地平线
Horizon Robotics

X3J3平台系统软件
媒体性能调试指南

v1.0

2021-02

免责声明

本文档信息仅用于帮助系统和软件使用人员使用地平线产品。本文档信息未以明示或暗示方式授权他人基于本文档信息设计或制造任何集成电路。

本文档中的信息如有更改，恕不另行通知。尽管本文档已尽可能确保内容的准确性，本文档中的所有声明、信息和建议均不构成任何明示或暗示的保证、陈述或担保。

本文档中的所有信息均按“原样”提供。地平线不就其产品在任何特定用途的适销性、适用性以及不侵犯任何第三方知识产权方面做出任何明示或暗示保证、陈述或担保。地平线不承担产品使用所引起的任何责任，包括但不限于直接或间接损失赔偿。

买方和正在基于地平线产品进行开发的其他方（以下统称为“用户”）理解并同意，用户在设计产品应用时应承担独立分析、评估和判断的责任。用户应对其应用（以及用于其应用的所有地平线产品）的安全性承担全部责任，并保证符合所有适用法规、法律和其它规定的要求。地平线产品简介和产品规格中提供的“典型”参数在不同应用下可能会不同，实际性能也可能随时间而变化。所有工作参数，包括“典型”参数，都必须由用户自己针对每项用户应用进行验证。

用户同意如因用户未经授权使用地平线产品或因不遵守本说明中的条款，造成任何索赔、损害、成本、损失和（或）责任，用户将为地平线及其代表提供全额赔偿。

修订记录

修订记录列出了各文档版本间发生的主要更改。下表列出了每次文档更新的技术内容。

版本	修订日期	修订说明
0.1	2020-6-23	提纲
0.2	2020-9-1	增加 Qos 控制相关内容
0.3	2020-9-5	增加常用场景 Qos 设置
1.0	2021-02	1.0 版本发布

1. 引言

1.1. 编写目的

撰写该文档主要是对于 XJ3 芯片方案中关于图像数据处理通路等模块的实际使用的常用场景中根据 DDR 带宽和延迟进行各处理模块 DDR 优先级和其它一些相关参数的调整说明。

在由于 DDR 瞬时带宽不足造成丢帧和每秒处理帧率之间根据场景选择一个最合适的值。

1.2. 术语约定

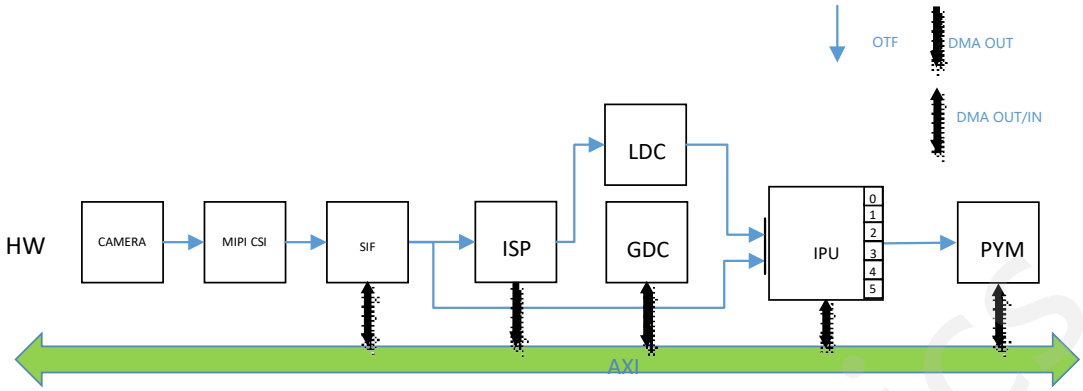
1. 缩写	1. 全称
2. VIO	2. Video in/out
3. ISP	3. Image Signal Processor
4. IPU	4. Image Process Unit
5. BPU	5. Brain Process Unit
6. HAL	6. Hardware Abstraction Layer
7. FW	7. Firmware
8. PYM	8. Pyramid
9. OTF	9. On The Fly

1.3. 读者对象和阅读建议

该文档针对需要使用到 VIO 以及 Camera 的应用工程师，测试工程师以及相关工程人员等，文档提供了该部分软件使用上的各层面的信息以及大致的概念，如果涉及实际接口开发，请参考对应的软件开发手册。

2. 总体概述

Camera 是图像数据的主要外部来源，VIO 部分软件是一个相对不透明的内部软件，主要面向提供内部应用软件提供相关的图像以及信息，XJ3 芯片内部图像处理 IP 信息大致如下：



输入方式	IP	输出方式
Online	MIPI	Online
Online/Offline	SIF	Online/Offline
Online	ISP	Online/Offline
Online	LDC	Online
Offline	GDC	Offline
Online/Offline	IPU	Online/Offline
Online/Offline	PYM	Offline

注：Online 指硬件通过片内 RAM 交换数据，Offline 指硬件通过 DDR 交换数据。

3. DDR Master 的 QoS

XJ3 各模块通过 AXI 接口访问 DDR，XJ3 有 8 个 AXI 接口，分别为 AXI_0 ~ AXI_7，XJ3 的模块使用 AXI 接口关系如下表：

端口号	AXI_0	AXI_1	AXI_2	AXI_3	AXI_4	AXI_5	AXI_6	AXI_7
模块名	CPU/R5	NOC	CNN0	CNN1	VIO0	VPU/JPU	VIO1	PERI

AXI_4 和 AXI_6 可配置，可以通过寄存器配置 VIO 子模块到 AXI_4 或者 AXI_6，AXI_6 有更高的优先级。

XJ3 VIO 包括如下子模块：SIF_W、ISP0_M0、ISP0_M2、GDC0、DIS、SIF_R、IPU0、PYM、IAR。

3.1. AXI QOS 控制

AXI Qos 优先级范围 0~15，值越大优先级越高。XJ3 系统启动后读写 QoS 默认配置为 0x2021100。

每个 Port 的优先级值通过 Perf Monitor 的 DDR_PORT_READ/WRITE_QOS_CTRL 寄存器设置，Perf Montior 再通过硬件的方式设置到 DDR 控制器中。软件无需设置 DDR 控制器。

DDR QoS 的值在 DDR_Monitor 驱动中通过 Sysfs 的属性文件的方式设置和查询。

可以通过 all 属性文件一次性设置，最低的 4bit 对应 P0_CPU，最高 4bit 对应 P7_PERI。

也可以通过 cpu、bifdma、bpu0、bpu1、vio0、vpu、vio1、peri 单独设置和查询各个端口的优先级，如下：

QoS sysfs 接口

```
#设置读 QoS，VIO1 为 3，VPU 为 1，VIO0 位 2，其余模块为 0
# echo 0x03120000 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/all
#设置写 QoS，VIO1 为 3，VPU 为 1，VIO0 位 2，其余模块为 0
# echo 0x03120000 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/all
#查询读 QoS：
# cat /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/all
03120000:
P0_CPU:      0
P1_BIFDMA: 0
P2_CNN0:     0
P3_CNN1:     0
P4_VIO0:     2
P5_VPU:      1
P6_VIO1:     3
P7_PERI:     0
#查询写 QoS：
# cat /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/all
03120000:
P0_CPU:      0
P1_BIFDMA: 0
P2_CNN0:     0
P3_CNN1:     0
P4_VIO0:     2
P5_VPU:      1
P6_VIO1:     3
P7_PERI:     0
#设置 cpu 读 QoS 为 1：
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/cpu
#设置 bifdma 读 QoS 为 2：
# echo 2 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/bifdma
#设置 bpu0 读 QoS 为 1：
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/bpu0
#设置 bpu1 读 QoS 为 1：
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/bpu1
#设置 vio0 读 QoS 为 2：
# echo 2 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/vio0
#设置 vpu 读 QoS 为 0：
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/vpu
```

```
#设置 vio1 读 QoS 为 3：
# echo 3 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/vio1
#设置 peri 读 QoS 为 0：
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/peri

#设置 cpu 写 QoS 为 1：
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/cpu
#设置 bifdma 写 QoS 为 2：
# echo 2 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/bifdma
#设置 bpu0 写 QoS 为 1：
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/bpu0
#设置 bpu1 写 QoS 为 1：
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/bpu1
#设置 vio0 写 QoS 为 2：
# echo 2 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/vio0
#设置 vpu 写 QoS 为 0：
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/vpu
#设置 vio1 写 QoS 为 3：
# echo 3 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/vio1
#设置 peri 写 QoS 为 0：
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/peri
```

3.2. VIO 子模块配置

XJ3 VIO 子模块包括 SIF_W、ISP0_M0、ISP0_M2、GDC0、DIS、SIF_R、IPU0、PYM、IAR，分别对应 SIF 模块写、ISP 写、ISP Temper 读写、GDC0 读写、DIS 写、SIF 模块读、IPU0 模块读写、PYM 模块读写、IAR 模块读写。

可以通过 AXIBUS 寄存器将这些子模块配置到 VIO0 或者 VIO1 上，XJ3 系统启动后默认配置 IAR 和 SIF_W 到 VIO1，其余模块配置到 VIO0。AXIBUS 寄存器 bit31~bit16 对应子模块如下图：

	VIO AXI Bus				單路online	單路online	單路offline	多路offline
	Slave / Master	DDR	Usage	online	IPU_Out	PYM_Out	SIF_out	SIF_out
0	n/a	0x0 ~ (0x8000_0000 - 1)	n/a	n/a	0	0	0	0
1	SIF_W	0x0 ~ (0x8000_0000 - 1)	SIF写入Sensor数据	online	1	1	1	1
2	ISP0_M0	0x0 ~ (0x8000_0000 - 1)	ISP0 fr dma write	online/offline	1	1	0	0
3	ISP0_M1	0x0 ~ (0x8000_0000 - 1)	ISP0 ds dma write	online/offline	1	1	0	0
4	ISP0_M2	0x0 ~ (0x8000_0000 - 1)	ISP0 temper dma read/write	online/offline	1	1	0	0
5	ISP1_M0	0x0 ~ (0x8000_0000 - 1)	ISP1 fr dma write	x (don't care)	0	0	0	0
6	ISP1_M1	0x0 ~ (0x8000_0000 - 1)	ISP1 ds dma write	x (don't care)	0	0	0	0
7	ISP1_M2	0x0 ~ (0x8000_0000 - 1)	ISP1 temper dma read/write	x (don't care)	0	0	0	0
8	DWE_GDC_0	0x0 ~ (0x8000_0000 - 1)	GDC读YUV和Table	offline	0	0	0	0
9	DWE_DIS	0x0 ~ (0x8000_0000 - 1)	写入DIS数据	online/offline	1	1	0	0
10	DWE_GDC_1	0x0 ~ (0x8000_0000 - 1)	GDC读YUV和Table	offline	0	0	0	0
11	SIF_R	0x0 ~ (0x8000_0000 - 1)	SIF读出Sensor数据	offline	0	0	0	0
12	IPU0	0x0 ~ (0x8000_0000 - 1)	Static Mux-out of IPU0 AXI (offline 工作模式)	online/offline	1	1	0	0
13	IPU1	0x0 ~ (0x8000_0000 - 1)	Static Mux-out of IPU1 AXI (offline 工作模式)	x (don't care)	0	0	0	0
14	PYM	0x0 ~ (0x8000_0000 - 1)	Static Mux-out of PYM AXI	online/offline	0	1	1	1
15	IAR	0x0 ~ (0x8000_0000 - 1)	IAR read/write axi	online	1	1	1	1
				0xA4000038[31:16]	921E	D21E	C002	C002

其中灰色部分模块不存在，对应 bit 设置为 1，该子模块被配置到 VIO1 上，否则配置到 VIO0 上。可以通过 all 属性一次性配置或查询，查询返回 vio1 上的模块，别的模块在 vio0 上。也可以通过子模块属性单独配置或查询。

AXIBUS sys 接口

```
# 设置，值为 1 配置到 vio1，值为 0 配置到 vio0
# echo 0xc0020000 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/all
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/sifr
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/isp0m0
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/isp0m1
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/isp0m2
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/t21
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/gdc0
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/gdc1
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/iar
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/pym
# echo 1 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/sifw
# echo 0 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/ipu0

#读取，打印所有配置在 vio1 上的模块，其余模块为 vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/all
axibus: 0xc0020000:
sifw: vio1
pym: vio1
iar: vio1
#模块读取
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/sifr
axibus: 0xc0020000:
sifr: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/sifw
axibus: 0xc0020000:
sif: vio1
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/isp0m0
axibus: 0xc0020000:
isp_0_m0: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/isp0m1
axibus: 0xc0020000:
isp_0_m1: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/isp0m2
axibus: 0xc0020000:
isp_0_m2: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/gdc0
axibus: 0xc0020000:
gdc_0: vio0
```

```
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/gdc1
axibus: 0xc0020000:
gdc_1: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/t21
axibus: 0xc0020000:
t21: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/ipu0
axibus: 0xc0020000:
ipu0: vio0
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/pym
axibus: 0xc0020000:
pym: vio1
# cat /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/iar
axibus: 0xc0020000:
iar: vio1
```

4. SIF 的 hblank 设置

SIF 将读取到的图像逐行送给 ISP 处理，可以通过增加行间隔 hblank 来延迟下一行的送出，以置换更多时间给 ISP 或后边的模块做处理。

XJ3 SIF 默认的 hblank 为 10。

hblank 会影响帧率，与帧率的关系为：

time = ((width+hblank*32)*hight）/(clock*1000)。 clock 为 ISP 频率，默认为 544M。

以 4K 为例计算如下：

time=((width+hblank*32)*high)/(clock*1000)					
image width	hblank(register)	image high	clock(MHz)	time(ms)	fps
3840	10	2160	544	16.5176	60.54
3840	40	2160	544	20.3294	49.19
3840	70	2160	544	24.1412	41.42
3840	120	2160	544	30.4941	32.79

提供 Sysfs 接口设置查询 hblank，如下：

设置 hblank：echo 120 > /sys/devices/platform/soc/a4001000.sif/hblank

查询 hblank：cat /sys/devices/platform/soc/a4001000.sif/hblank

5. IPU 的设置

5.1. IPU Line_delay wr_ddr_fifo_thred

IPU 有个 line_delay 设置，单位为 1 行。值越大，代表 IPU 可以忍受的总线延迟更大，对 offline 模式下降低 frame drop 有帮助。

同时 wr_ddr_fifo_thred 的值越小越能够降低 frame drop。

当 ipu 输出多个通道同时到 DDR 的时候，建议将 line_delay 设置为 255，wr_ddr_fifo_thred 设置为 0。

line_delay 默认值是 16，wr_fifo_thred0 默认值是 0x30323020，wr_fifo_thred1 默认值是 0x00003220。

提供 sysfs 接口设置如下:

echo 0x0 > /sys/devices/platform/soc/a4040000.ipu/wr_fifo_thred0

echo 0x0 > /sys/devices/platform/soc/a4040000.ipu/wr_fifo_thred1

echo 255 > /sys/devices/platform/soc/a4040000.ipu/line_delay

5.2. IPU Clock

IPU 的 clock 由 SIF mclk 提供，可以通过 sysfs 配置 SIF clock 来改变 IPU 的频率，IPU 频率默认为 544MHz，可以配置的频率有 544M、408M、326M、272M。

echo 544000000 > /sys/module/hobot_dev_ips/parameters/sif_mclk_freq

5.3. IPU 安全尺寸

IPU 多个通道的 FIFO 深度不同，安全尺寸如下

IPU Scaler #	Full 深度限制 (Bytes)	建议分辨率(像素)
Scaler 5(IPU US)	4096	8M
Scaler 2(DS2)	4096	8M
Scaler 1(DS1)	2048	2M
Scaler 3(DS3)	2048	2M
Scaler 4(DS4)	1280	1M
Scaler 0(DS0)	1280	1M

Scaler 0~4 对应 IPU 的 ds0~5，Scaler5 对应 IPU 的 us。如果输出尺寸超过安全尺寸，可能会造成硬件丢帧概率变大、输出数据中连续二三十字节出错的风险。

6. 典型场景的设置

6.1. 单路 4K 输入多通道编码

典型场景如下：4k DOL2 输入，SIF - offline - ISP - GDC - IPU，IPU 出 1 路 4k，2 路 1080P，2 路 D1 共 5 路送编码器编码。同时 IPU ds2 online 到 PYM，PYM 出 720P。

SIF hblank 和 QoS 建议配置如下：

echo 120 > /sys/devices/platform/soc/a4001000.sif/hblank

echo 0x10100000 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/all

echo 0x03120000 > /sys/bus/platform/drivers/ddr_monitor/read_qos_ctrl/all

echo 0x03120000 > /sys/bus/platform/drivers/ddr_monitor/write_qos_ctrl/all

6.2. 双路 1080P 输入

典型场景如下：两路 1080P 输入，SIF-offline-ISP-online-IPU-online-PYM->DDR (基础层)。

一路 pym 出来的去编码（1080P）+显示（1080P），另一路开 BPU。

SIF hblank 和 QoS 建议配置如下：

```
echo 120 > /sys/devices/platform/soc/a4001000.sif/hblank
echo 0x40000000 > /sys/bus/platform/drivers/ddr_monitor/axibus_ctrl/all
```

6.3. 单路 1080P 输入

典型场景如下：单路 1080P 输入，SIF-offline-ISP-online-IPU，IPU 6 通道 roi 打开。

SIF hblank 建议配置如下：

```
echo 64 > /sys/devices/platform/soc/a4001000.sif/hblank
```

7. 多进程共享配置

多进程共享目前最多支持 8 个进程共享一路 camera 数据，支持从 IPU 和 PYM 获取输出数据，多进程共享需要满足：

- 必须是全 online 的场景：SIF-online-ISP-online-IPU-online-PYM；
- 输出通道配置 BUF 个数需要大于等于 4，否则会有帧率较低的风险；