Zero suppression readout

2019年4月26日

- PAD列表中没有CLK PAD
- 关于PE接口与复位需求的改变
- FIFO空满标志位问题的修改与仿真

PAD 列表中没有CLK PAD

信号名称	PAD名称	PAD编号
ValidOut[3:0]	VALID_OUT<3:0>	6---9
rst_n	DIG_RO_RST_x	29
Inquiry[1:0]	INQUIRY<1:0>	33---34
Select[1:0]	BLK_SELECT<1:0>	31---32
ReadEn	FIFO_READEN	30
DataOut[7:0]	DATA_OUT<7:0>	16-19, 22-25
CLK	?????	

上表格中，第一列是阵列外围数字模块的对外端口在设计中的名称；
第二列是阵列外围数字模块的对外端口对应PAD列表中PAD名称；
第三列是阵列外围数字模块的对外端口对应PAD列表中PAD编号

PAD文档列表中没有发现CLK PAD，文档中PAD列表只是部分PAD的列表吗？还是CLK是芯片内部PLL产生？
若CLK不是内部PLL产生并且PAD列表是全部PAD，可以将ReadEn一直置高，去掉IFO_READEN PAD; 然后增加一个CLK PAD作为CLK时钟输入。

关于PE接口与复位需求的改变

- **PE接口改变：64bit---> 48bit**

目前解决办法：

1.将原来由8段(每一段8个像素)组成的64bit BK中每一段8个像素中最高2位直接写零(每一段即是00XX_XXXX，X为像素Hit)。

2. 直接去掉这8段中的2个段。（效率高）

初步的Layout规划

- PeripheralReadOut（图中阵列下方）的宽度需要与像素阵列的宽度相等（或者略微超出也可以）
- Pitch与像素阵列的pitch匹配
- 存在的问题：
  - PeripheralReadOut是按照每个sector有64个column设计的，现在每个sector只有48个column
  - 能否减少为48*4=192个输入端口？

- **PE复位改变：下降沿复位---> 高电平复位**

在PE最初设计时采用下降沿复位，而Cache_Bank是异步RESET(高电平有效)。因此将PE复位改成高电平复位，详细仿真结果见PPT第5页~第6页。

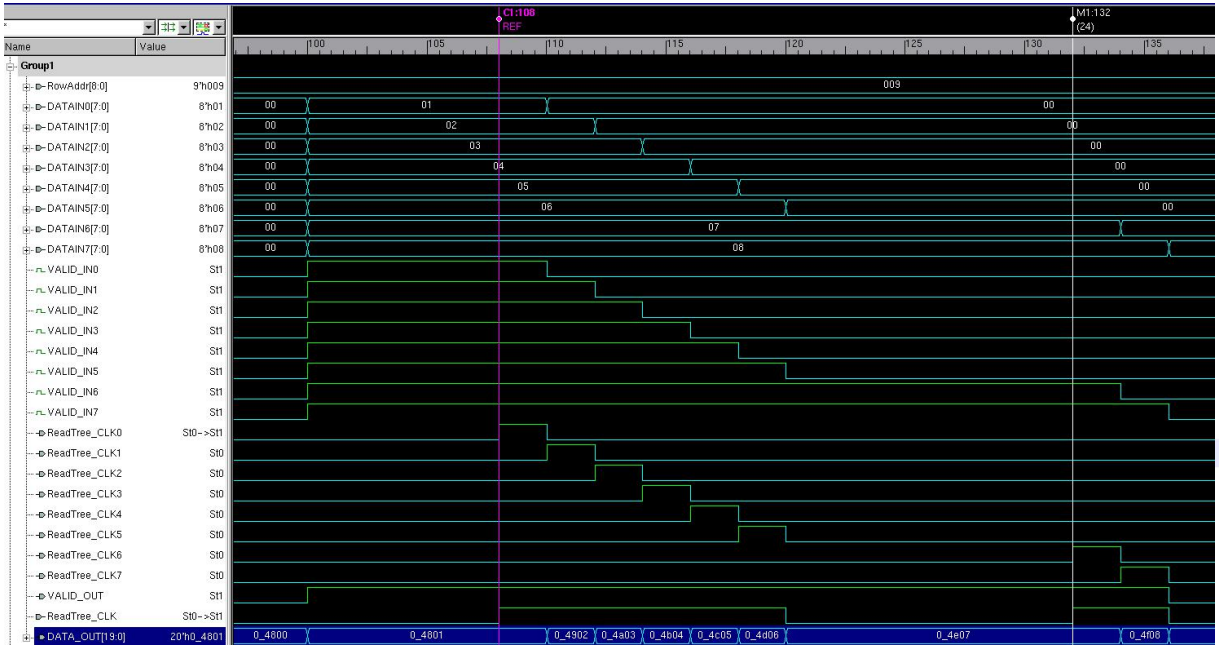
关于PE复位需求的改变

将TestBench中Cache_Bank模型下降沿复位改成高电平复位时，仿真原有设计发现**一个时钟内复位多个像素，**仿真结果如左下图所示。

仿真方法如下：高电平时对像素进行复位，2ns(2ns是人为随机给定)后像素单元复位完成，复位仿真代码如右下图所示。

一个时钟内复位多个像素是由于PE复位电平为高时，经过2ns PE完成当前像素复位，复位完成之后PE就开始新的优先级。此时PE复位电平还是为高，PE将直接对下一像素进行复位。**这将导致数据丢失**。

因此，需要对PE进行修改。

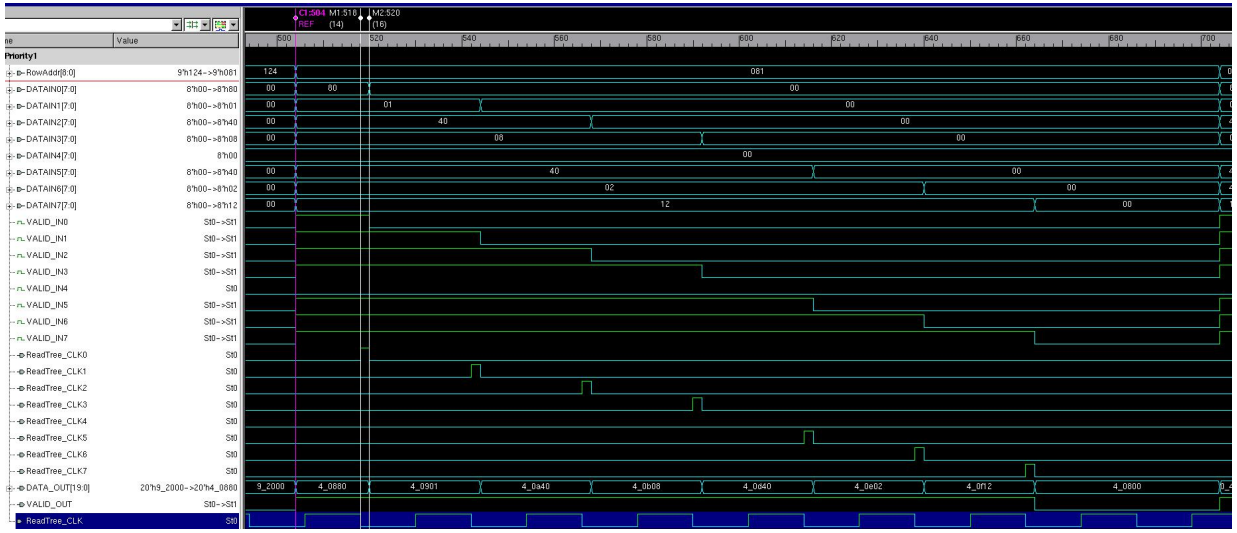



```
fork:thread
    wait(ReadTree_CLK0==1); DATAIN0 =#2 0;
    wait(ReadTree_CLK1==1); DATAIN1 =#2 0;
    wait(ReadTree_CLK2==1); DATAIN2 =#2 0;
    wait(ReadTree_CLK3==1); DATAIN3 =#2 0;
    wait(ReadTree_CLK4==1); DATAIN4 =#2 0;
    wait(ReadTree_CLK5==1); DATAIN5 =#2 0;
    wait(ReadTree_CLK6==1); DATAIN6 =#2 0;
    wait(ReadTree_CLK7==1); DATAIN7 =#2 0;
    #192; disable thread;
join
```

PE修改方法

为避免一个时钟内复位多个像素的问题，**在PE中增加辅助电路**。该辅助电路功能是：PE时钟下降沿到来时，将像素复位信号置高，该像素复位结束同时将复位信号置低。仿真结果如下图所示，仿真方法与上一页PPT中相同。

如下图所示，8段中像素同时被击中，VALID_OUT置高后第一个下降到来时将ReadTree_CLK0(PE复位接口)置高，经过2ns时间完成第一段像素的复位，与此同时将ReadTree_CLK0信号进行置低。之后才开始新的优先级，此时新的优先级直到下一周期的时钟下降沿时刻才能完成复位。因此每个时钟仅复位一段像素。



FIFO空满标志位问题的修改与仿真

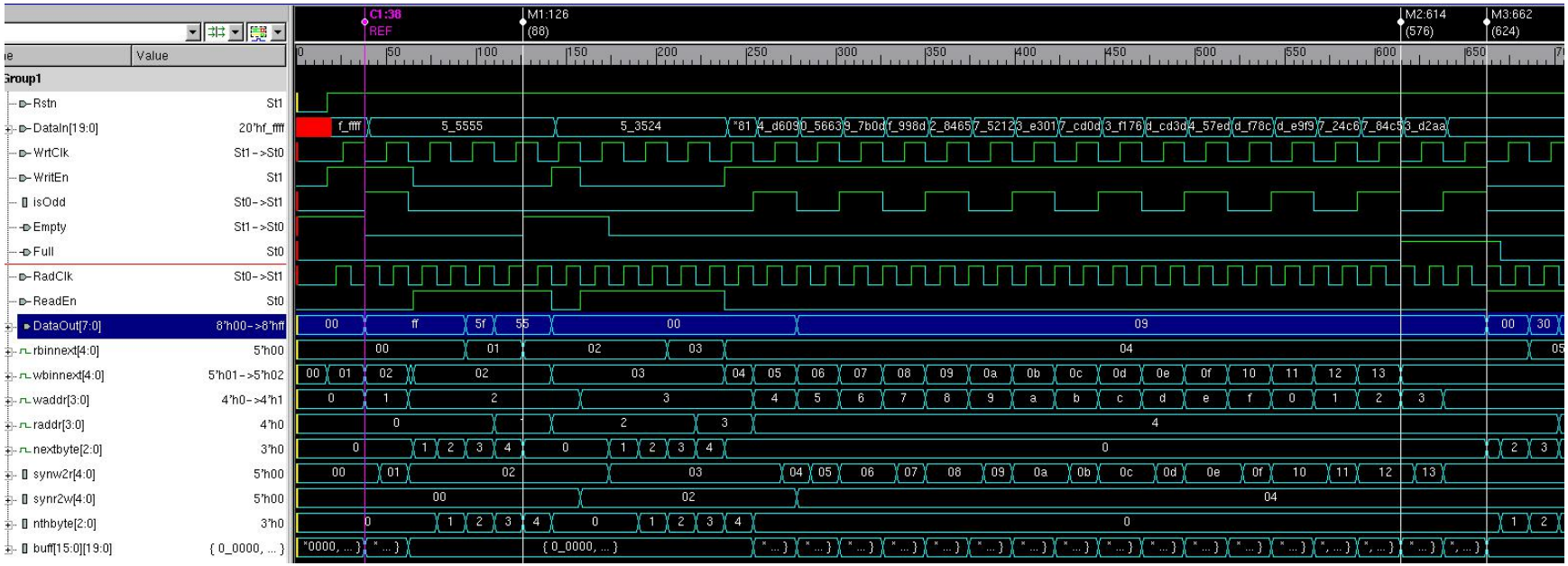
FIFO设计修改主要有：

1、为避免出现奇数个20bit数据，采用寄存器判断，若为奇数，则紧随其后写入FIFO中一个20bit零数据；若为偶数，则不用处理。

2、为避免Full的assert和Empty的de-assert还需要一个时钟周期的问题，在FIFO中采用的处理方法是：

- 在FIFO写使能有效之后第一个写时钟下降沿Empty de-assert;
- 在FIFO写使能有效时将下一写地址与从读时钟域同步到写时钟域的读地址(synr2w)相比较，若写地址与synr2w最高位不同则Full assert

FIFO空满标志位问题的修改与仿真



FIFO仿真：

1. 电路复位17ns,复位期间读写使能置低，复位结束之后，先写入2 * 20bit数据(F_FFFF和5_5555)，读出5 * 8byte数据（ff->ff->5f->55->55），再写入一个周期数据5_3524，此时会插入一个0_0000,读出5 * 8byte数据(24->35->05->00->00)；之后写入18个数据，再读出8个周期。随后FIFO同时进行读写操作。
2. 通过仿真结果可见，Empty信号会在写入数据之后立即拉低，Full信号会在FIFO下一写指针为最后一个可以写入地址时置高