文件编号：

模板版本：

Wrap\_wr\_logic详细说明书（FPGA）

制定：邢海涛

审核：

批准：

中国大恒（集团）有限公司北京图像视觉技术分公司

2016年09月14日

|  |
| --- |
| 文件状态： [ ]草稿 [√ ]正式发布[ ]正在修改 |

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 撰写人 | 变更内容 |
| V1.0.0 | 2016-09-14 | 邢海涛 | 初稿 |
|  |  |  |  |
|  |  |  |  |

目录

[1. wrap\_wr\_logic模块规格指标（Module Features） 4](#_Toc462763020)

[1.1. 概要设计指标 4](#_Toc462763021)

[1.2. 后级模块约束指标 5](#_Toc462763022)

[2. Wrap\_wr\_logic模块方案设计（Module Designs） 6](#_Toc462763023)

[2.1. Wrap\_wr\_logic结构 6](#_Toc462763024)

[2.2. Front\_buf设计 6](#_Toc462763025)

[2.2.1. fifo选择 6](#_Toc462763026)

[2.2.2. fifo复位 6](#_Toc462763027)

[2.2.3. fifo写 7](#_Toc462763028)

[2.2.4. fifo读 7](#_Toc462763029)

[2.3. 写逻辑设计 8](#_Toc462763030)

[2.3.1. 状态机 8](#_Toc462763031)

[2.3.2. mcb写模块整体波形图 10](#_Toc462763032)

[2.3.3. mcb写逻辑 10](#_Toc462763033)

[2.3.4. mcb写命令逻辑 10](#_Toc462763034)

[2.3.5. mcb写burst length 11](#_Toc462763035)

[2.3.6. mcb写地址 11](#_Toc462763036)

[2.4. 停采的处理方法 11](#_Toc462763037)

[2.5. 寄存器生效时机 11](#_Toc462763038)

[3. 模块测试项（Module Testings） 12](#_Toc462763039)

[3.1. 详细设计测试规格 12](#_Toc462763040)

[3.2. 其他测试规格 12](#_Toc462763041)

[4. 系统测试传递测试项（可选） 12](#_Toc462763042)

[5. 重点测试项（可选） 12](#_Toc462763043)

[6. 单元测试结果（Module Testings Result） 12](#_Toc462763044)

[7. 附录（可选） 12](#_Toc462763045)

# wrap\_wr\_logic模块规格指标（Module Features）

## 概要设计指标

| 指标 | 指标分解 |
| --- | --- |
| 1. SE=0设置完0.5ms之内，帧存模块要停下来 | 当se=0时，wrap\_wr\_logic立即停止写MCB WR FIFO。发出WR CMD命令，BL=64。这样会将MCB WR FIFO清空。但是会造成MCB WR FIFO underrun的现象，目前认为underrun不会影响到后面的传输，且一次发出64个数据量一定能够将MCB WR FIFO清空，这样做是比较安全的。 |
| 1. 设置SE=1后，场信号到来后，便允许图像数据进入DDR | 当se上升沿时，wrap\_wr\_logic的状态机不能立即启动，要看到一个fval上升沿才能认为是一帧开始 |
| 1. 传输阻塞时，FPGA抛弃当前曝光产生的所有ROI的图像，而不是抛弃其中某个ROI的图像 | 当写指针追上读指针的时候，写指针要跳过读指针，这样就会覆盖掉一帧数据。 |
| 1. 一次触发所有窗口都输出完整 | 一次触发，sensor输出一帧图像，一帧图像中可能包括多个ROI，帧存模块要将所有的ROI都读出 |
| 1. 切换采集模式，采集过程中，由连续切换触发模式，触发模式最多滞后n帧生效，生效后发送触发信号可以得到图像；触发切换连续模式，如果正在采集图像，连续模式最多滞后n帧生效，生效后得到连续图像，如果处于触发等待期间，连续模式可以立即生效；n表示当前设置的ROI的个数。 | 一次触发，sensor输出一帧图像，一帧图像中可能包括多个ROI，帧存模块要将所有的ROI都读出 |
| 1. FPGA后端传输帧率是Sensor输出帧率的n倍，n表示当前ROI的个数 | 一次触发，sensor输出一帧图像，一帧图像中可能包括多个ROI，帧存模块要将所有的ROI都读出 |

## 后级模块约束指标

| 信号 | 指标 |
| --- | --- |
| o\_wr\_ptr\_change | 1. 宽度为2个时钟周期 2. 在第二个时钟周期改变ov\_wr\_ptr |
| ov\_wr\_ptr | 在 o\_wr\_ptr\_change 第二个时钟周期改变 |
| ov\_wr\_addr | 写地址，在 o\_wr\_cmd\_en =1时改变 |
| o\_wr\_cmd\_en | 1. 正常情况下，每写入32组数据(256byte)之后，发出一个命令。 2. 当一个flag结束了，或者fval结束了，发出一个命令。 3. 当se下降沿，发出一个命令。 |
| ov\_wr\_cmd\_bl | 1. 正常情况下，bl=32 2. 当一个flag结束了，或者fval结束了，bl=写入的数据量。 3. 当se下降沿，bl=64。 |
| ov\_wr\_cmd\_instr | 固定为3’b0，不加precharge的写指令 |
| ov\_wr\_cmd\_mask | 固定为全0 |
| ov\_wr\_cmd\_byte\_addr | 写地址，在o\_wr\_cmd\_en 之后改变 |
| o\_wr\_en | Mcb wr fifo写使能，写使能为1的条件为   1. 处于写状态 2. 前端fifo不空 3. Wr mcb fifo不满 |
| ov\_wr\_data | 前端FIFO低64bit数据 |
| o\_writing | 正在写标志   1. 当开始写的时候，设为1 2. 当一帧写完的时候，设为0 |

# Wrap\_wr\_logic模块方案设计（Module Designs）

## Wrap\_wr\_logic结构



Wrap\_wr\_logic由两部分组成，front\_buf和wr\_logic。Front\_buf是一个fifo，完成缓存前端数据，隔离时钟域的作用。Wr\_logic完成写逻辑的操作，工作在clk\_frame\_buf时钟域。

## Front\_buf设计

### fifo选择

前端fifo的数据宽度为64+5，64表示数据位宽，5表示5个flag，分别是i\_leader\_flag、i\_trailer\_flag、i\_chunk\_flag、i\_image\_flag、i\_trailer\_final\_flag。



低64bit全部是数据，其中的数据既包括图像，也包括leader、chunk、trailer。Bit64为leader\_flag，Bit65为trailer\_flag，Bit66为chunk\_flag，Bit67为image\_flag，Bit68为trailer\_final\_flag。

fifo深度由sensor的行宽决定，一般来说，FIFO至少能容纳下一行的数据量，这样能够利用上行消隐的时间。

Python1300的行宽是1280，当10bit模式的时候，一行的数据量是2560byte，fifo深度至少是320才能装下一行的数据量，因此设置fifo的深度是512，这样肯定能够利用上sensor的行消隐。此时会用到4个M9K。

设置fifo的program empty为256，当帧存开始写一帧数据的时候，如果发现前端fifo中的数据超过一半，即promgram empty=0，认为前端fifo中已经放入了太多的数据，这一帧有可能会发生前端fifo溢出的现象，就不会把这一帧写入到DDR中。

### fifo复位

在以下条件下，fifo需要复位：（或操作）

* reset\_frame\_buf有效
* fval上升沿
* se=0直到fval上升沿

### fifo写

fifo写的条件是：（与操作）

* 前端fifo不满
* fval dval都是1

### fifo读

fifo读的条件是：（与操作）

* 前端fifo不空
* mcb wr fifo不满
* 处于写mcb的状态
* 开采信号有效

### fifo读写波形

## 写逻辑设计

### 状态机



**状态机状态说明**

* IDLE 空闲状态
* PTR 写指针改变状态
* WR 写一帧数据状态
* CMD 发出命令状态
* FLAG 切换flag状态

**状态机跳转**

* IDLE-> PTR

1. se=1，且
2. se=1之后有一个fval上升沿，且
3. 前端fifo半空信号=1，如果前端fifo中数据太多，则认为帧存无法把数据及时放到ddr中，要将这一帧丢掉，避免残帧。且
4. Calibration完成，且
5. 允许写
   1. 如果帧存深度为1，即单帧，只有当不读的时候，才能允许写
   2. 如果帧存深度大于1，任何时候都允许写

* PTR -> WR

1. 当wr\_ptr\_change=1宽度为2拍时，写指针改变完毕，进入写状态

* WR -> IDLE

1. 前端fifo空，且
2. Fval=0，且
3. 上一次发出cmd之后，没有从前端fifo中取数据

* WR -> CMD

1. 从前端fifo中读出的数据量是 BURST\_SIZE-1 且 正在读前端fifo。或
2. 前端fifo空，且Fval=0，且从前端fifo中取了一部分数据。或
3. Se=0

* WR -> FLAG

1. 当前flag有下降沿。

* CMD -> WR

1. wr\_cmd\_fifo不满，且
2. 当前flag下降沿=0

* CMD -> FLAG

1. 当前flag下降沿=1

* CMD -> IDLE

1. 当stream\_enable\_reg=1，且
2. 满足写cmd的条件

* FLAG -> WR

1. Wr cmd =1

* FLAG -> IDLE

1. 当stream\_enable\_reg=1，且
2. 满足写cmd的条件

### mcb写模块整体波形图



### mcb写逻辑

mcb wr fifo的写信号就是前端fifo的读信号。写信号有效的条件就是前端fifo读的条件。

### mcb写命令逻辑

在两个状态可能会发出写命令，分别是CMD和FLAG状态。

* CMD状态：

当mcb wr cmd fifo不满且cf\_fall=0的时候，发出写命令

* FLAG状态：

当mcb wr cmd fifo不满且在wr\_cmd=0

### mcb写burst length

mcb接口的burst\_length+1表示要写入的数据量，用word\_cnt寄存器表示当前的已经写入的数据量-1。在IDLE状态或者FLAG状态且wr\_cmd=1复位为最大值。

### mcb写地址

此处的写地址是寄存器wr\_addr，不是mcb的byte addr。

#### 切换写地址的时机

写地址分为三种情况改变

* IDLE，优先级最高

在IDLE状态，地址复位为0

* FLAG，优先级次高

在FLAG状态，且wr\_cmd=1，将写地址切换为下一个flag的起始地址。

* 其他状态，优先级最低

Wr\_cmd=1，写地址+1

#### 写地址的排布

目前共有5个flag，每个flag个需要一个起始地址。在frame\_buffer中已经根据每个flag数据量的大小，分配了每个flag的首地址。

## 停采的处理方法

停采时，地址、指针、正在写状态立即归零。

停采的时候，stream\_enable=1，立即把stream\_enable\_reg清为0。当stream\_enable=1，且fval上升沿的时候，stream\_enable\_reg设置为1。

在WR CMD FLAG状态才会判断是否停采。

* WR状态

当stream\_enable\_reg=0时，如果此时在WR状态，那么要进入到CMD状态

* CMD状态

当stream\_enable\_reg=0时，如果满足写命令的条件，那么进入到IDLE状态

* FLAG状态

当stream\_enable\_reg=0时，如果满足写命令的条件，那么进入到IDLE状态

## 寄存器生效时机

在停采的时候，寄存器生效

# 模块测试项（Module Testings）

## 详细设计测试规格

## 其他测试规格

系统测试传递测试项（可选）

无

重点测试项（可选）

单元测试结果（Module Testings Result）

附录（可选）