Rockchip Linux PCIe 开发指南

文件标识：RK-KF-YF-141

发布版本：V4.3.0

日期：2022-03-30

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

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
RK356X	4.19，5.10
RK3588	5.10

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

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

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

目录

1. 芯片资源介绍

1.1 RK3566

资源	模式	支持芯片互联	支持lane拆分	支持DMA	支持MMU	备注
PCIe Gen2 x 1 lane	RC only	否	否	否	否	内部时钟

1.2 RK3568

资源	模式	支持芯片互联	支持lane拆分	支持DMA	支持MMU	备注
PCIe Gen2 x 1 lane	RC only	否	否	否	否	内部时钟
PCIe Gen3 x 2 lane	RC/EP	是	1 lane RC+ 1 lane RC	是	否	外置晶振

1.3 RK3588

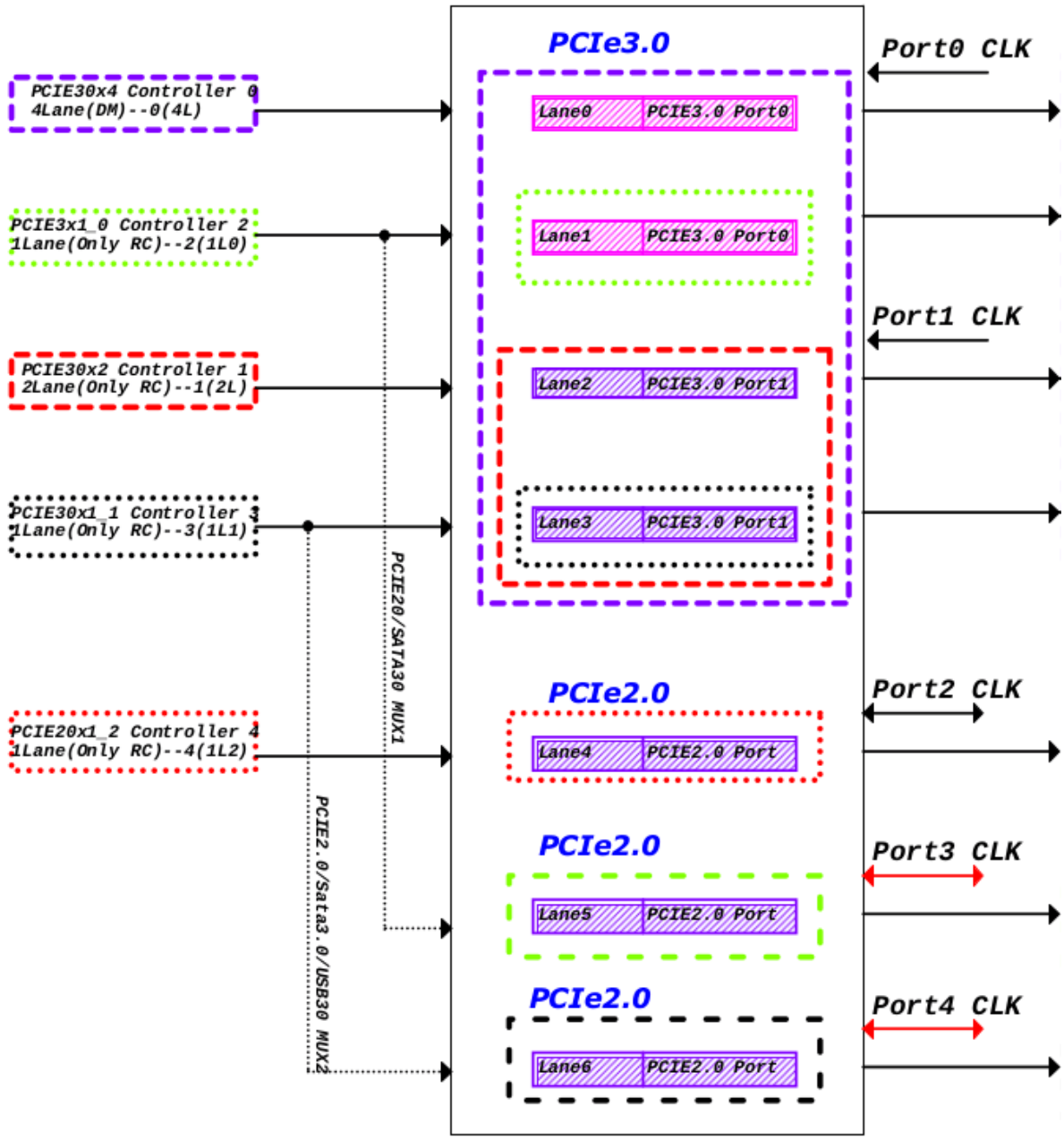
RK3588共有5个PCIe的控制器，硬件IP是一样的，配置不一样，其中一个4Lane DM模式可以支持作为EP使用，另外一个2Lane和3个1Lane控制器均只能作为RC使用。

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。



1.3.1 控制器

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

1.3.2 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

1.4 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。

1.4.1 控制器

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

1.4.2 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 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。

2.2.1 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

2.2.2 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

2.3 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。

2.3.1 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";
};
```

2.3.2 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";
};
```

2.3.3 示例3 pcie3.0phy拆分为4个1Lane, 1个使用PCIe 2.0 1 Lane(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";
};

&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";
};
```

2.4 DTS property说明

2.4.1 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RK3588，仅需将rk3568字段替换为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>;`

无需配置项：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。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。

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。

2.4.2 comboPHY dts配置

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。

3. `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。

4. `rockchip,compliance-mode`

特殊调试配置：此配置是针对compliance信号测试，使用PCIe控制器强制进入compliance测试模式。默认TX测试应该使用测试SMA夹具进入compliance, 而不需要强制进入。预留此配置是为了测试Gen3模式的loopback slave, 因为实验室测试可能Gen3的loopback测试需要进compliance模式。如果阅读者不理解什么是compliance测试，说明这不是你要找的配置，请勿针对此配置提问。

2.4.3 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 */
```

2.5 根据原理图填写DTS

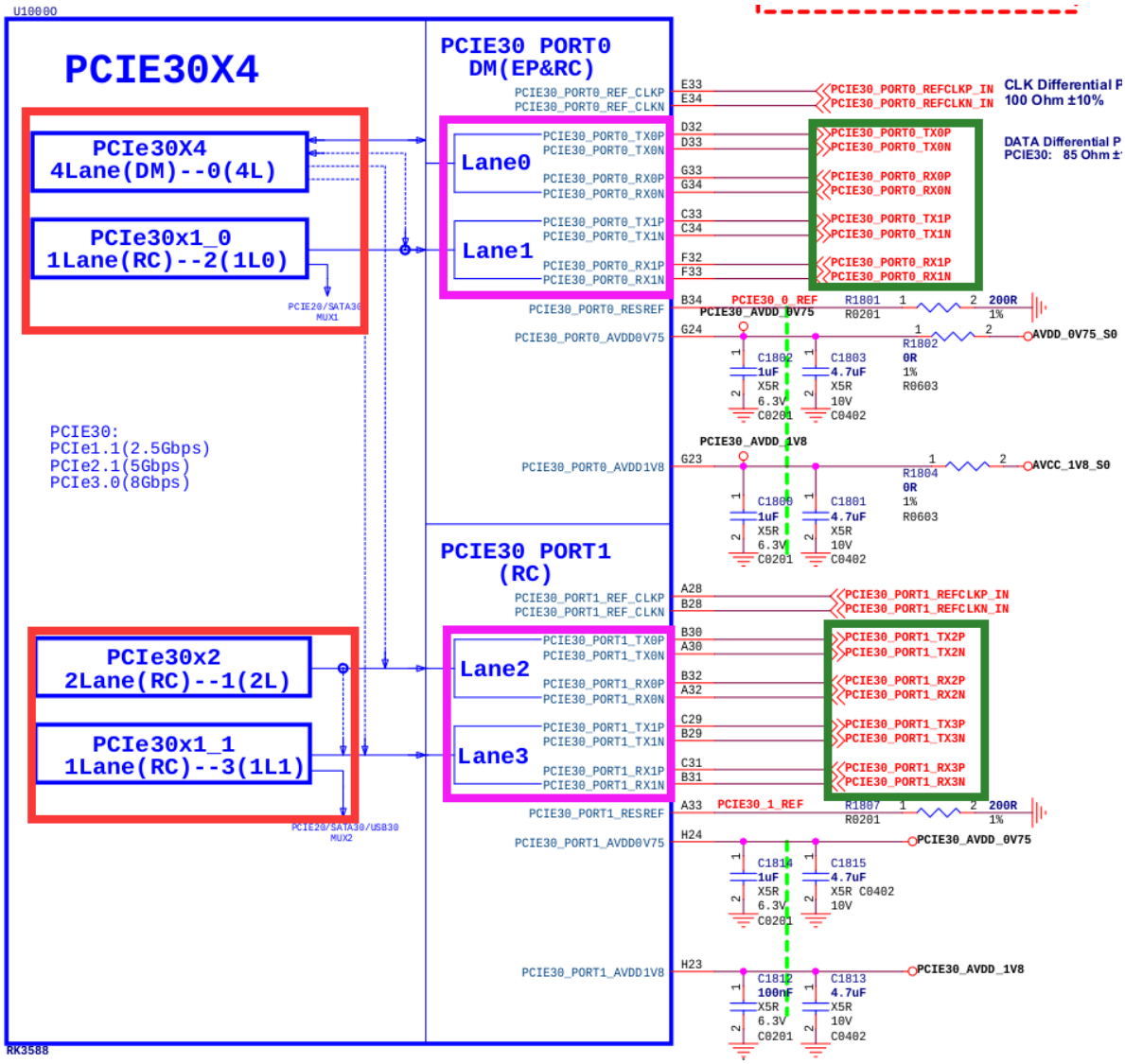
原理图是基于IO信号的视角来描述硬件，IO信号是跟PHY的index强相关的，前面提到RK3588的controller和PHY的index可能不一致，所以看原理图的时候需要特别注意这一点。这里给出一些填写建议，并通过示例说明如何将原理图中的PHY和控制器对应到dts的节点。

根据硬件原理图来填写dts的建议步骤：

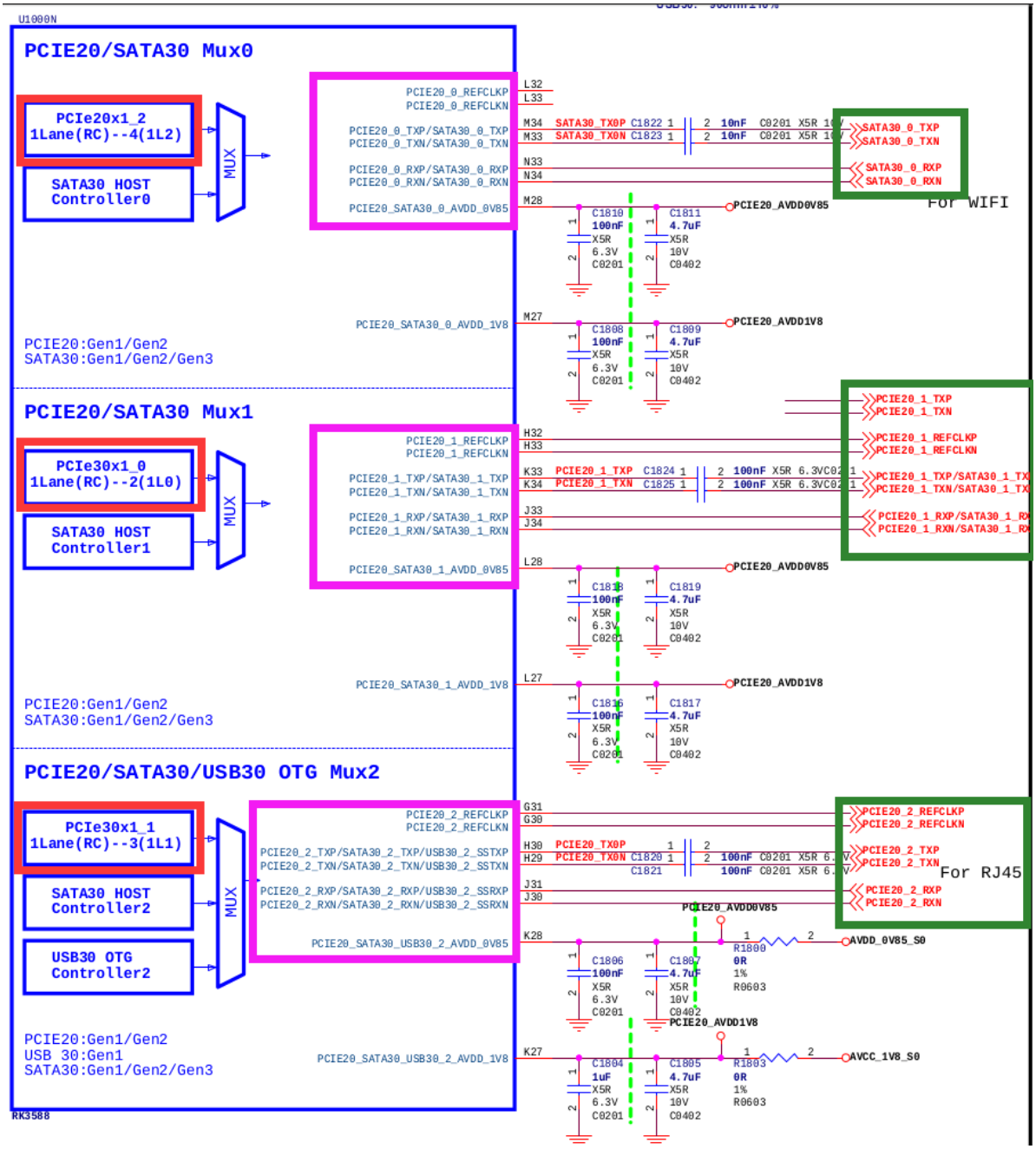
1. 跟硬件工程师确认使用了几个PCIe设备，芯片的多个PCIe接口是如何分配的；
2. 在原理图中分别查找某个设备使用的PCIe数据线对应到哪个PHY的输出；
3. 确定当前设备使用的分别是哪个控制器和PHY，在dts中使能；
4. 确定当前pcie接口使用的控制器dts "phy"属性及模式是否选择正确，如pcie2x1ln控制器需要选择comboPHY且指定为PHY_TYPE_PCIE;

5. 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；
6. 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；
7. 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)；
8. 配置其他外设工作所需的硬件；

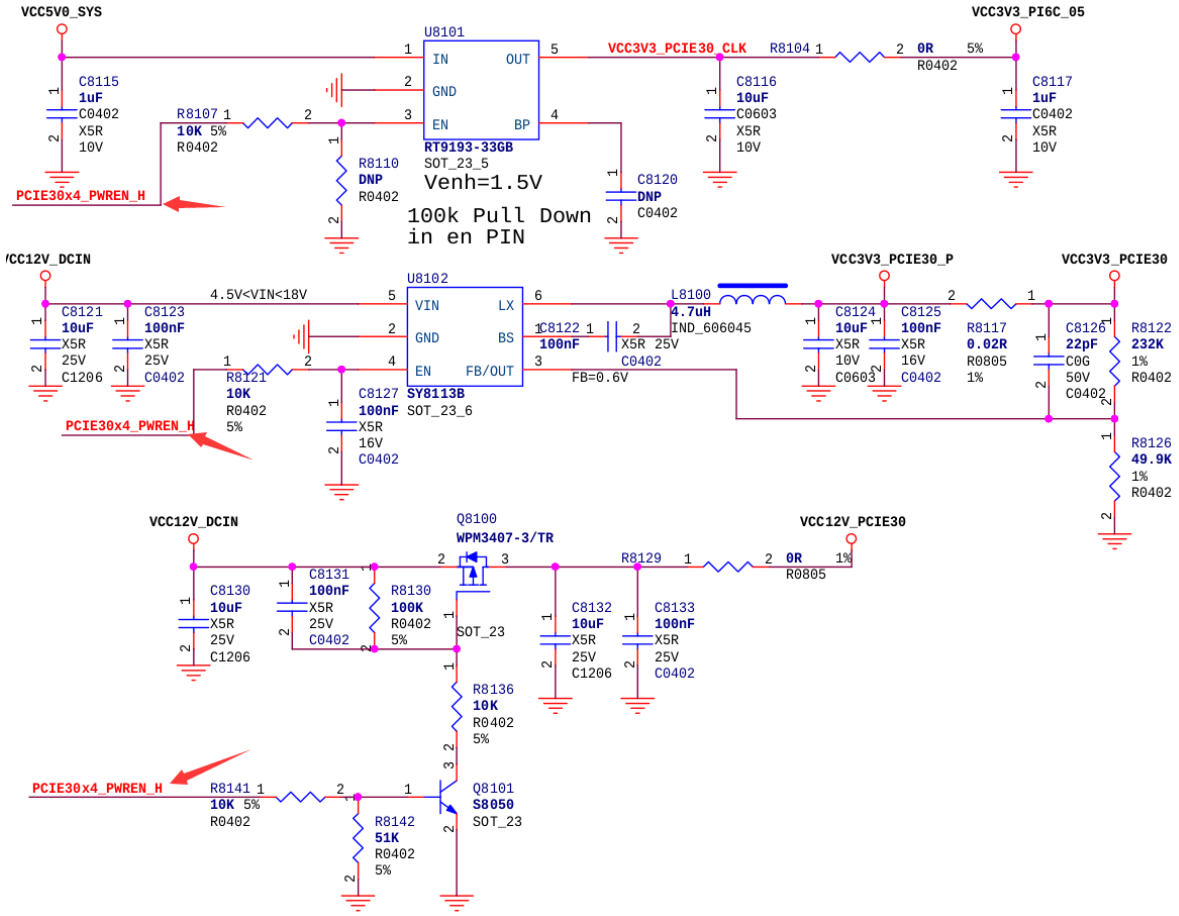
下图是RK3588 pcie30phy及其可能使用的controller，红色方框为controller，粉色方框为PHY信号，绿色方框为外设信号；实际使用哪个控制器可以通过外设信号连接来确认，也可以跟硬件工程师核对理解是否正确。下图是来自RK3588 evb1, 设备接的是一个pcie3.0 x4的slot，所以controller用的是PCIe30X4(dts命名pcie3x4)，其他几个controller都未跟这个PHY配合使用。



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。



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 4.19 开源内核仅支持 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. 常见应用问题

4.1 当走线位置不佳时，不同 lane 之间能否交织?

理论上可以交织，RC 的 lane[1-4]与 EP/switch 的 lane[1-4]随意对应，属于硬件协议行为，软件不需要改动。但我司EVB未验证，请谨慎使用，把控风险。

4.2 同一个 lane 的差分信号能否交织?

比如 RC 的 lane1 的 RX+ 与 EP/Switch 的 RX-对应，TX+与 EP/Switch 的 TX-对应。或者 RX 正负对应，TX 正负对应等等情况，怎么处理？理论上可以任意接，软件上不需要再额外处理。PCIe 的探测状态机已经考虑了这些所有情况。但我司EVB未验证，请谨慎使用，把控风险。

4.3 PCIe接口是否支持拆分或者合并?

RK3568的只有3.0的 RC有2个lane，能不能支持把这2个 lane 拆分成1+1模式，具体拆分方法查看DTS配置第七点。

RK3588的pcie3phy共2个Port4个Lane，支持拆分使用，具体方法参考示例和dts说明。

4.4 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。

4.5 在系统中如何确定控制器与设备的对应关系?

以RK356X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 << 16
            &its
            bus-range中的起始地址 << 16
            bus-range中分配的总线总数 << 16>
```

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。

4.6 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。

4.7 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，可分配的MSI或者MSI-X总数均是65535个。

4.8 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
```

4.9 芯片支持分配的BAR空间地址域有多大？

RK356X

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

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

控制器	高端BAR起始地址	长度	低端BAR起始地址	长度
pcie2x1l2	0xa00000000	1GB	0xf4000000	16MB

PCIe控制器支持最大1GB的64-bit的BAR空间，其中外设配置空间、IO空间和MEM空间共享这1GB的地址段，且MEM空间不支持预取功能。

4.10 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，RK356X芯片每个控制器的低位BAR地址仅有32MB，当RK356X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。

4.11 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。

4.12 如何对下游单个设备进行重扫描或者在线更换设备?

因为RK356X并不支持物理热拔插，所以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
```

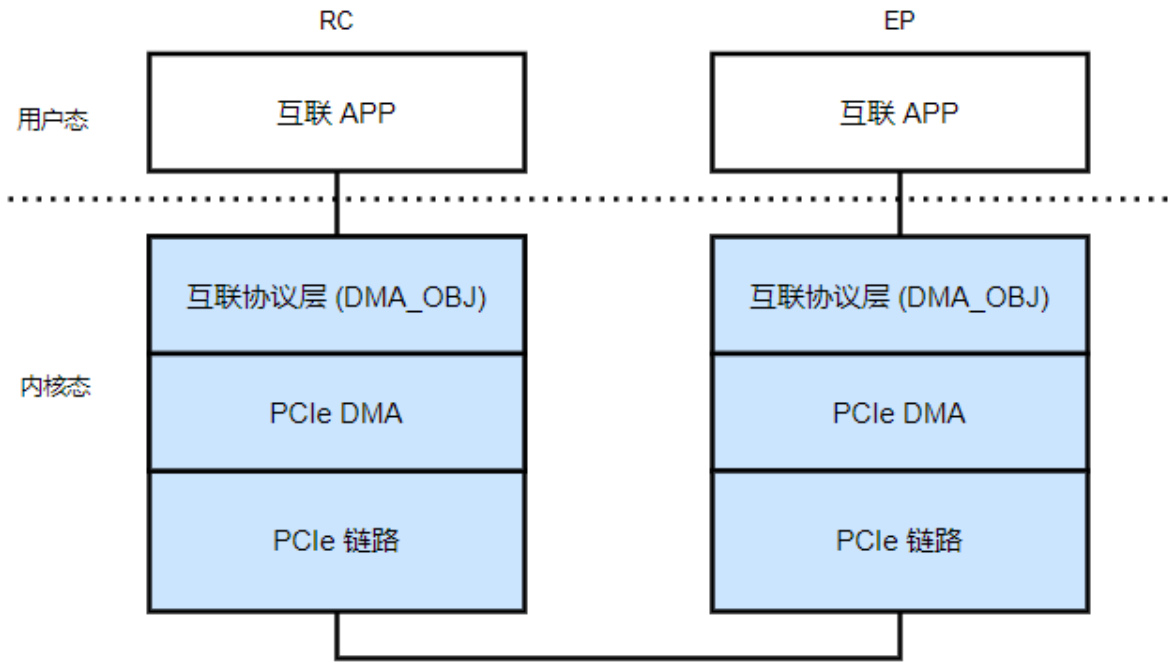
(4) 发起总线重扫描，可任选如下两条指令之一，执行后新设备恢复识别

```
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
```

5. 芯片互联功能

5.1 原理

芯片互联功能的原理是 DMA 的传输基础上添加相应的互联握手协议，所选 PCIe 控制器需支持DMA功能，相关内容查询“芯片资源介绍”章节。



5.2 互联协议层

互联协议层将预留的传输 buffer 规划为多个数据 buffer 和 ack buffer，提供用户接口来完成传输和握手流程。

5.2.1 内核配置

defconfig 配置：

```
CONFIG_ROCKCHIP_PCIE_DMA_OBJ=y
CONFIG_DEBUG_FS=y
```

RC dts 配置：

```
/ {
    reserved-memory {
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;

        dma_trans: dma_trans@3c000000 {
            reg = <0x0 0x3c000000 0x0 0x04000000>;
```

```
        };
        user_region: user_region@0x10000000 {
            reg = <0x0 0x10000000 0x0 0x1000>;
        };
    };
};

&pcie3x2 {
    memory-region = <&dma_trans>;                // 用户配置项: 传输 buffer 对应内存空间预留
    memory-region1 = <&user_region>;             // 固定配置项: 存放用户配置的内存空间buffer, 要求与 EP 内存地址一致
    busno = <0>;                                 // RC 配置为 0
    status = "okay";
};
```

EP dts 配置:

```
/ {
    reserved-memory {
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;

        dma_trans: dma_trans@3c000000 {
            reg = <0x0 0x3c000000 0x0 0x04000000>;
        };
        user_region: user_region@0x10000000 {
            reg = <0x0 0x10000000 0x0 0x1000>;
        };
    };
};

&pcie3x2 {
    compatible = "rockchip,rk3568-pcie-ep";
    memory-region = <&dma_trans>;                // 用户配置项: 传输 buffer 对应内存空间预留
    memory-region1 = <&user_region>;             // 固定配置项: 存放用户配置的内存空间buffer, 要求与 EP 内存地址一致
    busno = <1>;                                 // RC 配置为 1
    status = "okay";
};
```

5.2.2 用户接口

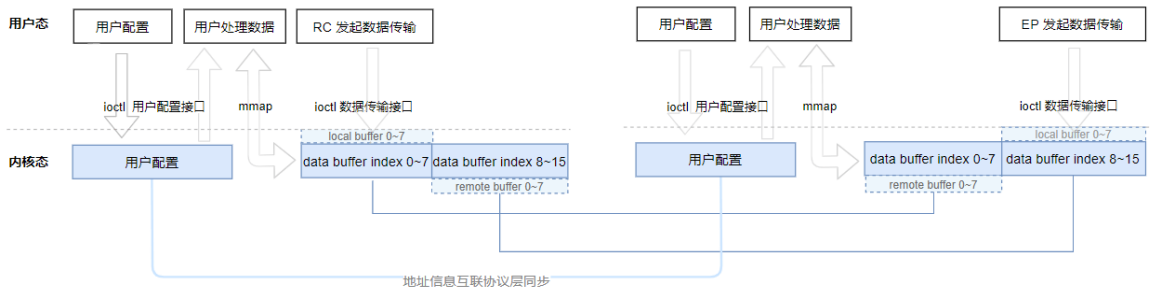
注册设备为 /dev/pcie-dev 设备，提供 poll mmap 和 ioctl 接口，其中 ioctl 主要接口如下:

./include/uapi/linux/rk-pcie-dma.h

```
#define PCIE_DMA_START                                  // 数据传输：启动 DMA 传输（通常发起写传输）
#define PCIE_DMA_GET_LOCAL_READ_BUFFER_INDEX            // 数据传输：获取 local read buffer 状态
#define PCIE_DMA_GET_LOCAL_REMOTE_WRITE_BUFFER_INDEX    // 数据传输：获取 local/remote write buffer 状态
#define PCIE_DMA_SET_LOCAL_READ_BUFFER_INDEX            // 数据传输：设置 local read buffer 状态
#define PCIE_DMA_SYNC_BUFFER_FOR_CPU
#define PCIE_DMA_SYNC_BUFFER_TO_DEVICE
#define PCIE_DMA_WAIT_TRANSFER_COMPLETE                 // 数据传输：等待 DMA 传输完成
#define PCIE_DMA_SET_LOOP_COUNT                         // 数据传输：设置传输循环次数（测试用）
#define PCIE_DMA_GET_TOTAL_BUFFER_SIZE                  // 用户配置项：获取传输 buffer 总大小
#define PCIE_DMA_SET_BUFFER_SIZE                        // 用户配置项：设置传输用单个 data buffer 的大小
#define PCIE_DMA_READ_FROM_REMOTE                       // 数据传输：启动 DMA 传输，读传输
#define PCIE_DMA_USER_SET_BUF_ADDR                      // 用户配置项：设置传输 buffer 起始地址
#define PCIE_DMA_GET_BUFFER_SIZE                        // 用户配置项：获取传输用单个 data buffer 的大小
```

5.2.3 缓存结构及传输

互联协议会根据用户配置自动规划由多个数据项组成的传输缓存，用户通过 mmap 处理数据项内容，通过 io_ctrl 数据传输接口发起相应的数据传输。



用户配置说明（no reverse）：

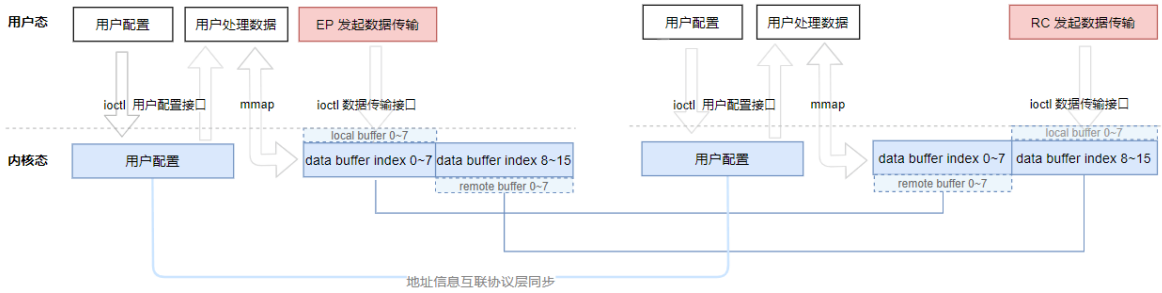
- 传输 buffer 的起始地址和单个传输 data buffer 的大小可以通过 ioctl 用户配置接口设定
- 单个传输 data buffer 尾部固定打包12B 冗余信息，所以有效空间为 buffer_size - 12B
- 传输 buffer 共有 16 个 data buffer 获取，分配给不同端口发起的传输：
  - RC index 0~7 -> data buffer index 0 ~ 7
  - EP index 0~7 -> data buffer index 8 ~ 15

传输说明：

- 传输 buffer 通过 mmap 映射虚拟地址，总长通过 PCIE_DMA_GET_TOTAL_BUFFER_SIZE 接口获取
- 用户处理数据时要寻址到对应 data buffer index 所在的 mmap 偏移

扩展配置：

- 支持 dts 配置 "reverse" property 调转 data buffer 方向，最终 buffer 整体布局如下



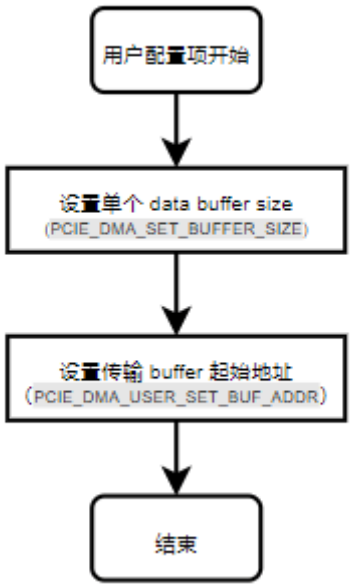
5.3 互联 APP

5.3.1 应用程序简介

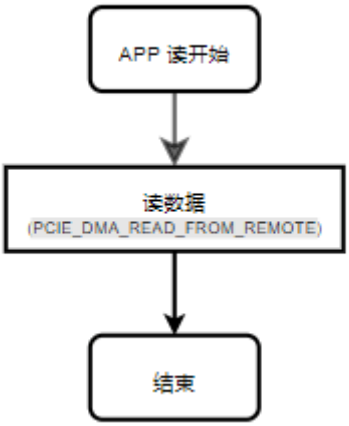
RK 提供的互联应用程序主要包括以下功能：

- test-pcie
  - 写数据传输（RC/EP）—— 本地发起端
  - 读数据传输（RC/EP）—— 本地发起端
- test-pcie-ep-new
  - 用户配置
  - 写数据传输（RC/EP） —— 远端的用户数据处理 deamon 程序，完成数据到达监控、数据处理和对应缓存释放

用户配置流程



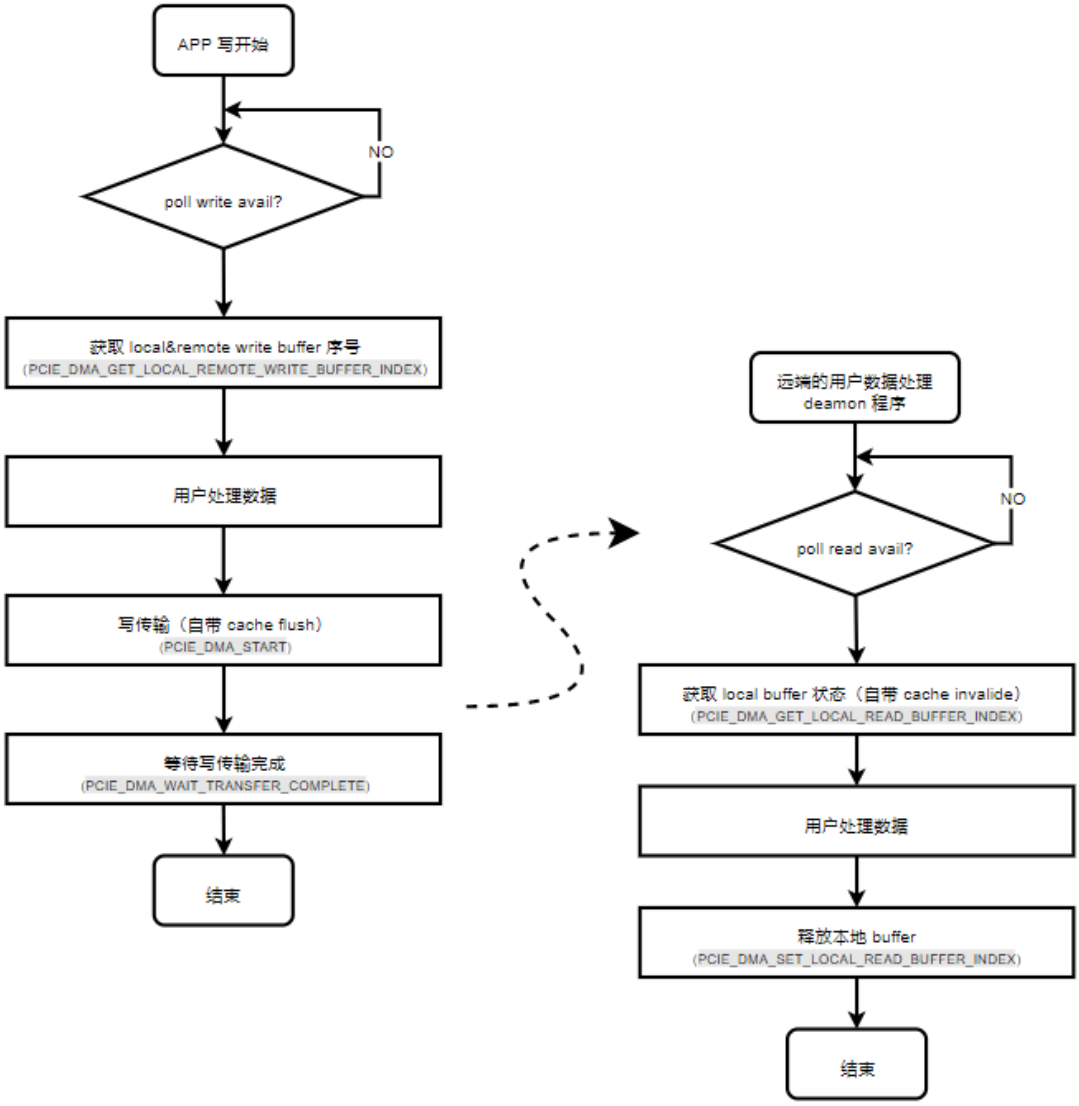
写数据传输



说明:

- 该流程假定远程 buffer 已有有效数据

读数据传输



5.3.2 应用

test-pcie-ep-new

```
test-pcie-ep-new <timeval> <buffer_size> <dma_chanel> <buffer_address>
```

参数:

- timeval：定长 usleep，单位 us，用以模拟远端收到有效数据后的用户行为耗时
- buffer_size：用户配置项，单个数据包上限，单位 KB
- dma_chanel：PCIe DMA 传输通道号
- buffer_address：用户配置项，传输buffer起始地址，仅支持十进制，设置为0则沿用dts设定的内核预留内存memory-region

test-pcie

```
test-pcie <rc> <loop_count> <transfer_size> <rw> <chn>
```

参数:

- rc：1 - RC，2 - EP
- loop_count：重复次数
- transfer_size：用户数据大小
- rw：1 - read，0 - write
- chn：DMA channel

5.3.3 应用实例

内核配置参考“内核配置”章节

1. 将 test-pcie-ep-new 和 test-pcie 拷贝到RC和EP板子中
2. 首先 RC 和 EP 的板子都运行以下命令完成用户配置和支持相应应答:

```
./test-pcie-ep-new 500 1024 0 1006632960 &    # 模拟用户耗时 500us，用户配置项 1M
buffer_size ，DMA chanel 0，用户配置项 0x3c000000（1006632960） 传输 buffer 起始地址
```

3. RC 端发起数据传输

```
./test-pcie 1 1000 512 1 0          # RC 端，1000 次传输，每次传输 512KB 数据，读传输，
DMA chanel 0
./test-pcie 1 1000 512 0 0          # RC 端，1000 次传输，每次传输 512KB 数据，写传输，
DMA chanel 0
```

4. EP 端发起数据传输

```
./test-pcie 2 1000 512 1 0          # EP 端，1000 次传输，每次传输 512KB 数据，读传输，
DMA chanel 0
./test-pcie 2 1000 512 0 0          # EP 端，1000 次传输，每次传输 512KB 数据，写传输，
DMA chanel 0
```

5.4 问题排查

互联模型的异常debug问题请提供下列两个信息

```
cat /sys/kernel/debug/pcie/pcie_trx
cat /proc/interrups | grep pcie
```

6. 标准EP功能件开发

将芯片PCIe接口作为EP设备与任意RK芯片互联，推荐使用我司提供的互联模型，性能与稳定性等均可得到有效的保证，详情可查看“芯片互联功能”章节。若熟悉PCIe EP设备驱动开发的技术人员需要使用RK芯片对接封闭芯片系统(如x86)，或者希望自行开发标准EP业务流程的，可参考本节内容进行二次开发。

标准EP功能件开发需要三个功能组件:

- RC端运行的针对RK3568 EP设备的function driver, 其功能是负责在RC系统中申请bar空间对应的虚拟内存，管理数据业务，注册处理各类中断，提供更上层业务态的业务接口。
- EP端(本例指RK3568芯片)运行的firmware driver, 其功能是负责配置bar内存的inbound和outbound, 提供DMA完成EP端与RC端内存数据搬移等功能。
- 快速建立链路的loader: 负责配置class code, ID, 修改bar需求大小以及迅速建立链路连接；因为例如x86的BIOS扫描总线时间较短，需要提前准备链路。

由于开发技术难度较高，我们提供了可运行的全部demo，希望降低阅读者二次开发的难度。此demo可以将RK3568芯片模拟成一个memory controller, 可对接任何芯片平台的linux系统。以x86为例，执行sudo lspci可以看到我司设备:

```
lt-HP-ProDesk-400-G5 -NT-ID5-APD:~$ sudo lspci
[sudo] password for lt:
00:00.0 Hest bridge: Intel Corporaton 8th Gen Core Processor Host Brldge/RAN
Regtsters
00:02.0 VGA compatible controller: Intel Corporation UHD Graphics 630 (Desktop)

...

02:00.0 Memory controller: Fuzhou Rockchip Electronics co. Ltd Device 356a Crev
01)
```

在RC端加载function driver模块之后，将出现/dev/rk-rmd设备节点，可使用echo/cat访问此节点。访问该节点实际将访问到EP端(本例指RK3568)的内存，而EP端被访问的内存地址在EP端的firmware driver中可配置。利用本demo可以实现最原始的数据交互，为封装更上层业务态提供支持。相关软件范例可以在附录章节所示redmine系统中获取。

注意事项:

- 对接x86设备时需要注意，由于较多市售x86设备主板的x16的插槽默认不支持低于4-lane的设备，烦请设计成x1的金手指接入其x1的槽。
- EP端的系统供电通过金手指由RC的PCIe插槽上提供，并将金手指的#PERST接到EP设备主控的PMU复位信号上，使得RC端可以控制它插槽上的#PERST信号，对EP进行芯片级的复位控制。
- EP端金手指的#PRSENT信号需要正确布置成x1模式。

- EP端的PCIe驱动必须配置Class code, 否则即便正确resize了BAR地址需求，也无法分配到正确的地址。典型的RC端lspci -vvv的输出log如下：

```
Region 0: Memory at <unassigned> (32-bit, non-prefetchable) [size=64M]
```

7. 异常排查

7.1 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。

7.2 trainning 失败

```
PCIe Link Fail的log如下一致重复，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

如果link成功，应该可以看到类似log，LTSSM状态机可能不同，重点看到link up了
[    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。RK356X芯片的PCIe ltssm状态机信息可参照文末附录部分。

7.3 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。
- 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。

7.4 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。请依次检查

- 检查 PCIE30_AVDD_0V9 和PCIE30_AVDD_1V8电压是否满足要求。
- 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，设置如下代码所示

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

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。

7.5 PCIe外设资源分配异常

```
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]
```

如常用应用问题Q4所述，RK356X的PCIe地址空间有限制。此log表明21号总线外设向RK356X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RK356X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。

7.6 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三种类型)，请采购使用对应型号的转接板。

7.7 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)
```

7.8 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
```

7.9 重新映射后访问PCIe设备的BAR地址空间异常

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
```

解决办法如下两种:

- 改用memremap(phys_addr, size, MEMREMAP_WC) 这类接口来替换mmap
- 改用memset_io或者memset_fromio/memset_toio等API

7.10 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 &&
```

7.11 PCIe 3.0设备休眠唤醒异常

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
- 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
```

7.12 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l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。

7.13 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 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。

7.14 设备IO BAR地址空间写入异常

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，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>;
```

7.15 PCIe设备性能抖动

RK356X的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。

```
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
```

7.16 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
```

7.17 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
```

8. 附录

8.1 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	-
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	-

状态描述符	状态代码 (6'h)	备注
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	-

8.2 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。

8.3 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