Contents

[1. Overview 2](#_Toc121321451)

[1.1. PCIe link: - 2](#_Toc121321452)

[1.2. PCIe Lane: - 2](#_Toc121321453)

[2. Features of PCIe in ExynosAuto9 v920: - 3](#_Toc121321454)

[2.1 PCIe channels: - 3](#_Toc121321455)

[2.2 PCIe channel supports bifurcation: - 3](#_Toc121321456)

[2.3 Specification: - 4](#_Toc121321457)

[2.4 PCIe dual mode: - 4](#_Toc121321458)

[3. Hardware connection of v920 SOC with WLAN chip: - 4](#_Toc121321459)

# Overview

## PCIe link: -

PCIe link represents a dual simplex communication between two components. Simplex communication means receive pair and transmit pair. It is named as pair because both receive and transmit is differential signal.

## PCIe Lane: -

Each PCIe lane consists of one Tx pair and one Rx pair. A link supports at least one lane.

To meet bandwidth requirements, a link employs multiple lanes. xN denotes these lanes, where N is width of any of the supported link.

The following lane widths are possible as per specification: -

x1, x2, x4, x8, x12, x16, and x32

For example, a “PCIe x8” connection has 8 data lanes. An x8 link that operates at transfer rate 2.5 GT/s represents an aggregate bandwidth of 2.5GT/s x 8bits/Transfer = 20 Gbps of raw bandwidth in each direction.

Note – “GT/s” means Giga Transfer per second.

Figure 1 – PCIe x1

In figure – 1 above, one Tx pair and 1 Rx pair combinely is called 1 lane. Figure – 1 also shows a PCIe link with width 1. If a link has N lanes, then link width is said N.

A PCIe link with width N will have 4xN wires.

# Features of PCIe in ExynosAuto9 v920: -

## 2.1 PCIe channels: -

PCIe has 3 channels and each channel have 2 links.

1. 4Lane Channel
2. 2Lane\_0 Channel
3. 2Lane\_1 Channel

**I. 4Lane Channel**

4Lane Channel comprises of below two links as below.

1. Max 4Lane link
2. Max 2Lane link

**II. 2Lane\_0 Channel**

2Lane\_0 Channel comprises of below two links as below.

1. Max 2Lane link
2. Max 1Lane link

**III. 2Lane\_1 Channel**

2Lane\_1 Channel comprises of below two links as below.

1. Max 2Lane link
2. Max 1Lane link

## 2.2 PCIe channel supports bifurcation: -

Each channel support Bifurcation as –

1. All lane mode or
2. Each lane mode

**I. 4Lane Channel**

4Lane channel can be configured as below.

4Lane All mode or 2Lane/2Lane bifurcation mode.

**II. 2Lane\_0 Channel**

2Lane\_0 channel can be configured as below.

2Lane All mode or 1Lane/1Lane Bifurcation mode.

**III. 2Lane\_1 Channel**

2Lane\_1 channel can be configured as below.

2Lane All mode or 1Lane/1Lane Bifurcation mode.

## 2.3 Specification: -

It supports PCI Express base 5.0 specification, revision 1.0.

PCIe suports upto Max Gen 5 having per lane data rate = 32 Gbps

## 2.4 PCIe dual mode: -

PCIe module supports dual mode as –

Root complex and

Endpoint

# Hardware connection of v920 SOC with WLAN chip: -

Diagram

Description automatically generated with low confidence

Figure 2 - WiFi chip connection wih PCIe link

WLAN chip – CYPRESS CYW8x570

WLAN interfaces to host processor through a PCIe v3.0 Gen2 interface.

# lspci command: -

Text

Description automatically generated

# Exynos PCI controller software manual

## Designware hardware: -

Exynos PCIe controller is based on the DesignWare hardware and therefore the driver re-uses the DesignWare core functions to implement the driver.

Reference - Android-kernel/exynos/drivers/pci/controller/dwc/Kconfig

## PCIe controller .dtsi files: -

File - Android-kernel/exynos/arch/arm64/boot/dts/exynos/exynosautov920-pci.dtsi

pcie\_2: pcie@161C0000 {

compatible = "samsung,exynos-v920-pcie-rc";

gpios = <&gph0 5 GPIO\_ACTIVE\_LOW>; /\* PERST \*/

reg = <0x0 0x161C0000 0x1000 /\* elbi base \*/

0x0 0x161C1304 0x4 /\* elbi base other \*/

0x0 0x161C6000 0x1000 /\* gen\_subsystem \*/

0x0 0x161C7000 0x1000 /\* e32\_phy \*/

0x0 0x16180000 0x2544 /\* phy base \*/

0x0 0x14000000 0x5000 /\* DBI base \*/

0x0 0x14200000 0x1000 /\* DBI Shadow base \*/

0x0 0x6FFFD000 0x2000 /\* configuration space \*/

0x0 0x16030000 0x1000 /\* sysreg base \*/

0x0 0x16320000 0xF04 /\* I/A space \*/

0x0 0x16003000 0x4 /\* CMU space \*/

0x0 0x14600000 0x2000>; /\* ATU base \*/

reg-names = "elbi", "elbi\_other", "gen\_subsys", "e32\_phy", "phy", "dbi", "dbi\_shadow", "config", "sysreg", "ia", "cmu", "atu";

……..

……..

};

Above platform resource description in dts file is read by PCIe platform/controller driver as below.

exynos\_v920\_pcie\_probe - > add\_pcie\_port\_v920 - > dw\_pcie\_host\_init().

The function dw\_pcie\_host\_init() does the following tasks.

**struct pcie\_port: -**

File - /drivers/pci/controller/dwc/pcie-designware.h

struct pcie\_port {

bool has\_msi\_ctrl:1;

u64 cfg0\_base;

void \_\_iomem \*va\_cfg0\_base;

u32 cfg0\_size;

resource\_size\_t io\_base;

phys\_addr\_t io\_bus\_addr;

u32 io\_size;

int irq;

const struct dw\_pcie\_host\_ops \*ops;

int msi\_irq;

struct irq\_domain \*irq\_domain;

struct irq\_domain \*msi\_domain;

dma\_addr\_t msi\_data;

struct page \*msi\_page;

struct irq\_chip \*msi\_irq\_chip;

u32 num\_vectors;

u32 irq\_mask[MAX\_MSI\_CTRLS];

struct pci\_host\_bridge \*bridge;

raw\_spinlock\_t lock;

DECLARE\_BITMAP(msi\_irq\_in\_use, MAX\_MSI\_IRQS);

};

struct dw\_pcie\_host\_ops {

int (\*host\_init)(struct pcie\_port \*pp);

int (\*msi\_host\_init)(struct pcie\_port \*pp);

};

enum dw\_pcie\_as\_type {

DW\_PCIE\_AS\_UNKNOWN,

DW\_PCIE\_AS\_MEM,

DW\_PCIE\_AS\_IO,

};

int dw\_pcie\_host\_init(struct pcie\_port \*pp)

{ ….

cfg\_res = platform\_get\_resource\_byname(pdev, IORESOURCE\_MEM, "config");

if (cfg\_res) {

pp->cfg0\_size = resource\_size(cfg\_res);

pp->cfg0\_base = cfg\_res->start;

pp->va\_cfg0\_base = devm\_pci\_remap\_cfg\_resource(dev, cfg\_res);

}

….

if (!pci->dbi\_base) {

struct resource \*dbi\_res = platform\_get\_resource\_byname(pdev, IORESOURCE\_MEM, "dbi");

pci->dbi\_base = devm\_pci\_remap\_cfg\_resource(dev, dbi\_res);

}

bridge = devm\_pci\_alloc\_host\_bridge(dev, 0);

pp->bridge = bridge;

pci->link\_gen = of\_pci\_get\_max\_link\_speed(np);

dma\_set\_mask(pci->dev, DMA\_BIT\_MASK(32));

pp->msi\_page = alloc\_page(GFP\_DMA32);

pp->msi\_data = dma\_map\_page(pci->dev, pp->msi\_page, 0, PAGE\_SIZE, DMA\_FROM\_DEVICE);

/\* Set default bus ops \*/

bridge->ops = &dw\_pcie\_ops;

bridge->child\_ops = &dw\_child\_pcie\_ops;

if (pp->ops->host\_init) {

ret = pp->ops->host\_init(pp);

}

dw\_pcie\_iatu\_detect(pci);

dw\_pcie\_setup\_rc(pp);

if (!dw\_pcie\_link\_up(pci) && pci->ops && pci->ops->start\_link) {

ret = pci->ops->start\_link(pci);

}

/\* Ignore errors, the link may come up later \*/

dw\_pcie\_wait\_for\_link(pci);

bridge->sysdata = pp;

ret = pci\_host\_probe(bridge);

…….

}