文件编号：

模板版本：

Frame\_buffer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20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 | 2016-09-20 | 邢海涛 | 初稿 |
|  |  |  |  |
|  |  |  |  |

目录

[1. Frame\_buffer模块规格指标（Module Features） 4](#_Toc462158020)

[2. Frame\_buffer模块方案设计（Module Designs） 5](#_Toc462158021)

[2.1. 帧缓存模块方案设计 5](#_Toc462158022)

[2.2. 帧缓存模块结构组成 5](#_Toc462158023)

[3. 模块测试项（Module Testings） 6](#_Toc462158024)

[4. 系统测试传递测试项（可选） 6](#_Toc462158025)

[5. 重点测试项（可选） 6](#_Toc462158026)

[6. 单元测试结果（Module Testings Result） 7](#_Toc462158027)

[7. 附录（可选） 7](#_Toc462158028)

# Frame\_buffer模块规格指标（Module Features）

在wrap\_wr\_logic wrap\_rd\_logic两个模块中详细描述指标。

# Frame\_buffer模块方案设计（Module Designs）

## 帧缓存模块方案设计

帧缓存模块的DDR3 SDRAM控制器采用XILINX的硬核(MCB)，该硬核对外接口是标准的FIFO接口。为了能提高DDR3 SDRAM的使用效率和简化设计，在MCB与图像输入和后级输出之间，添加了两个FIFO。

将MCB中的时钟模块提到帧缓存模块外部，使之成为一个独立的模块，这样做是为了方便FPGA中的其他模块使用PLL的端口。

## 帧缓存模块结构组成



帧存模块完成与DDR3芯片的交互工作，主要包括4部分。

* MCB是Xilinx自带的DDR3 SDRAM控制器，与用户逻辑的接口为通用的FIFO接口。在帧缓存模块中将MCB设置为两组64bit的端口P0和P1，P0与写逻辑相连，P1与读逻辑相连。

MCB包括硬核部分和软核部分两个子模块，硬核部分完成数据通道的功能，软核部分完成动态校正的功能。

* Wrap\_wr\_logic由前级FIFO和写逻辑组成，其中前级FIFO由FPGA中的BlockRAM9K构成，输入输出的位宽均为69bit，深度需要根据sensor行宽决定。写逻辑负责具体的写逻辑控制。
* Wrap\_rd\_loigc由读逻辑和后级FIFO组成，其中后级FIFO由FPGA中的BlockRAM9K构成，输入输出的位宽均为33bit，深度为256。目前控制数据只有一种，定义为帧结束符，即当后级模块检测到最高位是1时，应当认为是一帧的结束。读逻辑子模块负责具体的读逻辑控制。
* 与帧缓存模块平级的时钟模块输出高速时钟和帧缓存时钟。FPGA中的其他资源可以与帧缓存模块共享时钟模块中的PLL资源。



上图是帧缓存模块的时钟结构图。

* Wrap\_wr\_logic模块连接clk\_in时钟域和clk\_frame\_buf时钟域，将图像数据从输入时钟域转换到帧存时钟域。
* Wrap\_rd\_logic模块连接clk\_out时钟域和clk\_frame\_buf时钟域，将图像数据从帧存时钟域转换到输出时钟域。

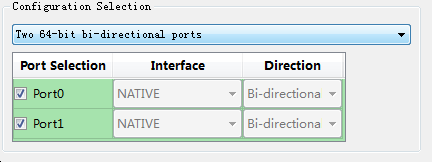
由于前后端FIFO的存在，wrap\_wr\_logic wrap\_rd\_loigc两个模块都工作在clk\_frame\_buf时钟域，这样方便了两个模块之间的交互，接口设计会更加简单。

* MCB硬核连接clk\_frame\_buf时钟域和高速时钟域，完成FPGA内部的数据与DDR3芯片中的数据交互的功能。
* MCB软核只工作在clk\_mcb\_drp时钟域，完成动态校准的工作。

## MCB模块设计

### ddr器件选型

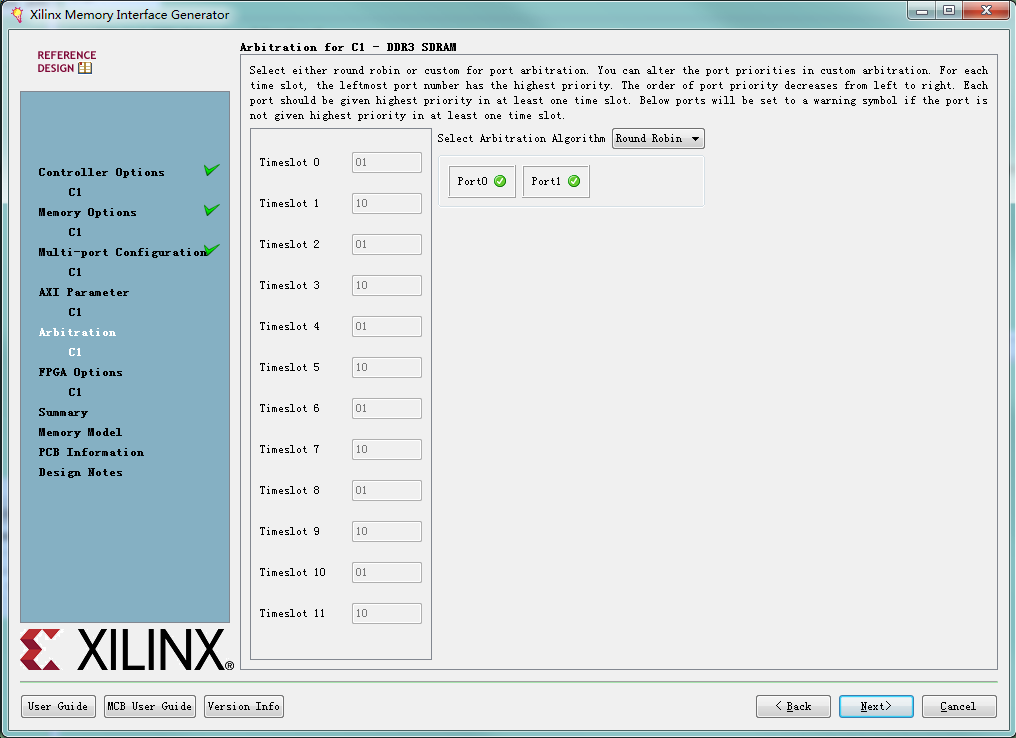
### 读写端口



例化两个64bit的端口，

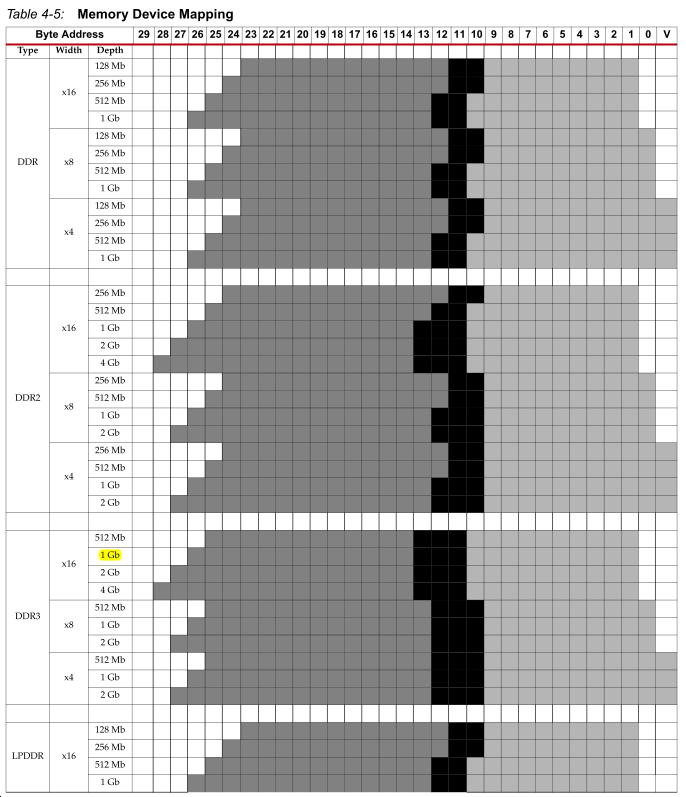
### 读写优先级

MCB的两个端口分别是读和写，可以设置MCB的对两个端口的仲裁逻辑，在本设计中，没有改变MCB的仲裁逻辑，还是使用 Round Robin算法，即MCB分时间片轮训。

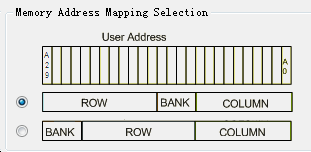


帧存逻辑的写优先级是高于读优先级的，为了达到这一目的，在读逻辑一侧做了如下操作：当MCB WR CMD FIFO不空的时候，就不能发出读命令。

### mcb地址划分



上图是MCB控制器地址划分的示意图。我们使用DDR3 x16 1Gb的芯片，地址以byte为单位，共有30位，其中bit29-27没有用到。1Gb的芯片，相当于128MByte=2^27Byte，因此使用27bit是足够的。



外部器件地址对应方式为ROW\_BANK\_COL。MCB地址的[26:14]对应SDRAM的row地址，MCB地址的[13:11]对应SDRAM的bank地址，MCB地址的[10:1]对应SDRAM的col地址。MCB地址最低位并没有使用。

帧缓存模块的读写指针映射到MCB地址的最高位，读写地址映射到MCB地址的中间位置，尾部的0的个数取决于每次读写的burst数据量。MCB地址与帧缓存模块读写指针和读写地址的对应关系是：

写地址映射：burst长度是256byte，m=8

mcb\_byte\_addr[29:0] = {**3’b0,读写指针,读写地址,m’b0**} （1Gb DDR3）

读地址映射：mcb fifo 位宽是8byte，n=3

mcb\_byte\_addr[29:0] = {**3’b0,读写指针,读写地址,n’b0**} （1Gb DDR3）

此处读写地址后面补0的个数是不一样的。写的时候，每一块的数据都是连续的，这样可以突发写一大批数据。但是读就不一样了，需要精细控制。比如说 ROI1的leader的起始地址是{**3’b0,读写指针,7 ,3’b0**}（56=0x38），在图像换行的时候，也需要精细控制。

### 帧存模块的物理位置



* 帧存模块所使用的PLL在芯片靠上的位置，目的是为了和MCB的距离较近。PLL输出的高速时钟并不会占用全局时钟资源
* BUFPLL\_MCB是高速时钟缓冲器，把PLL输出的高速时钟缓冲的MCB硬核当中
* MCB硬核在靠近芯片边缘的部分
* MCB软核是FPGA通用逻辑，由FPGA的综合工具布局布线

## 帧存深度设计

帧存深度分为三个方面，一个是帧存的最大深度，一个是帧存的设计深度，一个帧存的当前深度。

### 帧存的最大深度

帧存的最大深度和两个因素相关，一个是图像的最大分辨率，一个是DDR3的容量。在本项目中，DDR3的容量是固定的1Gb，因此图像分辨率是唯一的因素。图像分辨率越大，可容纳的帧存深度便越小。

### 帧存的设计深度

设计深度是指项目中实际可用的深度，定义的参数为 PTR\_WIDTH。设计深度要小于等于最大深度。

parameter PTR\_WIDTH = 2 //读写指针的位宽，1-最大1帧 2-最大3帧 3-最大7帧 4-最大15帧 5-最大31帧 ... 16-最大65535帧

设计深度越大，帧存模块的判断逻辑就会越多，资源消耗就会越大。该项目中，PTR\_WIDTH默认为2，最大三帧。

Why

1. 缓存能力最大
2. 实时性最高 √

### 帧存的当前深度

当前深度是指寄存器设置的数值，默认是2，2帧的深度。帧存当前深度会影响MCB的地址排布。



帧存深度越大，则每一帧可容纳的数据量就会越小。

## Mcb byte addr 设计

本次设计考虑了开小窗口的时候可以增加帧存深度的问题，mcb byte addr根据当前的帧存深度，选择读写指针、读写地址的分布。

以1Gb DDR3 写逻辑地址为例：

* 单帧：

{3'b0,wr\_addr[18:0],8'b0}

* 2帧：

{3'b0,wr\_ptr[0],wr\_addr[17:0],8'b0}

* 3、4帧：

{3'b0,wr\_ptr[1:0],wr\_addr[16:0],8'b0}

* 5-8帧：

{3'b0,wr\_ptr[2:0],wr\_addr[15:0],8'b0}

## 帧追赶策略

帧追赶的策略总共分为2种，分别是单帧追赶和多帧追赶。原则是相同的：

1. 保证读写的每一帧都是完整的。
2. 写指针不可以进入读指针。
3. 读指针可以进入写指针，但读地址不能超过写地址。

### 单帧追赶

当帧缓存深度为1时，只有一帧缓存可供读写，此时不会用到读写指针，只会用到读写地址。

当前端新来一帧图像的时候，如果此时读模块没有在读一帧数据，那么可以更新这一帧。如果此时读模块处于读状态，那么就不能更新这一帧，即丢弃当前帧。

当读模块要读一帧数据的时候，如果帧缓存中的数据没有刷新过，则不会将其读出来，如果帧缓存中的数据已经刷新过或正在刷新，则可以读数据。需要注意的是，如果写模块正在刷新，那读地址要小于写地址。

单帧追赶策略需要用到一个刷新标志，要来记录当前帧是否被刷新过。

与多帧缓存相比，单帧策略有如下缺点。

1. 如果读操作紧跟着写操作，那么读的速度是很慢的，相当于写入的速度。即使后端总线带宽很高，也只能一点点的读出数据。
2. 如果在帧消隐的时候开始读数据，一旦读的稍微慢一点，还没有读完的时候新的一帧又来了，那就必须把当前帧丢弃。因此单帧策略的情况下，丢帧了是比较高的。

### 多帧追赶

当前端新来一帧图像的时候，写指针需要移动。如果写指针累加之后不等于读指针，那么就正常写，如果写指针累加之后等于读指针，且读模块正在读，那么写指针=读指针+1，即写指针跳过了读指针。

当读模块要读一帧数据的时候，读指针累加。如果读指针累加之后等于写指针，读DDR3 SDRAM数据的时候，读地址要小于写地址。其他情况下，均可以正常读。

需要注意的是，目前的设计支持奇数帧的操作，当指针到达帧存底部的时候，需要返回0号指针。

比如帧缓存深度为4，从10号指针开始，写模块依次写入 a b c d四帧数据。读指针长时间驻留在a帧位置，当写模块准备写e帧的时候，10号帧缓存被占用，写指针越过读指针，将e帧写入11号指针。



当写指针在11号帧缓存时，读模块读完了10号帧缓存，此时读指针可以进入写指针所在的11号帧缓存。



如下图所示，读写指针都指向了11号指针，此时写指针正在将e帧数据写入其中。因此11号指针的前半部分是e帧数据，后半部分是b帧数据。为了将e帧数据完整的读出来，读地址要始终小于写地址。否则，就会把已经废弃的b帧也读出来。



注：奇数帧(3帧)的地址跳跃方法，以写指针为例

当写指针达到帧存末尾时(wr\_ptr=2)，如果读指针=0，则写指针=1，否则写指针=0。

当读指针到达帧存末尾时(rd\_ptr=2)，如果写指针+1=读指针，则写指针=0，否则写指针累加1。

## 顶层参数

待补充

## 顶层接口

待补充

## 性能描述

从以下几个方面描述：

1. 多ROI的数量
2. Sensor行宽
3. 图像分辨率大小
4. 前端带宽大小
5. 瞬时带宽大小

## 前端接口要求

## 后端接口要求

## DDR适应性描述

## 其他问题

### Ucf约束

### 校正输入阻抗

# 模块测试项（Module Testings）

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）