**XDMA图像采集应用寄存器接口定义**

**libudma-v0.1.0**

# FPGA中的寄存器映射表

|  |  |  |  |
| --- | --- | --- | --- |
| 地址/偏移量 | 名称 | 方向 | 功能 |
| 0x00 | CONTROL | W | 控制寄存器，包括使能、复位等 |
| 0x04 | STATUS | R | 状态寄存器，包括复位状态、时钟状态等 |
| 0x08 | IER | W | 中断控制寄存器，预留 |
| 0x0c | ISR | R | 中断状态寄存器，预留 |
| 0x10 ~ 0x3c | 保留 |  |  |
| 0x40 | C2H\_START | R | 接收缓冲区起始地址。目前作为FPGA的静态参数(parameter)，对软件是只读的。将来要考虑可通过软件设置。 |
| 0x44 | C2H\_END | R | 接收缓冲区结束地址，该地址为最大写入地址+1，例如起始地址0x00000000，结束地址0x10000000，则可写入地址范围0x00000000~0x0FFFFFFF，即256MB |
| 0x48 | C2H\_BUF\_SIZE | R | 接收缓冲区的每一帧大小，单位为字节。例如0x400000，表示每帧缓冲区4MB，这也是允许的最大图像尺寸。 |
| 0x4c | C2H\_RD\_NEXT | W | 接收数据读指针，表示软件下一次要读的帧的起始地址。软件每次读取一帧数据后，将C2H\_RD\_NEXT的新值写入。目前FPGA程序不使用这个指针，将来要考虑利用这个指针判断缓冲区是否已满，数据是否有溢出。 |
| 0x50 | C2H\_WR\_NEXT | R | 接收数据写指针，表示FPGA下一次要写入的帧的起始地址。FPGA每接收完一帧数据，将C2H\_WR\_NEXT的最新值更新。软件通过判断C2H\_WR\_NEXT的值，可以知道缓冲区中有多少数据。 |
| 0x54 | C2H\_FRM\_SIZE | R | 接收数据帧大小。单位为字节。FPGA将实际接收到的图像帧的大小更新到这个寄存器。软件可以读出，从而知道帧缓存中的实际数据大小。目前FPGA的帧大小采用静态参数，将来要考虑支持动态帧大小。 |
| 0x58 | C2H\_FRM\_WIDTH | R | 接收数据帧的宽度，单位为字节。 |
| 0x5c | C2H\_FRM\_HEIGHT | R | 接收数据帧的高度，单位为行数。 |
| 0x60 ~ 0x7c | 保留 |  |  |
| 0x80 | H2C\_BUF\_START | R | 发送缓冲区起始地址。 |
| 0x84 | H2C\_BUF\_END | R | 发送缓冲区结束地址。 |
| 0x88 | H2C\_BUF\_SIZE | R | 发送缓冲区每帧大小，单位为字节。 |
| 0x8c | H2C\_RD\_NEXT | R | 发送数据读指针。表示FPGA写一次要读取并发送的数据帧的起始地址。FPGA每发送一帧后，将H2C\_RD\_NEXT的值更新。 |
| 0x90 | H2C\_WR\_NEXT | W | 发送数据写指针。表示软件下一次要写入的帧的起始地址。FPGA通过比较软件写入的H2C\_WR\_NEXT和H2C\_RD\_NEXT，如果H2C\_RD\_NEXT != H2C\_WR\_NEXT，则读取一帧发送。 |
| 0x94 | H2C\_FRM\_SIZE | W | 发送数据帧大小，单位为字节。由软件将要发送的数据帧大小写入这个寄存器。其实FPGA程序仅使用FRM\_WIDTH和FRM\_HEIGHT就可以了，这里的FRM\_SIZE预留用作以后实现动态数据位宽。 |
| 0x98 | H2C\_FRM\_WIDTH | W | 发送数据帧的宽度，单位为字节。由软件将要发送的行长度写入这个寄存器，FPGA按照这里给出的大小将数据组成AXI-Stream的一个个Packet。 |
| 0x9c | H2C\_FRM\_HEIGHT | W | 发送数据帧的高度，单位为行数。由软件将要发送的行数写入这个寄存器，FPGA按照这里给出的数值发送给定数量的Packet。 |
| 0xa0 ~ 0xbc | 保留 |  |  |

# 基本操作流程

FPGA和软件驱动之间使用DDR中的缓冲区进行数据交换。DDR存储区域划分成两个部分，分别是接收缓冲区和发送缓冲区。缓冲区的起始和结束地址分别用BUF\_START和BUF\_END表示，每个缓冲区的大小是缓冲帧大小BUF\_SIZE的整数倍。

FPGA提供regfile，软件通过XDMA的user端口经AXI-lite读写regfile，通过XDMA的C2H和H2C端口读写DDR。

FPGA程序通过regfile的输入输出信号和软件之间传递寄存器的数值。

寄存器方向标注为W的，是软件从AXI-lite写入，FPGA程序从regfile的output获取寄存器值。寄存器方向标注为R的，是FPGA程序从regfile的input给出值，由软件经AXI-lite读取。

FPGA和软件驱动之间主要通过RD\_NEXT和WR\_NEXT实现缓冲队列的管理，接收时FPGA是写入方（负责维护C2H\_WR\_NEXT），软件是读取方（负责维护C2H\_RD\_NEXT）；发送时软件是写入方(负责维护H2C\_WR\_NEXT)，FPGA是读取方（负责维护H2C\_RD\_NEXT）。

## 图像接收流程：

### FPGA端的操作

以接收为例，FPGA接收到数据时，从C2H\_BUF\_START开始，按照C2H\_BUF\_SIZE为步进，依次写入每帧数据。每写入一帧后，将C2H\_WR\_NEXT的地址增加：

If(C2H\_WR\_NEXT + C2H\_BUF\_SIZE == C2H\_BUF\_END)

C2H\_WR\_NEXT = C2H\_BUF\_START;

else

C2H\_WR\_NEXT = C2H\_WR\_NEXT+C2H\_BUF\_SIZE;

### 软件端的操作

软件自己维护C2H\_RD\_NEXT的值，在读取到C2H\_WR\_NEXT变化后，发现C2H\_WR\_NEXT!=C2H\_RD\_NEXT，就认为缓冲区中存在有效数据。

If(C2H\_WR\_NEXT != C2H\_RD\_NEXT)

//发起XDMA从C2H\_RD\_NEXT地址读取一帧，然后增加C2H\_RD\_NEXT

If(C2H\_RD\_NEXT+C2H\_BUF\_SIZE == C2H\_BUF\_END)

C2H\_RD\_NEXT = C2H\_BUF\_START;

else

C2H\_RD\_NEXT = C2H\_RD\_NEXT + C2H\_BUF\_SIZE;

else

//没有数据，啥也不做

## 图像发送流程

对于发送，情况类似，只是操作角色互换。软件负责写入H2C\_WR\_NEXT，FPGA负责更新H2C\_RD\_NEXT。

### 软件端的操作

软件在写入数据前先检测H2C\_RD\_NEXT和H2C\_WR\_NEXT，判断是否还有空间。如果有空，则向H2C\_WR\_NEXT地址写入一帧图像，之后增加H2C\_WR\_NEXT。

If(H2C\_WR\_NEXT+H2C\_BUF\_SIZE == H2C\_RD\_NEXT || (H2C\_WR\_NEXT+H2C\_BUF\_SIZE==H2C\_BUF\_END && H2C\_RD\_NEXT == H2C\_BUF\_START))

//向H2C\_WR\_NEXT地址写入一帧图像，然后增加H2C\_WR\_NEXT

If(H2C\_WR\_NEXT+H2C\_BUF\_SIZE == H2C\_BUF\_END)

H2C\_WR\_NEXT = H2C\_BUF\_START;

else

H2C\_WR\_NEXT = H2C\_WR\_NEXT + H2C\_BUF\_SIZE;

### FPGA端的操作

FPGA在检测到H2C\_WR\_NEXT != H2C\_RD\_NEXT时，认为缓存区中有新的数据，从H2C\_RD\_NEXT地址读取一帧发送，然后增加H2C\_RD\_NEXT。不断循环直到H2C\_WR\_NEXT==H2C\_RD\_NEXT，这代表缓冲区中没有数据了。

while(H2C\_WR\_NEXT != H2C\_RD\_NEXT) {

//从H2C\_RD\_NEXT地址读取一帧并发送，然后更新H2C\_RD\_NEXT

If(H2C\_RD\_NEXT+H2C\_BUF\_SIZE == H2C\_BUF\_END)

H2C\_RD\_NEXT = H2C\_BUF\_START;

else

H2C\_RD\_NEXT = H2C\_RD\_NEXT+H2C\_BUF\_SIZE;

}

# 计划的扩展功能

## 接收时间戳

和LVDS测试项目一样，将接收时间记录下来写入帧缓冲的头或尾部。

## 接收溢出判断

如果出现C2H\_WR\_NEXT+C2H\_BUF\_SIZE==C2H\_RD\_NEXT，表示FPGA将会覆盖软件还未读取的数据，此时最好在状态寄存器中设一个标识位并发起中断让软件处理。