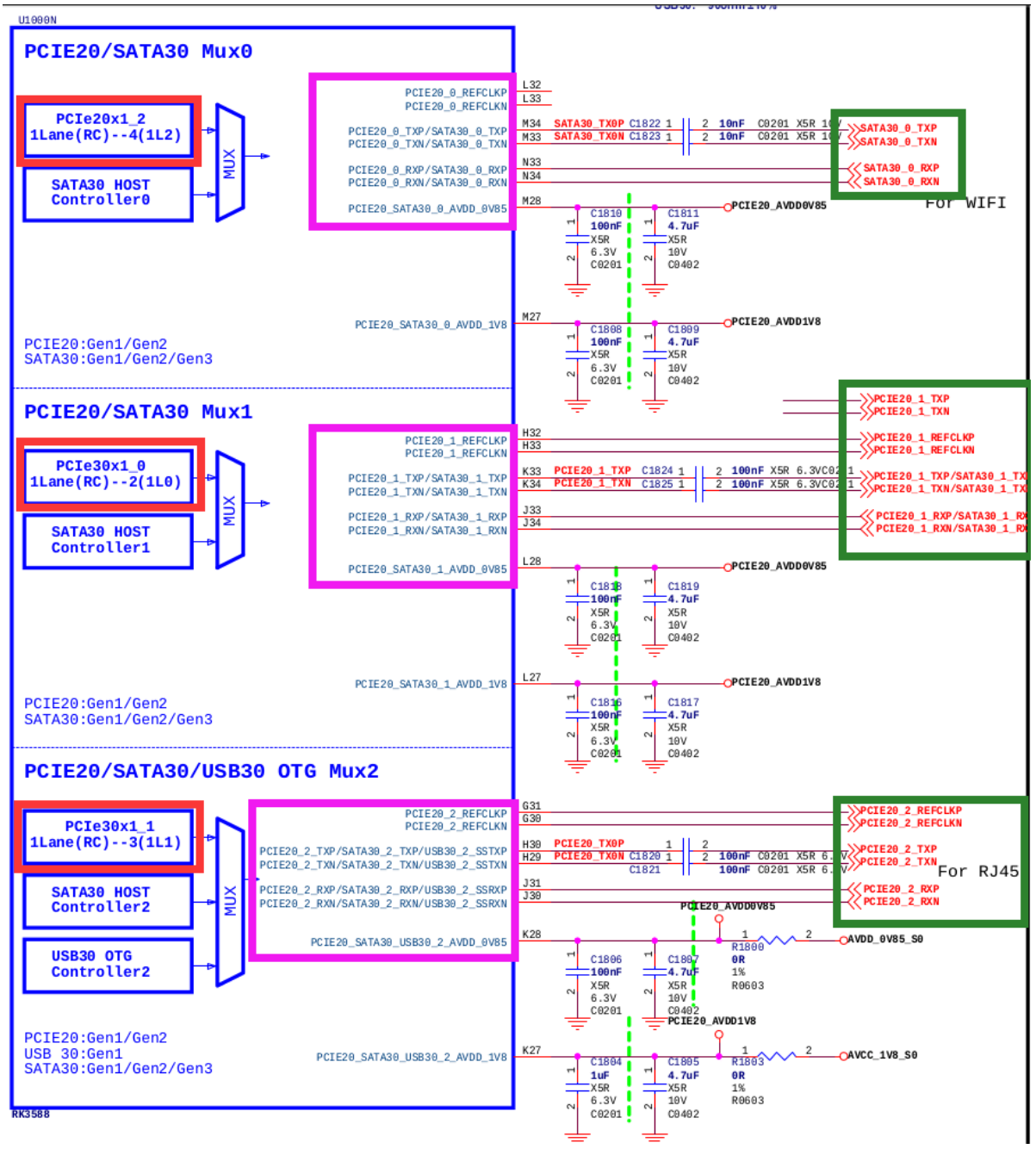
根据硬件原理图来填写dts的建议步骤：

1. 跟硬件工程师确认使用了几个PCIe设备，芯片的多个PCIe接口是如何分配的；
2. 在原理图中分别查找某个设备使用的PCIe数据线对应到哪个PHY的输出；
3. 确定当前设备使用的分别是哪个控制器和PHY，在dts中使能；
4. 确定当前pcie接口使用的控制器dts "phy"属性及模式是否选择正确，如pcie2x1ln控制器需要选择comboPHY且指定为PHY_TYPE_PCIE;
5. comboPHY可能被sata、usb、rgmii等多个控制器共用，请确认对应的其他控制器在dts中被disable；
6. 确定当前phy是否有多种工作模式，配置是否正确，如pcie30phy的不同拆分组合需要正确配置对应模式；
7. 确定当前pcie接口使用的"PERSTn"信号是哪个GPIO，正确配置到控制器dts节点；
8. 确定当前pcie接口使用的"PWREN"信号是哪个GPIO控制的，正确配置到控制器dts节点(这个配置也可以放到on board外设的dts中)；
9. 配置其他外设工作所需的硬件；

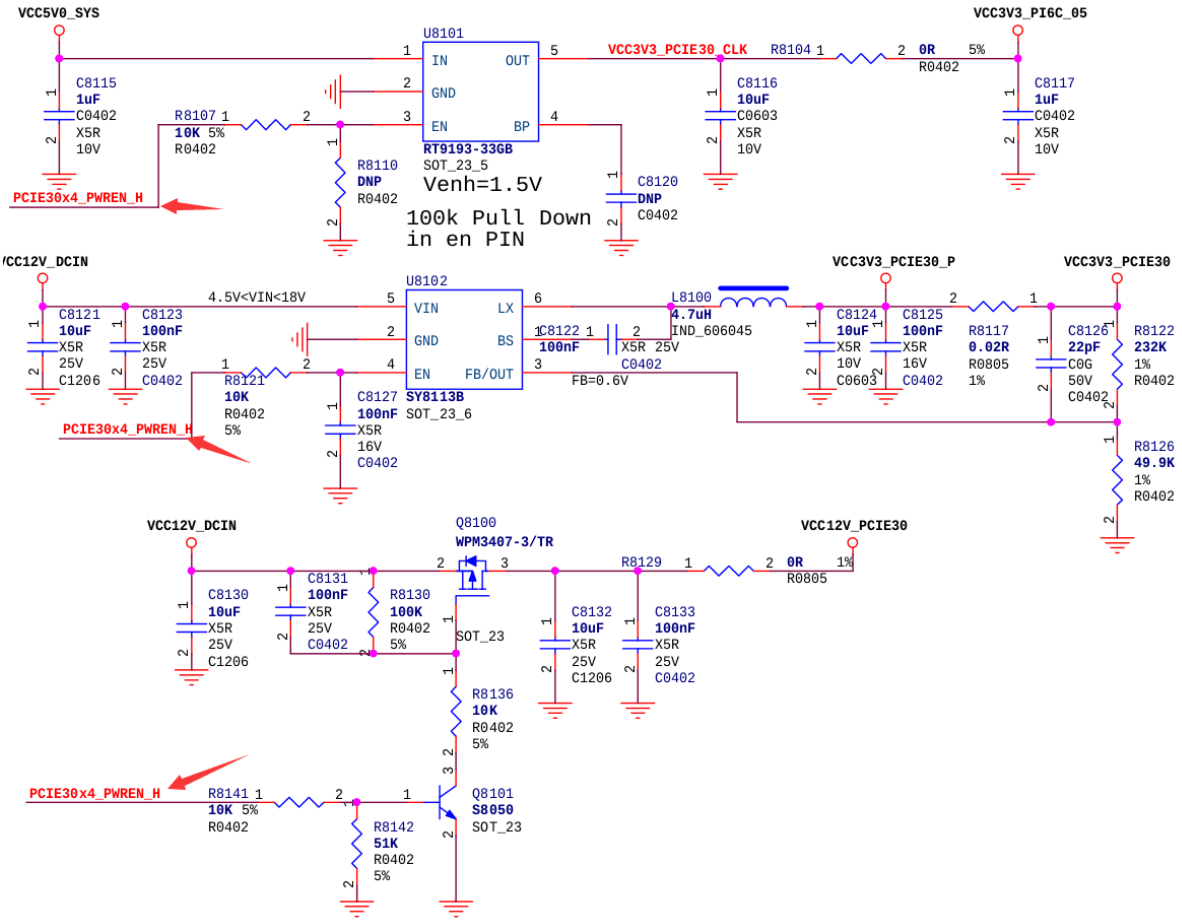
下图是RK3588 pcie30phy及其可能使用的controller，红色方框为controller，粉色方框为PHY信号，绿色方框为外设信号；实际使用哪个控制器可以通过外设信号连接来确认，也可以跟硬件工程师核对理解是否正确。下图是来自RK3588 evb1, 设备接的是一个pcie3.0 x4的slot，所以controller用的是PCIe30X4(dts命名pcie3x4)，其他几个controller都未跟这个PHY配合使用。



下图是RK3588 comboPHY及其可能使用的controller，红色方框为controller，粉色方框为PHY信号，绿色方框为外设信号；实际使用哪个控制器可以通过外设信号连接来确认，也可以跟硬件工程师核对理解是否正确。此图中Mux0的PHY(combphy0_ps)工作于SATA模式，未工作于PCIe；Mux1的PHY(combphy1_ps)配合PCIe30x1_0(dts命名为pcie2x1l0)可能工作于PCIe模式，需要由最终实际接的设备来确定；Mux2的PHY(combphy2_psu)配合PCIe30x1_1(dts命名为pcie2x1l1)工作于pcie模式用于连接一个PCIe网卡。



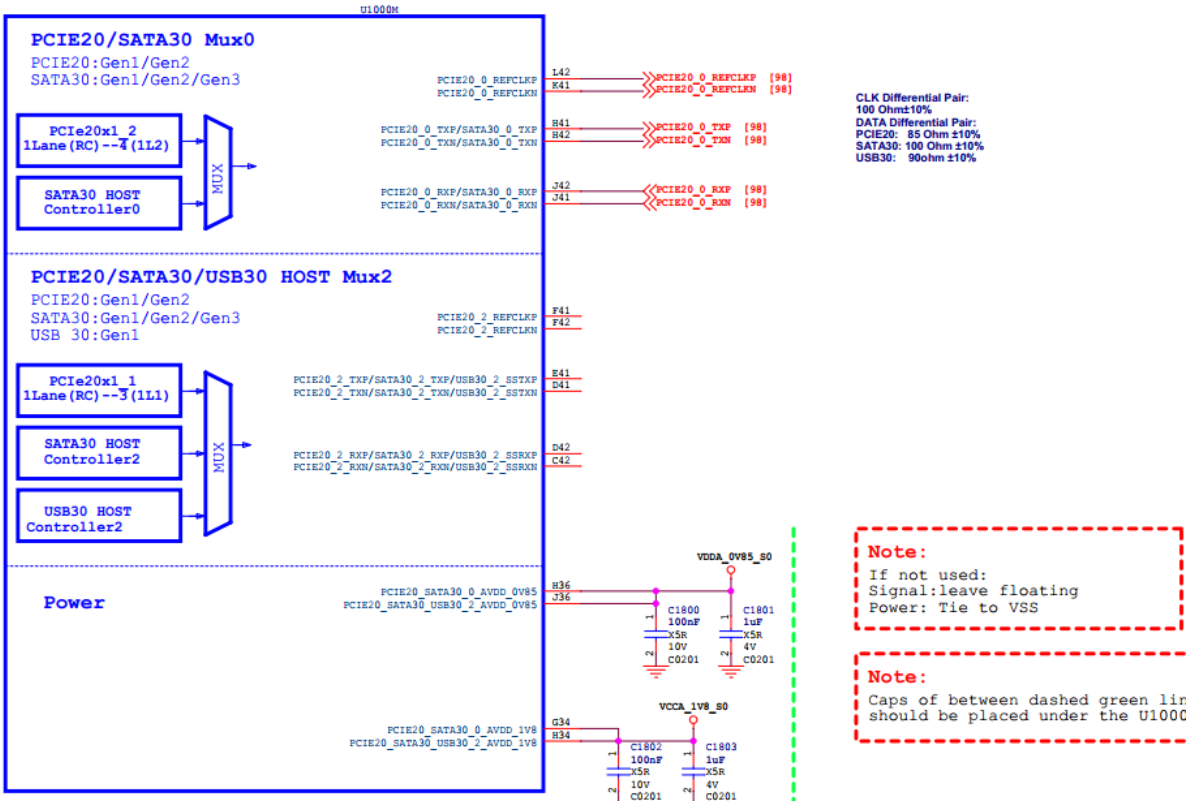
下图为RK3588 evb1的pcie3.0接口供电，可以通过PCIE30X4_PWREN_H这个信号来定位对应的RK3588 GPIO，然后填到pcie3x4控制器dts中。



2.8 Wi-Fi模块设备树编写范例

由于Wi-Fi模块一般接入PCIe2.0口，其使用的combphy复用关系复杂，且其电源使用方式、休眠模式、复位需求与其他设备有明显的不同，此处给出一个设备树编写范例，请酌情参考。

1. 查看原理图，弄清楚Wi-Fi模块所使用的combphy。combphy节点数字表示Mux关系，后缀表示复用关系，p、s、u、q分别表示PCIe、SATA、USB、 QSGMII。以此图为例，Wi-FI是接到了PCIe与SATA复用的PCIE20/SATA30 Mux0上，因此配置的应该是combphy0_ps节点。



2. 搜索dts文件，确保复用此combphy功能的其他控制器节点关闭，防止信号干扰。
3. 将wifi_reg_on信号从wireless_wlan节点挪到PCIe 3.3v电源控制节点中。
4. 如果Wi-Fi需要实现L1.x功耗模式，请参考”RC mode PM L1 Substates 支持“章节。
5. 如果Wi-Fi需要实现无线唤醒功能，需要确保 wifi_reg_on管脚在休眠时保持高电平，wifi_host_wake管脚连接到不断电的PMU IO上用于产生中断唤醒主控，具体硬件接发和软件修改详述请参考我司SDK发布文档中的《Rockchip_Developer_Guide_Linux_WIFI_BT_CN.pdf》相关章节。

```
+ vcc3v3_pcie20_wifi: vcc3v3-pcie20-wifi {
+     compatible = "regulator-fixed";
+     regulator-name = "vcc3v3_pcie20_wifi";
+     regulator-min-microvolt = <3300000>;
+     regulator-max-microvolt = <3300000>;
+     enable-active-high;
+    /*
+     * wifi_reg_on 是在vbat、vddio稳定后才使能，在reset-gpios前拉高，所以
+     * 放到PCIe的电源节点中才满足模块时序，引用wifi_poweren_gpio。
+     */
+     pinctrl-0 = <&wifi_poweren_gpio>;
+     startup-delay-us = <5000>;
+     vin-supply = <&vcc12v_dcin>;
+   };

    wireless_wlan: wireless-wlan {
        compatible = "wlan-platdata";
        wifi_chip_type = "ap6275p";
        pinctrl-names = "default";
        pinctrl-0 = <&wifi_host_wake_irq>;
        WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_HIGH>;
+       /* 注意：这里也需要配置wifi_reg_on管脚，给WiFi驱动来控制 */
+       WIFI,poweren_gpio = <&gpio0 RK_PC7 GPIO_ACTIVE_HIGH>;
        status = "okay";
```

```
    };
+
+&sata0 {
+    status = "disabled" /* sata0与pcie2x1l2复用了combphy0_ps, 需确保禁用 */
+}
+
+&combphy0_ps {
+    status = "okay"; /* 确保phy开启 */
+};
+
+&pcie2x1l2 {
+    reset-gpios = <&gpio3 RK_PD1 GPIO_ACTIVE_HIGH>;
+    rockchip,skip-scan-in-resume;
+    rockchip,perst-inactive-ms = <500>; /* 参考Wi-Fi模组手册，查询所需#PERST复位时间
*/
+    vpcie3v3-supply = <&vcc3v3_pcie20_wifi>;
+    status = "okay";
+};

&pinctrl {
     wireless-wlan {
         wifi_poweren_gpio: wifi-poweren-gpio {
+            //PCIE REG ON:  务必配置成上拉
+            rockchip,pins = <0 RK_PC7 RK_FUNC_GPIO &pcfg_pull_up>;
         };
     };
};
```

3. menuconfig 配置

1. 需要确保如下配置打开，方可正确的使用 PCIe 相关功能

```
CONFIG_PCI=y
CONFIG_PCI_DOMAINS=y
CONFIG_PCI_DOMAINS_GENERIC=y
CONFIG_PCI_SYSCALL=y
CONFIG_PCI_BUS_ADDR_T_64BIT=y
CONFIG_PCI_MSI=y
CONFIG_PCI_MSI_IRQ_DOMAIN=y
CONFIG_PHY_ROCKCHIP_SNPS_PCIE3=y
CONFIG_PHY_ROCKCHIP_NANENG_COMBO_PHY=y
CONFIG_PCIE_DW=y
CONFIG_PCIE_DW_HOST=y
CONFIG_PCIE_DW_ROCKCHIP=y
CONFIG_PCIEPORTBUS=y
CONFIG_PCIE_PME=y
CONFIG_GENERIC_MSI_IRQ=y
CONFIG_GENERIC_MSI_IRQ_DOMAIN=y
CONFIG_IRQ_DOMAIN=y
CONFIG_IRQ_DOMAIN_HIERARCHY=y
```

2. 使能 NVMe 设备(建立在 PCIe 接口的 SSD)，PCIe转接AHCI设备（SATA），PCIe转接USB设备(XHCI) 均已在默认config中打开，烦请确认。其他转接设备例如以太网卡，WiFi等请自行确认相关config配置。

```
CONFIG_BLK_DEV_NVME=y
CONFIG_SATA_PMP=y
CONFIG_SATA_AHCI=y
CONFIG_SATA_AHCI_PLATFORM=y
CONFIG_ATA_SFF=y
CONFIG_ATA=y
CONFIG_USB_XHCI_PCI=y
CONFIG_USB_XHCI_HCD=y
```

特别说明，默认内核仅支持 drivers/ata/ahci.c 中列表内的PCIe转接SATA设备，超出部分请找原厂或者代理商支持。

4. 标准EP功能件开发

RK部分芯片的PCIe控制器支持EP模式，可以将芯片开发为标准PCIe EP产品，针对EP功能的实现，请参考文档《Rockchip_Developer_Guide_PCIE_EP_Stardard_Card_CN》。

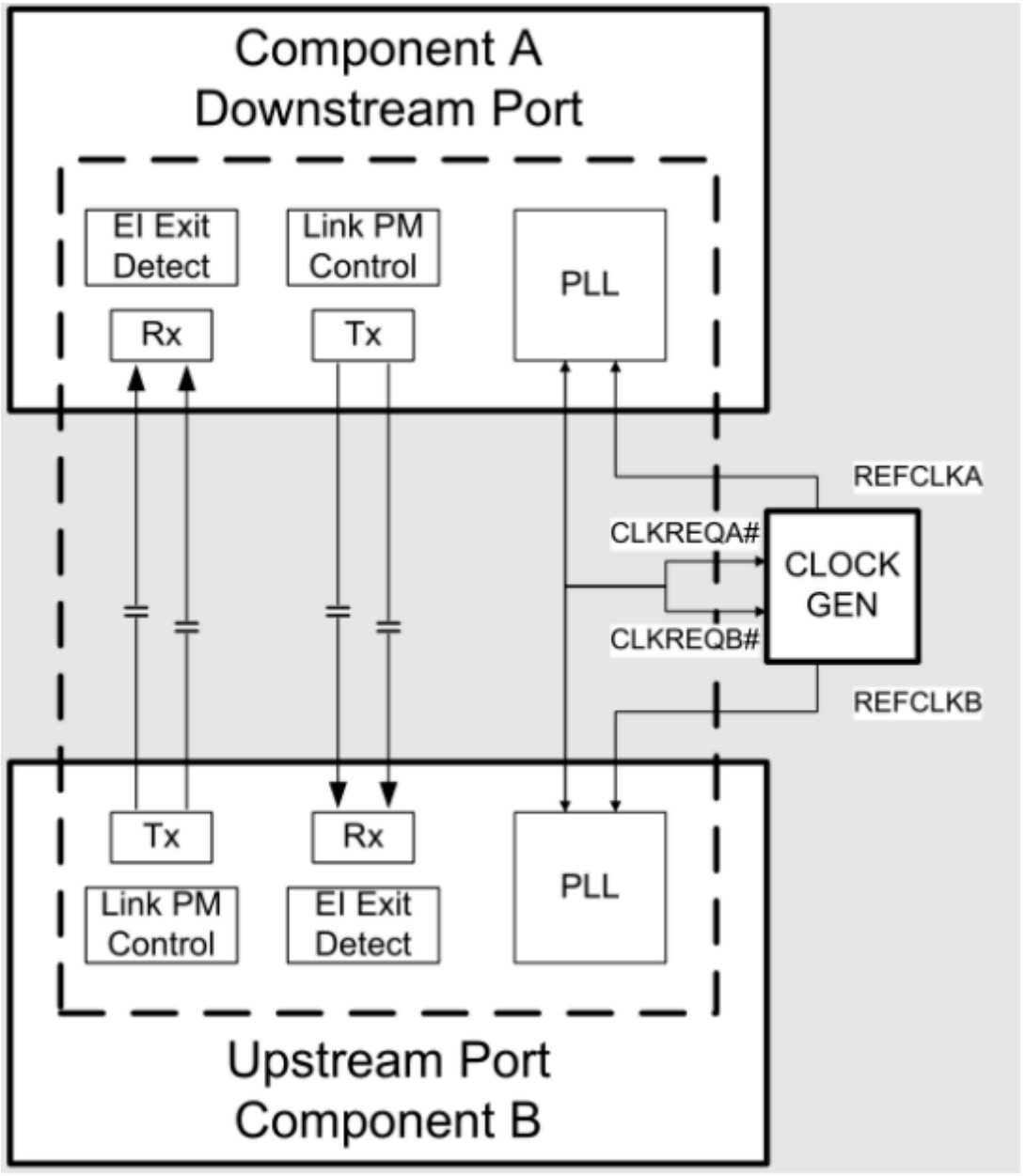
5. RC mode PM L1 Substates 支持

当确认所用 RK 主控及所接的外设都支持 PCIe PM L1 Substates，可以通过开启 PM L1 Substates 功能支持对功耗进一步优化。

进一步强调，如果目标外设中有不支持 PM L1 Substates 的设备，尤其是主板设计为 slot 外接非固定器件，请勿开启 PM L1 Substates 功能，否则部分设备将无法正常工作。

PM L1 Substates 支持的硬件电路设计：

- RC CLKREQ#、EP CLKREQ# 双端互联
- 可选优化：
    - 外部参考时钟方案，可以考虑 CLKREQ# 接入 CLOCK_GEN OE，如 CLOCK_GEN OE 为高有效，则应添加反相器



系统默认设置

关闭系统对 PM L1 Substates 支持：

- dts 未添加 ”supports-clkreq;“ 属性
- 内核宏配置 CONFIG_PCIEASPM_POWER_SUPERSAVE=n

系统开启 PM L1 Substates 支持

- 确认 RC/EP 都支持 PM L1 Substates，其中RC的支持情况可查阅”芯片资源介绍“章节之ASPM一栏。
- 硬件配置 CLKREQ# 信号：确认符合 “PM L1 Substates 支持的硬件电路设计”
- 软件配置 CLKREQ# 信号：设置 CLKREQ# iomux 为 function io
- 软件配置控制器节点添加 ”supports-clkreq;“ 属性，详细参考 "dts 可配置项 11" 说明
- 开启 PM L1 Substates 支持：
  - 方法1：内核宏配置 CONFIG_PCIEASPM_POWER_SUPERSAVE=y
  - 方法2：部分外设如 WIFI，支持驱动中使能 PM L1 Substates

说明：

- 如果不严格符合以上条件却要尝试通过开启内核宏配置，PCIe link 可能会进入异常状态，无法唤醒
- 请确认内核源码为较新代码，包含支持 L1SS 补丁：commit e18dfa93 PCI: rockchip: dw: Support PM L1 clock removing

6. 基于GPIO方式的拔插检测机制

6.1 硬件要求

1. PCIe slot的PRSNT#_1需要与主控的任意GPIO相连，作为检测脚
2. PCIe设备的电源需要软件可控制上下电

6.2 软件要求

1. 至少包含以下提交，如无请联系业务获取补丁：

```
commit 4de1a0c19e0f9804ba22e7f5e544fea317913957
Author: Shawn Lin <shawn.lin@rock-chips.com>
Date:   Tue Mar 12 16:38:46 2024 +0800

    PCI: rockchip: dw: Add gpio based hotplug
    Change-Id: I49c57755d11cc43bbf7cf9eb23542f5e1e11aaa3

commit 86f3010d7f523c9f5a2e88d9f8f1871ed89da098
Author: Vidya Sagar <vidyas@nvidia.com>
Date:   Sat Oct 1 00:57:45 2022 +0530

    FROMLIST: PCI/hotplug: Add GPIO PCIe hotplug driver
    Change-Id: Iafa798ee4d98f195f5d33d80120da0c569132548
```

2. 内核需确认打开如下配置：

```
CONFIG_HOTPLUG_PCI=y
CONFIG_HOTPLUG_PCI_GPIO=y
```

3. DTS配置参考如下

```
&pcie0 {
  reset-gpios = <&gpio4 RK_PC7 GPIO_ACTIVE_HIGH>;
  vpcie3v3-supply = <&vcc3v3_pcie0>;
  hotplug-gpios = <&gpio4 RK_PC4 IRQ_TYPE_EDGE_BOTH>;
  pinctrl-names = "default";
  pinctrl-0 = <&hot_plug0>;
  status = "okay";
};

&pinctrl {
  pcie {
```

```
      hot_plug0: hot-plug0 {
          rockchip,pins = <4 RK_PC4 RK_FUNC_GPIO &pcfg_pull_up>;
      };
  };
};
```

6.3 使用限制

1. PCIe设备带电拔插极易损坏设备和主控，在设备拔出后的卸载流程和断电流程需要一点时间，所以禁止快速热拔插。需要等到如下移除log后，再重新插入:

```
[   35.680289][  T134] pcieport 0000:00:00.0: Hot-UnPlug Event
[   35.680361][  T134] pcieport 0000:00:00.0: Power Status = 1
[   35.827183][  T134] rk-pcie 2a200000.pcie: rk_pcie_slot_disable
[   35.827303][  T134] pcieport 0000:00:00.0: Hot-UnPlug Event
[   35.827323][  T134] pcieport 0000:00:00.0: Power Status = 0
[   35.827334][  T134] pcieport 0000:00:00.0: Device is already removed
```

2. 为保证数据的完整性和系统的稳定性，需要确保系统停止访问待拔出设备。
3. 无法支持switch下游设备的单独拔插。如有此需求，首先需要确认switch支持下游设备打单独热拔插，然后参考常见应用问题中"如何对下游单个设备进行重扫描或者在线更换设备？"部分进行rescan处理，亦可达到同等效果
4. 不支持在休眠待机状态下，检测插入或者移除设备。

7. 内核 DMATEST

在进行开发前请确认目标PCIe控制器是否支持DMA传输，详细参考“芯片资源介绍”章节。

RK PCIe DMA提供基于内核module_para机制的测试机制，框架类似Linux dmatest，可以基于该框架进一步完成内核下的PCIe DMA应用。

内核版本要求

至少包含以下提交，如无请联系业务获取补丁：

```
commit a7c40cb119703e566d9d5befb8c1a7b0533dd7b7
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Tue Jan 17 17:46:48 2023 +0800

    PCI: rockchip: dw-dmatest: Suppport rc dma

    1.Set rc dma as default
    2.Changet to ep dma by sending command:
            echo 0 > ./sys/module/pcie_dw_dmatest/parameters/is_rc

    Change-Id: I9b16c328c08f220772e487c7c796b8898d74ae10
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

测试宏配置

```
CONFIG_PCIE_DW_DMATEST=y
CONFIG_ROCKCHIP_PCIE_DMA_OBJ=n
```

搭建环境

PCIe 互联模型：

- 客户自行搭建RK RC - FPGA EP环境
- RK本地测试搭建为RK RC（RK dmatest 配置）-RK EP（RK 芯片互联配置）

注意：

- MPS 配置为 256B
- RC EP都需要reserved memory，建议在0x3c000000预留64MB测试用内存（测试默认配置的地址）
- 建议关闭其他无关的 PCIe 控制器

测试

详细参考：pcie-dw-misc-dmatest.c源码

```
static int size = 0x20;
module_param(size, int, 0644);
MODULE_PARM_DESC(size, "each packet size in bytes");

static unsigned int cycles_count = 1;
module_param(cycles_count, uint, 0644);
MODULE_PARM_DESC(cycles_count, "how many erase cycles to do (default 1)");

static bool irq;
module_param(irq, bool, 0644);
MODULE_PARM_DESC(irq, "Using irq? (default: false)");

static unsigned int chn_en = 1;
module_param(chn_en, uint, 0644);
MODULE_PARM_DESC(chn_en, "Each bits for one dma channel, up to 2 channels,
(default enable chanel 0)");

static unsigned int rw_test = 3;
module_param(rw_test, uint, 0644);
MODULE_PARM_DESC(rw_test, "Read/Write test, 1-read 2-write 3-both(default 3)");

static unsigned int bus_addr = 0x3c000000;
module_param(bus_addr, uint, 0644);
MODULE_PARM_DESC(bus_addr, "Dmatest chn0 bus_addr(remote), chn1 add offset
0x100000, (default 0x3c000000)");

static unsigned int local_addr = 0x3c000000;
module_param(local_addr, uint, 0644);
MODULE_PARM_DESC(local_addr, "Dmatest chn0 local_addr(local), chn1 add offset
0x100000, (default 0x3c000000)");

static unsigned int test_dev;
module_param(test_dev, uint, 0644);
MODULE_PARM_DESC(test_dev, "Choose dma_obj device,(default 0)");
```

```
static bool is_rc = true;
module_param_named(is_rc, is_rc, bool, 0644);
MODULE_PARM_DESC(is_rc, "Test port is rc(default true)");
```

实例参考1：RC设备，channel 0，写数据，数据粒度1MB，循环次数1000，local 地址0x3c000000，remote地址0x3c000000：

```
echo 0 > ./sys/module/pcie_dw_dmatest/parameters/test_dev
echo 1 > ./sys/module/pcie_dw_dmatest/parameters/is_rc
echo 1 > ./sys/module/pcie_dw_dmatest/parameters/chn_en
echo 1 > ./sys/module/pcie_dw_dmatest/parameters/rw_test
echo 0x100000 > ./sys/module/pcie_dw_dmatest/parameters/size
echo 1000 > ./sys/module/pcie_dw_dmatest/parameters/cycles_count
echo 0x3c000000 > ./sys/module/pcie_dw_dmatest/parameters/local_addr
echo 0x3c000000 > ./sys/module/pcie_dw_dmatest/parameters/bus_addr
echo run > ./sys/module/pcie_dw_dmatest/parameters/dmatest
```

实例参考2：EP设备，channel 0/1（双线程同时运行），读写数据，数据粒度8KB，循环次数10000，local 地址 0x3c000000，remote 地址 0x3c000000：

```
echo 0 > ./sys/module/pcie_dw_dmatest/parameters/test_dev
echo 0 > ./sys/module/pcie_dw_dmatest/parameters/is_rc
echo 3 > ./sys/module/pcie_dw_dmatest/parameters/chn_en
echo 3 > ./sys/module/pcie_dw_dmatest/parameters/rw_test
echo 0x2000 > ./sys/module/pcie_dw_dmatest/parameters/size
echo 10000 > ./sys/module/pcie_dw_dmatest/parameters/cycles_count
echo 0x3c000000 > ./sys/module/pcie_dw_dmatest/parameters/local_addr
echo 0x3c000000 > ./sys/module/pcie_dw_dmatest/parameters/bus_addr
echo run > ./sys/module/pcie_dw_dmatest/parameters/dmatest
```

8. 内核 稳定性统计信息

PCIe 设备长期工作状态下出现异常，其运行状态不符合预期，可以尝试获取debugfs节点中的信息以便提供给我们进行分析。若需要启用此功能，先确保包含此提交（`pcie: rockchip: dw: Add debugfs support`），否则可以在附录章节所示redmine系统中获取`0001-pcie-rockchip-dw-Add-debugfs-support.patch`。

使用方法：

1. 明确出问题的设备所处的控制器地址节点，可从开机枚举log中查阅或者直接从dtsi中查看。
2. 以`fe16000.pcie`为例, 进入`/sys/kernel/debug/fe160000.pcie`目录
3. `echo disable > err_event` 关闭所有的事件统计功能
4. `echo clear > err_event` 清除所有的事件统计计数
5. `echo enable > err_event` 开启所有的事件统计功能
6. 开始设备老化，复现您的异常case, 复现后请执行`cat dumpfifo` 和`cat err_event`。
7. 将导出的信息与本文档附录中的Debugfs导出信息解析表进行对比，可以大致明确出现的问题。

9. 内核 错误注入测试支持

PCIe链路如果需要测试RC端funtion驱动、业务模型/EP端 firmware/双端硬件IP对错误的容错率，可以开启错误注入测试，模拟双方交互过程中可能出现的错误类型，评估双端软件、IP的稳定性。

使用方法：

```
(1) 需要包含以下提交
commit fe835d5fd3329ba629f8c4290c818ef4b8f9895d
Author: Shawn Lin <shawn.lin@rock-chips.com>
Date:   Wed Sep 4 17:04:37 2024 +0800

    PCI: rockchip: dw: Add fault injection support
    Change-Id: Ib214cc1be565bf16bafb6a847215572f35c43753

(2) 需要开启本文档中”内核 稳定性统计信息“章节所述功能，并进入对应控制器的目录

(3) echo "einj_number enable_or_disable error_type error_number" >
fault_injection
    |-------------------|-------------------------------------------------------
--|
    | 数值               |         含义
|
    |-------------------|-------------------------------------------------------
--|
    |einj_number:       |错误注入的组号，仅支持0到6，其他数值无效
  |
    |-------------------|-------------------------------------------------------
--|
    |enable_or_disable: |开启错误注入还是关闭错误注入，0表示关闭，1表示开启，其他数值无效
   |
    |-------------------|-------------------------------------------------------
--|
    |error_type:        |错误注入类型选择，根据附录所述选择einj_number所示组号所对应的错误
类型|
    |-------------------|-------------------------------------------------------
--|
    |error_number:      |错误注入数量，仅支持0到255，其他数值无效
  |
    |-------------------|-------------------------------------------------------
--|
    例如echo "2 1 2 128" > fault_injection ,代表开启einj2，注入128个NAK DLLP包

(4) 启动PCIe链路传输，例如NVMe: dd if=/dev/nvme0n1 of=/dev/null bs=1M count=5000

(5) 查看错误是否发生: cat err_event
    Rx Recovery Request: 0x1f
    ...
    Tx Nak DLLP: 0x80

(6) 分析双端软硬件，查看是否发生预期外的软硬件异常
```

10. 内核 PMU perf支持

10.1 软件与配置

```
(1)需包含以下五个提交:
commit 0270f32f207f5682a729c17e977eb87bba83823e
Author: Shuai Xue <xueshuai@linux.alibaba.com>
Date:   Fri Dec 8 10:56:50 2023 +0800

    UPSTREAM: PCI: Move pci_clear_and_set_dword() helper to PCI header
    Change-Id: I35125190a4dd8ba25e6ec14b4712750605c22285

commit 1b627c690ade9a72e3cd488e2e11edffb5d0e879
Author: Shuai Xue <xueshuai@linux.alibaba.com>
Date:   Fri Dec 8 10:56:49 2023 +0800

    UPSTREAM: PCI: Add Alibaba Vendor ID to linux/pci_ids.h
    Change-Id: I86188f119a42548ab777df0449f7d0a933f34d12

commit dcfa6c8947baeac74ab44ea8f03d3831a062c14b
Author: Shuai Xue <xueshuai@linux.alibaba.com>
Date:   Fri Dec 8 10:56:51 2023 +0800

    BACKPORT: drivers/perf: add DesignWare PCIe PMU driver
    Change-Id: I470f4dc2791168760517c77dd31a4dacd7dab591

commit 6cb6a00862fa29f815412634569e2015f86e397a
Author: Shawn Lin <shawn.lin@rock-chips.com>
Date:   Tue Sep 3 16:24:36 2024 +0800

    perf/dwc_pcie: Add support for Rockchip vendor devices
    Change-Id: I6fde80440d2fa058b38a7d927eb846f477812b5f

commit 50cb3fcd18fb9defe23ba95eb3962a287e957166
Author: Shawn Lin <shawn.lin@rock-chips.com>
Date:   Tue Sep 3 16:24:36 2024 +0800

    PCI: rockchip: dw: Add dwc pmu support for rockchip
    Change-Id: Ia27ee055aa3e63deeb7fd646411c3542b7019288

(2)内核需要开启CONFIG_PERF_EVENTS配置
(3)系统需要集成perf工具；若无，可以去本文档附录所述开发资源获取地址内下载。
```

10.2 使用说明

```
(1) 列出所有DWC PCIe PMU支持的配置
root@rk3576-buildroot:/# /userdata/perf list | grep dwc_rootport
  dwc_rootport_0/CFG_RCVRY/                          [Kernel PMU event] #链路rcvry
时间占比
```

```
  dwc_rootport_0/L0/                                [Kernel PMU event] #链路处于L0
占比
  dwc_rootport_0/L1/                                [Kernel PMU event] #链路处于L1
占比
  dwc_rootport_0/L1_1/                              [Kernel PMU event] #链路处于
L1.1占比
  dwc_rootport_0/L1_2/                              [Kernel PMU event] #链路处于
L1.2占比
  dwc_rootport_0/L1_AUX/                            [Kernel PMU event] #RK不支持状
态
  dwc_rootport_0/RX_L0S/                            [Kernel PMU event] #RX处于L0s
占比
  dwc_rootport_0/Rx_CCIX_TLP_Data_Payload/          [Kernel PMU event] #RK不支持
CCI数据统计
  dwc_rootport_0/Rx_PCIe_TLP_Data_Payload/          [Kernel PMU event] #RX TLP数
据量
  dwc_rootport_0/TX_L0S/                            [Kernel PMU event] #TX处于L0s
占比
  dwc_rootport_0/TX_RX_L0S/                         [Kernel PMU event] #TX/RX都处
于L0s占比
  dwc_rootport_0/Tx_CCIX_TLP_Data_Payload/          [Kernel PMU event] #RK不支持
CCI数据统计
  dwc_rootport_0/Tx_PCIe_TLP_Data_Payload/          [Kernel PMU event] #TX TLP数
据量
  dwc_rootport_0/one_cycle/                         [Kernel PMU event] #RK不支持
  dwc_rootport_0/rx_ack_dllp,lane=?/                [Kernel PMU event] #RX回复的
DLLP数
  dwc_rootport_0/rx_atomic,lane=?/                  [Kernel PMU event] #RK不支持
  dwc_rootport_0/rx_ccix_tlp,lane=?/                [Kernel PMU event] #RK不支持
  dwc_rootport_0/rx_completion_with_data,lane=?/    [Kernel PMU event] #RX cplt包
带数据
  dwc_rootport_0/rx_completion_without_data,lane=?/ [Kernel PMU event] #RX cplt包
不带数据
  dwc_rootport_0/rx_duplicate_tl,lane=?/            [Kernel PMU event] #RX/TL
dup错误数
  dwc_rootport_0/rx_io_read,lane=?/                 [Kernel PMU event] #RX上ior包
数
  dwc_rootport_0/rx_io_write,lane=?/                [Kernel PMU event] #RX上iow包
数
  dwc_rootport_0/rx_memory_read,lane=?/             [Kernel PMU event] #RX上memr
包数
  dwc_rootport_0/rx_memory_write,lane=?/            [Kernel PMU event] #RX上memw
包数
  dwc_rootport_0/rx_message_tlp,lane=?/             [Kernel PMU event] #RX上收到的
msg数
  dwc_rootport_0/rx_nulified_tlp,lane=?/            [Kernel PMU event] #RX上因错丢
弃的TLP数
  dwc_rootport_0/rx_tlp_with_prefix,lane=?/         [Kernel PMU event] #RX上带前缀
的TLP数
  dwc_rootport_0/rx_update_fc_dllp,lane=?/          [Kernel PMU event] #RX上收到的
流控包数
  dwc_rootport_0/tx_ack_dllp,lane=?/                [Kernel PMU event] #TX回复的
DLLP数
  dwc_rootport_0/tx_atomic,lane=?/                  [Kernel PMU event] #RK不支持
  dwc_rootport_0/tx_ccix_tlp,lane=?/                [Kernel PMU event] #RK不支持
```

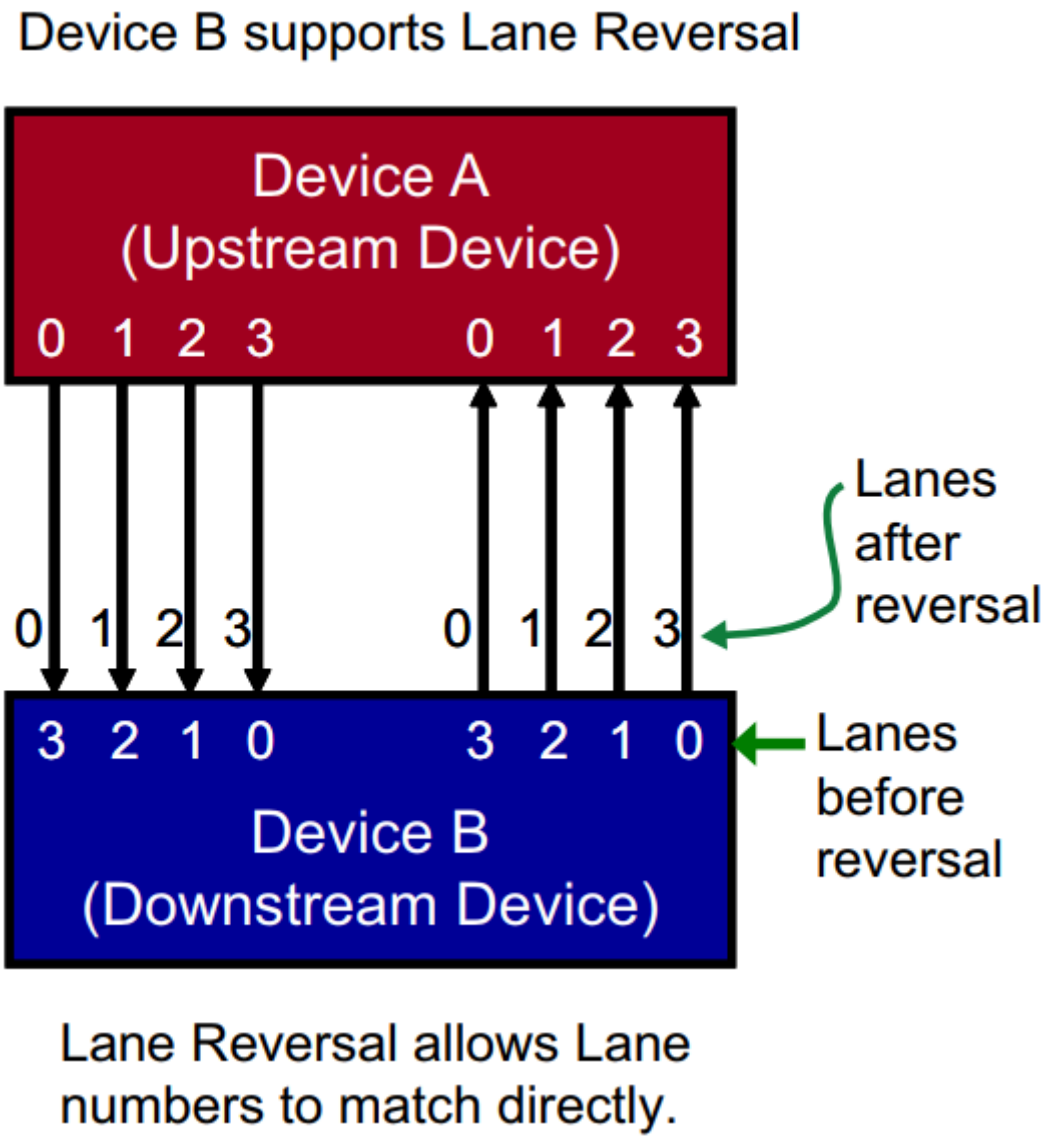
```
  dwc_rootport_0/tx_completion_with_data,lane=?/     [Kernel PMU event] #TX cplt包
带数据
  dwc_rootport_0/tx_completion_without_data,lane=?/  [Kernel PMU event] #TX cplt包
不带数据
  dwc_rootport_0/tx_configuration_read,lane=?/       [Kernel PMU event] #TX cfg-r
包数
  dwc_rootport_0/tx_configuration_write,lane=?/      [Kernel PMU event] #TX cfg-w
包数
  dwc_rootport_0/tx_io_read,lane=?/                  [Kernel PMU event] #TX上ior包
数
  dwc_rootport_0/tx_io_write,lane=?/                 [Kernel PMU event] #TX上iow包
数
  dwc_rootport_0/tx_memory_read,lane=?/              [Kernel PMU event] #TX上memr
包数
  dwc_rootport_0/tx_memory_write,lane=?/             [Kernel PMU event] #TX上memw
包数
  dwc_rootport_0/tx_message_tlp,lane=?/              [Kernel PMU event] #TX上收到的
msg数
  dwc_rootport_0/tx_nulified_tlp,lane=?/             [Kernel PMU event] #TX上因错丢
弃的TLP数
  dwc_rootport_0/tx_tlp_with_prefix,lane=?/          [Kernel PMU event] #TX发出的带
前缀TLP数
  dwc_rootport_0/tx_update_fc_dllp,lane=?/           [Kernel PMU event] #TX三发出的
流控包数

(2) 启动某项perf功能，以基于时间的统计RX TLP为例
 /userdata/perf stat -a -e dwc_rootport_0/Rx_PCIe_TLP_Data_Payload/

(3) 启动传输
root@rk3576-buildroot:/# dd if=/dev/nvme0n1 of=/dev/null bs=1M count=5000
dd if=/dev/nvme0n1 of=/dev/null bs=1M count=5000
5000+0 records in
5000+0 records out
5242880000 bytes (5.2 GB, 4.9 GiB) copied, 14.9016 s, 352 MB/s

(4) 查看统计
Performance counter stats for 'system wide':
5221423060      dwc_rootport_0/Rx_PCIe_TLP_Data_Payload/     (50.01%)
28.298528222 seconds time elapsed

(5) 同理可以测试TX TLP的数据量，则平均的RX/TX带宽计算
PCIe RX Bandwidth = Rx_PCIe_TLP_Data_Payload / 统计时长
PCIe TX Bandwidth = Tx_PCIe_TLP_Data_Payload / 统计时长

(6) Lane事件的统计
因为每个Lane都拥有相同的事件，为了避免产生大量冗余信息，建议指定Lane ID，例如:
 /userdata/perf stat -a -e dwc_rootport_0/rx_memory_read,lane=1/
```

11. 常见应用问题

11.1 当走线位置不佳时，不同lane之间能否交织?

1. 支持lane交织（Lane reversal），属于硬件协议行为，软件不需要改动。但是有如下限制：

- 4 Lane的情况下，目前RK平台仅支持全倒序交织，即RC的Lane[0.1.2.3] 分别对应 EP的Lane[3.2.1.0]，其余情况一概不支持。



- 2 Lane的情况下，同理支持RC的Lane[0.1]分别对应EP的Lane[1.0]。

2. 不支持同一侧不同组lane之间的信号的组合使用，如RC lane0 TX与lane1 RX无法组合为一组lane来外接设备。

11.2 同一个lane的差分信号能否交织?

支持PN翻转（Lane polarity），通常RC TX+接入EP RX+、RC TX-接入EP RX-，支持RC TX+接入EP RX-等PN翻转布线，且软件不需要额外处理，由PCIe控制器自动探测。

不支持TX/RX交织，例如RC的lane1 TX无法与EP的lane1TX对接。

11.3 同一个PCIe接口是否支持拆分或者合并?

RK芯片的部分PCIe口可以支持拆分和合并功能，请参阅"芯片资源介绍"章节，具体方法参考示例和dts说明。

11.4 PCIe 3.0接口支持哪些时钟输入模式?

PCIe 3.0 的PHY的输入时钟模式可以是HCSL、LPHCSL或者其他差分信号，例如支持LVDS外加电平转换等电路实现的输入时钟方案，以上所有以满足PHY指标为准。

Power Supplies:

vph Min = 1.8 V-10%; Typ = 1.8 V; Max = 1.8 V+10% or
vph Min = 1.5 V-5%; Typ = 1.5 V; Max = 1.5 V+10% or
vph Min = 1.2 V-5%; Typ = 1.2 V; Max = 1.2 V+10%
Note: 1.2 V support is pending silicon characterization and validation

vp/vpdig/vptxX Min = 0.9 V-7%; Typ = 0.9 V; Max = 0.9 V+10%

Symbol	Parameter	Min	Max	Units	Conditions
FREF_CLK	Reference clock frequency	19.2	200	MHz	Not continuous; per protocol as noted in the databook
FREF_OFFSET	Reference clock frequency offset	-100	100	ppm	All protocols except for SATA
RMSJREF_CLK <= 8 Gbps	Reference clock Random Jitter		1.6	ps_{rms}	Integrated RJ from 12 kHz to 20 MHz
			1.2		Integrated Rj from 2 MHz to 20 MHz
DJREF_CLK <= 8 Gbps	Reference clock deterministic jitter		2.8	ps, pp	0.75-10 MHz (offset)[a]
			5.6		0.2-50 MHz (offset)[a]
DCREF_CLK	Duty cycle	40	60	%	

Symbol	Parameter	Min	Max	Units	Conditions
VCMREF_CLk	Common mode input level	0	vp	V	Differential inputs
VDREF_CLK	Differential input swing	300	2*vp	mVppd	Differential input must meet both VDREF_CLK and VDSEREF_CLK
VDSEREF_CLK	Differential clock single ended voltage	0	vp		Differential input must meet both VDREF_CLK and VDSEREF_CLK
SKEWDIF_CLK	Differential clock input skew		100	ps	
SWREF_CLK	Differential clock input slew rate	0.1		V/ns	20 – 80%

Note:
For PCIe, follow the CEM specification requirements

a. DJ noise limits are for noise within an offset from the divided reference clock and harmonics of the divided reference clock

11.5 是否支持PCIe switch？贵司有没有推荐？

理论上支持，不需要任何补丁，且没有推荐列表。为了把控风险，请联系供应商借评估板，插在我司EVB上验证后再采购。

11.6 在系统中如何确定控制器与设备的对应关系？

以RK3568芯片为例：

PCIe2x1控制器给外设分配的Bus地址介于0x0$_{0xf}$，PCIe3x1控制器给外设分配的bus地址介于0x10^{0x1f}，PCIe3x2控制器给外设分配的bus地址介于0x20~0x2f。从lspci输出的信息中可以看到各设备分配到的bus地址（高位），即可确定对应关系。 第二列Class是设备类型，第三列VID:PID。Class类型请参考https://pci-ids.ucw.cz/read/PD/，厂商VID和产品PID请参考 http://pci-ids.ucw.cz/v2.2/pci.ids

```
console:/ # lspci
21:00.0 Class 0108: 144d:a808
20:00.0 Class 0604: 1d87:3566
11:00.0 Class 0c03: 1912:0014
10:00.0 Class 0604: 1d87:3566
01:00.0 Class 0c03: 1912:0014
00:00.0 Class 0604: 1d87:3566
```

我们可以看到每个控制器下游预留了16级bus来接设备，意味着每个控制器下游可以接16个设备(含switch), 一般可以满足需求，阅读者可以跳过下面的说明。如果确属需要调整，请调整rk3568.dtsi中三个控制器的bus-range分配，且务必确保不要重叠。另外，调整bus-range将导致设备的MSI(-X) RID区间变化，请同步调整msi-map。

```
bus-range = <起始地址    结束地址>

msi-map = < bus-range中的起始地址 << 8
            &its
            bus-range中的起始地址 << 8
            bus-range中分配的总线总数 << 8>
```

例如bus-range调整为0x30 ~ 0x60, 即该控制器下游设备分配的bus地址从0x30 到0x60, 总线总数 0x30个

则可配置 msi-map = <0x3000 &its 0x3000 0x3000>

依此类推，且一定要保证三个控制器的bus-range和msi-map互不重叠，且bus-range和msi-map相互适配。

11.7 如何确定PCIe设备的链路状态？

请使用服务器发布的lspci工具，执行lspci -vvv，找到对应设备的linkStat即可查看；其中Speed为速度，Width即为lane数。如需要解析其他信息，请查找搜索引擎，对照查看。

11.8 如何确定SoC针对PCIe设备可分配的MSI或者MSI-X数量?

SoC针对每个PCIe设备可分配的数量由中断控制器的资源决定。每个控制器的下游设备，RK1808/RK3566/RK3568/RK3588可分配的MSI或者MSI-X总数均是65535个，RK3528/RK3562/RK3576可分配的MSI或者MSI-X总数是8个。

11.9 是否支持Legacy INT方式？如何强制使用Legacy INTA ~ INTD的中断?

支持legacy INT方式。但Linux PCIe协议栈默认的优先级是MSI-X, MSI, Legacy INT，因此常规市售设备不会去申请Legacy INT。若调试测试需要，请参考内核中Documentation/admin-guide/kernel-parameters.txt文档，其中"pci=option[,option...] [PCI] various PCI subsystem options."描述了可以在cmdline中关闭MSI，则系统默认会强制使用Legacy INT分配机制。以RK356X安卓平台为例，可在arch/arm64/boot/dts/rockchip/rk3568-android.dtsi的cmdline参数中额外添加一项pci=nomsi，注意前后项需空格隔开：

```
bootargs = "...... pci=nomsi ......";
```

如果添加成功，则lspci -vvv可以看到此设备的MSI和MSI-X都是处于关闭状态(Enable-)，而分配了INT A中断，中断号是80。cat /proc/interrupts可查看到80中断的状态。

```
01:00.0 Class 0108: Device 14a4:22f1 (rev 01) (prog-if 02)
        Subsystem: Device 1b4b:1093
...
        Interrupt: pin A routed to IRQ 80
...
        Capabilities: [50] MSI: Enable- Count=1/1 Maskable+ 64bit+
                Address: 0000000000000000  Data: 0000
                Masking: 00000000  Pending: 00000000
...
        Capabilities: [b0] MSI-X: Enable- Count=19 Masked-
                Vector table: BAR=0 offset=00002000
                PBA: BAR=0 offset=00003000
```

11.10 芯片支持分配的BAR空间地址域有多大?

RK1808

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
PCIe2.0	-	-	0xF8000000	64MB

RK3528

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
PCIe2.0	0x100000000	1GB	0xFC000000	32MB

RK3562

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
PCIe2.0	0x300000000	1GB	0xFC000000	32MB

RK3566/RK3568

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
PCIe2.0	0x300000000	1GB	0xF4000000	32MB
PCIe3x1	0x340000000	1GB	0xF2000000	32MB
PCIe3x2	0x380000000	1GB	0xF0000000	32MB

RK3576

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
PCIe0	0x900000000	2GB	0x20000000	16MB
PCIe1	0x980000000	2GB	0x21000000	16MB

RK3588(s)

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
pcie3x4	0x900000000	1GB	0xf0000000	16MB
pcie3x2	0x940000000	1GB	0xf1000000	16MB
pcie2x1l0	0x980000000	1GB	0xf2000000	16MB
pcie2x1l1	0x9c0000000	1GB	0xf3000000	16MB
pcie2x1l2	0xa00000000	1GB	0xf4000000	16MB

PCIe控制器支持最大的64-bit的BAR空间见上图，其中外设配置空间、IO空间和MEM空间共享这地址段，且MEM空间不支持预取功能。

11.11 如果CPU运行在32位地址模式下，如何实现BAR空间的访问？

默认状态下，芯片给PCIe控制器分配的BAR空间均位于超出32位寻址的地址段。但是我们芯片有预留32位以下的一个BAR地址，对每个控制器有不同的限制。例如，RK3568芯片每个控制器的低位BAR地址仅有32MB，当RK3568的CPU运行在32位地址模式下，此时我们应该启用低位地址空间对每个PCIe节点的ranges进行重新分配。以下例子已经修改为配置空间1MB，IO空间1MB和32-bit MEM空间30MB：

```
&pcie2x1 {
        ranges = <0x00000800 0x0 0xF4000000 0x0 0xF4000000 0x0 0x100000
                  0x81000000 0x0 0xF4100000 0x0 0xF4100000 0x0 0x100000
                  0x82000000 0x0 0xF4200000 0x0 0xF4200000 0x0 0x1e00000>;
}
```

```
&pcie3x1 {
        ranges = <0x00000800 0x0 0xF2000000 0x0 0xF2000000 0x0 0x100000
                  0x81000000 0x0 0xF2100000 0x0 0xF2100000 0x0 0x100000
                  0x82000000 0x0 0xF2200000 0x0 0xF2200000 0x0 0x1e00000>;
}

&pcie3x2 {
        ranges = <0x00000800 0x0 0xF0000000 0x0 0xF0000000 0x0 0x100000
                  0x81000000 0x0 0xF0100000 0x0 0xF0100000 0x0 0x100000
                  0x82000000 0x0 0xF0200000 0x0 0xF0200000 0x0 0x1e00000>;
}
```

如需调整各个空间的大小，可参考附录中“关于PCIe地址空间配置详述”的部分。

11.12 如何查看芯片分配给外设的CPU域地址以及PCIe bus域地址，两者如何对应？

使用lspci命令可以看到各设备CPU域地址以及PCIe bus域地址

```
root@linaro-alip: /home/linaro# lspci -Vs 0002:21:00.0 -X
0002:21 :00.0 Non-Volatile memory controller: Intel Corporation SSD Pro 7600p/
760p/E 6100p Series (rev 03) (prog-if 02 [NVM Express] )
    Subsystem: Intel Corporation SSD Pro 7600p/ 760p/E 6100p Series
    Flags: bus master, fast devsel, latency 0, IRQ 114
    Memory at 380900000 (64-bit, non-prefetchable) [size=16K]
    Capabilities: [40] Power Management version 3
    ....

00: 86 80 a6 f1 06 04 10 00 03 02 08 01 00 00 00 00
10: 04 00 90 80 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 86 80 0b 39
30: 00 00 00 00 40 00 00 00 00 00 00 00 72 01 00 00
```

我们可以看出到，芯片给外设分配的CPU域地址是0x380900000；CPU若需要访问外设，可以直接用IO命令或者devmem命令读取0x380900000。而0x380900000这个CPU域地址对应的PCIe bus域地址，可从PCIe外设的BAR0地址(偏移0x10开始到0x14为止)读出“04 00 90 80”， 即为0x80900000（最低位04是表明此BAR的类型为64bit MEM）。因此当CPU发出访问0x380900000这个CPU域地址时，PCIe的地址转换服务(ATU)，通过所配置的outbound关系，可以发出0x80900000这个PCIe bus域地址，从而实现CPU访问到PCIe外设的内部信息。

而两者的对应关系在rk3568.dtsi中有定义，我们以pcie3x2为例：

```
ranges = <0x00000800 0x0 0x80000000 0x3 0x80000000 0x0 0x800000
          0x81000000 0x0 0x80800000 0x3 0x80800000 0x0 0x100000  IO空间
          0x83000000 0x0 0x80900000 0x3 0x80900000 0x0 0x3f700000>; Memory空间
```

例如我们可查看到Memory段的详细分配情况。 首段0x83000000表明此memory段为64-bit non-prefetch空间。
0x3 0x80900000为芯片分配的CPU域地址，即0x0000000380900000；0x0 0x80900000为分配的CPU域地址所对应的PCIe bus域地址，即0x0000000080900000；0x3f700000为此memory段的总大小。

因此CPU发出的CPU域访问地址与转换后的PCIe bus域地址存在0x300000000的偏移关系，这部分偏移关系在硬件上由ATU自动转化。

关于地址空间的配置详情，可参考附录中“关于PCIe地址空间配置详述”的部分。

11.13 如何对下游单个设备进行重扫描或者在线更换设备？

若有如下两点需求则需对下游设备进行重枚举。

1. 下游设备可能在不同阶段出现bar空间变化；
2. 下游设备损坏后进行在线更换；

```
(1) 找出下游所需更换或者重扫描设备，目前我们以BDF为01:00.0的这个设备为例
console:/ # /data/lspci -k
00:00.0 Class 0604: Device 1d87:3566 (rev 01)
        Kernel driver in use: pcieport
01:00.0 Class 0c03: Device 1912:0014 (rev 03)

(2) 对设备进行remove操作，可看到对应设备节点以及它所运行的pcie driver被反初始化
console:/ # echo 1 > /sys/bus/pci/devices/0000\:01\:00.0/remove
[   30.624938] xhci_hcd 0000:01:00.0: remove, state 4
[   30.624985] usb usb8: USB disconnect, device number 1
[   30.625741] xhci_hcd 0000:01:00.0: USB bus 8 deregistered
[   30.626115] xhci_hcd 0000:01:00.0: remove, state 4
[   30.626142] usb usb7: USB disconnect, device number 1
[   30.640977] xhci_hcd 0000:01:00.0: USB bus 7 deregistered
[   32.055886] vcc5v0_otg: disabling
[   32.055920] vcc3v3_lcd1_n: disabling

(3) 可再次查看，确定设备已经无法扫描到了
console:/ # /data/lspci -k
00:00.0 Class 0604: Device 1d87:3566 (rev 01)
        Kernel driver in use: pcieport

(4) 发起总线重扫描，可任选如下两条指令之一，执行后新设备恢复识别
console:/ #  echo 1 > /sys/bus/pci/devices/0000\:00\:00.0/rescan
console:/ #  echo 1 > /sys/bus/pci/rescan
[   33.222240] pci 0000:01:00.0: BAR 0: assigned [mem 0x300900000-0x300900fff
64bit]
[   33.222606] xhci_hcd 0000:01:00.0: xHCI Host Controller
[   33.224875] xhci_hcd 0000:01:00.0: new USB bus registered, assigned bus number
7
[   33.225468] xhci_hcd 0000:01:00.0: hcc params 0x002841eb hci version 0x100
quirks 0x0000000000000090
[   33.226318] usb usb7: New USB device found, idVendor=1d6b, idProduct=0002,
bcdDevice= 4.19
[   33.226329] usb usb7: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[   33.226345] usb usb7: Product: xHCI Host Controller
[   33.226355] usb usb7: Manufacturer: Linux 4.19.172 xhci-hcd
[   33.226364] usb usb7: SerialNumber: 0000:01:00.0
[   33.227661] hub 7-0:1.0: USB hub found
[   33.227716] hub 7-0:1.0: 1 port detected
[   33.228252] xhci_hcd 0000:01:00.0: xHCI Host Controller
```

```
[   33.228581] xhci_hcd 0000:01:00.0: new USB bus registered, assigned bus number
8
[   33.226816] xhci_hcd 0000:01:00.0: Host supports USB 3.0 SuperSpeed
[   33.228678] usb usb8: We don't know the algorithms for LPM for this host,
disabling LPM.
[   33.228783] usb usb8: New USB device found, idVendor=1d6b, idProduct=0003,
bcdDevice= 4.19
[   33.228796] usb usb8: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[   33.228803] usb usb8: Product: xHCI Host Controller
[   33.228809] usb usb8: Manufacturer: Linux 4.19.172 xhci-hcd
[   33.228814] usb usb8: SerialNumber: 0000:01:00.0
[   33.229360] hub 8-0:1.0: USB hub found
[   33.229406] hub 8-0:1.0: 4 ports detected
[   33.556216] usb 7-1: new high-speed USB device number 2 using xhci_hcd
[   33.700886] usb 7-1: New USB device found, idVendor=2109, idProduct=3431,
bcdDevice= 4.20
[   33.700913] usb 7-1: New USB device strings: Mfr=0, Product=1, SerialNumber=0
[   33.700920] usb 7-1: Product: USB2.0 Hub
[   33.702398] hub 7-1:1.0: USB hub found
[   33.702642] hub 7-1:1.0: 4 ports detected
```

11.14 PCIe外设及其function驱动如何处理cache一致性?

根据PCIe协议Snoop和No Snoop的定义，外设向主机系统进行访问请求，其TLP包需要设置No Snoop属性位，来表征是否需要主机系统的硬件协助其完成cache一致性的维护。

No Snoop位	Cache Coherency类型	管理模型	备注
0	Snoop	硬件保证cache一致性	Cfg包，I/O包，Message包，MSI(-x)包必须设置为0
1	No Snoop	外设驱动维护cache一致性	-

当外设发出写入主机内存的TLP包，若其No Snoop位为0，进入主机系统后，RC将检查所写入的地址是否落入其他CPU的cache内。若是，则需要在数据写入物理内存后，对所有含有此地址缓存的CPU执行invalidate操作。

当外设发来从主机内存取数据并送往外设的TLP包，若其No Snoop位为0，则RC将检查所有CPU的cache，确定该TLP包内数据所在的地址是否有被其他CPU缓存。若是，则在取数据之前进行flush操作。

RK平台的部分芯片体系并**不支持**Snoop类型，统一按照No Snoop为1处理。因此运行的function驱动需要自行维护cache一致性。支持Snoop类型的，芯片可以保证cache一致性。

不支持PCIe cache一致性的平台	支持PCIe cache一致性的平台
RK1808、RK3528、RK3562、RK3566、RK3568、RK3588(s)	RK3576

11.15 是否支持PCIe设备使用beacon方式唤醒主控?

RK平台不支持在L2的状态下，由PCIe设备发送beacon唤醒主控。如果有唤醒需求，请使用#WAKE信号当作GPIO唤醒源方式实现。

11.16 如何通过命令从RK PCIe EP发起MSI中断?

测试前确认：

- RC和EP都支持并使能MSI cap，RK RC和RK EP默认已使能MSI cap
- RC 上运行EP的function驱动，并已申请msi中断

RK PCIe EP client寄存器PCIE_CLIENT_MSI_GEN_CON 支持触发MSI中断，32bits寄存器分别对应32个msi 中断，以RK3588为例：

```
io -4 0xFE150038 1
```

除此之外，还可通过：

- 设置outbound atu，CPU写发起memory write
- 或通过DMA传输发起memory write

其他说明：

- 部分PC只能申请一个MSI中断，例如包括Linux 系统和Windows系统，主要受PC BIOS配置限制，例如：
  - Intel BIOS虚拟化及VT-d配置
  - Interrupt remapper support
- RK3588 RC上能申请32个MSI中断

11.17 如何通过命令从RK PCIe EP发起MSI-X中断?

原理

PC通过标准的MSI-X cap映射BAR4配置MSI table

RK3568实例

```
# 加载驱动后msix配置后状态，从设备端执行以下两条命令可以看到MSI-Xtable
io -4 0xfe280270 0x10001      # 测试时开启 Dbi writeable
io -4 -l 0x100 0xf6300000
io -4 0xfe280270 0x10000

f6300000:  fee02004 00000000 00000024 00000000
f6300010:  fee08004 00000000 00000024 00000000
f6300020:  fee01004 00000000 00000024 00000000
f6300030:  fee02004 00000000 00000025 00000000
f6300040:  fee08004 00000000 00000025 00000000
f6300050:  fee01004 00000000 00000025 00000000
f6300060:  fee02004 00000000 00000026 00000000
```

```
f6300070:  fee04004 00000000 00000026 00000000
f6300080:  00000000 00000000 00000000 00000001
f6300090:  00000000 00000000 00000000 00000001
f63000a0:  00000000 00000000 00000000 00000001
f63000b0:  00000000 00000000 00000000 00000001
f63000c0:  00000000 00000000 00000000 00000001
f63000d0:  00000000 00000000 00000000 00000001
f63000e0:  00000000 00000000 00000000 00000001
f63000f0:  00000000 00000000 00000000 00000001


# 由于当前业务没有提供 MSI-X 接口示例，可通过以下方式触发 MSI-X
# 设置设备 outbound，默认RK EP outbound 配置 CPU addr 为 0xf0000000，测试时未改动，或者通过内核接口完成 Bar outbound 配置
io -4 0xf6300014 0xfee00000    # 指向前面的 io -4 -l 0x100 0xf5300000 输出地址
io -4 0xf6300018 0x0

# 触发 MSI-X
io -4 0xf0002004 0x24
io -4 0xf0008004 0x24
io -4 0xf0001004 0x24
io -4 0xf0002004 0x25
io -4 0xf0008004 0x25
io -4 0xf0001004 0x25
io -4 0xf0002004 0x26
io -4 0xf0004004 0x26
```

RK3588实例

```
# 加载驱动后 msix 配置后状态，从设备端执行以下两条命令可以看到 MSI-X table
io -4 0xfe150270 0x10001       # 测试时开启 Dbi writeable
io -4 -l 0x100 0xf5300000
io -4 0xfe150270 0x10000

f5300000:  fee02004 00000000 00000024 00000000
f5300010:  fee08004 00000000 00000024 00000000
f5300020:  fee01004 00000000 00000024 00000000
f5300030:  fee02004 00000000 00000025 00000000
f5300040:  fee08004 00000000 00000025 00000000
f5300050:  fee01004 00000000 00000025 00000000
f5300060:  fee02004 00000000 00000026 00000000
f5300070:  fee04004 00000000 00000026 00000000
f5300080:  00000000 00000000 00000000 00000001
f5300090:  00000000 00000000 00000000 00000001
f53000a0:  00000000 00000000 00000000 00000001
f53000b0:  00000000 00000000 00000000 00000001
f53000c0:  00000000 00000000 00000000 00000001
f53000d0:  00000000 00000000 00000000 00000001
f53000e0:  00000000 00000000 00000000 00000001
f53000f0:  00000000 00000000 00000000 00000001


# 由于当前业务没有提供 MSI-X 接口示例，可通过以下方式触发 MSI-X
# 设置设备 outbound，默认RK EP outbound 配置 CPU addr 为 0xf0000000，测试时未改动，或者通过内核接口完成 Bar outbound 配置
```

```
io -4 0xf5300014 0xfee00000    # 指向前面的 io -4 -l 0x100 0xf5300000 输出地址
io -4 0xf5300018 0x0

# 触发 MSI-X
io -4 0xf0002004 0x24
io -4 0xf0008004 0x24
io -4 0xf0001004 0x24
io -4 0xf0002004 0x25
io -4 0xf0008004 0x25
io -4 0xf0001004 0x25
io -4 0xf0002004 0x26
io -4 0xf0004004 0x26
```

11.18 如何修改增加32bits-np映射地址空间?

背景

PCIe mmio空间为CPU发起PCIe传输的映射地址空间，物理地址专用，Dram不包括此段空间，每个PCIe控制器有专用的mmio空间，通常为32MB 32bits地址空间和1GB 64bits地址空间。

PCIe range为CPU访问地址和PCI域虚拟地址之间的映射，PCIe range分配的原则：

- 域地址与Dram物理地址不能重叠，避免EP端DMA地址混乱导致访问错误，所以;
  - 默认域地址分配在对应PCIe控制器mmio物理地址，与PCIe mmio空间一一对应
  - 扩展少量32bits-np域地址空间，建议跨用其他PCIe控制器mmio物理地址
  - 继续扩充更多32bits-np域地址空间，建议reserved 4GB内memory空间做进一步分配

默认配置提供确认，以RK3588为例：

```
[    5.325570] pci_bus 0000:00: root bus resource [mem 0xf0200000-0xf0ffffff]
                    # RC 14MB 32bits-np mem
[    5.325577] pci_bus 0000:00: root bus resource [mem 0x900000000-0x93fffffff
pref]                   # RC 1GB 64bits-pref mem
```

问题：默认分配32bits-np空间不足
log：

```
[   11.646077] pci 0000:01:00.0: reg 0x10: [mem 0x00000000-0x00ffffff 64bit]   #
Dev总共需求26MB 32bits-np，最终会对齐需求32MB空间
[   11.646113] pci 0000:01:00.0: reg 0x18: [mem 0x00000000-0x007fffff 64bit]
[   11.646150] pci 0000:01:00.0: reg 0x20: [mem 0x00000000-0x001fffff 64bit]
...
[   11.971710] pci 0000:01:00.0: BAR 0: no space for [mem size 0x01000000 64bit]
[   11.971713] pci 0000:01:00.0: BAR 0: failed to assign [mem size 0x01000000
64bit]
[   11.971717] pci 0000:01:00.0: BAR 2: no space for [mem size 0x00800000 64bit]
[   11.971720] pci 0000:01:00.0: BAR 2: failed to assign [mem size 0x00800000
64bit]
[   11.971723] pci 0000:01:00.0: BAR 4: no space for [mem size 0x00200000 64bit]
[   11.971726] pci 0000:01:00.0: BAR 4: failed to assign [mem size 0x00200000
64bit]
```

解决补丁参考

参考 TRM 确认 PCIe 专用的内存空间，然后新增所缺mem range资源。

实例1：RK3588 pcie3x4修改240MB 32bits-np映射空间配置，修改内核rk3588.dtsi对应节点range属性为：

```
                ranges = <0x00000800 0x0 0xff000000 0x0 0xf0000000 0x0 0x100000
                          0x81000000 0x0 0xff100000 0x0 0xf0100000 0x0 0x100000
                          0x82000000 0x0 0xf0000000 0x9 0x00000000 0x0 0x0f000000
                          0xc3000000 0x9 0x10000000 0x9 0x10000000 0x0
0x30000000>;
```

实例2：RK3588s pcie2x1l2修改240MB 32bits-np映射空间配置，修改内核rk3588s.dtsi对应节点range属性为：

```
                ranges = <0x00000800 0x0 0xff000000 0x0 0xf4000000 0x0 0x100000
                          0x81000000 0x0 0xff100000 0x0 0xf4100000 0x0 0x100000
                          0x82000000 0x0 0xf0000000 0xa 0x00000000 0x0 0x0f000000
                          0xc3000000 0xa 0x10000000 0xa 0x10000000 0x0
0x30000000>;
```

实例3：RK3588 pcie2x1l0修改240MB 32bits-np映射空间配置，修改内核rk3588.dtsi对应节点range属性为：

```
                ranges = <0x00000800 0x0 0xff000000 0x0 0xff000000 0x0 0x100000
                          0x81000000 0x0 0xff100000 0x0 0xff100000 0x0 0x100000
                          0x82000000 0x0 0xf0000000 0x9 0x80000000 0x0 0x0f000000
                          0xc3000000 0x9 0x90000000 0x9 0x90000000 0x0
0x30000000>;
```

实例3：RK3568 pcie3x2修改240MB 32bits-np映射空间配置，修改内核rk356x.dtsi对应节点range属性为：

```
                ranges = <0x00000800 0x0 0xff000000 0x0 0xf0000000 0x0 0x100000
                          0x81000000 0x0 0xff100000 0x0 0xf0100000 0x0 0x100000
                          0x82000000 0x0 0xf0000000 0x3 0x80000000 0x0 0x0f000000
                          0xc3000000 0x3 0x90000000 0x3 0x90000000 0x0
0x30000000>;
```

实例4：基于实例1新增0x30000000处的256M映射内存，并做按照要求做内存预留：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
index a10dad37f9cf..d1b16e13c459 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
@@ -233,6 +233,15 @@ wireless_wlan: wireless-wlan {
                WIFI,poweren_gpio = <&gpio3 RK_PB1 GPIO_ACTIVE_HIGH>;
                status = "okay";
        };
+
+       reserved-memory {
+               #address-cells = <2>;
```

```
+               #size-cells = <2>;
+               ranges;
+               pcie3x4_range: pcie3x4-range@30000000 {
+                       reg = <0x0 0xdfe00000 0x0 0x10200000>;
+               };
+       };
 };

 &backlight {
diff --git a/arch/arm64/boot/dts/rockchip/rk3588.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588.dtsi
index ad414c61fd38..096f16740e11 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588.dtsi
@@ -601,10 +601,11 @@ pcie3x4: pcie@fe150000 {
                phys = <&pcie30phy>;
                phy-names = "pcie-phy";
                power-domains = <&power RK3588_PD_PCIE>;
-               ranges = <0x00000800 0x0 0xf0000000 0x0 0xf0000000 0x0 0x100000
-                         0x81000000 0x0 0xf0100000 0x0 0xf0100000 0x0 0x100000
-                         0x82000000 0x0 0xf0200000 0x0 0xf0200000 0x0 0xe00000
-                         0xc3000000 0x9 0x00000000 0x9 0x00000000 0x0
0x40000000>;
+                ranges = <0x00000800 0x0 0xdfe00000 0x0 0xf0000000 0x0 0x100000
+                          0x81000000 0x0 0xdff00000 0x0 0xf0100000 0x0 0x100000
+                          0x82000000 0x0 0xe0000000 0x9 0x00000000 0x0
0x20000000 // 扩展至512MB
+                          0xc3000000 0x9 0x20000000 0x9 0x20000000 0x0
0x20000000>;
                reg = <0x0 0xfe150000 0x0 0x10000>,
                      <0xa 0x40000000 0x0 0x400000>;
                reg-names = "pcie-apb", "pcie-dbi";
```

11.19 如何配置max payload size?

PCIe以TLP的形式发送数据，而max payload size(简称mps)决定了pcie设备的tlp能够传输的最大字节数。mps的大小是由PCIe链路两端的设备协商决定的，PCIe设备发送TLP时，其最大payload不能超过mps的值。

内核提供基于dts的"[PCI] various PCI subsystem options."配置，详细参考“Documentation/admin-guide/kernel-parameters.txt”文档，相关配置如下：

```
pcie_bus_tune_off  #关闭mps调整，使用设备自身默认值
pcie_bus_safe      #开启mps调整，设置所有设备都支持的最大mps值
pcie_bus_perf      #开启mps调整，根据parent bus及自身capability设置为最大mps
pcie_bus_peer2peer #开启mps调整，设置所有设备为mps为128B
```

说明：

- 内核默认max payload size配置机制为pcie_bus_tune_off
- 通常可以考虑直接在dts bootargs添加pci=pcie_bus_safe属性

11.20 如何固定被枚举设备的ID号?

当系统中出现多路同类型的PCIe设备，如多个网卡，由于初始化顺序不固定的情况，所以eth0代表哪一路设备实际并不固定。同样的情况也适用于NVMe等。用户如果希望将被枚举设备的ID进行固定，需要修改对应的function驱动，修改的理论基础是设备的bus number是DTS固定分配的。以下两个例子可供参考：

```
如果RK3588接了三个网卡，想按下面顺序固定：
pcie2x1l0: pcie@fe170000 => eth1
pcie2x1l2: pcie@fe190000 => eth2
pcie2x1l1: pcie@fe180000 => eth3

查询dtsi可知：
pcie2x1l0节点：bus-range = <0x20 0x2f> -> 网卡分配的bus number为0x21 -> eth1
pcie2x1l2节点：bus-range = <0x40 0x4f> -> 网卡分配的bus number为0x41 -> eth2
pcie2x1l1节点：bus-range = <0x30 0x3f> -> 网卡分配的bus number为0x31 -> eth3

修改对应的function驱动，在register_netdev函数被调用前修改注册到网络子系统的名字。
--- a/drivers/net/ethernet/realtek/r8168/r8168_n.c
+++ b/drivers/net/ethernet/realtek/r8168/r8168_n.c
@@ -25481,6 +25481,18 @@ static const struct net_device_ops rtl8168_netdev_ops =
{
 };
 #endif

+static void pci_bus_nr_2_id(struct pci_dev *pdev, struct net_device *ndev )
+{
+
+       dev_info(&pdev->dev, "%s pdev->bus->number = 0x%x\n",
+                __func__, pdev->bus->number);
+
+       if(pdev->bus->number == 0x21)
+               strcpy(ndev->name, "eth1");
+       if(pdev->bus->number == 0x41)
+               strcpy(ndev->name, "eth2");
+       if(pdev->bus->number == 0x31)
+               strcpy(ndev->name, "eth3");
+}
+
 static int __devinit
 rtl8168_init_one(struct pci_dev *pdev,
                  const struct pci_device_id *ent)
@@ -25624,7 +25636,7 @@ rtl8168_init_one(struct pci_dev *pdev,
                 rtl8168_set_eeprom_sel_low(tp);

         rtl8168_get_mac_address(dev);
-
+       pci_bus_nr_2_id(pdev, dev); // 修改位置一定要在register_netdev()之前
         tp->fw_name = rtl_chip_fw_infos[tp->mcfg].fw_name;

同理NVMe存储器可参考类似修改
--- a/drivers/nvme/host/core.c
+++ b/drivers/nvme/host/core.c
@@ -5169,6 +5169,19 @@ static void nvme_free_ctrl(struct device *dev)
```

```
                nvme_put_subsystem(subsys);
 }

+static void pci_bus_nr_2_id(struct pci_dev *pdev, struct nvme_ctrl *ctrl)
+{
+       dev_info(&pdev->dev, "%s pdev->bus->number = 0x%x\n",
+                __func__, pdev->bus->number);
+
+       if(pdev->bus->number == 0x21)
+               ctrl->instance = 1; //pcie2x1l0 -> nvme1
+       if(pdev->bus->number == 0x41)
+               ctrl->instance = 2; //pcie2x1l2 -> nvme2
+       if(pdev->bus->number == 0x31)
+               ctrl->instance = 3; //pcie2x1l1 -> nvme3
+}
+
 /*
  * Initialize a NVMe controller structures.  This needs to be called during
  * earliest initialization so that we have the initialized structured around
@@ -5178,6 +5191,7 @@ int nvme_init_ctrl(struct nvme_ctrl *ctrl, struct device
*dev,
                const struct nvme_ctrl_ops *ops, unsigned long quirks)
 {
        int ret;
+       struct pci_dev *pdev = container_of(dev, struct pci_dev, dev);

        ctrl->state = NVME_CTRL_NEW;
        clear_bit(NVME_CTRL_FAILFAST_EXPIRED, &ctrl->flags);
@@ -5214,6 +5228,8 @@ int nvme_init_ctrl(struct nvme_ctrl *ctrl, struct device
*dev,
                goto out;
        ctrl->instance = ret;

+       pci_bus_nr_2_id(pdev, ctrl);
        device_initialize(&ctrl->ctrl_device);
        ctrl->device = &ctrl->ctrl_device;
        ctrl->device->devt = MKDEV(MAJOR(nvme_ctrl_base_chr_devt),
```

12. 异常排查

12.1 驱动加载失败

```
[    0.417008] rk-pcie 3c0000000.pcie: Linked as a consumer to regulator.14
[    0.417477] rk-pcie 3c0800000.pcie: Linked as a consumer to regulator.14
[    0.417648] rk-pcie 3c0800000.pcie: phy init failed
```

异常原因：dts中未正确开启此控制器所对应的phy节点。

```
[    0.195567] rochchip_p3phy_init: lock failed 0x6890000, check input refclk and
power supply
[    0.195585] phy phy-fe8c0000.phy.8: phy init failed --> -110
[    0.195599] rk-pcie 3c0800000.pcie: fail to init phy, err -110
[    0.195611] rk-pcie 3c0800000.pcie: phy init failed
```

异常原因：PCIe 3.0 PHY工作电源或者输入时钟异常，导致phy没有正常工作。

12.2 trainning 失败

```
PCIe Link Fail的log如下一直重复”PCIe Linking...“，LTSSM状态机可能不同

rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x0
rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x0
rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x0

或者出现"PCIe Link up"的打印， 但LTSSM状态机的bit16与bit17不等于0x3, bit0到bit8数值不大于
0x11
[    3.108325] rk-pcie fe150000.pcie: Looking up vpcie3v3-supply from device tree
[    3.126926] rk-pcie fe150000.pcie: missing legacy IRQ resource
[    3.126940] rk-pcie fe150000.pcie: IRQ msi not found
[    3.126944] rk-pcie fe150000.pcie: use outband MSI support
[    3.126947] rk-pcie fe150000.pcie: Missing *config* reg space
[    3.126954] rk-pcie fe150000.pcie: host bridge /pcie@fe150000 ranges:
[    3.126965] rk-pcie fe150000.pcie:      err 0x00f0000000..0x00f00fffff ->
0x00f0000000
[    3.126972] rk-pcie fe150000.pcie:       IO 0x00f0100000..0x00f01fffff ->
0x00f0100000
[    3.126979] rk-pcie fe150000.pcie:      MEM 0x00f0200000..0x00f0ffffff ->
0x00f0200000
[    3.126984] rk-pcie fe150000.pcie:      MEM 0x0900000000..0x090fffffff ->
0x0900000000
[    3.127007] rk-pcie fe150000.pcie: Missing *config* reg space
[    3.127028] rk-pcie fe150000.pcie: invalid resource
[    3.387304] rk-pcie fe150000.pcie: PCIe Link up, LTSSM is 0x0

真正如果link成功，应该可以看到类似log, 出现"PCIe Link up"的打印，且LTSSM状态机的bit16与
bit17等于0x3, bit0到bit8数值大等于0x11
[    2.410536] rk-pcie 3c0000000.pcie: PCIe Link up, LTSSM is 0x130011
```

异常原因：trainning 失败，外设没有处于工作状态或者信号异常。首先检测下 reset-gpios 这个是否配置对了。其次，检测下外设的3V3供电是否有，是否足够，部分外设需要12V电源。最后测试复位信号与电源的时序是否与此设备的spec冲突。如果都无法解决，大概率需要定位信号完整性，需要拿出测试眼图和PCB给到我司硬件，并且最好我们建议贵司找实验室提供一份测试TX兼容性信号测试报告。

另外还建议客户打开pcie-dw-rockchip.c中的RK_PCIE_DBG，抓一份log以便分析。请阅读者注意，如果有多个控制器同时使用，抓log前请先把不使用或者没问题的设备对应的控制器disable掉，这样log会好分析一点。所抓取的log中，将会出现类似“rk-pcie 3c0000000.pcie: fifo_status = 0x144001”等信息。fifo_status的末尾两位是PCIe链路的ltssm状态机，可以根据状态机信息判断异常发生的大致情况。芯片的PCIe ltssm状态机信息可参照文末附录部分。

12.3 PCIe3.0控制器初始化设备系统异常

```
[   21.523506] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
[   21.523557] rcu:     1-...0: (0 ticks this GP) idle=652/1/0x4000000000000000
softirq=30/30 fqs=2097
[   21.523579] rcu:     3-...0: (5 ticks this GP) idle=4fa/1/0x4000000000000000
softirq=35/36 fqs=2097
[   21.523590] rcu:     (detected by 2, t=6302 jiffies, g=-1151, q=98)
[   21.523610] Task dump for CPU 1:
[   21.523622] rk-pcie         R  running task        0    55      2 0x0000002a
[   21.523640] Call trace:
[   21.523666]  __switch_to+0xe0/0x128
[   21.523682]  0x43752cfcfe820900
[   21.523694] Task dump for CPU 3:
[   21.523704] kworker/u8:0    R  running task        0     7      2 0x0000002a
[   21.523737] Workqueue: events_unbound enable_ptr_key_workfn
[   21.523751] Call trace:
[   21.523767]  __switch_to+0xe0/0x128
[   21.523786]  event_xdp_redirect+0x8/0x90
[   21.523816] rcu: INFO: rcu_sched detected stalls on CPUs/tasks:
[   21.523840] rcu:     1-...0: (50 ticks this GP) idle=652/1/0x4000000000000000
softirq=7/30 fqs=2099
[   21.523859] rcu:     3-...0: (55 ticks this GP) idle=4fa/1/0x4000000000000000
softirq=5/36 fqs=2099
[   21.523870] rcu:     (detected by 2, t=6302 jiffies, g=-1183, q=1)
[   21.523887] Task dump for CPU 1:
[   21.523898] rk-pcie         R  running task        0    55      2 0x0000002a
[   21.523915] Call trace:
[   21.523931]  __switch_to+0xe0/0x128
[   21.523944]  0x43752cfcfe820900
[   21.523955] Task dump for CPU 3:
[   21.523965] kworker/u8:0    R  running task        0     7      2 0x0000002a
[   21.523990] Workqueue: events_unbound enable_ptr_key_workfn
[   21.524004] Call trace:
```

异常原因：如果系统卡住此log附近，则表明PCIe3.0的PHY工作异常。请依次检查

- 外部晶振芯片的时钟输入是否异常，如果无时钟或者幅度异常，将导致phy无法锁定。
- 检查 PCIE30_AVDD_0V9 和PCIE30_AVDD_1V8电压是否满足要求。
- 检查板级DTS中是否有将PCIe的低速IO进行配置，如有请删除再测试；并参考'低速IO说明'部分了解这些IO的具体使用方式。
- RK3588 pcie30phy 如果只使用其中一个port，另一个port也需要供电，检查是否符合要求。

12.4 PCIe2.0控制器初始化设备系统异常

```
[   21.523870] rcu:     (detected by 2, t=6302 jiffies, g=-1183, q=1)
[   21.523887] Task dump for CPU 1:
[   21.523898] rk-pcie         R  running task        0    55      2 0x0000002a
[   21.523915] Call trace:
[   21.523931]  __switch_to+0xe0/0x128
[   21.523944]  0x43752cfcfe820900
[   21.523955] Task dump for CPU 3:
[   21.523965] kworker/u8:0    R  running task        0     7      2 0x0000002a
[   21.523990] Workqueue: events_unbound enable_ptr_key_workfn
[   21.524004] Call trace:
```

异常原因：如果系统卡住此log附近，则表明PCIe2.0的PHY工作异常。以RK3568的combphy2_psq这个PHY为例，请依次检查

- 检查 PCIE30_AVDD_0V9 和PCIE30_AVDD_1V8电压是否满足要求。
- 检查板级DTS中是否有将PCIe的低速IO进行配置，如有请删除再测试；并参考'低速IO说明'部分了解这些IO的具体使用方式。
- 修改combphy2_psq的驱动phy-rockchip-naneng-combphy.c，在rockchip_combphy_init函数的末尾增加如下代码，检查PHY内部的一些配置：

```
val = readl(priv->mmio + (0x27 << 2));
dev_err(priv->dev, "TXPLL_LOCK is 0x%x PWON_PLL is 0x%x\n",
val & BIT(0), val & BIT(1));
val = readl(priv->mmio + (0x28 << 2));
dev_err(priv->dev, "PWON_IREF is 0x%x\n", val & BIT(7));
```

首先查看TXPLL_LOCK是否为1，如果不是，表明PHY没有lock完成。其次查看PWON_IREF是否为1，如果不为1，则表明PHY时钟异常。此时尝试切换combophy的时钟频率，修改rk3568.dtsi中的combphy2_psq的assigned-clock-rates，依次调整为25M或者100M进行尝试。

- 如果调整以上步骤均无效，请将PHY内部的时钟bypass到refclk差分信号脚上，进行测量。bypass加在rockchip_combphy_pcie_init函数的末尾，根据芯片的不同，设置如下代码所示

```
/* For RK356X, RK3588 */
u32 val;
val = readl(priv->mmio + (0xd << 2));
val |= BIT(5);
writel(val,priv->mmio + (0xd << 2));

/* For RK3528 */
u32 val;
val = readl(priv->mmio + 0x108);
val |= BIT(29);
writel(val, priv->mmio + 0x108);
```

设置完成后，请依次配置combphy2_psq的时钟频率为24M,25M以及100M，用示波器从PCIe的refclk差分信号脚上测量时钟情况，检查频率和幅值、抖动是否满足要求。

还需特别注意：由于PCIe 2.0接口与SATA2接口复用，如果两者被错误地同时打开，开机过程或者休眠唤醒过程也会出现类似log。

12.5 PCIe外设Memory BAR资源分配异常

外设分配Memory资源异常主要分为三类：

- **地址空间超出平台限制**。各平台地址空间大小可查阅**常见应用问题**章节之"芯片支持分配的BAR空间地址域有多大"。此类异常的典型log如下所示，其特征关键字为`no space for`, 表明21号总线外设向RK平台申请3GB的64bit memory空间，超出了限制导致无法分配资源。若为市售设备，将不受RK芯片支持；若为定制设备，请联系设备vendor确认是否可以修改其BAR空间容量编码。

```
3.286864] pci 0002:20:00.0: bridge configuration invalid ([bus 01-ff]),
reconfiguring
3.286886] scanning [bus 00-00] behind bridge, pass 1
3.288165] pci 0002:21 :00.0: supports D1 D2
3.288170] pci 0002:21 :00.0: PME# supported from DO D1 D3hot
3.298238] pci bus 0002:21: busn res: [bus 21-2f] end is updated to 21
3.298441] pci 0002:21:00.0: BAR 1: no space for [mem size 0xe0000000 ]
3.298456] pci 0002:21:00.0: BAR 1: failed to assign [mem size 0xe0000000 ]
3.298473] pci 0002:21:00.0: BAR 2: assigned [mem 0x380900000- 0x38090ffff pref ]
3.298488] pci 0002:21:00.0: PCI bridge to [bus 21]
```

- **class类型非法**。其特征关键字为`class 0x000000`, 且lspci输出可见其BAR资源均显示unassigned状态。应对此问题的正确做法是联系模块供应商，对模块的class类型进行正确的配置。如果无法修改或者作为临时性的解决方案，可以参考所示补丁尝试对模块的class类型进行软件修复, 需要填入此异常模块的VID和PID, 并且class修复为其真实类型(请参考**常见应用问题**章节之"在系统中如何确定控制器与设备的对应关系"部分关于class类型的描述进行配置)。

```
[    2.335899] pci 0000:02:04.0: [13fe:1730] type 00 class 0x000000
[    2.335986] pci 0000:02:04.0: reg 0x10: [io  0x0000-0x00ff]
[    2.336023] pci 0000:02:04.0: reg 0x14: [mem 0x00000000-0x0000007f]
[    2.336058] pci 0000:02:04.0: reg 0x18: [mem 0x00000000-0x000000ff]
[    2.342340] pci_bus 0000:02: busn_res: [bus 02-ff] end is updated to 02
[    2.342444] pci 0000:00:00.0: BAR 8: assigned [mem 0xf4200000-0xf42fffff]
[    2.342484] pci 0000:00:00.0: BAR 6: assigned [mem 0xf4300000-0xf430ffff pref]
[    2.342496] pci 0000:00:00.0: BAR 7: assigned [io  0x1000-0x1fff]
[    2.342510] pci 0000:01:00.0: BAR 8: assigned [mem 0xf4200000-0xf42fffff]
[    2.342519] pci 0000:01:00.0: BAR 7: assigned [io  0x1000-0x1fff]
[    2.342529] pci 0000:01:00.0: PCI bridge to [bus 02]
[    2.342546] pci 0000:01:00.0:   bridge window [io  0x1000-0x1fff]
[    2.342572] pci 0000:01:00.0:   bridge window [mem 0xf4200000-0xf42fffff]
[    2.342616] pci 0000:00:00.0: PCI bridge to [bus 01-ff]
[    2.342631] pci 0000:00:00.0:   bridge window [io  0x1000-0x1fff]
[    2.342645] pci 0000:00:00.0:   bridge window [mem 0xf4200000-0xf42fffff]
[    2.344896] pcieport 0000:00:00.0: Signaling PME with IRQ 87

lspci -vvv:

0000: 02:04.0 Non-VGA unclassified device: Advantech Co. Ltd Device 1730
Subsystem: Advantech Co. Ltd Device 1730
F1ags: medium devsel,  IRQ 255
I/0 ports at[<unassigned> _disabled] [size=256]
Memory at <unass1gned> (3z-bit,  non-prefetchabie) [disabled] [size=128]
Memory at <unassigned> (32-bit, non-prefetchable) [disab1ed] [size=256]
```

```
--- a/drivers/pci/quirks.c
+++ b/drivers/pci/quirks.c
@@ -207,6 +207,13 @@ static void quirk_mmio_always_on(struct pci_dev *dev)
 DECLARE_PCI_FIXUP_CLASS_EARLY(PCI_ANY_ID, PCI_ANY_ID,
                               PCI_CLASS_BRIDGE_HOST, 8, quirk_mmio_always_on);

+static void quirk_class_id_fixup(struct pci_dev *dev)
+{
+       dev->class = 0x123456; /* Fixup your own class id ! */
+}
+
+DECLARE_PCI_FIXUP_CLASS_EARLY(PCI_VENDOR_ID_FOO, PCI_DEVICE_ID_BAR, 8,
quirk_class_id_fixup); /* Please Add Correct Vendor ID and Device ID ! */
```

- **外设的BAR寄存器校验异常**。其特征关键字为`assigned [mem`和`error updating`，且lspci输出可见其BAR资源均显示unassigned状态。出现此问题的原因是外设工作状态异常，导致BAR地址写入外设后，回读校验失败。其可能的因素是低功耗支持异常，或可能是外设对于非Gen1的RC支持存在问题，亦或者是其他类型的模块工作异常。

```
[    2.460092] pci 0002:21:00.0: calc_l1ss_pwron: Invalid T_PwrOn scale: 3
[    2.472049] pci_bus 0002:21: busn_res: [bus 21-2f] end is updated to 21
[    2.472089] pci 0002:20:00.0: BAR 8: assigned [mem 0xf2200000-0xf23fffff]
[    2.472105] pci 0002:20:00.0: BAR 6: assigned [mem 0xf2400000-0xf240ffff pref]
[    2.472124] pci 0002:21:00.0: BAR 0: assigned [mem 0xf2200000-0xf23fffff
64bit]
[    2.472139] pci 0002:21:00.0: BAR 0: error updating (0xf2200004 != 0xffffffff)
[    2.472153] pci 0002:21:00.0: BAR 0: error updating (high 0x000000 !=
0xffffffff)
```

此类问题，首先请测量模组供电、#PERST复位信号和100M参考时钟三者的控制时序是否满足模组手册要求，例如QCNFA765模块如果使用常供电方式，就使得时序异常进而导致模块工作异常，产生此问题。如果硬件外围设计与接口时序均满足模块手册要求，则建议依次尝试下列的补丁。若依然无效，联系供应商协助分析模块异常原因。

1. 禁止此模块的ASPM功能，其中PID和VID需要填写设备真实的ID。

```
--- a/drivers/pci/quirks.c
+++ b/drivers/pci/quirks.c
@ -2356,6 +2356,7 @ static void quirk_disable_aspm_l0s_l1(struct pci_dev *dev) *
disable both L0s and L1 for now to be safe.
*/
DECLARE_PCI_FIXUP_FINAL(PCI_VENDOR_ID_ASMEDIA, 0x1080,
quirk_disable_aspm_l0s_l1);
+DECLARE_PCI_FIXUP_FINAL(PID, VID, quirk_disable_aspm_l0s_l1);  /* Please Add
Correct Vendor ID and Device ID to disable L0s and L1 ASPM */
```

2. 限制对应控制的模式为Gen1，在dtsi中修改对应节点的max-link-speed，示例如下：

```
--- a/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
@@ -4552,7 +4552,7 @@
        num-ib-windows = <8>;
        num-ob-windows = <8>;
        num-viewport = <4>;
-       max-link-speed = <2>;
+       max-link-speed = <1>;
        msi-map = <0x4000 &its0 0x4000 0x1000>;
                num-lanes = <1>;
                phys = <&combphy0_ps PHY_TYPE_PCIE>;
```

3. 增加外设供电稳定时间以及#PERST复位时间，示例如下：

```
--- a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
@@ -159,7 +159,7 @@
                regulator-max-microvolt = <3300000>;
                enable-active-high;
                gpios = <&gpio3 RK_PC3 GPIO_ACTIVE_HIGH>;
-               startup-delay-us = <5000>;
+               startup-delay-us = <500000>;
                vin-supply = <&vcc12v_dcin>;
        }; /* vcc3v3_pcie30 */

@@ -551,6 +551,7 @@
 &pcie3x4 {
        reset-gpios = <&gpio4 RK_PB6 GPIO_ACTIVE_HIGH>;
        vpcie3v3-supply = <&vcc3v3_pcie30>;
+       rockchip,perst-inactive-ms = <1000>;
        status = "okay";
 };
```

4. 已知高通部分系列的wifi模块存在异常，受影响的模块包AR9xxx系列和QCA9xxx系列。非高通系列模块无需此补丁。

```
--- a/drivers/pci/pcie/aspm.c
+++ b/drivers/pci/pcie/aspm.c
@@ -192,12 +192,56 @@ static void pcie_clkpm_cap_init(struct pcie_link_state
*link, int blacklist)
    link->clkpm_disable = blacklist ? 1 : 0;
 }

+static int pcie_downgrade_link_to_gen1(struct pci_dev *parent)
+{
+   u16 reg16;
+   u32 reg32;
+   int ret;
+
+   /* Check if link is capable of higher speed than 2.5 GT/s */
+   pcie_capability_read_dword(parent, PCI_EXP_LNKCAP, &reg32);
+   if ((reg32 & PCI_EXP_LNKCAP_SLS) <= PCI_EXP_LNKCAP_SLS_2_5GB)
+       return 0;
+
```

```
+	/* Check if link speed can be downgraded to 2.5 GT/s */
+	pcie_capability_read_dword(parent, PCI_EXP_LNKCAP2, &reg32);
+	if (!(reg32 & PCI_EXP_LNKCAP2_SLS_2_5GB)) {
+		pci_err(parent, "ASPM: Bridge does not support changing Link Speed to 2.5
GT/s\n");
+		return -EOPNOTSUPP;
+	}
+
+	/* Force link speed to 2.5 GT/s */
+	ret = pcie_capability_clear_and_set_word(parent, PCI_EXP_LNKCTL2,
+						 PCI_EXP_LNKCTL2_TLS,
+						 PCI_EXP_LNKCTL2_TLS_2_5GT);
+	if (!ret) {
+		/* Verify that new value was really set */
+		pcie_capability_read_word(parent, PCI_EXP_LNKCTL2, &reg16);
+		if ((reg16 & PCI_EXP_LNKCTL2_TLS) != PCI_EXP_LNKCTL2_TLS_2_5GT)
+			ret = -EINVAL;
+	}
+
+	if (ret) {
+		pci_err(parent, "ASPM: Changing Target Link Speed to 2.5 GT/s failed:
%d\n", ret);
+		return ret;
+	}
+
+	pci_info(parent, "ASPM: Target Link Speed changed to 2.5 GT/s due to
quirk\n");
+	return 0;
+}
+
 static bool pcie_retrain_link(struct pcie_link_state *link)
 {
 	struct pci_dev *parent = link->pdev;
 	unsigned long end_jiffies;
 	u16 reg16;

+	if ((link->downstream->dev_flags &
PCI_DEV_FLAGS_NO_RETRAIN_LINK_WHEN_NOT_GEN1) &&
+	    pcie_downgrade_link_to_gen1(parent)) {
+		pci_err(parent, "ASPM: Retrain Link at higher speed is disallowed by
quirk\n");
+		return false;
+	}
+
 	pcie_capability_read_word(parent, PCI_EXP_LNKCTL, &reg16);
 	reg16 |= PCI_EXP_LNKCTL_RL;
 	pcie_capability_write_word(parent, PCI_EXP_LNKCTL, reg16);
diff --git a/drivers/pci/quirks.c b/drivers/pci/quirks.c
index 653660e3ba9e..4999ad9d08b8 100644
--- a/drivers/pci/quirks.c
+++ b/drivers/pci/quirks.c
@@ -3553,23 +3553,46 @@ static void mellanox_check_broken_intx_masking(struct
pci_dev *pdev)
 DECLARE_PCI_FIXUP_FINAL(PCI_VENDOR_ID_MELLANOX, PCI_ANY_ID,
 			mellanox_check_broken_intx_masking);
```

```
-static void quirk_no_bus_reset(struct pci_dev *dev)
+static void quirk_no_bus_reset_and_no_retrain_link(struct pci_dev *dev)
 {
-   dev->dev_flags |= PCI_DEV_FLAGS_NO_BUS_RESET;
+   dev->dev_flags |= PCI_DEV_FLAGS_NO_BUS_RESET |
+             PCI_DEV_FLAGS_NO_RETRAIN_LINK_WHEN_NOT_GEN1;
 }

 /*
- * Some Atheros AR9xxx and QCA988x chips do not behave after a bus reset.
+ * Atheros AR9xxx and QCA9xxx chips do not behave after a bus reset and also
+ * after retrain link when PCIe bridge is not in GEN1 mode at 2.5 GT/s speed.
  * The device will throw a Link Down error on AER-capable systems and
  * regardless of AER, config space of the device is never accessible again
  * and typically causes the system to hang or reset when access is attempted.
+ * Or if config space is accessible again then it contains only dummy values
+ * like fixed PCI device ID 0xABCD or values not initialized at all.
+ * Retrain link can be called only when using GEN1 PCIe bridge or when
+ * PCIe bridge has forced link speed to 2.5 GT/s via PCI_EXP_LNKCTL2 register.
+ * To reset these cards it is required to do PCIe Warm Reset via PERST# pin.
  * https://lore.kernel.org/r/20140923210318.498dacbd@dualc.maya.org/
+ * https://lore.kernel.org/r/87h7l8axqp.fsf@toke.dk/
+ * https://www.mail-archive.com/ath9k-devel@lists.ath9k.org/msg07529.html
  */
-DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0030, quirk_no_bus_reset);
-DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0032, quirk_no_bus_reset);
-DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x003c, quirk_no_bus_reset);
-DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0033, quirk_no_bus_reset);
-DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0034, quirk_no_bus_reset);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x002e,
+            quirk_no_bus_reset_and_no_retrain_link);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0030,
+            quirk_no_bus_reset_and_no_retrain_link);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0032,
+            quirk_no_bus_reset_and_no_retrain_link);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0033,
+            quirk_no_bus_reset_and_no_retrain_link);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0034,
+            quirk_no_bus_reset_and_no_retrain_link);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x003c,
+            quirk_no_bus_reset_and_no_retrain_link);
+DECLARE_PCI_FIXUP_HEADER(PCI_VENDOR_ID_ATHEROS, 0x0042,
+            quirk_no_bus_reset_and_no_retrain_link);
+
+static void quirk_no_bus_reset(struct pci_dev *dev)
+{
+   dev->dev_flags |= PCI_DEV_FLAGS_NO_BUS_RESET;
+}

 /*
  * Root port on some Cavium CN8xxx chips do not successfully complete a bus
diff --git a/include/linux/pci.h b/include/linux/pci.h
index 86c799c97b77..fdbf7254e4ab 100644
--- a/include/linux/pci.h
+++ b/include/linux/pci.h
@@ -227,6 +227,8 @@ enum pci_dev_flags {
```

```
    PCI_DEV_FLAGS_NO_FLR_RESET = (__force pci_dev_flags_t) (1 << 10),
    /* Don't use Relaxed Ordering for TLPs directed at this device */
    PCI_DEV_FLAGS_NO_RELAXED_ORDERING = (__force pci_dev_flags_t) (1 << 11),
    /* Device does honor MSI masking despite saying otherwise */
    PCI_DEV_FLAGS_HAS_MSI_MASKING = (__force pci_dev_flags_t) (1 << 12),
+   /* Don't Retrain Link for device when bridge is not in GEN1 mode */
+   PCI_DEV_FLAGS_NO_RETRAIN_LINK_WHEN_NOT_GEN1 = (__force pci_dev_flags_t) (1 <<
13),
 };
```

- **switch的无效端口占用了部分资源**。以RK3588为例，其每个root port可分配的32-bit BAR空间仅为16MB。有的switch，如ASM2812，不论port的下游是否有设备，都会如下面log所示，申请2MB的32-bit BAR空间。根据lspci结果我们可以看到，bus 12的switch下的2/3/a/b这四个port下面并无设备，但却额外占用了总共8MB的32-bit BAR空间，导致正常设备的32-bit BAR资源不够分配。

```
[   22.005666] pci 0001:12:00.0: BAR 8: no space for [mem size 0x00600000]
[   22.040400] pci 0001:12:00.0: BAR 8: failed to assign [mem size 0x00600000]
[   22.040402] pci 0001:12:08.0: BAR 8: no space for [mem size 0x00600000]
[   22.040406] pci 0001:12:08.0: BAR 8: failed to assign [mem size 0x00600000]
[   22.067370] pci 0001:12:00.0: BAR 9: assigned [mem 0x940000000-0x9401fffff
64bit pref]
[   22.067373] pci 0001:12:02.0: BAR 8: no space for [mem size 0x00200000]
[   22.080018] pci 0001:12:02.0: BAR 8: failed to assign [mem size 0x00200000]
[   22.080022] pci 0001:12:02.0: BAR 9: assigned [mem 0x940200000-0x9403fffff
64bit pref]
[   22.093658] pci 0001:12:03.0: BAR 8: no space for [mem size 0x00200000]
[   22.093661] pci 0001:12:03.0: BAR 8: failed to assign [mem size 0x00200000]
[   22.093663] pci 0001:12:03.0: BAR 9: assigned [mem 0x940400000-0x9405fffff
64bit pref]
[   22.106310] pci 0001:12:08.0: BAR 9: assigned [mem 0x940600000-0x9407fffff
64bit pref]
[   22.106313] pci 0001:12:0a.0: BAR 8: no space for [mem size 0x00200000]
[   22.120756] pci 0001:12:0a.0: BAR 8: failed to assign [mem size 0x00200000]
[   22.130702] pci 0001:12:0a.0: BAR 9: assigned [mem 0x940800000-0x9409fffff
64bit pref]
[   22.130705] pci 0001:12:0b.0: BAR 8: no space for [mem size 0x00200000]
[   22.144573] pci 0001:12:0b.0: BAR 8: failed to assign [mem size 0x00200000]

lspci -vvt
-+-[0003:30]---00.0-[31]----00.0  Realtek Semiconductor Co., Ltd. Device
[10ec:8125]
 +-[0001:10]---00.0-[11-18]----00.0-[12-18]--+-00.0-[13]----00.0  Aquantia Corp.
Device
 |                                          +-02.0-[14]--
 |                                          +-03.0-[15]--
 |                                          +-08.0-[16]----00.0  Aquantia Corp.
Device
 |                                          +-0a.0-[17]--
 |                                          \-0b.0-[18]--
 \-[0000:00]---00.0-[01-ff]----00.0  ASMedia Technology Inc. Device [1b21:1164]

lspci -nn
0000:00:00.0 PCI bridge [0604]: Fuzhou Rockchip Electronics Co., Ltd Device
[1d87:3588] (rev 01)
```

```
0000:01:00.0 SATA controller [0106]: ASMedia Technology Inc. Device [1b21:1164]
(rev 02)
0001:10:00.0 PCI bridge [0604]: Fuzhou Rockchip Electronics Co., Ltd Device
[1d87:3588] (rev 01)
0001:11:00.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:12:00.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:12:02.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:12:03.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:12:08.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:12:0a.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:12:0b.0 PCI bridge [0604]: ASMedia Technology Inc. Device [1b21:2812] (rev
01)
0001:13:00.0 Ethernet controller [0200]: Aquantia Corp. Device [1d6a:14c0] (rev
03)
0001:16:00.0 Ethernet controller [0200]: Aquantia Corp. Device [1d6a:14c0] (rev
03)
0003:30:00.0 PCI bridge [0604]: Fuzhou Rockchip Electronics Co., Ltd Device
[1d87:3588] (rev 01)
0003:31:00.0 Ethernet controller [0200]: Realtek Semiconductor Co., Ltd. Device
[10ec:8125] (rev 05)
```

处理这类问题，建议同switch厂家协商，是否可以修改switch的固件配置，关闭不存在下游设备的端口的BAR空间。如果switch的固件无法关闭的情况下，且产品的拓扑结构相对固定（即不存在switch的各插槽随机插入设备的情况），可以尝试下列补丁进行过滤：

```
diff --git a/drivers/pci/probe.c b/drivers/pci/probe.c
index ece90a2..53bef6b 100644
--- a/drivers/pci/probe.c
+++ b/drivers/pci/probe.c
@@ -2794,6 +2794,37 @@ void __weak pcibios_fixup_bus(struct pci_bus *bus)
        /* nothing to do, expected to be removed in the future */
 }

+struct scan_blacklist_devfn {
+       u8 bus; /* bus number */
+       u8 dev; /* device number */
+       u8 func; /* function number */
+};
+
+#define ANY 0xff
+
+/* 请在sbl表中填写实际要过滤的设备bdf，以上述log为例，过滤的是bus12的2/3/a/b四个端口*/
+static struct scan_blacklist_devfn sbl[] = {
+       {12, 0x2, ANY},
+       {12, 0x3, ANY},
+       {12, 0xa, ANY},
+       {12, 0xb, ANY},
+};
+
```

```
+static bool pci_scan_check_blacklist(u8 bus, unsigned int devfn)
+{
+       int i;
+
+       for (i = 0; i < ARRAY_SIZE(sbl); i++) {
+               if (bus == sbl[i].bus && PCI_SLOT(devfn) == sbl[i].dev) {
+                       if (sbl[i].func == ANY)
+                               return true;
+                       else
+                               return sbl[i].func == PCI_FUNC(devfn);
+               }
+       }
+
+       return false;
+}
+
 /**
  * pci_scan_child_bus_extend() - Scan devices below a bus
  * @bus: Bus to scan for devices
@@ -2819,6 +2850,9 @@ static unsigned int pci_scan_child_bus_extend(struct
pci_bus *bus,

        /* Go find them, Rover! */
        for (devfn = 0; devfn < 256; devfn += 8) {
+               if (pci_scan_check_blacklist(bus->number, devfn))
+                       continue;
+
                nr_devs = pci_scan_slot(bus, devfn);
```

12.6 PCIe外设IO BAR资源分配异常

在枚举过程中出现以下IO BAR地址空间写入异常，并报有类似错误log，其特征关键字为`assigned [io`和`error updating`，或者`firmware Bug`，则建议核对设备手册中关于IO BAR地址空间的限制：

```
dmesg | grep "0002:22"
[    2.324600] pci_bus 0002:22: extended config space not accessible
[    2.324764] pci 0002:22:00.0: [13f6:0111] type 00 class 0x040100
[    2.324873] pci 0002:22:00.0: [Firmware Bug]: reg 0x10: invalid BAR (can't
size)
[    2.325445] pci 0002:22:00.0: supports D1 D2
[    2.331041] pci_bus 0002:22: busn_res: [bus 22-2f] end is updated to 22
```

或：

```
console:/ $ dmesg | grep "0002:22"
[    2.434085] [  T117] pci_bus 0002:22: extended config space not accessible
[    2.434629] [  T117] pci 0002:22:00.0: [13f6:0111] type 00 class 0x040100
[    2.434842] [  T117] pci 0002:22:00.0: reg 0x10: initial BAR value 0x0000d000
invalid
[    2.434869] [  T117] pci 0002:22:00.0: reg 0x10: [io  size 0x0100]
[    2.435536] [  T117] pci 0002:22:00.0: supports D1 D2
[    2.458229] [  T117] pci_bus 0002:22: busn_res: [bus 22-2f] end is updated to
22
[    2.458542] [  T117] pci 0002:22:00.0: BAR 0: assigned [io  0x1000-0x10ff]
[    2.458581] [  T117] pci 0002:22:00.0: BAR 0: error updating (0x80801001 !=
0x001001)
[    2.463002] [  T117] snd_cmipci 0002:22:00.0: enabling device (0080 -> 0081)
```

注释：

- 以上log为使用PCI声卡CMI8738时的异常打印。PCI原生设计是针对X86体系，所以IO BAR地址长度通常较小。CMI8738声卡IO BAR0地址限制在 0xFF00 以内，所以可参考以下补丁：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568.dtsi
index 9dc057637177..a060dcbf7df5 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568.dtsi
@@ -2311,7 +2311,7 @@
                phy-names = "pcie-phy";
                power-domains = <&power RK3568_PD_PIPE>;
                ranges = <0x00000800 0x0 0x80000000 0x3 0x80000000 0x0 0x800000
-                         0x81000000 0x0 0x80800000 0x3 0x80800000 0x0 0x100000
+                         0x81000000 0x0 0x00000000 0x3 0x80800000 0x0 0x100000
                          0x83000000 0x0 0x80900000 0x3 0x80900000 0x0
0x3f700000>;
                reg = <0x3 0xc0800000 0x0 0x400000>,
                      <0x0 0xfe280000 0x0 0x10000>;
```

12.7 MSI/MSI-X无法使用

在移植外设驱动的开发过程中(主要指的是WiFi)，认为主机端的function driver因无法使用MSI或者MSI-X中断而导致流程不正常，按如下流程进行排查

- 确认前述menuconfig 中提到的配置，尤其是MSI相关配置是否都有正确勾选
- 确认rk3568.dtsi中，its节点是否被设置为disabled
- 执行lspci -vvv，查看对应设备的MSI或者MSI-X是否有支持并被使能。以此设备为例，其上报的capabilities显示其支持32个64 bit MSI，目前仅使用1个，但是 Enable-表示未使能。若正确使能应该看到Enable+， 且Address应该能看到类似为0x00000000fd4400XX的地址。此情况一般是设备驱动还未加载或者加载时申请MSI或者MSI-X失败导致，请参考其他驱动，使用pci_alloc_irq_vectors等函数进行申请，详情可结合其他成熟的PCIe外设驱动做法以及参考内核中的Documentation/PCI/MSI-HOWTO.txt文档进行编写和排查异常。

```
Capabilities: [58] MSI: Enable- Count=1/32 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
```

- 如果MSI或者MSI-X有正确申请，可用如下命令导出中断计数，查看是否正常： cat /proc/interrupts 。在其中找到对应驱动申请的ITS-MSI中断(依据最后一列申请者驱动名称，例如此处为xhci_hcd驱动申请了这些MSI中断)。理论上每一笔的通信传输都会增加ITS，如果设备没有通信或者通信不正常，就会看到中断计数为0，或者有数值但发起通信后不再增加中断计数的情况。

```
229: 0 0 0 0 0 0 ITS-MSI 524288 Edge xhci_hcd
```

- 如果是概率性事件导致function driver无法收到MSI或者MSI-X中断，可以进行如下尝试。首先执行cat /proc/interrupts 查看相应中断号，以上述229为例，将中断迁移导其他CPU测试。例如切换导CPU2，则使用命令echo 2 > /proc/irq/229/smp_affinity_list 。
- 使用协议分析仪器抓取协议信号，查看流程中外设是否有概率性没有向主机发送MSI或者MSI-X中断，而导致的异常。需注意，目前协议分析仪一般都难以支持焊贴设备的信号采集，需向设备vendor购买金手指的板卡，在我司EVB上进行测试和信号采集。另需注意我司EVB仅支持标准接口的金手指板卡，若待测设备为M.2接口的设备(常见key A, key B, key M, key B+M, key E 四种类型)，请参考附录中"M.2 硬件接口"章节所介绍的关于不同key的外观尺寸，采购使用对应型号的转接板。

12.8 外设枚举后通信过程中报错

以下是NVMe在RK3566-EVB2上进行正常枚举之后，通信过程中突然设备异常报错的log。不论是什么设备，如果可以正常枚举并使能，则可以看到类似nvme 0000:01:00.0: enabling device (0000 -> 0002)的log。此后通信过程中设备报错，需要考虑如下三个方面：

- 利用示波器测量触发外设的电源，排除是否有跌落的情况发生
- 利用示波器测量触发外设的#PERST信号，排除是否被人误操作导致设备被复位的情况发生
- 利用示波器测量触发PCIe PHY的0v9和1v8两路电源，排除是否PHY的电源异常

特别提醒：RK EVB有较多的信号复用，利用拨码将PCIe的#PERST控制信号和其他外设的IO进行复用，请配合硬件重点确认。例如目前已知有部分RK3566-EVB2的拨码有异常，需要修正。

```
[    2.426038] pci 0000:00:00.0:   bridge window [mem 0x300900000-0x3009fffff]
[    2.426183] pcieport 0000:00:00.0: of_irq_parse_pci: failed with rc=-22
[    2.427493] pcieport 0000:00:00.0: Signaling PME with IRQ 106
[    2.427712] pcieport 0000:00:00.0: AER enabled with IRQ 115
[    2.427899] pcieport 0000:00:00.0: of_irq_parse_pci: failed with rc=-22
[    2.428202] nvme nvme0: pci function 0000:01:00.0
[    2.428259] nvme 0000:01:00.0: enabling device (0000 -> 0002)
[    2.535404] nvme nvme0: missing or invalid SUBNQN field.
[    2.535522] nvme nvme0: Shutdown timeout set to 8 seconds
...
[   48.129408] print_req_error: I/O error, dev nvme0n1, sector 0
[   48.137197] nvme 0000:01:00.0: enabling device (0000 -> 0002)
[   48.137299] nvme nvme0: Removing after probe failure status: -19
[   48.147182] Buffer I/O error on dev nvme0n1, logical block 0, async page read
[   48.162900] nvme nvme0: failed to set APST feature (-19)
```

12.9 外设枚举过程报FW异常

如设备在枚举过程分配BAR空间报如下两类错误，一般问题是设备的BAR空间与协议不兼容，需要特殊处理。需要修改drivers/pci/quirks.c中，增加对应quirk处理。具体信息应该咨询设备厂商。目前已知JMB585芯片给出的解决办法是需要重复读取BAR空间，才可以解决他们Fireware的异常，那么可以使用echo 1 > /sys/bus/pci/rescan重新对链路进行扫描，可以修复。

```
类型1:
[    2.379768] rk-pcie 3c0000000.pcie: PCIe Link up, LTSSM is 0x30011
[    2.380155] rk-pcie 3c0000000.pcie: PCI host bridge to bus 0000:00
[    2.380187] pci_bus 0000:00: root bus resource [bus 00-0f]
[    2.380204] pci_bus 0000:00: root bus resource [??? 0x300000000-0x3007fffff
flags 0x0] (bus address [0x00000000-0x007fffff])
[    2.380217] pci_bus 0000:00: root bus resource [io  0x0000-0xfffff] (bus
address [0x800000-0x8fffff])
[    2.380230] pci_bus 0000:00: root bus resource [mem 0x300900000-0x33fffffff]
(bus address [0x00900000-0x3fffffff])
[    2.394983] pci 0000:01:00.0: [Firmware Bug] reg 0x10: invalid BAR (can't
size)

类型2:
[    2.548219] pci 0000:01:00.0: [10ec:b723] type 00 class 0x028000
[    2.548389] pci 0000:01:00.0: reg 0x10: initial BAR value 0x00000000 invalid
[    2.548426] pci 0000:01:00.0: reg 0x10: [io  size 0x0100]
[    2.548549] pci 0000:01:00.0: reg 0x18: [mem 0x00000000-0x00003fff 64bit]
[    2.549132] pci 0000:01:00.0: supports D1 D2
[    2.549138] pci 0000:01:00.0: PME# supported from D0 D1 D2 D3hot D3cold
```

12.10 重新映射后访问PCIe设备的BAR地址空间异常

1. 首先，如果内核中利用ioremap将分配给PCIe外设的BAR地址进行映射后，使用memset或者memcpy来读写，会产生alignment fault错误。亦或者利用mmap将分配给PCIe外设的BAR地址映射到用户态进行访问，使用memset或者memcpy来读写，会产生sigbug错误。原因是memcpy有对齐优化问题或者memset在ARM64上会使用类似DC ZVA等指令，这些指令不支持Device memory type(nGnRE)。

```
[    69.195811] Unhandled fault: alignment fault (0x96000061) at
0xffffff8009800000
[    69.195829] Internal error: : 96000061 [#1] PREEMPT SMP
[    69.363352] Modules linked in:
[    69.363655] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.19.172 #691
[    69.364205] Hardware name: Rockchip rk3568 evb board (DT)
[    69.364688] task: ffffffc00a300000 task.stack: ffffffc00a2dc000
[    69.365227] PC is at __memset+0x16c/0x190
[    69.365593] LR is at snda_alloc_res+0xac/0xfc
[    69.366054] pc : [<ffffff800839a2ac>] lr : [<ffffff80085055b8>] pstate:
404000c5
[    69.366713] sp : ffffffc00a2df810
```

解决办法：改用memset_io或者memset_fromio/memset_toio等API。如果需要在用户态操作，请使用循环赋值方式来拷贝或者清理内存。另外，由于对齐访问的要求，如果您的程序触发了上述异常，强烈建议在内核配置中开启这个配置CONFIG_ROCKCHIP_ARM64_ALIGN_FAULT_FIX。

2. 其次，针对RK3588如果对BAR空间赋值操作触发以下类似异常，这是由于编译器将四个地址连续的访问优化成一个平台不支持的16字节的访问。

```
[14037.477507][    C3] SError Interrupt on CPU3, code 0xbe000011 -- SError
[14037.477512][    C3] CPU: 3 PID: 2037 Comm: memtest Not tainted 5.10.110 #1690
[14037.477514][    C3] Hardware name: Rockchip RK3588 EVB1 LP4 V10 Board (DT)
[14037.477516][    C3] pstate: 00001000 (nzcv daif -PAN -UAO -TCO BTYPE=--)
[14037.477518][    C3] pc : 0000000000405d58
[14037.477519][    C3] lr : 0000000000405fb4
[14037.477521][    C3] sp : 0000007fc48effd0
[14037.477522][    C3] x29: 0000007fc48effd0 x28: 0000000000000000
[14037.477529][    C3] x27: 0000000000452000 x26: 000000000048b000
[14037.477533][    C3] x25: 000000000048b000 x24: 0000000000000018
[14037.477537][    C3] x23: 0000000000489030 x22: 0000000000400280
[14037.477541][    C3] x21: 0000000000000000 x20: 0000000000400eb8
[14037.477545][    C3] x19: 0000000000400df0 x18: 0000000000000000
[14037.477549][    C3] x17: 0000000000417e00 x16: 0000000000489030
[14037.477554][    C3] x15: 000000000572c738 x14: 0000000000000000
[14037.477558][    C3] x13: 0000000000000150 x12: 000000000048b000
[14037.477562][    C3] x11: 0000000000000000 x10: 0000200000000000
[14037.477566][    C3] x9 : 00003fffffffffff x8 : 00000000000000de
[14037.477570][    C3] x7 : 0000000000000000 x6 : 000000000048af30
[14037.477574][    C3] x5 : 00000000f0000000 x4 : 0000000000000000
[14037.477578][    C3] x3 : 0000000000000001 x2 : 0000000000000001
[14037.477582][    C3] x1 : 0000000000489058 x0 : 0000000000000000
[14037.477587][    C3] Kernel panic - not syncing: Asynchronous SError Interrupt
[14037.477589][    C3] CPU: 3 PID: 2037 Comm: memtest Not tainted 5.10.110 #1690
[14037.477590][    C3] Hardware name: Rockchip RK3588 EVB1 LP4 V10 Board (DT)
[14037.477592][    C3] Call trace:
[14037.477593][    C3]  dump_backtrace+0x0/0x1a8
[14037.477594][    C3]  show_stack+0x18/0x28
[14037.477596][    C3]  dump_stack_lvl+0xcc/0xf4
[14037.477598][    C3]  dump_stack+0x18/0x58
[14037.477600][    C3]  panic+0x170/0x36c
[14037.477602][    C3]  nmi_panic+0x8c/0x90
[14037.477603][    C3]  arm64_serror_panic+0x78/0x84
[14037.477605][    C3]  arm64_is_fatal_ras_serror+0x34/0xc8
[14037.477607][    C3]  do_serror+0x6c/0x98
[14037.477608][    C3]  el0_error_naked+0x14/0x1c
[14037.477622][    C2] CPU2: stopping
[14037.477684][    C0] CPU0: stopping
[14037.477711][    C1] CPU1: stopping
[14037.477718][    C1] CPU: 1 PID: 0 Comm: swapper/1 Not tainted 5.10.110 #1690
[14037.477720][    C1] Hardware name: Rockchip RK3588 EVB1 LP4 V10 Board (DT)
[14037.477722][    C1] Call trace:
[14037.477729][    C1]  dump_backtrace+0x0/0x1a8
[14037.477734][    C1]  show_stack+0x18/0x28
[14037.477739][    C7] CPU7: stopping
[14037.477744][    C1]  dump_stack_lvl+0xcc/0xf4
[14037.477749][    C6] CPU6: stopping
[14037.477753][    C1]  dump_stack+0x18/0x58
```

```
[14037.477760][    C1]  local_cpu_stop+0x60/0x70
[14037.477764][    C1]  ipi_handler+0x1a4/0x310
[14037.477769][    C4] CPU4: stopping
[14037.477778][    C1]  handle_percpu_devid_fasteoi_ipi+0x7c/0x1c8
```

假设addr是您程序重映射的PCIe设备BAR地址，您的程序对这个地址进行四个连续地址的赋值操作，请在代码点1、2、3处任意一点加入一个内存屏障 barrier();

```
/* 若在用户态使用，需要添加如下barrier的实现 */
# define barrier() __asm__ __volatile__("": : :"memory")

*((u32 *)((void *)addr + 0x0)) = 0x11111111;
/* 代码点 1 */
*((u32 *)((void *)addr + 0x4)) = 0x44444444;
/* 代码点 2 */
*((u32 *)((void *)addr + 0x8)) = 0x88888888;
/* 代码点 3 */
*((u32 *)((void *)addr + 0xc)) = 0xcccccccc;
```

12.11 PCIe转USB设备驱动(xhci)加载异常

部分市售PCIe转USB芯片，如VL805，在链路建立之后，设备驱动加载异常。主要异常点就是等待xHCI芯片复位没有完成，大概率是转接芯片的固件需要升级。可先对接PC平台测试，若确定需要升级固件可联系供应商。

```
[ 6.289987] pci 0000:01:00.0: xHCI HW not ready after 5 sec (HC bug?) status =
0x811
[ 6.531098] xhci_hcd 0000:01:00.0: xHCI Host Controller
[ 6.531803] xhci_hcd 0000:01:00.0: new USB bus registered, assigned bus number 3
[ 16.532539] xhci_hcd 0000:01:00.0: can't setup: -110
[ 16.533033] xhci_hcd 0000:01:00.0: USB bus 3 deregistered
[ 16.533712] xhci_hcd 0000:01:00.0: init 0000:01:00.0 fail, -110
[ 16.534281] xhci_hcd: probe of 0000:01:00.0 failed with error -110
```

若仍无法解决，可以尝试下列补丁drivers/usb/host/pci-quirks.c

```
diff --git a/drivers/usb/host/pci-quirks.c b/drivers/usb/host/pci-quirks.c
index 3ea435c..cca536d 100644
--- a/drivers/usb/host/pci-quirks.c
+++ b/drivers/usb/host/pci-quirks.c
@@ -1085,8 +1085,11 @@ static void quirk_usb_early_handoff(struct pci_dev *pdev)
     /* Skip Netlogic mips SoC's internal PCI USB controller.
      * This device does not need/support EHCI/OHCI handoff
      */
-    if (pdev->vendor == 0x184e) /* vendor Netlogic */
+    if ((pdev->vendor == 0x184e) ||
+        (pdev->vendor == PCI_VENDOR_ID_VIA && pdev->device == 0x3483)) {
+        /* 以VL805为例，其他芯片请填写正确的厂商ID和设备ID */
+        dev_warn(&pdev->dev, "bypass xhci quirk for VL805\n");
         return;
+    }
     if (pdev->class != PCI_CLASS_SERIAL_USB_UHCI &&
```

```
            pdev->class != PCI_CLASS_SERIAL_USB_OHCI &&
            pdev->class != PCI_CLASS_SERIAL_USB_EHCI &&
```

12.12 PCIe 3.0接口休眠唤醒时系统异常

休眠唤醒测试如见下列log，原因是休眠时候关闭3.3v电源时导致了时钟晶振的电源异常。请从三个方面着手：

- dts中vpcie3v3-supply的电源配置，是否电源的max和min等配置不合理，导致电源操作异常
- 测量时钟晶振，是否在休眠前提前关闭了，或者休眠失败后就没有再次开启，亦或者掉电不充分导致时钟晶振芯片异常
- 将3.3v电源和晶振的供电飞线改为外部供电，排除异常

```
[   17.406781] PM: suspend entry (deep)
[   17.406839] PM: Syncing filesystems ... done.
[   17.471710] Freezing user space processes ... (elapsed 0.002 seconds) done.
[   17.474337] OOM killer disabled.
[   17.474343] Freezing remaining freezable tasks ... (elapsed 0.001 seconds)
done.
[   17.476200] Suspending console(s) (use no_console_suspend to debug)
[   17.479152] android_work: sent uevent USB_STATE=DISCONNECTED
[   17.480290] [WLAN_RFKILL]: Enter rfkill_wlan_suspend
[   17.501382] rk-pcie 3c0000000.pcie: fail to set vpcie3v3 regulator
[   17.501406] dpm_run_callback(): genpd_suspend_noirq+0x0/0x18 returns -22
[   17.501418] PM: Device 3c0000000.pcie failed to suspend noirq: error -22
[   38.506580] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
[   38.506601] rcu:  1-...0: (1 GPs behind) idle=25a/1/0x4000000000000000
softirq=4657/4657 fqs=2100
[   38.506604] rcu:  (detected by 0, t=6302 jiffies, g=4609, q=17)
[   38.506613] Task dump for CPU 1:
[   38.506617] kworker/u8:4    R  running task        0  1380      2 0x0000002a
[   38.506642] Workqueue: events_unbound async_run_entry_fn
[   38.506647] Call trace:
[   38.506657]  __switch_to+0xe4/0x138
[   38.506667]  pci_pm_resume_noirq+0x0/0x120
[  101.523233] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
[  101.523250] rcu:  1-...0: (1 GPs behind) idle=25a/1/0x4000000000000000
softirq=4657/4657 fqs=8402
[  101.523253] rcu:  (detected by 0, t=25207 jiffies, g=4609, q=17)
[  101.523260] Task dump for CPU 1:
[  101.523264] kworker/u8:4    R  running task        0  1380      2 0x0000002a
[  101.523284] Workqueue: events_unbound async_run_entry_fn
[  101.523288] Call trace:
[  101.523297]  __switch_to+0xe4/0x138
[  101.523307]  pci_pm_resume_noirq+0x0/0x120
```

12.13 PCIe设备切换PM模式时模块异常

目前有发现个别Wi-Fi模块休眠唤醒的时候，协议栈无法将其设置从D3 hot切换为D0状态，导致休眠唤醒流程失败。由于考虑到PCIe 控制器驱动会强制链路进入L2并复位，唤醒后链路和设备能回到L0和D0状态，所以协议栈是否能成功将模块设置进入D0状态并不影响唤醒后的使用。因此可以使用所提供的补丁依次尝试进行修复。

```
[ 848.869840] PM: Some devices failed to suspend, or early wake event detected
[ 848.871600] rtl88x2ce 0000:01:00.0: Refused to change power state, currently in
D3
[ 848.946128] rtl88x2ce 0000:01:00.0: Refused to change power state, currently in
D3
[ 848.962770] rtl88x2ce 0000:01:00.0: Refused to change power state, currently in
D3
```

```
--- a/drivers/pci/quirks.c
+++ b/drivers/pci/quirks.c
@@ -1859,6 +1865,15 @@ static void quirk_d3hot_delay(struct pci_dev *dev,
unsigned int delay)
                 dev->d3_delay);
 }

+ /* Some devices report short delay it actually need, fix them! */
+static void quirk_wifi_pm(struct pci_dev *dev)
+{
+       /* Ask your vendor how long it takes for D3 to D0 !!! */
+       if (dev->vendor == PCI_VENDOR_ID_FOO && dev->device == PCI_DEVICE_ID_BAR)
+               quirk_d3hot_delay(dev, 200); /* e.g 200ms, can be more for test
*/
+}
+
+DECLARE_PCI_FIXUP_FINAL(PCI_VENDOR_ID_FOO, PCI_DEVICE_ID_BAR, quirk_wifi_pm);
+/* Please Add your own PID and VID */
```

```
--- a/drivers/pci/quirks.c
+++ b/drivers/pci/quirks.c
@@ -1334,6 +1334,12 @@ DECLARE_PCI_FIXUP_CLASS_EARLY(PCI_VENDOR_ID_AL,
PCI_ANY_ID,
 DECLARE_PCI_FIXUP_CLASS_EARLY(PCI_VENDOR_ID_VIA, PCI_ANY_ID,
                               PCI_CLASS_STORAGE_IDE, 8, quirk_no_ata_d3);

+/* Some Wireless devices cannot transit from D3 to D0 by writing PCI_PM_CTRL */
+DECLARE_PCI_FIXUP_CLASS_EARLY(PCI_VENDOR_ID_FOO, PCI_DEVICE_ID_BAR,
+                             PCI_CLASS_NOT_DEFINED, 8, quirk_no_ata_d3);
+/* Please Add your own PID and VID */
```

```
--- a/drivers/pci/pci.c
+++ b/drivers/pci/pci.c
@@ -818,7 +818,7 @@ static int pci_raw_set_power_state(struct pci_dev *dev,
pci_power_t state)
        if (!dev->pm_cap)
            return -EIO;
+       /* Block all requests to D3hot */
+       if (state == PCI_D3hot)
+           return 0;
```

12.14 PCIe 外设休眠唤醒时模块无法连接或者工作异常

原则上PCIe外设在休眠时候需要进行断电。如果由于硬件上的限制导致掉电过程过于缓慢，那么如果快速唤醒就很容易导致外设掉电不充分又被上电，引起模块工作异常。此问题常见但不限于NVMe模块，由于NVMe为了防止异常掉电而在电源增加了一些大的耦合电容，而显得问题相对突出。可以打上如下风险测试补丁，若存在上述问题，一般可以快速复现。

```
--- a/kernel/cpu.c
+++ b/kernel/cpu.c
@@ -1724,11 +1724,11 @@ int freeze_secondary_cpus(int primary)
                if (cpu == primary)
                        continue;

-               if (pm_wakeup_pending()) {
+               //if (pm_wakeup_pending()) {
                        pr_info("Wakeup pending. Abort CPU freeze\n");
                        error = -EBUSY;
                        break;
-               }
+               //}

                trace_suspend_resume(TPS("CPU_OFF"), cpu, true);
                error = _cpu_down(cpu, 1, CPUHP_OFFLINE);
```

此类问题的处理需要从两方面着手解决：

- 硬件上修改PCIe模块的放电电路，保证合理的放电时长，给出优化后的最长放电时长t。
- 软件上增加外设电源上下电的延时，确保唤醒系统时再次上电之前经过t的时间，范例如下：

```
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
@@ -63,6 +63,7 @@
                enable-active-high;
                gpio = <&gpio0 RK_PD4 GPIO_ACTIVE_HIGH>;
                startup-delay-us = <5000>;
+               off-on-delay-us = <500000>; //硬件给出的最大时长t
                vin-supply = <&dc_12v>;
        };
```

12.15 设备分配到legacy中断号为0

```
0002:21:00.0 Serial controller: Device 1c00:3853 (rev 10) (prog-if 05 [16850])
        Subsystem: Device 1c00:3853
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr-
Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
        Interrupt: pin A routed to IRQ 0
        Region 0: I/O ports at 1000 [disabled] [size=256]
        Region 1: Memory at 380900000 (32-bit, prefetchable) [disabled]
[size=32K]
        Region 2: I/O ports at 100100 [disabled] [size=4]
        [virtual] Expansion ROM at 380908000 [disabled] [size=32K]
        Capabilities: [60] Power Management version 3
```

可以看到pin A分配到的legacy中断号为0，实际是默认未分配状态。原理上，不论内核cmdline中是否禁用msi中断，PCIe协议栈都会在pci bus driver加载的时候调用pci_assign_irq函数读取外设配置空间的0x3d寄存器，确定pin的数值。针对所读取的pin数值进行映射并分配映射后的虚拟中断号，写入外设配置空间的0x3c寄存器。所以一般不会出现查询不到分配legacy中断号的情况。此情况下，外设将无法正常使用I发出legacy类型中断，影响设备驱动的正常执行。目前有发现部分客户从非Linux平台移植PCIe设备驱动到Linux平台上，没有适配好pci bus driver模型，导致pci_assign_irq函数未被调用，导致legacy中断未分配。

12.16 无法读取分配给外设的IO地址空间

```
0002:21:00.0 Serial controller: Device 1c00:3853 (rev 10) (prog-if 05 [16850])
        Subsystem: Device 1c00:3853
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr-
Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
        Interrupt: pin A routed to IRQ 0
        Region 0: I/O ports at 1000 [disabled] [size=256]
        Region 1: Memory at 380900000 (32-bit, prefetchable) [disabled]
[size=32K]
        Region 2: I/O ports at 100100 [disabled] [size=4]
        [virtual] Expansion ROM at 380908000 [disabled] [size=32K]
        Capabilities: [60] Power Management version 3

0002:21:00.0 Serial Controler; pevice 1c00:3853 (rev 10)
00: 00 1c 53 38 00 00 10 00 10 05 00 07 00 00 00 00
10: 01 00 80 80 08 00 90 80 01 01 80 80 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 1c 53 38
30: 00 00 00 00 60 00 00 00 00 00 00 00 00 00 01 00 00

console:/ # cat /proc/ioports
00000000-000fffff : I/O
  00001000-00001fff : PCI Bus 0002:21
    00001000-00001007 : 0002:21:00.0
    00001008-0000100f : 0002:21:00.2
```

```
    00001010-00001017 : 0002:21:00.2
```

可以看到，lspci显示出此外设的IO端口分配地址是0x1000，而对应的pcie bus为0x80800000。这与rk3568.dtsi中定义的不符。以pcie3x2为例，其定义的IO端口的物理起始地址是0x380800000，所对应的pci bus起始地址为0x80800000。因此不能直接用IO命令或者devmem命令访问0x1000这个物理地址，或者软件访问其ioremap后的地址，否则会发生异常。

出现这种情况是由于历史原因，x86的PCIe中IO端口地址与内存端口地址是分离的，16M以下的地址段为IO端口地址。而ARM平台使用的PCIe IO地址端口是由内存端口模拟而来的。因此PCIe协议栈在计算IO端口地址的时候，为了将其地址也限制在16M以下，形式上看起来与x86平台更一致化，故而做了一层转换。转换的原理是将4K对齐处的0x1000作为IO端口的起始地址。换句话说，lspci中看到的IO端口地址0x1000即对应0x380800000这个物理地址。所以如果用IO命令或者devmem命令，可直接访问0x380800000。以此类推，如果分配到的IO端口地址为0x1010, 则根据该原理可得出所需访问的真实物理地址为0x380800010。如果是自行编写的驱动中，想要获取真实的IO端口地址进行访问，可以参考8250_port.c与8250_pci.c 串口驱动，使用pci_ioremap_bar和request_region函数组合，获取IO端口的真实物理地址以及iomap后的CPU地址。

12.17 PCIe设备性能抖动

PCIe控制器外接实时性需求比较高的外设时，如寒武纪MLU220 AI加速卡，若产品有显示输出需求或高带宽网络访问的情况下，可能使得AI加速卡性能抖动。此时，可以尝试提高对应PCIe接口的内存访问优先级。以RK3566/RK3568芯片为例：

```
根据实际产品测试确认所需提高性能的PCIe接口以及是否需要降低对应GMAC和VOP IP的内存访问优先级。其中，末尾数字越大表明优先级越高，即0x80000303 > 0x80000202 > 0x80000101

提高pcie2x1接口为第一优先级: io -4 0xfe190008 0x80000303
提高pcie3x1接口为第一优先级: io -4 0xfe190088 0x80000303
提高pcie3x2接口为第一优先级: io -4 0xfe190108 0x80000303
降低GMAC1接口为第二优先级:   io -4 0xfe130008 0x80000202
降低GMAC0接口为第二优先级:   io -4 0xfe188008 0x80000202
降低VOP_M0接口为第二优先级:  io -4 0xfe1a8088 0x80000202
降低VOP_M1接口为第二优先级:  io -4 0xfe1a8108 0x80000202

修改后的实际测试环节，需要观察受影响模块在AI加速卡运行过程中的具体指标，例如GMAC网络的性能抖动影响，或者观察是否出现下列VOP带宽不足的提示，综合考虑。
rockchip-vop2 fe040000.vop: [drm:vop2_isr] ERROR POST_BUF_EMPTY irq err at vp0
```

其他芯片的PCIe接口内存访问优先级调整寄存器请参阅附录中" 各SoC中PCIe控制器QoS调节寄存器"部分。

12.18 PCIe转SATA设备盘号不固定

因为驱动的初始化是采用线程化，所以多个PCIe控制器下外接的转接SATA芯片的初始化顺序不固定。实际上，每个转接SATA芯片下的多个硬盘口号是固定的，但是多个SATA芯片之间的硬盘口号是不固定的。例如某次启动，转接芯片1下的四个硬盘分别为sda~sdd, 转接芯片2下的四个硬盘分别为sde$_{sdh}$；下一次启动可能变成转接芯片1下的四个硬盘分别为sde^{sdh}, 转接芯片2下的四个硬盘分别为sda~sdd的情况。为了完全固定这些硬盘的顺序，可以采用应用层遍历硬盘路径，按照控制器地址(例如fe160000.pcie)来固定转接

芯片，进而创建软连接来固定顺序。如果想简化应用层，也可以采用关闭线程初始化，将CONFIG_PCIE_RK_THREADED_INIT去除即可。

```
ls -al /sys/block/sd*

rwxrwxrwx    1 root     root             0 Jan  1 00:00 /sys/block/sda ->
../devices/platform/fe160000.pcie/pci0001:10/0001:10:00.0/0001:11:00.0/ata2/host1
/target1:0:0/1:0:0:0/block/sda

lrwxrwxrwx    1 root     root             0 Jan  1 00:00 /sys/block/sdb ->
../devices/platform/fe160000.pcie/pci0001:10/0001:10:00.0/0001:11:00.0/ata3/host2
/target2:0:0/2:0:0:0/block/sdb

lrwxrwxrwx    1 root     root             0 Jan  1 00:03 /sys/block/sdc ->
../devices/platform/fe160000.pcie/pci0001:10/0001:10:00.0/0001:11:00.0/ata4/host3
/target3:0:0/3:0:0:0/block/sdc

lrwxrwxrwx    1 root     root             0 Jan  1 00:00 /sys/block/sdd ->
../devices/platform/fe160000.pcie/pci0001:10/0001:10:00.0/0001:11:00.0/ata5/host4
/target4:0:0/4:0:0:0/block/sdd
```

12.19 PCIe 设备可以Link却无法枚举出设备

若出现设备可以Link up但是却没有枚举出设备，例如lspci可见root port所在bus 0却看不到下一级bus 1等。

```
rk-pcie fe150000.pcie: PCIe Link up, LTSSM is 0x30011
rk-pcie fe150000.pcie: PCI host bridge to bus 0000:00
pci_bus 0000:00: root bus resource [bus 00-0f]
pci_bus 0000:00: root bus resource [??? 0xf0000000-0xf00fffff flags 0x0]
pci_bus 0000:00: root bus resource [io  0x300000-0x3fffff] (bus address
[0xf0100000-0xf01fffff])
pci_bus 0000:00: root bus resource [mem 0xf0200000-0xf0ffffff]
pci_bus 0000:00: root bus resource [mem 0x900000000-0x93fffffff pref]
pci 0000:00:00.0: [1d87:3588] type 01 class 0x060400
pci 0000:00:00.0: reg 0x10: [mem 0x00000000-0x3fffffff]
pci 0000:00:00.0: reg 0x14: [mem 0x00000000-0x3fffffff]
pci 0000:00:00.0: reg 0x38: [mem 0x00000000-0x0000ffff pref]
pci 0000:00:00.0: supports D1 D2
pci 0000:00:00.0: PME# supported from D0 D1 D3hot

lspci
0000:00:00.0 PCI bridge: Fuzhou Rockchip Electronics Co., Ltd Device 3588 (rev
01) (prog-if 00 [Normal decode])
```

此原因是读取的下一级设备，即bus 1设备的vendor ID 为非法数值。请在drivers/pci/probe.c文件的pci_bus_generic_read_dev_vendor_id函数中添加打印确认，协议栈默认了四种非法ID类型。

```
--- a/drivers/pci/probe.c
+++ b/drivers/pci/probe.c
@@ -2314,6 +2314,8 @@ bool pci_bus_generic_read_dev_vendor_id(struct pci_bus
*bus, int devfn, u32 *l,
        if (pci_bus_read_config_dword(bus, devfn, PCI_VENDOR_ID, l))
                return false;

+       dev_info(&bus->dev, "vendor id is 0x%x\n", *l);
+
        /* Some broken boards return 0 or ~0 if a slot is empty: */
        if (*l == 0xffffffff || *l == 0x00000000 ||
            *l == 0x0000ffff || *l == 0xffff0000)
```

目前已知ZX-200这款switch芯片在释放复位后需要加载bin并运行的时间较长，可能出现link up之后还未完全运行好其内部程序，导致RC去枚举时候读取到0的非法ID。如有类似情况请加上下列补丁尝试修复，如补丁加入仍无法解决问题，则建议联系设备厂家协助排查具体原因。

```
--- a/drivers/pci/controller/dwc/pcie-dw-rockchip.c
+++ b/drivers/pci/controller/dwc/pcie-dw-rockchip.c
@@ -814,7 +814,7 @@ static int rk_pcie_establish_link(struct dw_pcie *pci)
                         * that LTSSM max timeout is 24ms per period, we can wait
a bit
                         * more for Gen switch.
                         */
-                       msleep(50);
+                       msleep(1000);
                        /* In case link drop after linkup, double check it */
                        if (dw_pcie_link_up(pci)) {
                                dev_info(pci->dev, "PCIe Link up, LTSSM is
0x%x\n",
```

12.20 PCIe 设备可以枚举但访问异常

部分设备#PERST复位时间不够，可能概率性导致其虽然可以被系统枚举，但是实际工作不正常。例如有客户使用PCIe switch扩展后再连RTL8111网卡，出现如下的异常信息。此时需要增加#PERST的复位时间，在DTS中增加rockchip,perst-inactive-ms属性。详情请参照DTS property说明章节。

```
[    8.739794] enP4p67s0f0: 0xffffffc0127bd000, 86:41:ff:d2:48:a0, IRQ 133
[   10.178084] ------------[ cut here ]------------
[   10.178100] WARNING: CPU: 6 PID: 691 at
drivers/net/ethernet/realtek/r8168/r8168_n.c:7291
rtl8168_wait_phy_ups_resume+0x6c/0x88
[   10.178104] Modules linked in:
[   10.178112] CPU: 6 PID: 691 Comm: dhcpcd Not tainted 5.10.66 #1
[   10.178116] Hardware name: Rockchip RK3588 NVR DEMO LP4 V10 Board (DT)
[   10.178121] pstate: 60400089 (nZCv daIf +PAN -UAO -TCO BTYPE=--)
[   10.178127] pc : rtl8168_wait_phy_ups_resume+0x6c/0x88
[   10.178132] lr : rtl8168_wait_phy_ups_resume+0x54/0x88
[   10.178135] sp : ffffffc01358ba30
[   10.178139] x29: ffffffc01358ba30 x28: 0000000000001043
[   10.178145] x27: 0000000000000000 x26: 0000000000008914
[   10.178151] x25: 0000000000000000 x24: ffffff8102e10c38
```

```
[   10.178157] x23: 0000000000418958 x22: 0000000000000002
[   10.178163] x21: ffffff8102e109c0 x20: 0000000000000064
[   10.178168] x19: 0000000000000007 x18: 0000000000000000
[   10.178174] x17: 0000000000000000 x16: 0000000000000000
[   10.178179] x15: 000000000000000a x14: 00000000000b49d2
[   10.178186] x13: 0000000000000006 x12: ffffffffffffffff
[   10.178191] x11: 0000000000000010 x10: ffffffc09358b767
[   10.178197] x9 : ffffffc0104d3868 x8 : 0000000000000000
[   10.178202] x7 : 663a31343a363820 x6 : 00000000ffff0000
[   10.178208] x5 : 0000000000000004 x4 : 000000000000ffff
[   10.178213] x3 : 0000000000000006 x2 : ffffffc011dcbbb8
[   10.178219] x1 : ffffffc01358b9f0 x0 : 0000000000005dc3
[   10.178225] Call trace:
[   10.178231]  rtl8168_wait_phy_ups_resume+0x6c/0x88
[   10.178236]  rtl8168_exit_oob+0x2e8/0x36c
[   10.178241]  rtl8168_open+0x1c8/0x37c
[   10.178247]  __dev_open+0x154/0x17c
[   10.178252]  __dev_change_flags+0xfc/0x1ac
[   10.178257]  dev_change_flags+0x30/0x70
[   10.178263]  devinet_ioctl+0x288/0x51c
[   10.178268]  inet_ioctl+0x1b8/0x1e8

lspci -vvv的输出结果:
0003:31:00.0 Class 0108: Device 8086:f1a5 (rev ff) (prog-if ff)
        !!! Unknown header type 7f
```

12.21 PCIe 设备驱动使用Producer-Consumer模型导致系统卡死

此类卡死一般发生在PCIe设备老化过程中，系统卡死后能看到CPU都停在访问PCIe外设的某寄存器上。

```
ffffff80083c2b9c:       b4000224        cbz     x4, ffffff80083c2be0
<igb_rd32+0x58>
ffffff80083c2ba0:       8b214082        add     x2, x4, w1, uxtw
ffffff80083c2ba4:       b9400042        ldr     w2, [x2]   #死机位置，访问PCIE映射的
寄存器
或者
ffffffc010a7a990:       d50331bf        dmb     oshld
ffffffc010a7a994:       92403c08        and     x8, x0, #0xffff
ffffffc010a7a998:       ca080108        eor     x8, x8, x8
ffffffc010a7a99c:       b5000008        cbnz    x8, ffffffc010a7a99c
<os_pci_read16+0x34>
ffffffc010a7a9a0:       f9400bf3        ldr     x19, [sp,#16]
ffffffc010a7a9a4:       a8c27bfd        ldp     x29, x30, [sp],#32
ffffffc010a7a9a8:       f85f8e5e        ldr     x30, [x18,#-8]!#死机位置，访问PCIe映
射的寄存器
ffffffc010a7a9ac:       d50323bf        hint    #0x1d
ffffffc010a7a9b0:       d65f03c0        ret
又或者
ffffffc01068193c:       7100069f        cmp     w20, #0x1
ffffffc010681940:       54000281        b.ne    ffffffc010681990
<pci_generic_config_read+0x8c>
ffffffc010681944:       39400108        ldrb    w8, [x8]   #死机位置，访问PCIe映射的
寄存器
```

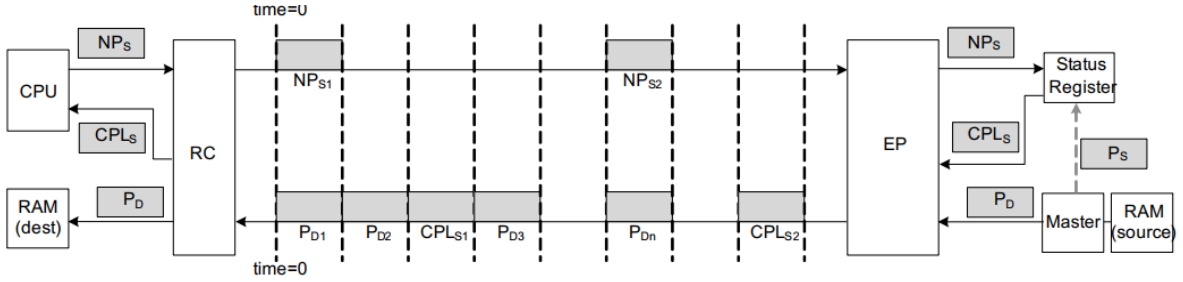
```
ffffffc010681948:       d50331bf        dmb     oshld
ffffffc01068194c:       92401d09        and     x9, x8, #0xff
ffffffc010681950:       ca090129        eor     x9, x9, x9
ffffffc010681954:       b5000009        cbnz    x9, ffffffc010681954
```

问题的产生一般是设备驱动使用了Producer-Consumer模型，且设备端存在数据发送断流的情况。下图是严格的PCIe协议上的数据包排序模型，其中有严格约束的是Row B/C/D与Col 2，尤其是D2a，即Completion包不能跨过Posted请求。

Row Pass Column?		Posted Request (Col 2)	Non-Posted Request		Completion (Col 5)
			Read Request (Col 3)	NPR with Data (Col 4)	
Posted Request (Row A)		a) No b) Y/N	Yes	Yes	a) Y/N b) Yes
Non-Posted Request	**Read Request** (Row B)	a) No b) Y/N	Y/N	Y/N	Y/N
	NPR with Data (Row C)	a) No b) Y/N	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

更通俗一点说，在Producer-Consumer模型中，作为消费者的一端，在生产者完全将数据写入某内存之前，不能使用该内存的数据；且生产者提供的状态查询寄存器也不允许在数据未完全写出去之前，提前响应消费者的状态读取请求。下图中EP设备是一个生产者，RC端是消费者：

1. EP设备从source RAM中获得数据，并采用post方式写入RC端的dest RAM，我们记为P_D。写完之后EP写一个完成标志位到它的状态寄存器(Status Register)。
2. RC端的CPU一直在轮询EP端的这个状态寄存器，来检查是否生产的数据已经写完。这个轮询的读过程我们记为NP_S。其所对应的从EP端获得的寄存器数据我们记为CPL_S。
3. 在这个流程中，CPL_S绝对不允许超过任何P_D之前发出，或者RC的CPU端在数据完全落入dest RAM之前对CPL_S进行响应。这个流程称为严格的Producer-Consumer模型。



前述问题的发生的根源，是由于设备驱动的不当业务模型所致：

- EP设备驱动和EP Firmware在使用Producer-Consumer模型时未遵守协议，当EP端向dest RAM发送P_D间隙，RC端的设备驱动恰好发出了NP_S读完成标志的请求(包括CfgRd或MemRd类型)，而EP端在P_D未完全发完之前提前允许返回了CPL_S。
- 此时，RC端因为严格遵守Producer-Consumer协议模型，等不到完整的P_D而无法接收CPL_S，且没有超时返回的途径，导致等待CPL_S的CPU卡住。

- 还需要特别注意，EP设备驱动如果不是发起读取完成标志的TLP请求，即不希望此包进入严格Producer-Consumer模型排序的包，都应该配置成Relaxed Ordering, 即TLP的header的Attr置上RO位。否则不仅可能会因为排序而导致业务性能下降，也会导致误进入排序规则而引起不必要的异常风险。

因此，为了要打破这种协议约束下的严格模型，我们RC端提供了可修改排序规则的寄存器，允许在Producer-Consumer模型下使得CPL_S 超越 P_D ，让CPU端提前获得Status Register的数据，用于确认问题和提供临时规避方案。但需要指出，使用我们下方提供的规避方案，在Producer-Consumer模型下虽然不会因为数据推送间隙中提前返回CPL_S而卡住RC端CPU，但可能会发生数据完整性异常。例如明显CPL_S 抢先返回了完成标志位，然而P_{Dn} 却还没有完全落入dest RAM的情况，那么RC端CPU去获取dest RAM内的数据就**可能**是不完整的，缺失P_{Dn}部分的数据。这个可能性大小受CPU运算速度，驱动软件获得完成标志后去取数据流的软件延时，芯片内Memory latency/QoS等多种因素影响。目前已知一些网络设备存在这类问题，规避方案配合低概率的数据错误率、网络协议栈数据包的校验和出错处理机制，并不会最终导致使用异常。但是其他设备并不能保证，这依赖于这些设备驱动完善的校验和恢复机制。因此我们对于使用这种协议模型又无法满足约束的EP设备的建议：

- 设备驱动可以抛弃传统的Producer-Consumer模型，即以读取寄存器来确认数据发送完成的方法，改用EP端触发MSI(-x)中断机制进行通知，这是目前绝大多数PCIe设备的处理共识。
- 若确实因为技术支持层面无法修改的，请与Vendor确认其非严格约束的Producer-Consumer模式采用我们的规避方案之后，需评估驱动对数据完整性出错的处理机制是否完善，是否会严重影响业务生态的使用。

```
1. 查询PCIe控制器，获得dbi寄存器基地址，我们以RK3588 pcie2x1l0为例
pcie2x1l0: pcie@fe170000 {
           ......
           reg = <0x0 0xfe170000 0x0 0x10000>,
                 <0xa 0x40800000 0x0 0x400000>; //dbi地址为前两个cell高低位数值，
0xa40800000
           reg-names = "pcie-apb", "pcie-dbi";
}

2. 临时调整Ordering rule寄存器，允许CPLs超过Pd，寄存器为dbi地址+0x8b4：
io -4 0xa408008b4 0xff00  # 0xa40800000 + 0x8b4

3. Ordering rule寄存器详解：
_____________________________________________________________________________
__
|[0: 7] | Determines if NP can pass halted P queue. (Completion Passing Posted规
则) |
|       | 0: NP can not pass P (recommended)
|
|       | 1: NP can pass P
|
|_______|_____________________________________________________________________
__ |
|[8:15] | Determines if CPL can pass halted P queue.(Non-Posted Passing Posted规
则)  |
|       | 0: CPL can not pass P (recommended)
  |
|       | 1: CPL can pass P
  |
|_______|_____________________________________________________________________
__ |
```

12.22 NVMe 设备长期工作状态下出现异常

```
[186825.261896] nvme nvme0: Abort status: 0x0
[187047.435152] nvme nvme0: Abort status: 0x0
[187052.740828] nvme nvme0: I/O 256 QID 1 timeout, aborting
[187082.743811] nvme nvme0: I/O 256 QID 1 timeout, reset controller
[187092.649150] nvme nvme0: Abort status: 0x0
[187389.590726] nvme nvme0: I/O 709 QID 2 timeout, aborting
[187396.088472] nvme nvme0: Abort status: 0x0
[187419.670448] nvme nvme0: I/O 736 QID 2 timeout, aborting
[187430.034886] nvme nvme0: Abort status: 0x0
[187449.750258] nvme nvme0: I/O 736 QID 2 timeout, reset controller
```

重载测试下NVMe设备出现异常，从上述log可以看到NVMe设备返回的设备状态为0，因为如果链路异常，则返回应该是0xf，所以此时可排除链路问题。这种情况下，请重点从设备工作温度考虑。

- 首先需内核开启NVMe温控配置CONFIG_NVME_HWMON
- 其次遍历hwmon的ID以寻找NVMe注册的温控节点，以8为例`cat /sys/class/hwmon/hwmon8/name`，得到nvme字样即为nvme设备的温控节点。
- 读取NVMe各设备的温传计数`cat /sys/class/hwmon/hwmon8/*input`，观察重载测试下其温度变化，是否超过该设备额定工作温度。
- 若确定工作温度超出限制，需改善扇热条件或者在用户态执行相应的硬盘温控策略。

```
[ 3836.614828] sysrq: Show Blocked State
[ 3836.614921] task:binder:306_2    state:D stack:    0 pid:  329 ppid:     1
flags:0x04000009
[ 3836.614925] Call trace:
[ 3836.614932] __switch_to+0x118/0x148
[ 3836.614936] __schedule+0x538/0x7a4
[ 3836.614938] schedule+0x9c/0xe0
[ 3836.614940] schedule_timeout+0x8c/0xf4
[ 3836.614942] io_schedule_timeout+0x48/0x6c
[ 3836.614944] wait_for_common_io+0x7c/0x100
[ 3836.614945] wait_for_completion_io_timeout+0x10/0x1c
[ 3836.614949] submit_bio_wait+0x70/0xb4
[ 3836.614952] blkdev_issue_discard+0x88/0xd8
[ 3836.614954] blk_ioctl_discard+0x100/0x108
[ 3836.614955] blkdev_common_ioctl+0x388/0x664
[ 3836.614957] blkdev_ioctl+0x16c/0x20c
[ 3836.614959] block_ioctl+0x34/0x44
[ 3836.614962] __arm64_sys_ioctl+0x90/0xc8
[ 3836.614964] el0_svc_common+0xac/0x1ac
[ 3836.614965] do_el0_svc+0x1c/0x28
[ 3836.614968] el0_svc+0x10/0x1c
[ 3836.614970] el0_sync_handler+0x68/0xac
[ 3836.614972] el0_sync+0x160/0x180
```

除`nvme nvme0: I/O 736 QID 2 timeout, aborting`之外，如果还触发如上打印。此时应分析是否在异常前进行了全盘格式化或者同步批量删除等操作，尤其是大容量NVMe在较满的状态下更易触发。解决办法如下：

```
(1) 将CONFIG_DEFAULT_HUNG_TASK_TIMEOUT配置一个稍大秒数，例如120，以规避block-mq的阻塞检测
(2) 调整NVMe协议栈关于admin queue和io queue的软件超时时间
--- a/drivers/nvme/host/core.c
+++ b/drivers/nvme/host/core.c
@@ -41,12 +41,12 @@ struct nvme_ns_info {
        bool is_removed;
 };

-unsigned int admin_timeout = 60;
+unsigned int admin_timeout = 120;
 module_param(admin_timeout, uint, 0644);
 MODULE_PARM_DESC(admin_timeout, "timeout in seconds for admin commands");
 EXPORT_SYMBOL_GPL(admin_timeout);

-unsigned int nvme_io_timeout = 30;
+unsigned int nvme_io_timeout = 120;
 module_param_named(io_timeout, nvme_io_timeout, uint, 0644);
```

13. 附录

13.1 LTSSM状态机

状态描述符	状态代码 (6'h)	备注
S_DETECT_QUIET	0x00	未检到外设RX端接电阻，设备未正常工作
S_DETECT_ACTIVE	0x01	-
S_POLL_ACTIVE	0x02	-
S_POLL_COMPLIANCE	0x03	兼容性测试模式。非测试模式下因信号异常原因错误进入
S_POLL_CONFIG	0x04	-
S_PRE_DETECT_QUIET	0x05	-
S_DETECT_WAIT	0x06	-
S_CFG_LINKWD_START	0x07	-
S_CFG_LINKWD_ACEPT	0x08	Lane 顺序检测过程
S_CFG_LANENUM_WAIT	0x09	-
S_CFG_LANENUM_ACEPT	0x0a	-
S_CFG_COMPLETE	0x0b	物理层检测完毕
S_CFG_IDLE	0x0c	-

状态描述符	状态代码 (6'h)	备注
S_RCVRY_LOCK	0x0d	速率切换过程
S_RCVRY_SPEED	0x0e	-
S_RCVRY_RCVRCFG	0x0f	-
S_RCVRY_IDLE	0x10	-
S_RCVRY_EQ0	0x20	-
S_RCVRY_EQ1	0x21	-
S_RCVRY_EQ2	0x22	-
S_RCVRY_EQ3	0x23	-
S_L0	0x11	链路正常工作过状态L0
S_L0S	0x12	-
S_L123_SEND_EIDLE	0x13	-
S_L1_IDLE	0x14	-
S_L2_IDLE	0x15	-
S_L2_WAKE	0x16	-
S_DISABLED_ENTRY	0x17	-
S_DISABLED_IDLE	0x18	-
S_DISABLED	0x19	-
S_LPBK_ENTRY	0x1a	-
S_LPBK_ACTIVE	0x1b	loopback测试模式，一般在测试环境下会进入
S_LPBK_EXIT	0x1c	-
S_LPBK_EXIT_TIMEOUT	0x1d	-
S_HOT_RESET_ENTRY	0x1e	外设主动发其hot reset流程
S_HOT_RESET	0x1f	-

13.2 Debugfs导出信息解析表

事件符号	含义
EBUF Overflow	-
EBUF Under-run	-
Decode Error	包解码错误，信号异常
Running Disparity Error	极性偏差错误，8bit/10bit编码中0与1数量比例失衡，Gen2信号异常
SKP OS Parity Error	SKP 序列奇偶校验错误，Gen3信号异常
SYNC Header Error	异步包头错误，信号问题
CTL SKP OS Parity Error	控制的SKP 序列奇偶校验错误，Gen3信号异常
Detect EI Infer	检测到信号串扰
Receiver Error	接收端错误
Rx Recovery Request	RX信号收到外设请求，进入rescoery状态进行信号矫正
N_FTS Timeout	外设的n_fts不达标，L0s回到L0异常进入recovery状态
Framing Error	帧格式解码错误，信号异常
Deskew Error	deskew错误，信号异常
BAD TLP	收到错误TLP包
LCRC Error	link CRC，信号异常
BAD DLLP	错误的DLLP包，信号异常
Replay Number Rollover	重传包积累太多，replay buffer冲了
Replay Timeout	误码包重传超时。
Rx Nak DLLP	收到来自下游的DLLP而拒绝其访问
Tx Nak DLLP	发往下游的DLLP被拒绝
Retry TLP	重传的TLP包数量，可以表征误码率
FC Timeout	更新流控超时
Poisoned TLP	上报一个错误的TLP类型，可能是因为链路原因或RAM的位跳变引起
ECRC Error	end CRC，信号异常
Unsupported Request	收到不支持的请求类型的TLP包，可能是访问外设被拒绝
Completer Abort	设备接收到了RC发来的请求，但是发生了其内部错误，要求终止该访问请求。

事件符号	含义
Completion Timeout	读取外设相关资源，TLP返回超时
Common event signal status	读取目前控制器的PM模式

13.3 错误注入配置对照表

组号 einj_number	组描述	类型号 error_type	类型含义
0	CRC错误	0x0	TLP LCRC错包
0	CRC错误	0x1	16b CRC of ACK/NAK DLLP 错包
0	CRC错误	0x2	16b CRC of Update-FC DLLP错包
0	CRC错误	0x3	TLP ECRC错包
0	CRC错误	0x4	TLP FCRC错包(128b/130bit编码)
0	CRC错误	0x5	TX极性错误TSOS(128b/130bit编码)
0	CRC错误	0x6	TX极性错误SKPOS(128b/130bit编码)
0	CRC错误	0x8	RX LCRC错包
0	CRC错误	0xb	RX ECRC错包
1	包编号顺序错误	0x0	TLP SEQ#错误
1	包编号顺序错误	0x1	DLLP SEQ#错误 ACK_NAK_DLLP
2	DLLP错误	0x0	阻止传输DLLP ACK/NAK 包[1]
2	DLLP错误	0x1	阻止传输Update FC DLLP包[1]
2	DLLP错误	0x2	总是发送NAK DLLP包[1]
3	符号错误	0x0	Invert sync header(128bit/130bit编码)
3	符号错误	0x1	TS1 order set错误
3	符号错误	0x2	TS2 order set错误
3	符号错误	0x3	FTS order set错误
3	符号错误	0x4	E-idle order set错误
3	符号错误	0x5	END/EDB symbol错误
3	符号错误	0x6	STP/SDP symbol错误
3	符号错误	0x7	SKP order set错误
4	流控FC错误	0x0	Posted TLP header credit
4	流控FC错误	0x1	Non-Posted TLP header credit
4	流控FC错误	0x2	Commpletion TLP header credit

组号 einj_number	组描述	类型号 error_type	类型含义
4	流控FC错误	0x4	Posted TLP Data credit
4	流控FC错误	0x5	Non-Posted TLP Data credit
4	流控FC错误	0x6	Commpletion TLP data credit
5	特殊TLP错误	0x0	将ACK DLLP当成NAK DLLP来产生duplicate TLP包
5	特殊TLP错误	0x1	制造无效TLP包，原始包放入retry流程
6	包头检查错误	0x0	产生TLP header错误
6	包头检查错误	0x1	产生TLP prefix 中第一组四个dword错误
6	包头检查错误	0x2	TLP prefix 中第二组dword错误

[1] 不依赖于计数到零而停止，需要命令手动停止

13.4 关于PCIe TX加重预设值对照表

Preset Number	Preshoot(dB)	De-emphasis(dB)
P0	0	-6.0 ± 1.5 dB
P1	0	-3.5 ± 1 dB
P2	0	-4.4 ± 1.5 dB
P3	0	-2.5 ± 1 dB
P4	0	0
P5	1.9 ± 1 dB	0
P6	2.5 ± 1 dB	0
P7	3.5 ± 1 dB	-6.0 ± 1.5 dB
P8	3.5 ± 1 dB	-3.5 ± 1 dB
P9	3.5 ± 1 dB	0
P10	0	x

13.5 开发资源获取地址

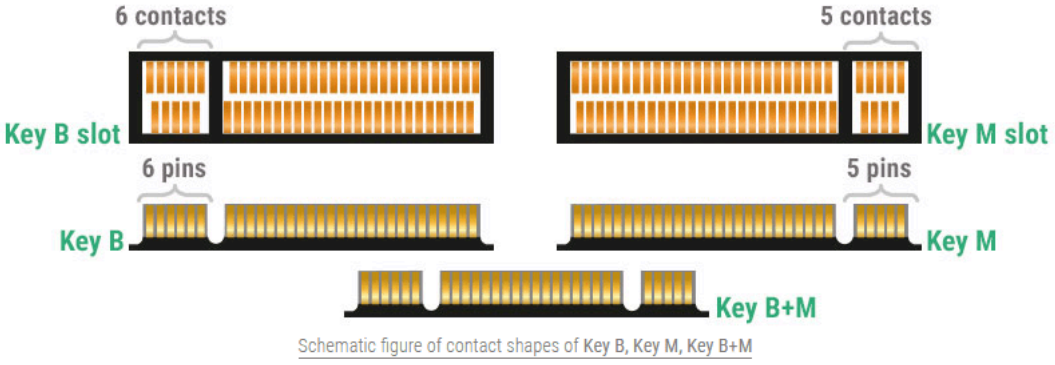
取得redmine权限后可以直接访问https://redmine.rock-chips.com/documents/107获取前文所述各类资料。

13.6 PCIe地址空间配置详述

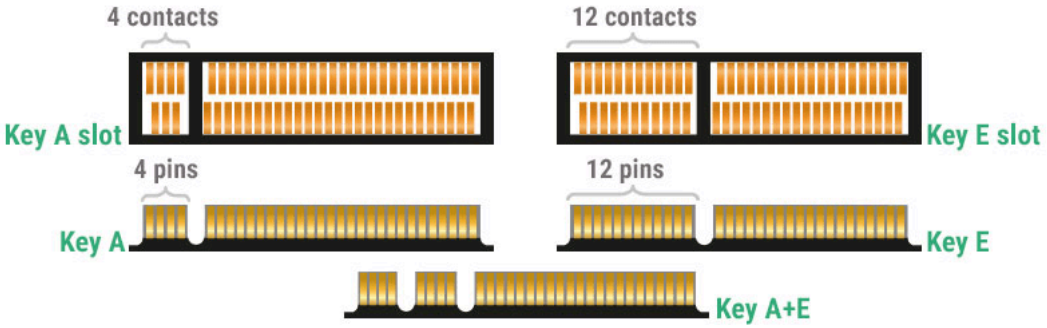
针对DTS中PCIe地址空间的详细配置，可参考如下链接进行解读或者修改：https://elinux.org/Device_Tree_Usage#PCI_Host_Bridge

13.7 M.2接口硬件

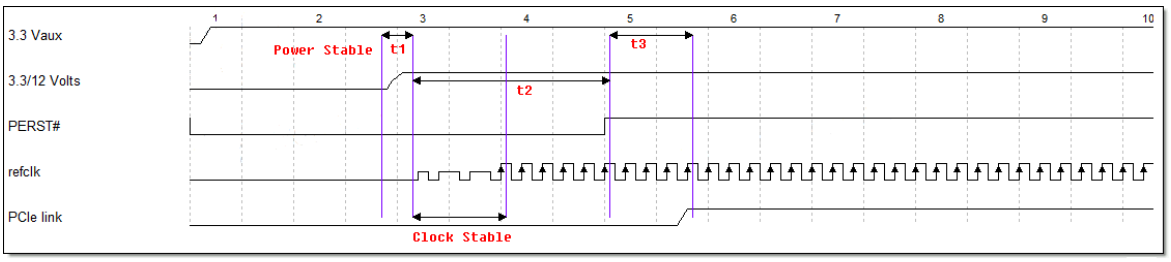
Key B+M的金手指可以使用key B或者key M的转接板slot，key A+E的金手指可以使用key A或者key E的转接板slot。其余类型的金手指板请严格按照接口选择对应型号的转接板slot。



Schematic figure of contact shapes of Key B, Key M, Key B+M



13.8 板级可配置时序



可调时间	图标	dts属性	说明（Linux 内核阶段）
Power Stable	t1	startup-delay-us	此属性是配置到PCIe节点的vpcie3v3-supply所引用的电源节点中的。
Tpvperl	t2	rockchip,perst-inactive-ms	此属性是配置到PCIe节点中的，若不配置默认为200ms，在开机时候生效。
Tpvperl	t2	rockchip,s2r-perst-inactive-ms	此属性是配置到PCIe节点中的，若不配置默认等同于rockchip,perst-inactive-ms的配置数值，在休眠唤醒时生效。
out of electrical idle	t3	rockchip,wait-for-link-ms	此属性是配置到PCIe节点中的，若不配置默认为1ms, 用于等待每次发起link之前的等待时间。

13.9 各SoC中PCIe控制器QoS调节寄存器

下表中的寄存器命令范例为配置成所允许的最高优先级,实际请根据测试情况酌情调整

芯片	控制器	寄存器命令
RK1808	pcie0	io -4 0xfe880008 0x303
RK3528	pcie2x1	io -4 0xff280188 0x404
RK3562	pcie2x1	io -4 0xfeea0008 0x404
RK3566 RK3568	pcie2x1	io -4 0xfe190008 0x80000303
	pcie3x1	io -4 0xfe190088 0x80000303
	pcie3x2	io -4 0xfe190108 0x80000303
RK3576	pcie0	io -4 0x27f0a008 0x303
	pcie1	io -4 0x27f0a088 0x303
RK3588	PCIe1L0/1/2	io -4 0xfdf3a008 0x404
	PCIe3x4 PCIe3x2	o -4 0xfdf3a208 0x404