

Multi-Channel PCIe QDMA&RDMA Subsystem

User Guide

V1.5

July 28, 2024

Revision History

The following table shows the revision history for this document.

Data	Version	Revision
2021.10.23	1.0	Initial Release
2022.7.5	1.1	增加C2H和H2C DMA的逻辑复位
2022.7.31	1.2	增加Display的FPS Timing控制和寄存器
2022.12.25	1.3	增加RDMA功能
2023.3.27	1.4	完善RDMA的Ring指针
2024.7.28	1.5	优化RAM使用量，提高C2H和H2C性能

目 录

1	介绍.....	1
1.1	特性	1
1.2	应用	2
2	概述.....	2
2.1	特性概要	3
3	产品规格	4
3.1	性能	4
3.2	资源	4
3.3	内核 COMPONENT	5
3.3.1	<i>Target Bridge</i>	5
3.3.2	<i>H2C 通道</i>	6
3.3.3	<i>C2H 通道</i>	6
3.3.4	<i>AXI4-Lite Master</i>	6
3.3.5	<i>IRQ Module</i>	6
3.3.6	<i>DMA 操作</i>	6
3.4	端口描述	7
3.5	寄存器空间	13
3.5.1	<i>PCIe to AXI4-Lite Master (BAR1) 地址映射</i>	13
3.5.2	<i>PCIe to DMA (BAR0) 地址映射</i>	13
3.6	DMA FLOW	25
3.6.1	<i>Global Reset</i>	25
3.6.2	<i>CH#i H2C Flow for SGDMA</i>	26
3.6.3	<i>CH#i C2H Flow for SGDMA</i>	27
3.6.4	<i>CH#i H2C Flow for CDMA</i>	27
3.6.5	<i>CH#i C2H Flow for CDMA</i>	28
3.6.6	<i>CH#i H2C Flow for RDMA</i>	29
3.6.7	<i>CH#i C2H Flow for RDMA</i>	30

3.7	设备驱动	31
3.7.1	<i>Windows WDF (Queue or Non-Queue, Ring).....</i>	<i>31</i>
3.7.2	<i>Linux (Queue or Non-Queue, Ring).....</i>	<i>31</i>
3.7.3	<i>V4L2.....</i>	<i>31</i>
4	附录(视频存储队列管理说明).....	31
4.1	视频采集队列存储管理	31
4.2	视频显示队列存储管理	32

1 介绍

基于 PCI Express Integrated Block, Multi-Channel PCIe QDMA Subsystem 实现了使用 DMA 地址队列的独立多通道、高性能 Continuous (CDMA) 或 Scatter Gather DMA (SGDMA), 提供 FIFO/AXI4-Stream 用户接口。

基于 PCI Express Integrated Block, Multi-Channel PCIe RDMA Subsystem 实现了使用 DMA Ring 缓冲的独立多通道、高性能/超低延时/超低抖动 Continuous Ring DMA, 提供 FIFO/AXI4-Stream 用户接口。

1.1 特性

- 支持 Ultrascale+, Ultrascale, 7 Series 的 PCI Express Integrated Block
- 支持 PCIe 2.0, 3.0, 4.0 速度; PCIe x4, x8, x16 链路宽度
- 支持 64, 128, 256, 512-bit 数据路径
- 64-bit 源地址, 目的地址, 和描述符地址
- 多达 8 个独立的 host-to-card (H2C/Read) 数据通道或 H2C DMA, H2C DMA 效率高达 90% 以上
- 多达 8 个独立的 card-to-host (C2H/Write) 数据通道或 C2H DMA, C2H DMA 效率高达 90% 以上
- AXI4-Stream/FIFO 用户接口(每个通道都有自己的 AXI4-Stream/FIFO 接口)
- 每个 DMA 引擎支持 DMA 地址队列, 队列深度可达 32
- 每个 DMA 引擎支持 DMA Ring 缓冲, Ring 缓冲深度和个数可配置
- RDMA 的超低延时和超低抖动性
- H2C DMA 支持视频显示定时时序输入控制
- AXI4-Lite Master 接口允许 PCIe 通信绕过 DMA 引擎
- Scatter Gather 描述符列表支持无限列表大小
- 每个描述符的最大传输长度为 4GB
- MSI 中断
- 连续描述符的块获取, 支持描述符预取
- 中断或查询模式

1.2 应用

本内核体系结构支持广泛的计算和通信目标程序应用，强调性能、成本、可扩展性、功能可扩展性和关键任务可靠性。典型应用包括：

- 数据通信网络
- 电信网络
- 宽带有线和无线应用
- 网络接口卡
- 用于各种应用程序的服务器 add-in card

典型应用如下图所示：

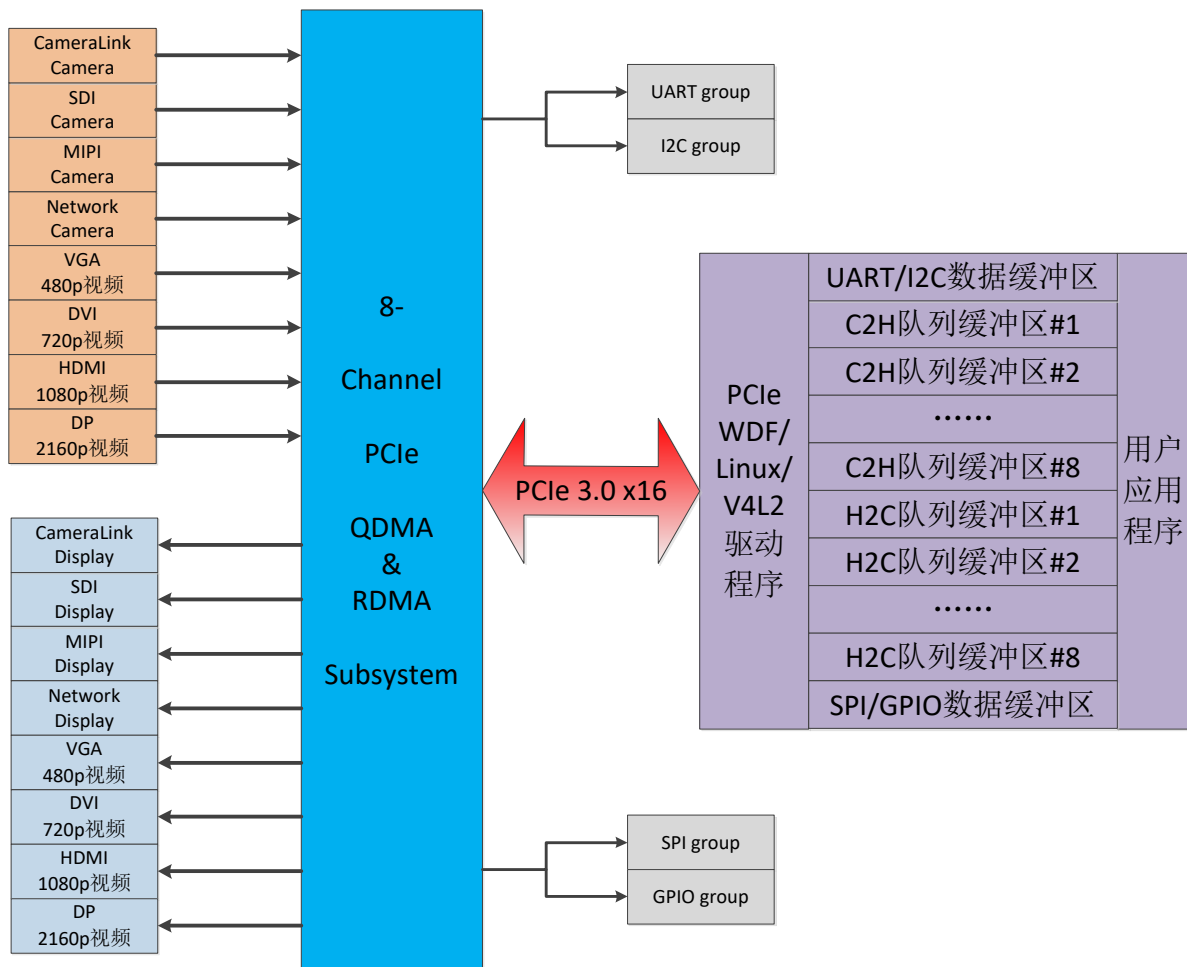


图 1 Multi-Channel PCIe QDMA&RDMA Subsystem 典型应用：8 通道视频采集和视频显示

2 概述

Multi-Channel PCIe QDMA&RDMA Subsystem 作为一个高性能 DMA 数据搬运器，

内核通过 AXI4-Stream/FIFO 接口直接连接 RTL 逻辑。使用提供的字符驱动程序，AXI4-Stream/FIFO 接口可用于 PCIe 地址空间和 AXI 地址空间之间的高性能数据搬移。除了基本的 DMA 功能，DMA 支持多达 8 个独立的 upstream 和 downstream 通道，每个通道支持深度为 32 的 DMA 地址队列，以及深度和个数可配置的 DMA Ring 缓冲，另外还允许 PCIe 通信绕过 DMA 引擎。

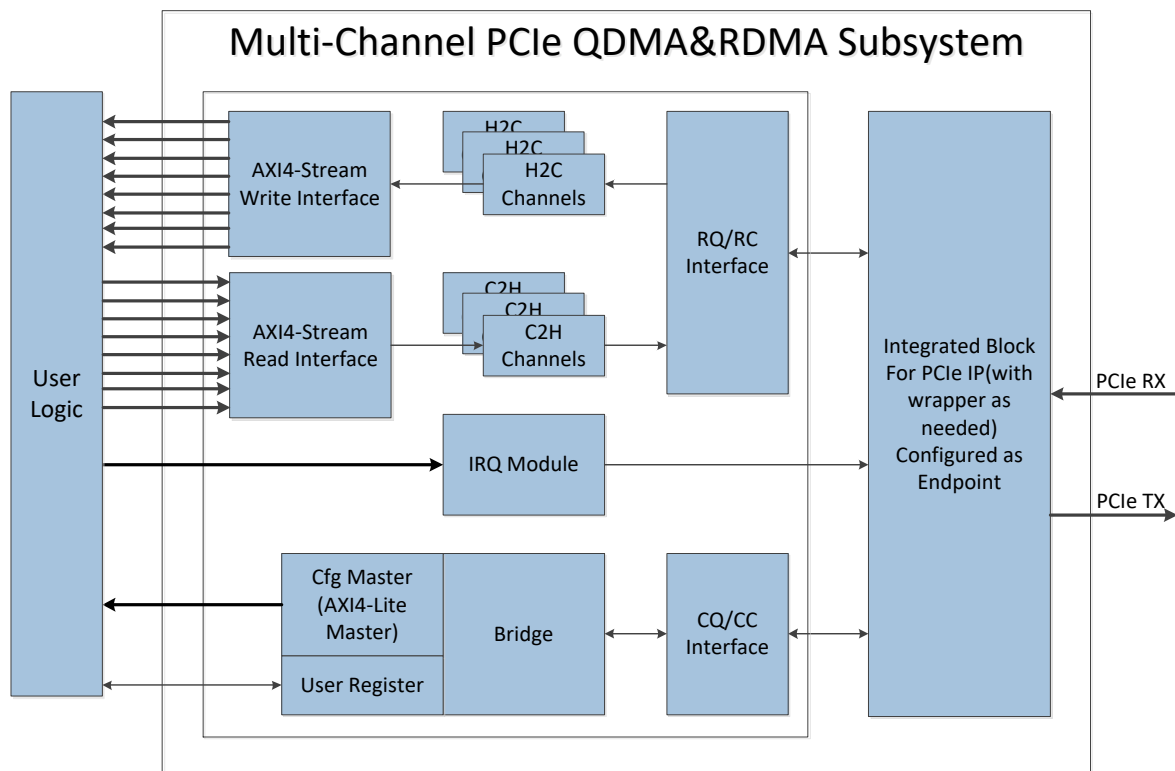


图 2 Multi-Channel PCIe QDMA&RDMA Subsystem 概述

2.1 特性概要

基于描述符提供的信息：源地址，目的地址和传输数据长度，Multi-Channel PCIe QDMA&RDMA Subsystem 实现 Host 存储器和 PCIe DMA 子系统之间的数据搬移。这些 DMA 可以同时是 Host to Card (H2C) 和 Card to Host (C2H) 传输。每个 DMA 通道对应各自的 AX4-Stream/FIFO 接口，DMA 从 Host 存储器获取并解析描述符链表，基于描述符链表信息完成自己通道的数据传输，然后使用 MSI 中断发出描述符完成或错误的信令。内核也提供多达 16 个输出到 Host 的用户中断信号。

主机可以通过以下 2 个接口访问用户逻辑：

- AXI4-Lite Master 配置接口：这个接口是一个固定的 32-bit 端口，用于对性能要求不高的用户配置和状态寄存器的访问
- User Register：这个接口是多个 32-bit 向量信号和 1-bit 信号，这些信号来自

对应 DMA 通道数据搬移过程中产生的控制或状态信号

3 产品规格

结合 Integrated Block for PCI Express IP, Multi-Channel PCIe QDMA&RDMA Subsystem 为 PCIe 提供了一个高性能的 DMA 解决方案。

3.1 性能

Endpoint 配置参数:

1. Max Payload Size=256-byte
2. Max Read Request Size=512-byte
3. DMA Transfer Length = 4MB

C2H & H2C DMA 性能:

1. PCIe 3.0 x16, C2H DMA 和 H2C DMA 速度大于 14GB/s
2. PCIe 3.0 x8, C2H DMA 和 H2C DMA 速度大于 7GB/s
3. PCIe 2.0 x8, C2H DMA 和 H2C DMA 速度大于 3.6GB/s
4. PCIe 2.0 x4, C2H DMA 和 H2C DMA 速度大于 1.7GB/s

3.2 资源

表 1 PCIe 3.0 x16 DMA Subsystem 资源 (8 通道)

	LUTs	FFs	BRAMs	PCIe
资源	53424	95356	142	1

表 2 PCIe 3.0 x8 DMA Subsystem 资源 (8 通道)

	LUTs	FFs	BRAMs	PCIe
资源	25488	41741	73	1

表 3 PCIe 2.0 x8 DMA Subsystem 资源 (8 通道)

	LUTs	FFs	BRAMs	PCIe
资源	24372	32342	36	1

表 4 PCIe 2.0 x4 DMA Subsystem 资源 (8 通道)

	LUTs	FFs	BRAMs	PCIe
资源	23595	27989	33	1

表 5 PCIe 3.0 x16 DMA Subsystem 资源 (1 通道)

	LUTs	FFs	BRAMs	PCIe
资源	22107	46116	30	1

表 6 PCIe 3.0 x8 DMA Subsystem 资源（1 通道）

	LUTs	FFs	BRAMs	PCIe
资源	8690	18174	17	1

表 7 PCIe 2.0 x8 DMA Subsystem 资源（1 通道）

	LUTs	FFs	BRAMs	PCIe
资源	8500	12343	8	1

表 8 PCIe 2.0 x4 DMA Subsystem 资源（1 通道）

	LUTs	FFs	BRAMs	PCIe
资源	6451	7999	12	1

3.3 内核 Component

内核内部实现多达 16 个独立的物理 DMA 引擎（多达 8 个 C2H 和 8 个 H2C）。这些 DMA 引擎被映射到各自的 AXI4-Stream/FIFO 用户应用接口。AXI4-Stream/FIFO 接口只传递数据。

通道的类型配置决定了在哪个总线上进行事务传输。

- Host-to-Card（H2C）通道向 PCIe 产生读请求，然后给用户应用提供数据
- 同样，Card-to-Host（C2H）通道等待用户侧的数据，然后向 PCIe 产生写请求（包含接收到的数据）

Host 通过 AXI4-Lite Master 接口访问用户逻辑的配置和状态寄存器。这些请求是 32-bit 的读或写。

3.3.1 Target Bridge

Target bridge 从 Host 接收请求。基于 BARs，这些请求通过 AXI4-Lite Master 接口传递到内部的目标用户。下行用户逻辑返回 non-post 请求的数据后，target bridge 产生一个读返回 TLP，然后通过 CC 总线发送给 PCIe IP。

PCIe BARs 配置如下表所示。

表 19 32-bit BARs

BAR0(32-bit)	BAR1(32-bit)
DMA	PCIe to AXI4-Lite Master

3.3.2 H2C 通道

H2C 通道处理 host-to-card 的 DMA 传输。根据最大读请求大小和可用的内部资源，H2C 通道负责拆分这些读请求。读取请求的每个拆分（如果有）都会消耗一条额外的读取请求条目。DMA 通道向 PCIe RQ 模块发出读请求后，直到按顺序接收到用户接口的写完成确认，这个读请求才算完成。在数据传输完成后，DMA 通道向 Host 发出一个中断。

H2C 通道会根据配置的最大读请求大小和数据 FIFO 可用空间来拆分 Host 读接口上的数据传输。从 PCIe RC 模块输出的读请求返回完成数据包会存储到已分配的数据 FIFO 中。为了减少数据传输延时，一旦接收到任何一个完成数据包，H2C 通道就开始把读取的数据输出到 AXI4-Stream/FIFO Write 接口。

3.3.3 C2H 通道

C2H 通道处理 card-to-host 的 DMA 传输。在 AXI4-Stream/FIFO Read 接口接收数据之前，C2H 通道首先接收 DMA 描述符，建立 DMA 传输的环境，然后在准备好请求 ID 和启用 C2H 通道后，通道的 AXI4-Stream/FIFO 接口可以接收数据并对主机执行 DMA。在数据传输完成后，DMA 通道向 Host 发出一个中断。

3.3.4 AXI4-Lite Master

Host 使用这个接口向用户逻辑发出 32-bit 读和 32-bit 写请求。通过 PCIe 总线接收读或者写请求，路由到 AXI4-Lite Master BAR 中，target bridge 通过 PCIe IP CC 总线返回读取的完成数据。

3.3.5 IRQ Module

IRQ module 接收来自用户逻辑中断请求和每个 DMA 通道的独立中断线。这个模块通过 PCIe 向 Host 产生 MSI 中断。

3.3.6 DMA 操作

从 DMA 的原理出发，PCIe DMA 引擎通常在 Host 内存和驻留在 FPGA 中的内存之间搬移数据，FPGA 的内存通常（但不总是）位于 add-in 板卡。当数据从 Host 内存搬移到 FPGA 内存时，称为 Host to Card（H2C）传输。相反，当数据从 FPGA 内存搬移到 Host 内存，称为 Card to Host（C2H）传输。

在通常的操作中，Host 中的应用程序必须在 FPGA 和 Host 内存之间搬移数据。为了完成此次传输，Host 在系统内存中设置缓冲区空间，并创建 DMA 引擎用于搬移数据的描述符。

Multi-Channel PCIe QDMA&RDMA Subsystem 使用连续的描述符链表，单个描述符指定了地址和 DMA 传输长度。Host 驱动创建描述符链表，并存储在 Host 内存。Host 驱动只需要少量控制寄存器就可以初始化 DMA 通道，接着该 DMA 通道就开始去获取描述符链表并执行 DMA 操作。

描述符描述了 Multi-Channel PCIe QDMA&RDMA Subsystem 执行的内存传输。每个 DMA 通道有自己的描述符链表。Host 驱动通过硬件寄存器初始化每个 DMA 通道描述符链表的起始地址。在启动 DMA 通道后，该 DMA 通道开始从起始地址获取描述符。

EOP 控制比特显示描述符链表的终止。当 DMA 通道检测到某个描述符的 EOP 控制比特后，会停止获取该描述符链表。EOP 控制比特只能在描述符链表的最后一个描述符中被设置。

描述符格式如下表所示。

表 20 描述符格式

Offset	Fields
0x0	Bit 31: EOP Bit 30~16: 保留 Bit 15~0: Magic, 常数 16'hAD4B
0x4	DMA Transfer Length
0x8	Address[31:0]
0xC	Address[63:32]

Address: 64-bit, Destination address for C2H 或 Source address for H2C

EOP: 1-bit, End of packet for stream data

DMA Transfer Length: 32-bit, Length of data in bytes

3.4 端口描述

Multi-Channel PCIe QDMA&RDMA Subsystem 直接和 integrated block for PCIe 连接。和 PCIe integrated block IP 数据路径接口的宽度是 64, 128, 256 或 512-bit，时钟频率最高可达 250MHz。除了 AXI4-Lite Master 接口，数据路径宽度适用于所有数据接口。AXI4-Lite Master 接口的宽度固定为 32-bit。

以下列表描述了这个 IP 的端口（PCIe 接口是 x16）。

表 21 参数定义

参数名称	描述	默认值
CNUM	H2C 和 C2H 的通道数量 1, 4, 8	8
DATW	数据路径的数据宽度 64, 128, 256, 512	256
R_OUTSTANDING	Read Request Outstanding Number 8, 16, 32	16
C2H_BUF_USE_BRAM	C2H 数据缓冲区是否使用 BRAM 0: 不使用 URAM 1: 使用 URAM	1
C2H_BUF_BRAM_CASCADE_DEPTH	C2H 数据缓冲区的 BRAM 级联深度 0: BRAM 深度= $2^9=512$ 1: BRAM 深度= $2^{9+1}=1024$ N: BRAM 深度= 2^{9+N}	0
C2H_BUF_USE_URAM	C2H 数据缓冲区是否使用 URAM 0: 不使用 URAM 1: 使用 URAM	0
C2H_BUF_URAM_CASCADE_DEPTH	H2C 数据缓冲区的 URAM 级联深度 0: URAM 深度= $2^{12}=4096$ 1: URAM 深度= $2^{12+1}=8192$ N: URAM 深度= 2^{12+N}	0
H2C_BUF_BRAM_CASCADE_DEPTH	H2C 数据缓冲区的 BRAM 级联深度 0: BRAM 深度= $2^9=512$ 1: BRAM 深度= $2^{9+1}=1024$ N: BRAM 深度= 2^{9+N}	0
H2C_BUF_USE_URAM	H2C 数据缓冲区是否使用 URAM	0

	0: 不使用 URAM 1: 使用 URAM	
H2C_BUF_URAM_CASCADE_DEPTH	C2H 数据缓冲区的 URAM 级联深度 0: URAM 深度= $2^{12}=4096$ 1: URAM 深度= $2^{12+1}=8192$ N: URAM 深度= 2^{12+N}	0

表 22 顶层接口信号

信号名称	方向	描述
pcie_trn_clk	输出	PCI Express Transaction 接口时钟
pcie_trn_reset_n	输出	PCI Express Transaction 复位，低有效
trn_lnk_up	输出	PCI Express Transaction Link Up 信号，高有效

表 23 PCIe 接口信号

信号名称	方向	描述
pcie_refclk_p	输入	PCI Express 接口参考时钟+
pcie_refclk_n	输入	PCI Express 接口参考时钟-
pcie_perst_n	输入	PCI Express 接口基本复位，低有效
pci_exp_txp[15:0]	输出	PCI Express 串行差分输出+，16 通道
pci_exp_txn[15:0]	输出	PCI Express 串行差分输出-，16 通道
pci_exp_rxp[15:0]	输入	PCI Express 串行差分输入+，16 通道
pci_exp_rxn[15:0]	输入	PCI Express 串行差分输入-，16 通道

表 24 H2C 通道 0-CNUM-1 FIFO (FWFT) 接口信号

信号名称	方向	描述
fifo_rdclk_disp [CNUM-1:0]	输入	FIFO 读时钟 Bit i, 表示 H2C 通道 i 的读时钟
fifo_rdrstn_disp [CNUM-1:0]	输入	FIFO 读复位，低有效 Bit i, 表示 H2C 通道 i 的读复位
fifo_rdreq_disp [CNUM-1:0]	输入	FIFO 读使能，高有效 Bit i, 表示 H2C 通道 i 的读使能
fifo_q_disp [DATW*CNUM-1:0]	输出	FIFO 读数据 Bit DATW*(i+1)-1~DATW*i, 表示 H2C 通道 i 的读数据
fifo_empty_disp [CNUM-1:0]	输出	FIFO 空，高有效 Bit i, 表示 H2C 通道 i 的缓存空信号

fifo_prog_empty_disp [CNUM-1:0]	输出	FIFO 可编程空 (阈值等于 16), 高有效 Bit i, 表示 H2C 通道 i 的缓存可编程空信号
------------------------------------	----	----------------------------------------------------------

表 25 C2H 通道 0-CNUM-1 FIFO 接口信号

信号名称	方向	描述
fifo_wrclk_acq [CNUM-1:0]	输入	FIFO 写时钟 Bit i, 表示 C2H 通道 i 的写时钟
fifo_wrrstn_acq [CNUM-1:0]	输入	FIFO 写复位, 低有效 Bit i, 表示 C2H 通道 i 的写复位
fifo_wrreq_acq [CNUM-1:0]	输入	FIFO 写使能, 高有效 Bit i, 表示 C2H 通道 i 的写使能
fifo_data_acq [DATW*CNUM-1:0]	输入	FIFO 写数据 Bit DATW*(i+1)-1~DATW*i, 表示 C2H 通道 i 的写数据
fifo_prog_full_acq [CNUM-1:0]	输出	FIFO 可编程满 (阈值等于深度-16), 高有效 Bit i, 表示 C2H 通道 i 的缓存可编程满信号

表 26 Config AXI4-Lite Master 接口信号

信号名称	方向	描述
m_axil_awaddr[31:0]	输出	write address
m_axil_awprot[2:0]	输出	write protection type
m_axil_awvalid	输出	write address valid
m_axil_awready	输入	write address ready
m_axil_wdata[31:0]	输出	write data
m_axil_wstrb[3:0]	输入	write strobes
m_axil_wvalid	输出	write valid
m_axil_wready	输入	write ready
m_axil_bresp[1:0]	输入	write response
m_axil_bvalid	输入	write response valid
m_axil_bready	输出	response ready
m_axil_araddr[31:0]	输出	read address
m_axil_arprot[2:0]	输出	read protection type
m_axil_arvalid	输出	read address valid
m_axil_arready	输入	read address ready
m_axil_rdata[31:0]	输入	read data
m_axil_rresp[1:0]	输入	read response
m_axil_rvalid	输入	read valid
m_axil_rready	输出	read ready

表 27 中断接口信号

信号名称	方向	描述
------	----	----

usr_intr_pos[15:0]	输入	用户中断输入。 Bit i: 用户中断#i 输入，上升沿有效。
--------------------	----	------------------------------------

表 28 软复位接口信号

信号名称	方向	描述
c2h_dma_grst_n	输出	C2H DMA 全局复位输出，低有效。
c2h_dma_fsm_srst_n [CNUM-1:0]	输出	C2H 通道 i 的 DMA FSM 复位输出，低有效。
c2h_dma_buf_srst_n [CNUM-1:0]	输出	C2H 通道 i 的 DMA Buffer 复位输出，低有效。
h2c_dma_grst_n	输出	H2C DMA 全局复位输出，低有效。
h2c_dma_fsm_srst_n [CNUM-1:0]	输出	H2C 通道 i 的 DMA FSM 复位输出，低有效。
h2c_dma_buf_srst_n [CNUM-1:0]	输出	H2C 通道 i 的 DMA Buffer 复位输出，低有效。

表 29 用户寄存器接口信号

信号名称	方向	描述
acquisition_stat [32*CNUM-1:0]	输入	CNUM 个 C2H 通道状态 Bit 32*(i+1)-1~32*i, 表示 C2H 通道 i 的状态
display_stat [32*CNUM-1:0]	输入	CNUM 个 H2C 通道状态 Bit 32*(i+1)-1~32*i, 表示 H2C 通道 i 的状态
acquisition_res_vld [CNUM-1:0]	输出	CNUM 个 C2H 通道的分辨率设置有效，高有效 Bit i = 1, 表示 C2H 通道 i 分辨率设置有效
display_res_vld [CNUM-1:0]	输出	CNUM 个 H2C 通道的分辨率设置有效，高有效 Bit i = 1, 表示 H2C 通道 i 分辨率设置有效
acquisition_fps_vld [CNUM-1:0]	输出	CNUM 个 C2H 通道的 FPS 设置有效，高有效 Bit i = 1, 表示 C2H 通道 i FPS 设置有效
display_fps_vld [CNUM-1:0]	输出	CNUM 个 H2C 通道的 FPS 设置有效，高有效 Bit i = 1, 表示 H2C 通道 i FPS 设置有效
acquisition_res [32*CNUM-1:0]	输出	CNUM 个 C2H 通道的分辨率设置 Bit 32*(i+1)-1~32*i, 表示 C2H 通道 i 的分辨率
display_res [32*CNUM-1:0]	输出	CNUM 个 H2C 通道的分辨率设置 Bit 32*(i+1)-1~32*i, 表示 H2C 通道 i 的分辨率
acquisition_fps [32*CNUM-1:0]	输出	CNUM 个 C2H 通道的帧率设置 Bit 32*(i+1)-1~32*i, 表示 C2H 通道 i 的帧率
display_fps [32*CNUM-1:0]	输出	CNUM 个 H2C 通道的帧率设置 Bit 32*(i+1)-1~32*i, 表示 H2C 通道 i 的帧率
acquisition_xfer_size [32*CNUM-1:0]	输出	CNUM 个 C2H 通道的传输大小设置 Bit 32*(i+1)-1~32*i, 表示 C2H 通道 i 的传输长度
display_xfer_size [32*CNUM-1:0]	输出	CNUM 个 H2C 通道的传输大小设置 Bit 32*(i+1)-1~32*i, 表示 H2C 通道 i 的传输长度

acquisition_xfer_num [32*CNUM-1:0]	输出	CNUM 个 C2H 通道的传输总数设置 Bit 32*(i+1)-1~32*i, 表示 C2H 通道 i 的传输总数
display_xfer_num [32*CNUM-1:0]	输出	CNUM 个 H2C 通道的传输总数设置 Bit 32*(i+1)-1~32*i, 表示 H2C 通道 i 的传输总数
acquisition_enable [CNUM-1:0]	输出	CNUM 个 C2H 通道的采集使能, 高有效 Bit i = 1, 表示使能 C2H 通道 i 采集
display_enable [CNUM-1:0]	输出	CNUM 个 H2C 通道的显示使能, 高有效 Bit i = 1, 表示使能 H2C 通道 i 显示
display_timing_enable [CNUM-1:0]	输出	CNUM 个 H2C 通道的内部显示定时使能, 高有效。 Bit i = 1, 表示使能 H2C 通道 i 内部显示定时
display_timing_ext_enable [CNUM-1:0]	输出	CNUM 个 H2C 通道的外部显示定时使能, 高有效。 Bit i = 1, 表示使能 H2C 通道 i 外部显示定时
acq_usr_reset[CNUM-1:0]	输出	CNUM 个 C2H 通道的用户复位信号 Bit i, 表示 C2H 通道 i 用户复位信号
disp_usr_reset[CNUM-1:0]	输出	CNUM 个 H2C 通道的用户复位信号 Bit i, 表示 H2C 通道 i 用户复位信号
usr_ctrl[31:0]	输出	用户控制信号输出
usr_ctrl2[31:0]	输出	用户控制#2 信号输出
usr_stat[31:0]	输入	用户状态信号输入
acq_blk_baddr_l[32*CNUM-1:0]	输出	CNUM 个 VID-ACQ 数据块基址 LSB Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的数据块基址
disp_blk_baddr_l[32*CNUM-1:0]	输出	CNUM 个 VID-DISP 数据块基址 LSB Bit 32*(i+1)-1~32*i, 表示 DISP#i 的数据块基址
acq_blk_baddr_h[32*CNUM-1:0]	输出	CNUM 个 VID-ACQ 数据块基址 MSB Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的数据块基址
disp_blk_baddr_h[32*CNUM-1:0]	输出	CNUM 个 VID-DISP 数据块基址 MSB Bit 32*(i+1)-1~32*i, 表示 DISP#i 的数据块基址
acq_blk_size[32*CNUM-1:0]	输出	CNUM 个 VID-ACQ 数据块大小 Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的数据块总数
disp_blk_size[32*CNUM-1:0]	输出	CNUM 个 VID-DISP 数据块大小 Bit 32*(i+1)-1~32*i, 表示 DISP#i 的数据块总数
acq_blk_num[32*CNUM-1:0]	输出	CNUM 个 VID-ACQ 数据块总数 Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的数据块大小
disp_blk_num[32*CNUM-1:0]	输出	CNUM 个 VID-DISP 数据块总数 Bit 32*(i+1)-1~32*i, 表示 DISP#i 的数据块大小
acq_usr_param [32*CNUM-1:0]	输出	CNUM 个 VID-ACQ 用户参数 Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的用户参数
disp_usr_param [32*CNUM-1:0]	输出	CNUM 个 VID-DISP 用户参数 Bit 32*(i+1)-1~32*i, 表示 DISP#i 的用户参数
acq_pkt_num [32*CNUM-1:0]	输入	CNUM 个 VID-ACQ 数据包总数 Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的数据包总数
disp_pkt_num [32*CNUM-1:0]	输入	CNUM 个 VID-DISP 数据包总数 Bit 32*(i+1)-1~32*i, 表示 DISP#i 的数据包总数
acq_dsc_pkt_num	输入	CNUM 个 VID-ACQ 丢弃数据包总数

[32*CNUM-1:0]		Bit 32*(i+1)-1~32*i, 表示 ACQ#i 的丢弃数据包总数
disp_dpl_pkt_num [32*CNUM-1:0]	输入	CNUM 个 VID-DISP 复制数据包总数 Bit 32*(i+1)-1~32*i, 表示 DISP#i 的复制数据包总数

表 30 Display FPS Timing 通道 0-CNUM-1 接口信号

信号名称	方向	描述
fifo_wrclk_disp_fps [CNUM-1:0]	输入	FIFO 写时钟 Bit i, 表示 Display FPS Timing 通道 i 的写时钟
fifo_wrrstn_disp_fps [CNUM-1:0]	输入	FIFO 写复位, 低有效 Bit i, 表示 Display FPS Timing 通道 i 的写复位
fifo_wrrreq_disp_fps [CNUM-1:0]	输入	FIFO 写使能, 高有效 Bit i, 表示 Display FPS Timing 通道 i 的写使能
fifo_prog_full_disp_fps [CNUM-1:0]	输出	FIFO 可编程满, 高有效 Bit i, 表示 Display FPS Timing 通道 i 的缓存可编程满信号

3.5 寄存器空间

Host 可以通过将读或写请求映射到 BAR 方式访问 Multi-Channel PCIe QDMA&RDMA Subsystem 内部的配置和状态寄存器以及用户逻辑中的配置和状态寄存器。根据 BAR 的命中, 读写请求被路由到相应的位置上。对于 PCIe BAR 分配, 请参考 Target Bridge。

3.5.1 PCIe to AXI4-Lite Master (BAR1) 地址映射

命中 PCIe to AXI4-Lite Master 空间的事务被路由到 AXI4-Lite Memory Mapped 用户接口上。这个接口支持 32-bit 地址空间和 32-bit 读写请求。PCIe to AXI4-Lite Master 的地址映射由用户逻辑定义。

3.5.2 PCIe to DMA (BAR0) 地址映射

命中 PCIe to DMA 空间的事务被路由到 Multi-Channel PCIe QDMA&RDMA Subsystem 内部的配置寄存器总线。这个接口支持 32-bit 地址空间和 32-bit 读写请求。这些寄存器应用于 DMA 编程和状态检查。

表 31 PCIe to DMA 地址映射

名称	位宽	偏移	描述	复位值
CH0_C2H_ADDR_L	32	0x000	通道 0 C2H 目的地址 Lower, RW。 SGDMA 和 RDMA 模式下, 表示 DMA 描述符基地址。 CDMA 模式下, 表示 DMA 缓冲区基地址。	0x00000000

CH1_C2H_ADDR_L	32	0x004	通道 1 C2H 目的地址 Lower, RW。	0x00000000
CH2_C2H_ADDR_L	32	0x008	通道 2 C2H 目的地址 Lower, RW。	0x00000000
CH3_C2H_ADDR_L	32	0x00C	通道 3 C2H 目的地址 Lower, RW。	0x00000000
CH4_C2H_ADDR_L	32	0x010	通道 4 C2H 目的地址 Lower, RW。	0x00000000
CH5_C2H_ADDR_L	32	0x014	通道 5 C2H 目的地址 Lower, RW。	0x00000000
CH6_C2H_ADDR_L	32	0x018	通道 6 C2H 目的地址 Lower, RW。	0x00000000
CH7_C2H_ADDR_L	32	0x01C	通道 7 C2H 目的地址 Lower, RW。	0x00000000
CH0_C2H_ADDR_U	32	0x020	通道 0 C2H 目的地址 Upper, RW。 SGDMA 和 RDMA 模式下, 表示 DMA 描述符基地址。 CDMA 模式下, 表示 DMA 缓冲区基地址。	0x00000000
CH1_C2H_ADDR_U	32	0x024	通道 1 C2H 目的地址 Upper, RW。	0x00000000
CH2_C2H_ADDR_U	32	0x028	通道 2 C2H 目的地址 Upper, RW。	0x00000000
CH3_C2H_ADDR_U	32	0x02C	通道 3 C2H 目的地址 Upper, RW。	0x00000000
CH4_C2H_ADDR_U	32	0x030	通道 4 C2H 目的地址 Upper, RW。	0x00000000
CH5_C2H_ADDR_U	32	0x034	通道 5 C2H 目的地址 Upper, RW。	0x00000000
CH6_C2H_ADDR_U	32	0x038	通道 6 C2H 目的地址 Upper, RW。	0x00000000
CH7_C2H_ADDR_U	32	0x03C	通道 7 C2H 目的地址 Upper, RW。	0x00000000
CH0_H2C_ADDR_L	32	0x040	通道 0 H2C 源地址 Lower, RW。 SGDMA 和 RDMA 模式下, 表示 DMA 描述符基地址。 CDMA 模式下, 表示 DMA 缓冲区基地址。	0x00000000
CH1_H2C_ADDR_L	32	0x044	通道 1 H2C 源地址 Lower, RW。	0x00000000
CH2_H2C_ADDR_L	32	0x048	通道 2 H2C 源地址 Lower, RW。	0x00000000
CH3_H2C_ADDR_L	32	0x04C	通道 3 H2C 源地址 Lower, RW。	0x00000000
CH4_H2C_ADDR_L	32	0x050	通道 4 H2C 源地址 Lower, RW。	0x00000000
CH5_H2C_ADDR_L	32	0x054	通道 5 H2C 源地址 Lower, RW。	0x00000000
CH6_H2C_ADDR_L	32	0x058	通道 6 H2C 源地址 Lower, RW。	0x00000000
CH7_H2C_ADDR_L	32	0x05C	通道 7 H2C 源地址 Lower, RW。	0x00000000
CH0_H2C_ADDR_U	32	0x060	通道 0 H2C 源地址 Upper, RW。 SGDMA 和 RDMA 模式下, 表示 DMA 描述符基地址。 CDMA 模式下, 表示 DMA 缓冲区基地址。	0x00000000
CH1_H2C_ADDR_U	32	0x064	通道 1 H2C 源地址 Upper, RW。	0x00000000
CH2_H2C_ADDR_U	32	0x068	通道 2 H2C 源地址 Upper, RW。	0x00000000
CH3_H2C_ADDR_U	32	0x06C	通道 3 H2C 源地址 Upper, RW。	0x00000000
CH4_H2C_ADDR_U	32	0x070	通道 4 H2C 源地址 Upper, RW。	0x00000000
CH5_H2C_ADDR_U	32	0x074	通道 5 H2C 源地址 Upper, RW。	0x00000000
CH6_H2C_ADDR_U	32	0x078	通道 6 H2C 源地址 Upper, RW。	0x00000000
CH7_H2C_ADDR_U	32	0x07C	通道 7 H2C 源地址 Upper, RW。	0x00000000
CH0_C2H_XFER_SIZE	32	0x080	通道 0 C2H 传输长度, RW。 SGDMA 模式下, 不使用。 CDMA 模式下, 表示 DMA 传输长度。 RDMA 模式下: Ring Buffer 尾指针寄存器 (CPU 控制), WO。 Ring Buffer 头指针寄存器 (FPGA 控制), RO。	0x00000000

CH1_C2H_XFER_SIZE	32	0x084	通道 1 C2H 传输长度, RW。	0x00000000
CH2_C2H_XFER_SIZE	32	0x088	通道 2 C2H 传输长度, RW。	0x00000000
CH3_C2H_XFER_SIZE	32	0x08C	通道 3 C2H 传输长度, RW。	0x00000000
CH4_C2H_XFER_SIZE	32	0x090	通道 4 C2H 传输长度, RW。	0x00000000
CH5_C2H_XFER_SIZE	32	0x094	通道 5 C2H 传输长度, RW。	0x00000000
CH6_C2H_XFER_SIZE	32	0x098	通道 6 C2H 传输长度, RW。	0x00000000
CH7_C2H_XFER_SIZE	32	0x09C	通道 7 C2H 传输长度, RW。	0x00000000
CH0_H2C_XFER_SIZE	32	0x0A0	通道 0 H2C 传输长度, RW。 SGDMA 模式下, 不使用。 CDMA 模式下, 表示 DMA 传输长度。 RDMA 模式下: Ring Buffer 尾指针寄存器 (CPU 控制), WO。 Ring Buffer 头指针寄存器 (FPGA 控制), RO。	0x00000000
CH1_H2C_XFER_SIZE	32	0x0A4	通道 1 H2C 传输长度, RW。	0x00000000
CH2_H2C_XFER_SIZE	32	0x0A8	通道 2 H2C 传输长度, RW。	0x00000000
CH3_H2C_XFER_SIZE	32	0x0AC	通道 3 H2C 传输长度, RW。	0x00000000
CH4_H2C_XFER_SIZE	32	0x0B0	通道 4 H2C 传输长度, RW。	0x00000000
CH5_H2C_XFER_SIZE	32	0x0B4	通道 5 H2C 传输长度, RW。	0x00000000
CH6_H2C_XFER_SIZE	32	0x0B8	通道 6 H2C 传输长度, RW。	0x00000000
CH7_H2C_XFER_SIZE	32	0x0BC	通道 7 H2C 传输长度, RW。	0x00000000
CH0_C2H_FPS	32	0x0C0	通道 0 C2H 帧率, RW。	0x00000000
CH1_C2H_FPS	32	0x0C4	通道 1 C2H 帧率, RW。	0x00000000
CH2_C2H_FPS	32	0x0C8	通道 2 C2H 帧率, RW。	0x00000000
CH3_C2H_FPS	32	0x0CC	通道 3 C2H 帧率, RW。	0x00000000
CH4_C2H_FPS	32	0x0D0	通道 4 C2H 帧率, RW。	0x00000000
CH5_C2H_FPS	32	0x0D4	通道 5 C2H 帧率, RW。	0x00000000
CH6_C2H_FPS	32	0x0D8	通道 6 C2H 帧率, RW。	0x00000000
CH7_C2H_FPS	32	0x0DC	通道 7 C2H 帧率, RW。	0x00000000
CH0_H2C_FPS	32	0x0E0	通道 0 H2C 帧率, RW。	0x00000000
CH1_H2C_FPS	32	0x0E4	通道 1 H2C 帧率, RW。	0x00000000
CH2_H2C_FPS	32	0x0E8	通道 2 H2C 帧率, RW。	0x00000000
CH3_H2C_FPS	32	0x0EC	通道 3 H2C 帧率, RW。	0x00000000
CH4_H2C_FPS	32	0x0F0	通道 4 H2C 帧率, RW。	0x00000000
CH5_H2C_FPS	32	0x0F4	通道 5 H2C 帧率, RW。	0x00000000
CH6_H2C_FPS	32	0x0F8	通道 6 H2C 帧率, RW。	0x00000000
CH7_H2C_FPS	32	0x0FC	通道 7 H2C 帧率, RW。	0x00000000
CH0_C2H_CTRL	32	0x100	通道 0 C2H 控制, RW。 Bit 0: 设置为 '1' 启动 C2H-DMA 引擎, '0' 停止 C2H-DMA 引擎 Bit 1: 设置为 '1' 开始 C2H 传输, '0' 停止 C2H 传输 Bit 3~2: 保留 Bit 4: Reserved Bit 5: 设置为 '1' 使能 C2H-DMA 中断间隔 (过滤) 控制,	0x00000000

			‘0’ 禁止 C2H-DMA 中断间隔（过滤）控制；适用于 RDMA Bit 6: 设置为 1, RDMA Ring Buffer 满的时候采用覆盖策略 Bit 31: 设置为 ‘1’ 复位 C2H DMA	
CH1_C2H_CTRL	32	0x104	通道 1 C2H 控制, RW。	0x00000000
CH2_C2H_CTRL	32	0x108	通道 2 C2H 控制, RW。	0x00000000
CH3_C2H_CTRL	32	0x10C	通道 3 C2H 控制, RW。	0x00000000
CH4_C2H_CTRL	32	0x110	通道 4 C2H 控制, RW。	0x00000000
CH5_C2H_CTRL	32	0x114	通道 5 C2H 控制, RW。	0x00000000
CH6_C2H_CTRL	32	0x118	通道 6 C2H 控制, RW。	0x00000000
CH7_C2H_CTRL	32	0x11C	通道 7 C2H 控制, RW。	0x00000000
CH0_H2C_CTRL	32	0x120	通道 0 H2C 控制, RW。 Bit 0: 设置为 ‘1’ 启动 H2C-DMA 引擎, ‘0’ 停止 H2C-DMA 引擎 Bit 1: 设置为 ‘1’ 开始 H2C 传输, ‘0’ 停止 H2C 传输 Bit 2: 设置为 ‘1’ 使能内部显示定时, ‘0’ 禁止内部显示定时 Bit 3: 设置为 ‘1’ 使能外部显示定时, ‘0’ 禁止外部显示定时 Bit 4: Reserved Bit 5: 设置为 ‘1’ 使能 H2C-DMA 中断间隔（过滤）控制, ‘0’ 禁止 H2C-DMA 中断间隔（过滤）控制；适用于 RDMA Bit 31: 设置为 ‘1’ 复位 H2C DMA	0x00000000
CH1_H2C_CTRL	32	0x124	通道 1 H2C 控制, RW。	0x00000000
CH2_H2C_CTRL	32	0x128	通道 2 H2C 控制, RW。	0x00000000
CH3_H2C_CTRL	32	0x12C	通道 3 H2C 控制, RW。	0x00000000
CH4_H2C_CTRL	32	0x130	通道 4 H2C 控制, RW。	0x00000000
CH5_H2C_CTRL	32	0x134	通道 5 H2C 控制, RW。	0x00000000
CH6_H2C_CTRL	32	0x138	通道 6 H2C 控制, RW。	0x00000000
CH7_H2C_CTRL	32	0x13C	通道 7 H2C 控制, RW。	0x00000000
CH0_C2H_STAT	32	0x140	通道 0 C2H 状态, RO。	0x00000000
CH1_C2H_STAT	32	0x144	通道 1 C2H 状态, RO。	0x00000000
CH2_C2H_STAT	32	0x148	通道 2 C2H 状态, RO。	0x00000000
CH3_C2H_STAT	32	0x14C	通道 3 C2H 状态, RO。	0x00000000
CH4_C2H_STAT	32	0x150	通道 4 C2H 状态, RO。	0x00000000
CH5_C2H_STAT	32	0x154	通道 5 C2H 状态, RO。	0x00000000
CH6_C2H_STAT	32	0x158	通道 6 C2H 状态, RO。	0x00000000
CH7_C2H_STAT	32	0x15C	通道 7 C2H 状态, RO。	0x00000000
CH0_H2C_STAT	32	0x160	通道 0 H2C 状态, RO。	0x00000000
CH1_H2C_STAT	32	0x164	通道 1 H2C 状态, RO。	0x00000000
CH2_H2C_STAT	32	0x168	通道 2 H2C 状态, RO。	0x00000000
CH3_H2C_STAT	32	0x16C	通道 3 H2C 状态, RO。	0x00000000
CH4_H2C_STAT	32	0x170	通道 4 H2C 状态, RO。	0x00000000
CH5_H2C_STAT	32	0x174	通道 5 H2C 状态, RO。	0x00000000

CH6_H2C_STAT	32	0x178	通道 6 H2C 状态, RO。	0x00000000
CH7_H2C_STAT	32	0x17C	通道 7 H2C 状态, RO。	0x00000000
CH0_C2H_RES	32	0x180	通道 0 C2H 分辨率, RW。	0x00000000
CH1_C2H_RES	32	0x184	通道 1 C2H 分辨率, RW。	0x00000000
CH2_C2H_RES	32	0x188	通道 2 C2H 分辨率, RW。	0x00000000
CH3_C2H_RES	32	0x18C	通道 3 C2H 分辨率, RW。	0x00000000
CH4_C2H_RES	32	0x190	通道 4 C2H 分辨率, RW。	0x00000000
CH5_C2H_RES	32	0x194	通道 5 C2H 分辨率, RW。	0x00000000
CH6_C2H_RES	32	0x198	通道 6 C2H 分辨率, RW。	0x00000000
CH7_C2H_RES	32	0x19C	通道 7 C2H 分辨率, RW。	0x00000000
CH0_H2C_RES	32	0x1A0	通道 0 H2C 分辨率, RW。	0x00000000
CH1_H2C_RES	32	0x1A4	通道 1 H2C 分辨率, RW。	0x00000000
CH2_H2C_RES	32	0x1A8	通道 2 H2C 分辨率, RW。	0x00000000
CH3_H2C_RES	32	0x1AC	通道 3 H2C 分辨率, RW。	0x00000000
CH4_H2C_RES	32	0x1B0	通道 4 H2C 分辨率, RW。	0x00000000
CH5_H2C_RES	32	0x1B4	通道 5 H2C 分辨率, RW。	0x00000000
CH6_H2C_RES	32	0x1B8	通道 6 H2C 分辨率, RW。	0x00000000
CH7_H2C_RES	32	0x1BC	通道 7 H2C 分辨率, RW。	0x00000000
CH0_INT_GAP	32	0x1C0	通道 0 中断间隔数值, RW。 Bit 15~0: C2H 中断间隔数值 (实际数值减 1), 以 us 为单位。 Bit 32~16: H2C 中断间隔数值 (实际数值减 1), 以 us 为单位。	0x00000000
CH1_INT_GAP	32	0x1C4	通道 1 中断间隔数值, RW。	0x00000000
CH2_INT_GAP	32	0x1C8	通道 2 中断间隔数值, RW。	0x00000000
CH3_INT_GAP	32	0x1CC	通道 3 中断间隔数值, RW。	0x00000000
CH4_INT_GAP	32	0x1D0	通道 4 中断间隔数值, RW。	0x00000000
CH5_INT_GAP	32	0x1D4	通道 5 中断间隔数值, RW。	0x00000000
CH6_INT_GAP	32	0x1D8	通道 6 中断间隔数值, RW。	0x00000000
CH7_INT_GAP	32	0x1DC	通道 7 中断间隔数值, RW。	0x00000000
CTRL	32	0x1E0	控制, RW。	0x00000000
CTRL2	32	0x1E4	控制 2, RW。	0x00000000
SRST	32	0x1E8	软复位, RW。 Bit 0: 设置为 '1' 复位所有 C2H DMA Bit 1: 设置为 '1' 复位所有 H2C DMA Bit 16: 设置为 '1' 复位 C2H#0 用户逻辑 Bit 17: 设置为 '1' 复位 C2H#1 用户逻辑 Bit 18: 设置为 '1' 复位 C2H#2 用户逻辑 Bit 19: 设置为 '1' 复位 C2H#3 用户逻辑 Bit 20: 设置为 '1' 复位 C2H#4 用户逻辑 Bit 21: 设置为 '1' 复位 C2H#5 用户逻辑 Bit 22: 设置为 '1' 复位 C2H#6 用户逻辑 Bit 23: 设置为 '1' 复位 C2H#7 用户逻辑	0x00000000

			Bit 24: 设置为 ‘1’ 复位 H2C#0 用户逻辑 Bit 25: 设置为 ‘1’ 复位 H2C#1 用户逻辑 Bit 26: 设置为 ‘1’ 复位 H2C#2 用户逻辑 Bit 27: 设置为 ‘1’ 复位 H2C#3 用户逻辑 Bit 28: 设置为 ‘1’ 复位 H2C#4 用户逻辑 Bit 29: 设置为 ‘1’ 复位 H2C#5 用户逻辑 Bit 31: 设置为 ‘1’ 复位 H2C#6 用户逻辑 Bit 24: 设置为 ‘1’ 复位 H2C#7 用户逻辑	
INT_MASK	32	0x1EC	中断屏蔽，RW。 Bit 0: 设置为 ‘0’ 使能 CH0 C2H 中断 Bit 1: 设置为 ‘0’ 使能 CH1 C2H 中断 Bit 2: 设置为 ‘0’ 使能 CH2 C2H 中断 Bit 3: 设置为 ‘0’ 使能 CH3 C2H 中断 Bit 4: 设置为 ‘0’ 使能 CH4 C2H 中断 Bit 5: 设置为 ‘0’ 使能 CH5 C2H 中断 Bit 6: 设置为 ‘0’ 使能 CH6 C2H 中断 Bit 7: 设置为 ‘0’ 使能 CH7 C2H 中断 Bit 8: 设置为 ‘0’ 使能 CH0 H2C 中断 Bit 9: 设置为 ‘0’ 使能 CH1 H2C 中断 Bit 10: 设置为 ‘0’ 使能 CH2 H2C 中断 Bit 11: 设置为 ‘0’ 使能 CH3 H2C 中断 Bit 12: 设置为 ‘0’ 使能 CH4 H2C 中断 Bit 13: 设置为 ‘0’ 使能 CH5 H2C 中断 Bit 14: 设置为 ‘0’ 使能 CH6 H2C 中断 Bit 15: 设置为 ‘0’ 使能 CH7 H2C 中断 Bit 16: 设置为 ‘0’ 使能 User#0 中断 Bit 17: 设置为 ‘0’ 使能 User#1 中断 Bit 18: 设置为 ‘0’ 使能 User#2 中断 Bit 19: 设置为 ‘0’ 使能 User#3 中断 Bit 20: 设置为 ‘0’ 使能 User#4 中断 Bit 21: 设置为 ‘0’ 使能 User#5 中断 Bit 22: 设置为 ‘0’ 使能 User#6 中断 Bit 23: 设置为 ‘0’ 使能 User#7 中断 Bit 24: 设置为 ‘0’ 使能 User#8 中断 Bit 25: 设置为 ‘0’ 使能 User#9 中断 Bit 26: 设置为 ‘0’ 使能 User#10 中断 Bit 27: 设置为 ‘0’ 使能 User#11 中断 Bit 28: 设置为 ‘0’ 使能 User#12 中断 Bit 29: 设置为 ‘0’ 使能 User#13 中断 Bit 30: 设置为 ‘0’ 使能 User#14 中断 Bit 31: 设置为 ‘0’ 使能 User#15 中断	0xFFFFFFFF F
INT_STAT	32	0x1F0	中断状态，RW1C。 Bit 0: ‘1’ 表示 CH0 C2H 中断 Bit 1: ‘1’ 表示 CH1 C2H 中断	0x00000000

			Bit 2: '1' 表示 CH2 C2H 中断 Bit 3: '1' 表示 CH3 C2H 中断 Bit 4: '1' 表示 CH4 C2H 中断 Bit 5: '1' 表示 CH5 C2H 中断 Bit 6: '1' 表示 CH6 C2H 中断 Bit 7: '1' 表示 CH7 C2H 中断 Bit 8: '1' 表示 CH0 H2C 中断 Bit 9: '1' 表示 CH1 H2C 中断 Bit 10: '1' 表示 CH2 H2C 中断 Bit 11: '1' 表示 CH3 H2C 中断 Bit 12: '1' 表示 CH4 H2C 中断 Bit 13: '1' 表示 CH5 H2C 中断 Bit 14: '1' 表示 CH6 H2C 中断 Bit 15: '1' 表示 CH7 H2C 中断 Bit 16: '1' 表示 User#0 中断 Bit 17: '1' 表示 User#1 中断 Bit 18: '1' 表示 User#2 中断 Bit 19: '1' 表示 User#3 中断 Bit 20: '1' 表示 User#4 中断 Bit 21: '1' 表示 User#5 中断 Bit 22: '1' 表示 User#6 中断 Bit 23: '1' 表示 User#7 中断 Bit 24: '1' 表示 User#8 中断 Bit 25: '1' 表示 User#9 中断 Bit 26: '1' 表示 User#10 中断 Bit 27: '1' 表示 User#11 中断 Bit 28: '1' 表示 User#12 中断 Bit 29: '1' 表示 User#13 中断 Bit 30: '1' 表示 User#14 中断 Bit 31: '1' 表示 User#15 中断	
INT_DLY	32	0x1F4	中断延时, RW。单位: 4ns。	0x00000000
STAT	32	0x1F8	状态, RO。	
SCFG, EPS	32	0x1FC	System Config 寄存器, WO。 Bit 0: C2H Channel 绝对优先级使能, 高有效。 注: C2H#0 最高优先级, C2H#7 最低优先级 Bit 1: H2C Channel 绝对优先级使能, 高有效。 注: H2C#0 最高优先级, H2C#7 最低优先级 Endpoint 状态寄存器, RO。 Bit 0: Link Down 0b: Link is Up, 1b: Link is Down Bit 3~1: Link Status	

			<p>VU+, KU and V7:</p> <ul style="list-style-type: none"> 00b: No receivers detected 01b: Link training in progress 10b: Link up, DL initialization in progress 11b: Link up, DL initialization completed <p>K7:</p> <ul style="list-style-type: none"> 000b: L0 001b: PPM L1 010b: PPM L2/L3 Ready 011b: PM_PME 100b: in or transitioning to/from ASPM L0s 101b: transitioning to/from PPM L1 110b: transition to PPM L2/L3 Ready 111b: Reserved <p>Bit 7~4 : Current Link Speed</p> <p>VU+:</p> <ul style="list-style-type: none"> 00b: 2.5 GT/s PCI Express Link 01b: 5.0 GT/s PCI Express Link 10b: 8.0 GT/s PCI Express Link 11b: Reserved <p>KU and V7:</p> <ul style="list-style-type: none"> 001b: 2.5 GT/s PCI Express Link 010b: 5.0 GT/s PCI Express Link 100b: 8.0 GT/s PCI Express Link Other values are reserved <p>K7:</p> <ul style="list-style-type: none"> 01b: 2.5 GT/s PCI Express Link 10b: 5.0 GT/s PCI Express Link Other values are reserved <p>Bit 11~8 : Negotiated Link Width</p> <p>VU+:</p> <ul style="list-style-type: none"> 000b: x1 001b: x2 010b: x4 011b: x8 100b: x16 Other values are reserved <p>KU, V7 and K7:</p> <ul style="list-style-type: none"> 0001b: x1 0010b: x2 0100b: x4 1000b: x8 	
--	--	--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

			<p>Other values are reserved</p> <p>Bit 14~12 : Max Payload Size。 VU+: 00b: 128 byte 01b: 256 byte 10b: 512 byte 11b: 1024 byte</p> <p>KU, V7 and K7: 000b: 128 byte 001b: 256 byte 010b: 512 byte 011b: 1024 byte Other values are reserved</p> <p>Bit 17~15 : Max Read Request Size 000b: 128 byte 001b: 256 byte 010b: 512 byte 011b: 1024 byte 100b: 2048 byte 101b: 4096 byte Other values are reserved</p> <p>Bit 18 : IO 地址空间译码使能。 0b: 禁止 IO 译码 1b: 使能 IO 译码</p> <p>Bit 19 : Memory 地址空间译码使能。 0b: 禁止 Memory 译码 1b: 使能 Memory 译码</p> <p>Bit 20 : Master 使能。 0b: 禁止 Master 1b: 使能 Master</p> <p>Bit 21 : INTx 中断消息禁止。 0b: 使能 INTx 中断消息 1b: 禁止 INTx 中断消息</p> <p>Bit 27~22 : Current LTSSM State</p> <p>Bit 28: RCB Status 0b: an RCB of 64 bytes</p>	
--	--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

			<p>1b: an RCB of 128 bytes</p> <p>Bit 29 : MSI 中断使能。 0b: 禁止 MSI 中断 1b: 使能 MSI 中断</p> <p>Bit 31~30: power state of the PCI Express link VU+, KU and V7: 00b: L0 01b: L0s 10b: L1 11b: L2 (Note: L2 state is not supported)</p> <p>K7: 00: RX Not in L0s 01: RX L0s Entry 10: RX L0s Idle 11: RX L0s FTS</p>	
ACQ_BLK_BADDR_L#0	32	0x200	VID-ACQ#0 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#1	32	0x204	VID-ACQ#1 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#2	32	0x208	VID-ACQ#2 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#3	32	0x20C	VID-ACQ#3 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#4	32	0x210	VID-ACQ#4 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#5	32	0x214	VID-ACQ#5 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#6	32	0x218	VID-ACQ#6 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_L#7	32	0x21C	VID-ACQ#7 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#0	32	0x220	VID-ACQ#0 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#1	32	0x224	VID-ACQ#1 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#2	32	0x228	VID-ACQ#2 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#3	32	0x22C	VID-ACQ#3 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#4	32	0x230	VID-ACQ#4 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#5	32	0x234	VID-ACQ#5 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#6	32	0x238	VID-ACQ#6 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_BADDR_H#7	32	0x23C	VID-ACQ#7 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#0	32	0x240	VID-DISP#0 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#1	32	0x244	VID-DISP#1 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#2	32	0x248	VID-DISP#2 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#3	32	0x24C	VID-DISP#3 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#4	32	0x250	VID-DISP#4 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#5	32	0x254	VID-DISP#5 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#6	32	0x258	VID-DISP#6 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_L#7	32	0x25C	VID-DISP#7 数据块基址寄存器 LSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#0	32	0x260	VID-DISP#0 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000

DISP_BLK_BADDR_H#1	32	0x264	VID-DISP#1 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#2	32	0x268	VID-DISP#2 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#3	32	0x26C	VID-DISP#3 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#4	32	0x270	VID-DISP#4 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#5	32	0x274	VID-DISP#5 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#6	32	0x278	VID-DISP#6 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
DISP_BLK_BADDR_H#7	32	0x27C	VID-DISP#7 数据块基址寄存器 MSB, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#0	32	0x280	VID-ACQ#0 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#1	32	0x284	VID-ACQ#1 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#2	32	0x288	VID-ACQ#2 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#3	32	0x28C	VID-ACQ#3 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#4	32	0x290	VID-ACQ#4 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#5	32	0x294	VID-ACQ#5 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#6	32	0x298	VID-ACQ#6 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_SIZE#7	32	0x29C	VID-ACQ#7 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#0	32	0x2A0	VID-DISP#0 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#1	32	0x2A4	VID-DISP#1 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#2	32	0x2A8	VID-DISP#2 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#3	32	0x2AC	VID-DISP#3 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#4	32	0x2B0	VID-DISP#4 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#5	32	0x2B4	VID-DISP#5 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#6	32	0x2B8	VID-DISP#6 数据块大小寄存器, RW。单位: 字节	0x00000000
DISP_BLK_SIZE#7	32	0x2BC	VID-DISP#7 数据块大小寄存器, RW。单位: 字节	0x00000000
ACQ_BLK_NUM#0	32	0x2C0	VID-ACQ#0 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#1	32	0x2C4	VID-ACQ#1 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#2	32	0x2C8	VID-ACQ#2 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#3	32	0x2CC	VID-ACQ#3 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#4	32	0x2D0	VID-ACQ#4 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#5	32	0x2D4	VID-ACQ#5 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#6	32	0x2D8	VID-ACQ#6 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_BLK_NUM#7	32	0x2DC	VID-ACQ#7 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#0	32	0x2E0	VID-DISP#0 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#1	32	0x2E4	VID-DISP#1 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#2	32	0x2E8	VID-DISP#2 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#3	32	0x2EC	VID-DISP#3 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#4	32	0x2F0	VID-DISP#4 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#5	32	0x2F4	VID-DISP#5 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#6	32	0x2F8	VID-DISP#6 数据块总数寄存器, RW。单位: 数据块	0x00000000
DISP_BLK_NUM#7	32	0x2FC	VID-DISP#7 数据块总数寄存器, RW。单位: 数据块	0x00000000
ACQ_PKT_NUM#0	32	0x300	VID-ACQ#0 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_PKT_NUM#1	32	0x304	VID-ACQ#1 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_PKT_NUM#2	32	0x308	VID-ACQ#2 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_PKT_NUM#3	32	0x30C	VID-ACQ#3 数据包总数寄存器, RO。单位: 数据包	0x00000000

ACQ_PKT_NUM#4	32	0x310	VID-ACQ#4 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_PKT_NUM#5	32	0x314	VID-ACQ#5 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_PKT_NUM#6	32	0x318	VID-ACQ#6 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_PKT_NUM#7	32	0x31C	VID-ACQ#7 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#0	32	0x320	VID-DISP#0 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#1	32	0x324	VID-DISP#1 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#2	32	0x328	VID-DISP#2 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#3	32	0x32C	VID-DISP#3 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#4	32	0x330	VID-DISP#4 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#5	32	0x334	VID-DISP#5 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#6	32	0x338	VID-DISP#6 数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_PKT_NUM#7	32	0x33C	VID-DISP#7 数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#0	32	0x340	VID-ACQ#0 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#1	32	0x344	VID-ACQ#1 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#2	32	0x348	VID-ACQ#2 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#3	32	0x34C	VID-ACQ#3 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#4	32	0x350	VID-ACQ#4 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#5	32	0x354	VID-ACQ#5 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#6	32	0x358	VID-ACQ#6 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_DSC_PKT_NUM#7	32	0x35C	VID-ACQ#7 丢弃数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#0	32	0x360	VID-DISP#0 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#1	32	0x364	VID-DISP#1 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#2	32	0x368	VID-DISP#2 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#3	32	0x36C	VID-DISP#3 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#4	32	0x370	VID-DISP#4 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#5	32	0x374	VID-DISP#5 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#6	32	0x378	VID-DISP#6 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
DISP_DPL_PKT_NUM#7	32	0x37C	VID-DISP#7 复制数据包总数寄存器, RO。单位: 数据包	0x00000000
ACQ_USR_PARAM#0	32	0x380	VID-ACQ#0 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#1	32	0x384	VID-ACQ#1 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#2	32	0x388	VID-ACQ#2 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#3	32	0x38C	VID-ACQ#3 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#4	32	0x390	VID-ACQ#4 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#5	32	0x394	VID-ACQ#5 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#6	32	0x398	VID-ACQ#6 用户参数寄存器, RW。	0x00000000
ACQ_USR_PARAM#7	32	0x39C	VID-ACQ#7 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#0	32	0x3A0	VID-DISP#0 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#1	32	0x3A4	VID-DISP#1 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#2	32	0x3A8	VID-DISP#2 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#3	32	0x3AC	VID-DISP#3 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#4	32	0x3B0	VID-DISP#4 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#5	32	0x3B4	VID-DISP#5 用户参数寄存器, RW。	0x00000000
DISP_USR_PARAM#6	32	0x3B8	VID-DISP#6 用户参数寄存器, RW。	0x00000000

DISP_USR_PARAM#7	32	0x3BC	VID-DISP#7 用户参数寄存器, RW。	0x00000000
CH0_C2H_XFER_NUM	32	0x3C0	通道 0 C2H 传输总数, RW。 RDMA 模式下, 表示 Ring Buffer 的长度。 SGDMA 和 CDMA 模式下, 不使用。	0x00000000
CH1_C2H_XFER_NUM	32	0x3C4	通道 1 C2H 传输总数, RW。	0x00000000
CH2_C2H_XFER_NUM	32	0x3C8	通道 2 C2H 传输总数, RW。	0x00000000
CH3_C2H_XFER_NUM	32	0x3CC	通道 3 C2H 传输总数, RW。	0x00000000
CH4_C2H_XFER_NUM	32	0x3D0	通道 4 C2H 传输总数, RW。	0x00000000
CH5_C2H_XFER_NUM	32	0x3D4	通道 5 C2H 传输总数, RW。	0x00000000
CH6_C2H_XFER_NUM	32	0x3D8	通道 6 C2H 传输总数, RW。	0x00000000
CH7_C2H_XFER_NUM	32	0x3DC	通道 7 C2H 传输总数, RW。	0x00000000
CH0_H2C_XFER_NUM	32	0x3E0	通道 0 H2C 传输总数, RW。 RDMA 模式下, 表示 Ring Buffer 的长度。 SGDMA 和 CDMA 模式下, 不使用。	0x00000000
CH1_H2C_XFER_NUM	32	0x3E4	通道 1 H2C 传输总数, RW。	0x00000000
CH2_H2C_XFER_NUM	32	0x3E8	通道 2 H2C 传输总数, RW。	0x00000000
CH3_H2C_XFER_NUM	32	0x3EC	通道 3 H2C 传输总数, RW。	0x00000000
CH4_H2C_XFER_NUM	32	0x3F0	通道 4 H2C 传输总数, RW。	0x00000000
CH5_H2C_XFER_NUM	32	0x3F4	通道 5 H2C 传输总数, RW。	0x00000000
CH6_H2C_XFER_NUM	32	0x3F8	通道 6 H2C 传输总数, RW。	0x00000000
CH7_H2C_XFER_NUM	32	0x3FC	通道 7 H2C 传输总数, RW。	0x00000000

3.6 DMA Flow

3.6.1 Global Reset

3.6.1.1 Global Reset for SGDMA

- 停止 CH#i H2C DMA: 0 写入 CH#i_H2C_CTRL, 遍历 i
- 停止 CH#i C2H DMA: 0 写入 CH#i_C2H_CTRL, 遍历 i
- 施加 C2H 和 H2C DMA 总复位: SRST 设置为 3, 延时 10us
- 释放 C2H 和 H2C DMA 总复位: SRST 设置为 0, 延时 10us
- 中断使能: INT_MASK 设置为 0

3.6.1.2 Global Reset for CDMA-C2H and RDMA-C2H

- 停止 CH#i C2H DMA: 0 写入 CH#i_C2H_CTRL, 遍历 i
- 施加 C2H DMA 总复位: SRST 设置为 1, 延时 10us
- 释放 C2H DMA 总复位: SRST 设置为 0, 延时 10us
- 中断使能: INT_MASK 设置为 0

3.6.1.3 Global Reset for CDMA-H2C and RDMA-H2C

- 停止 CH#i H2C DMA: 0 写入 CH#i_H2C_CTRL, 遍历 i

- b. 施加 H2C DMA 总复位：SRST 设置为 2，延时 10us
- c. 释放 H2C DMA 总复位：SRST 设置为 0，延时 10us
- d. 中断使能：INT_MASK 设置为 0

3.6.2 CH#i H2C Flow for SGDMA

3.6.2.1 初始化 CH#i H2C

- a. 停止 CH#i H2C DMA：0 写入 CH#i_H2C_CTRL
- b. 施加 CH#i H2C DMA 复位：0x80000000 写入 CH#i_H2C_CTRL，延时 10us
- c. 释放 CH#i H2C DMA 复位：0 写入 CH#i_H2C_CTRL，延时 10us
- d. CH#i H2C 中断清除：1<<(i+8)写入 INT_STAT
- e. Host 配置 Card 的 CH#i 显示定时（可选）：如果使能内部显示定时 CH#i_H2C_CTRL[2]设置为 1；如果禁止内部显示定时 CH#i_H2C_CTRL[2]设置为 0；如果使能外部显示定时 CH#i_H2C_CTRL[3]设置为 1；如果禁止外部显示定时 CH#i_H2C_CTRL[3]设置为 0
- f. Host 配置 Card 的 CH#i VID-DISP 参数（可选）：配置 CH#i_H2C_FPS、CH#i_H2C_RES、DISP_BLK_BADDR_L#i、DISP_BLK_BADDR_H#i、DISP_BLK_SIZE#i、DISP_BLK_NUM#i、DISP_USR_PARAM#i 寄存器

3.6.2.2 启动 CH#i H2C

- a. Host 启动 Card 的 CH#i H2C DMA 引擎：CH#i_H2C_CTRL[0]设置为 1
- b. Host 启动 Card 的 CH#i H2C 工作：CH#i_H2C_CTRL[1]设置为 1

3.6.2.3 执行 CH#i H2C DMA

- a. Host 向 Card 的 CH#i H2C 地址队列写入一个或多个 DMA 内存地址：配置 CH#i_H2C_ADDR_U 和 CH#i_H2C_ADDR_L 寄存器
- b. Card 完成 CH#i H2C DMA，Card 向 Host 发出 MSI 中断
- c. Host 读取中断状态寄存器：读取 INT_STAT
- d. Host 清除该中断：回写 INT_STAT

3.6.2.4 停止 CH#i H2C

- a. Host 关闭 Card 的 CH#i H2C DMA 引擎以及禁止 CH#i H2C 工作：0 写入 CH#i_H2C_CTRL

3.6.3 CH#i C2H Flow for SGDMA

3.6.3.1 初始化 CH#i C2H

- a. 停止 CH#i C2H DMA: 0 写入 CH#i_C2H_CTRL
- b. 施加 CH#i C2H DMA 复位: 0x80000000 写入 CH#i_C2H_CTRL, 延时 10us
- c. 释放 CH#i C2H DMA 复位: 0 写入 CH#i_C2H_CTRL, 延时 10us
- d. CH#i C2H 中断清除: $1 \ll i$ 写入 INT_STAT
- e. Host 配置 Card 的 CH#i VID-ACQ 参数 (可选): 配置 CH#i_C2H_FPS、CH#i_C2H_RES、ACQ_BLK_BADDR_L#i、ACQ_BLK_BADDR_H#i、ACQ_BLK_SIZE#i、ACQ_BLK_NUM#i、ACQ_USR_PARAM#i 寄存器

3.6.3.2 启动 CH#i C2H

- a. Host 启动 Card 的 CH#i C2H DMA 引擎: CH#i_C2H_CTRL[0] 设置为 1
- b. Host 启动 Card 的 CH#i C2H 工作: CH#i_C2H_CTRL[1] 设置为 1

3.6.3.3 执行 CH#i C2H DMA

- a. Host 向 Card 的 CH#i C2H 地址队列写入一个或多个 DMA 内存地址: 配置 CH#i_C2H_ADDR_U 和 CH#i_C2H_ADDR_L 寄存器
- b. Card 完成 CH#i C2H DMA, Card 向 Host 发出 MSI 中断
- c. Host 读取中断状态寄存器: 读取 INT_STAT
- d. Host 清除该中断: 回写 INT_STAT

3.6.3.4 停止 CH#i C2H

- a. Host 关闭 Card 的 CH#i C2H DMA 引擎以及禁止 CH#i C2H 工作: 0 写入 CH#i_C2H_CTRL

3.6.4 CH#i H2C Flow for CDMA

3.6.4.1 初始化 CH#i H2C

- a. 停止 CH#i H2C DMA: 0 写入 CH#i_H2C_CTRL
- b. 施加 CH#i H2C DMA 复位: 0x80000000 写入 CH#i_H2C_CTRL, 延时 10us
- c. 释放 CH#i H2C DMA 复位: 0 写入 CH#i_H2C_CTRL, 延时 10us
- d. CH#i H2C 中断清除: $1 \ll (i+8)$ 写入 INT_STAT
- e. Host 配置 CH#i 显示定时 (可选): 如果使能内部显示定时 CH#i_H2C_CTRL[2] 设置为 1; 如果禁止内部显示定时 CH#i_H2C_CTRL[2] 设置为 0; 如果使能外部显示定时 CH#i_H2C_CTRL[3] 设置为 1; 如果禁止外部显示定时 CH#i_H2C_CTRL[3]

设置为 0

- f. Host 配置 Card 的 CH#i VID-DISP 参数（可选）：配置 CH#i_H2C_FPS、CH#i_H2C_RES、DISP_BLK_BADDR_L#i、DISP_BLK_BADDR_H#i、DISP_BLK_SIZE#i、DISP_BLK_NUM#i、DISP_USR_PARAM#i 寄存器

3.6.4.2 启动 CH#i H2C

- a. Host 启动 Card 的 CH#i H2C DMA 引擎：CH#i_H2C_CTRL[0]设置为 1
- b. Host 启动 Card 的 CH#i H2C 工作：CH#i_H2C_CTRL[1]设置为 1

3.6.4.3 执行 CH#i H2C DMA

- a. Host 向 Card 的 CH#i H2C 地址队列写入一个或多个 DMA 内存地址和 DMA 传输长度：配置 CH#i_H2C_ADDR_U 和 CH#i_H2C_ADDR_L 寄存器以及 CH#i_H2C_XFER_SIZE 寄存器
- b. Card 完成 CH#i H2C DMA，Card 向 Host 发出 MSI 中断
- c. Host 读取中断状态寄存器：读取 INT_STAT
- d. Host 清除该中断：回写 INT_STAT

3.6.4.4 停止 CH#i H2C

- a. Host 关闭 Card 的 CH#i H2C DMA 引擎以及禁止 CH#i H2C 工作：0 写入 CH#i_H2C_CTRL

3.6.5 CH#i C2H Flow for CDMA

3.6.5.1 初始化 CH#i C2H

- a. 停止 CH#i C2H DMA：0 写入 CH#i_C2H_CTRL
- b. 施加 CH#i C2H DMA 复位：0x80000000 写入 CH#i_C2H_CTRL，延时 10us
- c. 释放 CH#i C2H DMA 复位：0 写入 CH#i_C2H_CTRL，延时 10us
- d. CH#i C2H 中断清除：1<<i 写入 INT_STAT
- e. Host 配置 Card 的 CH#i VID-ACQ 参数（可选）：配置 CH#i_C2H_FPS、CH#i_C2H_RES、ACQ_BLK_BADDR_L#i、ACQ_BLK_BADDR_H#i、ACQ_BLK_SIZE#i、ACQ_BLK_NUM#i、ACQ_USR_PARAM#i 寄存器

3.6.5.2 启动 CH#i C2H

- a. Host 启动 Card 的 CH#i C2H DMA 引擎：CH#i_C2H_CTRL[0]设置为 1
- b. Host 启动 Card 的 CH#i C2H 工作：CH#i_C2H_CTRL[1]设置为 1

3.6.5.3 执行 CH#i C2H DMA

- a. Host 向 Card 的 CH#i C2H 地址队列写入一个或多个 DMA 内存地址和 DMA 传输长度：配置 CH#i_C2H_ADDR_U 和 CH#i_C2H_ADDR_L 寄存器以及 CH#i_C2H_XFER_SIZE 寄存器
- b. Card 完成 CH#i C2H DMA，Card 向 Host 发出 MSI 中断
- c. Host 读取中断状态寄存器：读取 INT_STAT
- d. Host 清除该中断：回写 INT_STAT

3.6.5.4 停止 CH#i C2H

- a. Host 关闭 Card 的 CH#i C2H DMA 引擎以及禁止 CH#i C2H 工作：0 写入 CH#i_C2H_CTRL

3.6.6 CH#i H2C Flow for RDMA

3.6.6.1 初始化 CH#i H2C

- a. 停止 CH#i H2C DMA：0 写入 CH#i_H2C_CTRL
- b. 施加 CH#i H2C DMA 复位：0x80000000 写入 CH#i_H2C_CTRL，延时 10us
- c. 释放 CH#i H2C DMA 复位：0 写入 CH#i_H2C_CTRL，延时 10us
- d. CH#i H2C 中断清除：1<<(i+8)写入 INT_STAT
- e. Host 配置 Card 的 CH#i H2C Ring 缓冲区参数：配置 CH#i_H2C_ADDR_U 和 CH#i_H2C_ADDR_L 寄存器以及 CH#i_H2C_XFER_SIZE 和 CH#i_H2C_XFER_NUM 寄存器
- f. 复位 CH#i H2C Ring 缓冲区的尾指针：0 写入 CH#i_H2C_XFER_SIZE 寄存器
- g. Host 配置 Card 的 CH#i VID-DISP 参数（可选）：配置 CH#i_H2C_FPS、CH#i_H2C_RES、DISP_BLK_BADDR_L#i、DISP_BLK_BADDR_H#i、DISP_BLK_SIZE#i、DISP_BLK_NUM#i、DISP_USR_PARAM#i 寄存器

3.6.6.2 启动 CH#i H2C

- a. Host 启动 Card 的 CH#i H2C DMA 引擎：CH#i_H2C_CTRL[0]设置为 1
- b. Host 启动 Card 的 CH#i H2C 工作：CH#i_H2C_CTRL[1]设置为 1

3.6.6.3 执行 CH#i H2C DMA

- a. Host 更新 CH#i H2C Ring 缓冲区的尾指针：写入 DISP_RING_PTR#i 寄存器
- b. Card 完成 CH#i H2C DMA，Card 向 Host 发出 MSI 中断
- c. Host 读取中断状态寄存器：读取 INT_STAT

- d. Host 清除该中断：回写 INT_STAT
- e. Host 读取 CH#i H2C Ring 缓冲区的头指针（可选）：读取 CH#i_H2C_XFER_SIZE 寄存器

3.6.6.4 停止 CH#i H2C

- a. Host 关闭 Card 的 CH#i H2C DMA 引擎以及禁止 CH#i H2C 工作：0 写入 CH#i_H2C_CTRL

3.6.7 CH#i C2H Flow for RDMA

3.6.7.1 初始化 CH#i C2H

- a. 停止 CH#i C2H DMA：0 写入 CH#i_C2H_CTRL
- b. 施加 CH#i C2H DMA 复位：0x80000000 写入 CH#i_C2H_CTRL，延时 10us
- c. 释放 CH#i C2H DMA 复位：0 写入 CH#i_C2H_CTRL，延时 10us
- d. CH#i C2H 中断清除：1<<i 写入 INT_STAT
- e. Host 配置 Card 的 CH#i H2C Ring 缓冲区参数：配置 CH#i_C2H_ADDR_U 和 CH#i_C2H_ADDR_L 寄存器以及 CH#i_C2H_XFER_SIZE 和 CH#i_C2H_XFER_NUM 寄存器
- f. 复位 CH#i C2H Ring 缓冲区的尾指针：0 写入 CH#i_C2H_XFER_SIZE 寄存器
- g. Host 配置 Card 的 CH#i VID-ACQ 参数（可选）：配置 CH#i_C2H_FPS、CH#i_C2H_RES、ACQ_BLK_BADDR_L#i、ACQ_BLK_BADDR_H#i、ACQ_BLK_SIZE#i、ACQ_BLK_NUM#i、ACQ_USR_PARAM#i 寄存器

3.6.7.2 启动 CH#i C2H

- a. Host 启动 Card 的 CH#i C2H DMA 引擎：CH#i_C2H_CTRL[0]设置为 1
- b. Host 启动 Card 的 CH#i C2H 工作：CH#i_C2H_CTRL[1]设置为 1

3.6.7.3 CH#i C2H DMA

- a. Host 更新 CH#i C2H Ring 缓冲区的尾指针：写入 CH#i_C2H_XFER_SIZE 寄存器
- b. Card 完成 CH#i C2H DMA，Card 向 Host 发出 MSI 中断
- c. Host 读取中断状态寄存器：读取 INT_STAT
- d. Host 清除该中断：回写 INT_STAT
- e. Host 读取 CH#i C2H Ring 缓冲区的头指针（可选）：读取 CH#i_C2H_XFER_SIZE 寄存器

3.6.7.4 停止 CH#i C2H

- a. Host 关闭 Card 的 CH#i C2H DMA 引擎以及禁止 CH#i C2H 工作：0 写入 CH#i_C2H_CTRL

3.7 设备驱动

3.7.1 Windows WDF (Queue or Non-Queue , Ring)

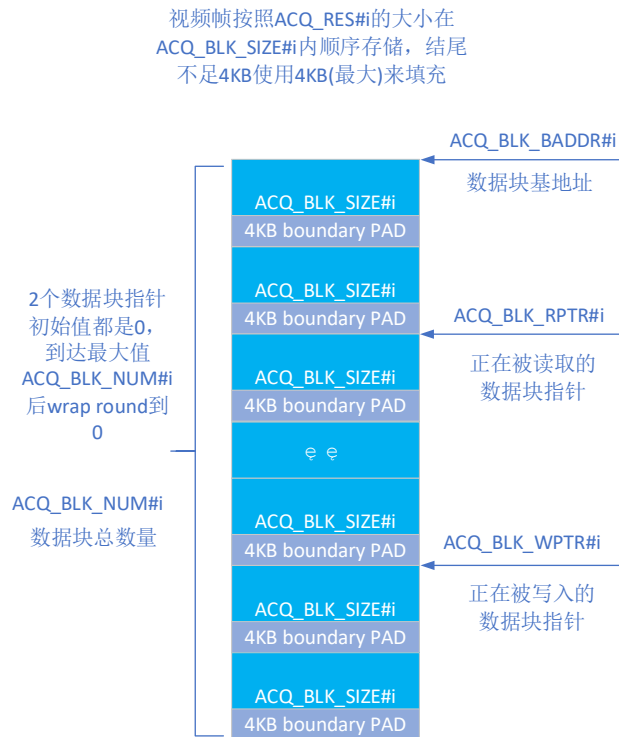
3.7.2 Linux (Queue or Non-Queue , Ring)

3.7.3 V4L2

4 附录(视频存储队列管理说明)

4.1 视频采集队列存储管理

每路视频采集在 DDR 有独立的帧缓冲区，存储空间如下所示：



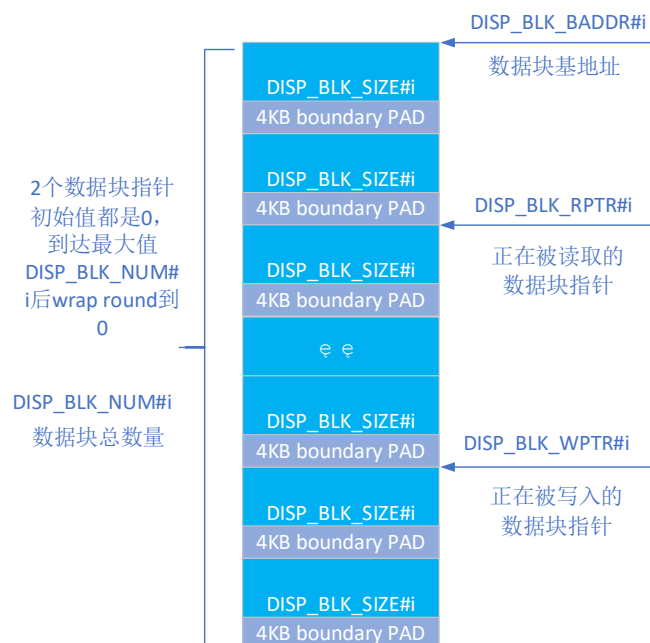
视频采集通道的存储空间配置示例：

	ACQ#0	ACQ#1	ACQ#2	ACQ#3	ACQ#4	ACQ#5	ACQ#6	ACQ#7
ACQ_BLK_BADDR#i	0M	256M	512M	768M	1024M	1280M	1536M	1792M
ACQ_BLK_SIZE#i	8M	8M	8M	8M	8M	8M	8M	8M
ACQ_BLK_NUM#i	32	32	32	32	32	32	32	32

4.2 视频显示队列存储管理

每路视频显示在 DDR 有独立的帧缓冲区，存储空间如下所示：

视频帧按照DISP_RES#i的大小在
DISP_BLK_SIZE#i内顺序存储，结尾
不足4KB使用4KB(最大)来填充



视频显示通道的存储空间配置示例：

	DISP#0	DISP#1	DISP#2	DISP#3	DISP#4	DISP#5	DISP#6	DISP#7
DISP_BLK_BADDR#i	2048M	2304M	2560M	2816M	3072M	3328M	3584M	3840M
DISP_BLK_SIZE#i	8M	8M	8M	8M	8M	8M	8M	8M
DISP_BLK_NUM#i	32	32	32	32	32	32	32	32