# 系统设计报告

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档版本** | **修订内容** | **修订日期** | **设计者** | **审核人** |
| **V1.0** | **创建文档** | **2024.4.28** | **赵泽煊** |  |
|  | **添加内容** | **2024.5.7** | **赵泽煊** |  |

# 系统设计报告

## 概述

## 设计指标

1. **芯片管脚**
2. **功能描述**

4.1超级像素

4.1.1单像素

表4-1 单像素内部信号

|  |  |
| --- | --- |
| **信号名称** | **描述** |
| clk\_gating\_single\_pixel\_40MHz | 超级像素管理逻辑传给单像素的40MHz门控时钟 |
| clk\_gating\_single\_pixel\_640MHz | VCO给单像素的640M时钟 |
| hit\_pixel | 经过两次同步后的hit信号 |
| out\_flag | 超级像素管理逻辑传给单像素的复位信号 |
| shutter | 光子计数模式开启标志 |
| TimeStamp | 时间戳 |
| hit\_pixel\_edge | hit\_pixel信号上升沿标志 |
| hit\_or | FTOA计数使能信号 |
| hit\_over | 为高表示击中过程结束 |
| ToT\_data | 过阈值时间 |
| timestamp\_hit | TOA，到达时间 |
| FTOA | 细量化计时 |

每个超级像素由8个单像素和1个超级像素管理逻辑组成，单像素中内置9bitTOA寄存器、8bitToT寄存器和5bitFTOA寄存器。在粒子径迹检测模式下，当单像素收到超级像素管理逻辑传来的40MHz门控时钟后在每个时钟上升沿进行判断，如果检测到hit\_pixel上升沿，TOA寄存器记录此刻TimeStamp，ToT寄存器在每个时钟上升沿加1，而FTOA检测VCO传到单像素的640MHz时钟，在时钟上升沿若hit\_or为高则寄存器加1。在光子计数模式下，PC为6bit，iToT为14bit，由于使用了寄存器重用，TOA的第0位和第1位、FTOA的0~3位和ToT的8位构成14bit的iToT寄存器，TOA的2~7位构成6bit的PC，FTOA的最高位和TOA最高位未使用。为了实现全帧读出的功能，在shutter下降沿会强制开启一个时钟周期的40MHz门控时钟，使每个单像素中都有至少一个周期的能量信息，在后续处理时需要把PC数为0的数据去掉，剩下的每个iToT减一即可。

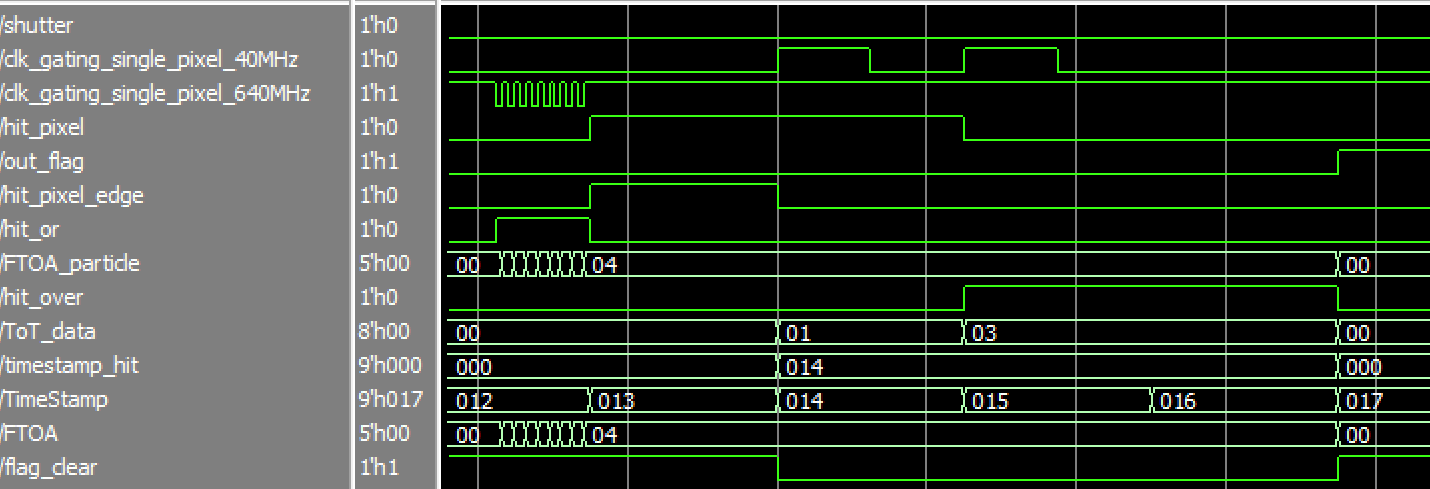


图4-1 粒子径迹检测时序图

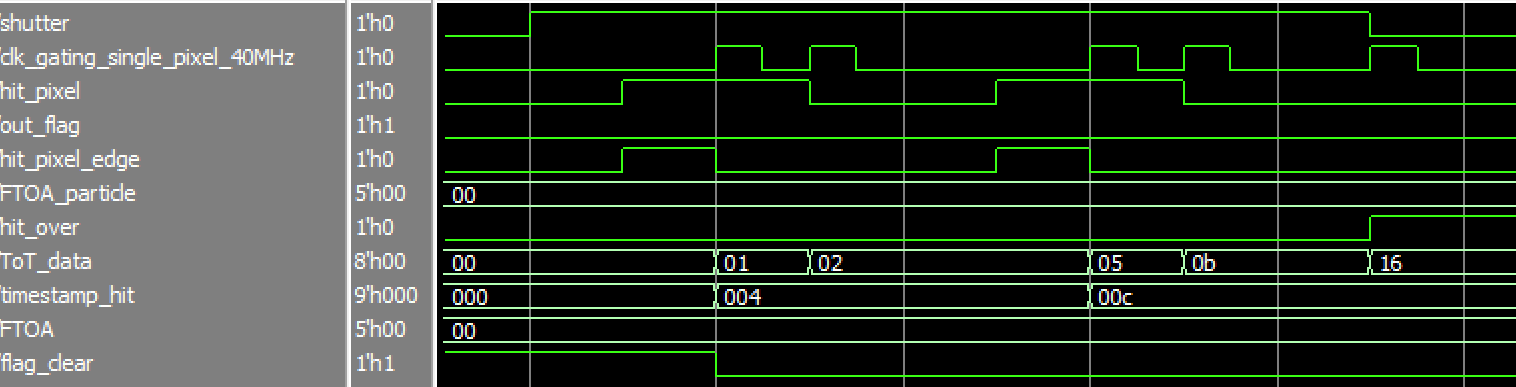


图4-2 光子计数时序图

4.1.2超级像素管理模块

表4-2 超级像素内部信号

|  |  |
| --- | --- |
| **信号名称** | **描述** |
| clk\_40MHz | 40MHz系统时钟 |
| push\_clk | 配置时钟 |
| rst\_n | 系统复位 |
| rst\_n\_pixel | 像素复位（只有像素复位时不清除DAC信息） |
| Dpulse | 从pad引入的Dpulse |
| hit | 模拟前端传来的击中信号 |
| hit\_over | 单像素数据记录结束信号 |
| shutter | 光子计数模式使能 |
| mode | 粒子径迹追踪/光子计数模式切换开关 |
| config\_info | 配置信息，其中包括4位DAC信息，1位Mask和1位Dpulse\_en |
| ToT\_data | 单像素传来的ToT信息 |
| timestamp\_hit | 单像素传来的TOA信息 |
| FTOA | 单像素传来的FTOA信息 |
| addr\_col | 该超级像素在双列中的位置 |
| shake\_hands\_next | 握手协议，由下一级超级像素仲裁器发出 |
| hit\_pixel | 两次同步之后的hit信号 |
| hit\_or\_super\_pixel | 8位单像素hit\_or经过或运算传到VCO的信号 |
| next\_config\_info | 传到下个超级像素的配置信息 |
| arbiter\_data | 仲裁数据 |
| clk\_gating\_single\_pixel\_40MHz | 传到单像素中的40MHz门控 |
| shake\_hands\_last | 发给上一级超级像素的握手信息 |
| out\_flag | 告知单像素复位的信号 |
| shutter\_temp | 经过同步后的shutter信号，传给单像素中 |
| config\_DAC | 单像素的DAC配置信息 |
| hit\_or | 传给单像素的hit\_or |
| hit\_pixel\_edge | hit\_pixel上升沿检测 |

超级像素管理模块与8个单像素相连，超级像素管理模块的主要功能是击中信息处理、击中信号同步和控制、数据存储、超级像素间数据仲裁和DAC配置等。超级像素有两个复位信号，分别是rst\_n和rst\_n\_pixel，rst\_n是电路整体复位，rst\_n\_pixel只清除像素中数据，不清除DAC配置。

表4-3 像素电路工作模式

|  |  |  |
| --- | --- | --- |
| mode | shutter | 工作阶段 |
| 0 | 0 | 粒子径迹检测 |
| 0 | 1 | 非法操作 |
| 1 | 0 | 光子计数帧读出阶段 |
| 1 | 1 | 光子计数数据记录阶段 |

由Mask和Dpulse\_en来对电路功能进行测试，关闭坏掉的像素点。每个单像素有六位配置信息，其中第0位是Dpulse\_en，第1位是Mask，当Mask为1时，屏蔽这个像素，当为0时正常工作，当Dpulse\_en为1时，从pad传来的Dpulse会产生击中信号。从模拟部分传来的hit也会产生击中信号。

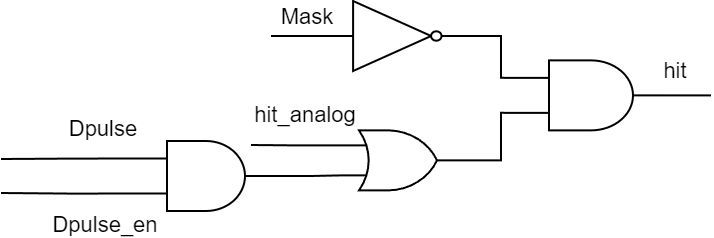


图4-3 击中信号处理

当击中信号到来时，在时钟下降沿先对信号进行同步获得hit\_negedge，再在时钟上升沿时对hit\_negedge信号进行同步获得hit\_pixel，对hit\_pixel再打一拍获得hit\_pixel\_temp，将hit\_pixel和hit\_pixel\_temp进行与操作获得hit\_pixel上升沿信号hit\_pixel\_edge。Hit\_or信号记录从hit到hit\_pixel中间的时间，为单像素的FTOA计时提供判断。将hit\_pixel作为门控时钟使能信号，产生40MHz门控时钟传入单像素中。

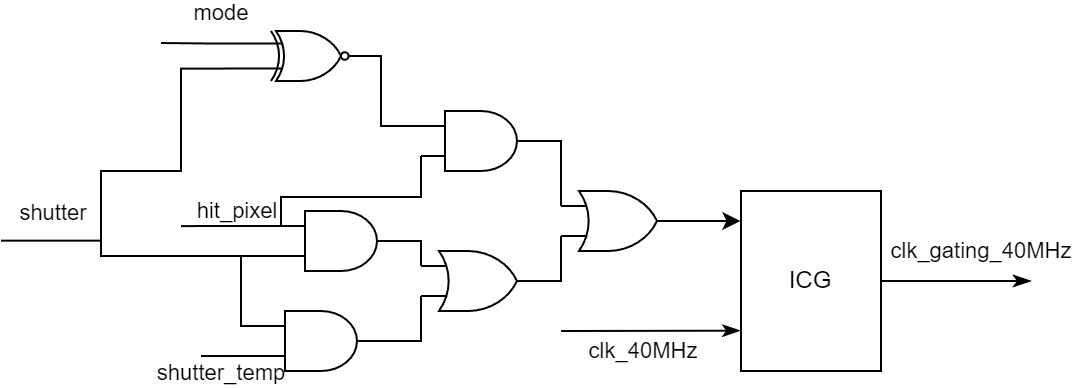
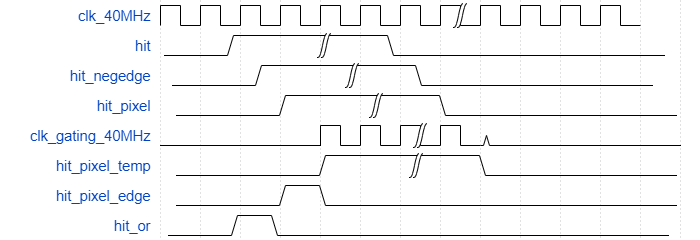


图4-4 门控时钟产生图4-5 击中信号处理、同步与控制时序图

每个超级像素管理模块中有48bit配置寄存器，每个单像素6bit。进行DAC配置时，在每个配置时钟上升沿接受从外面传入的6bit配置信息，并且寄存器左移6位，将配置信息存在第0~5位上。

当单像素获取完数据后hit\_over信号升高，超级像素管理模块通过仲裁确定取走哪个单像素的数据并传给外围读出电路。仲裁的规则是如果首次传输时上个超级像素有数据，那么先传它的，然后传自身的数据，当都有数据时交替传输。当只有上一个超级像素或自身有数据时直接选择进行数据传输。

表4-4 超级像素26bit数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 寄存器位分组 | 25~17 | 16~12 | 11~4 | 3 | 2~0 |
|  | TOA(9bit gray) | FTOA  (5bit LFSR) | TOT  (8bit LFSR) | Addr\_col  (1bit) | Addr\_single\_pixel(3bit) |

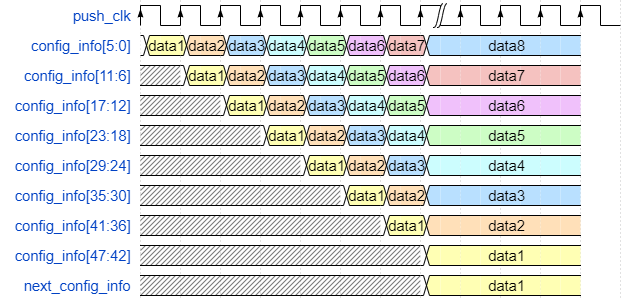


图4-6 DAC配置时序图

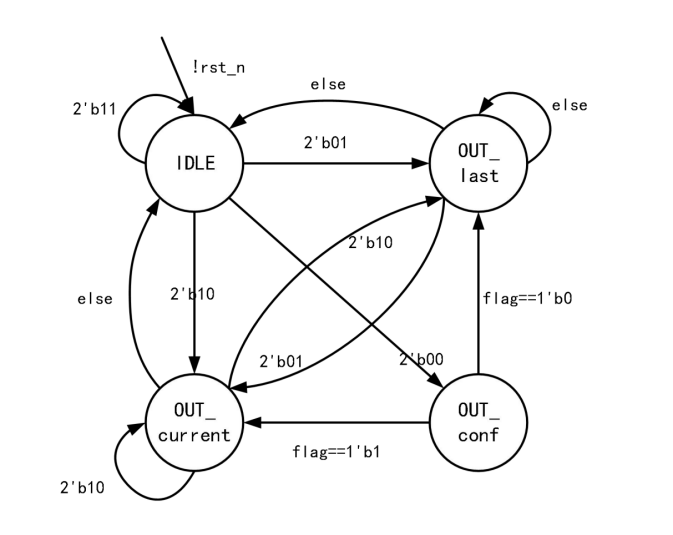


图4-7 超级像素仲裁状态机

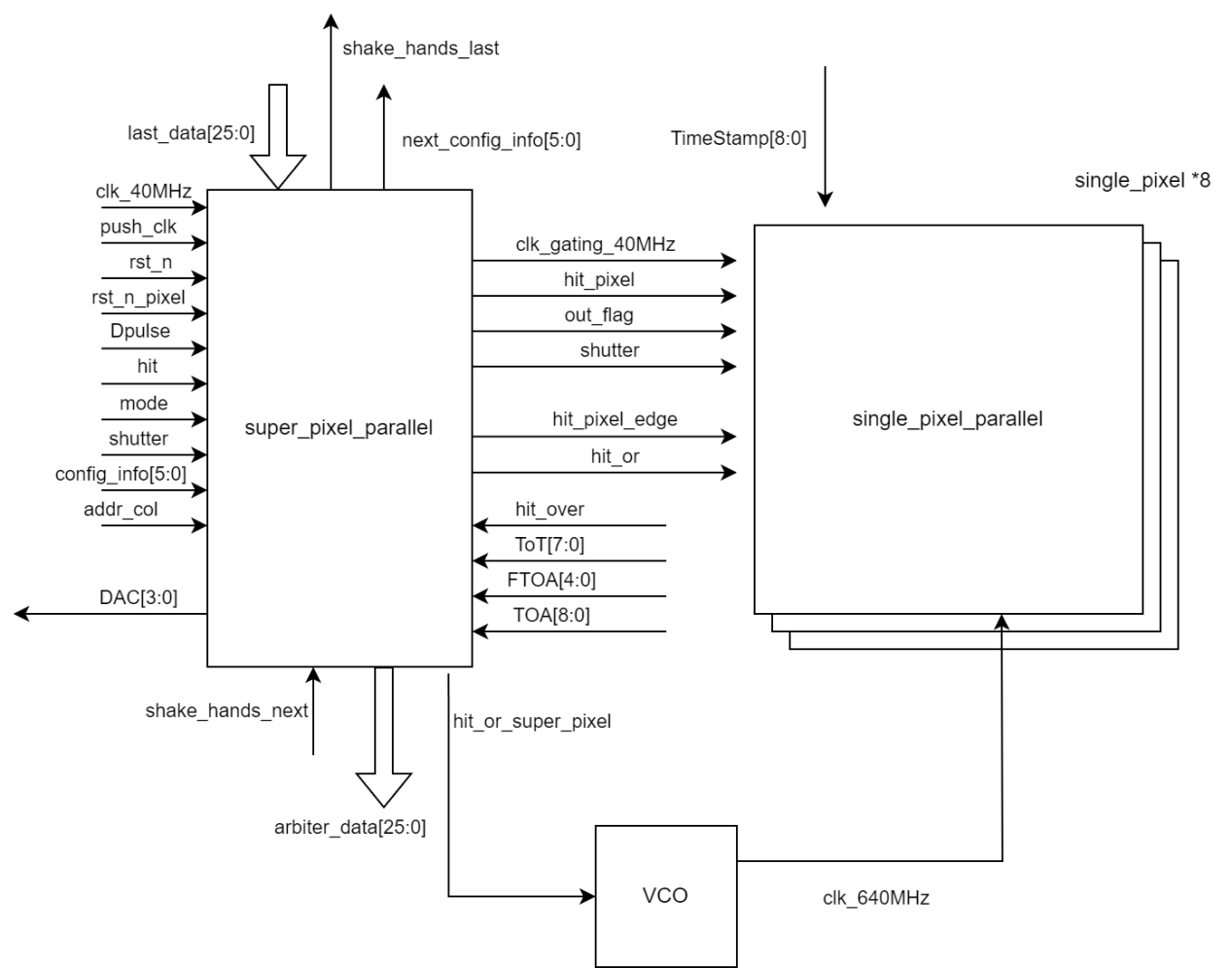


图4-8 超级像素总体框图

4.2外围电路

4.2.1像素阵列拥塞控制电路

一个拥塞缓解控制电路column\_pixel\_peri连接两个双列，其中主要包括三个部分，分别是EOC列尾数据处理模块、FIFO和拥塞缓解处理模块。

拥塞缓解处理模块dealing\_with\_congestion负责对双列中传出来的数据进行拥塞控制，选择其中一列的数据进行传输，并在数据末尾加上一位表示是左双列还是右双列，左为0，右为1。之后数据传到EOC中附加两位地址信息表示是哪两个双列的数据，得到完整的28位数据然后传到FIFO中。EOC还负责为像素阵列提供时间戳计数器、握手信息、配置时钟和FIFO写使能等。我们使用的FIFO是宽28bit，深度为2的同步FIFO，FIFO与二级仲裁进行连接。

二级仲裁data\_route\_merge\_proc连接两个拥塞缓解控制电路，对这两个节点不断进行循环仲裁，选择一个节点并从其FIFO中取出数据进行输出。

data\_route\_merge\_proc输出的最终的28位数据进入PIS中，PIS模块负责将这并行的28位数据转化成串行最终从数字顶层输出，vliad\_out用来表示有效数据，route\_data\_proc用来输出数据。当valid\_out为高时表示数据线包括这一时刻和之后的共28个时钟周期的值就是最终输出的有效数据。

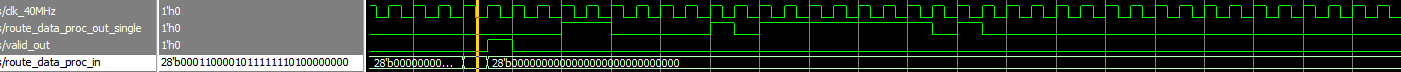


图4-9 PIS模块时序图

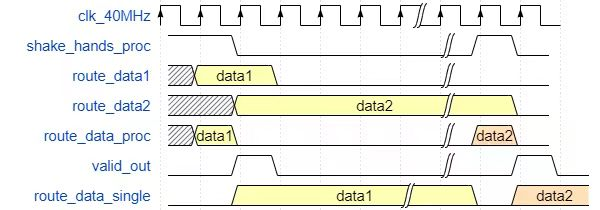


图4-10 循环仲裁与数据输出时序图

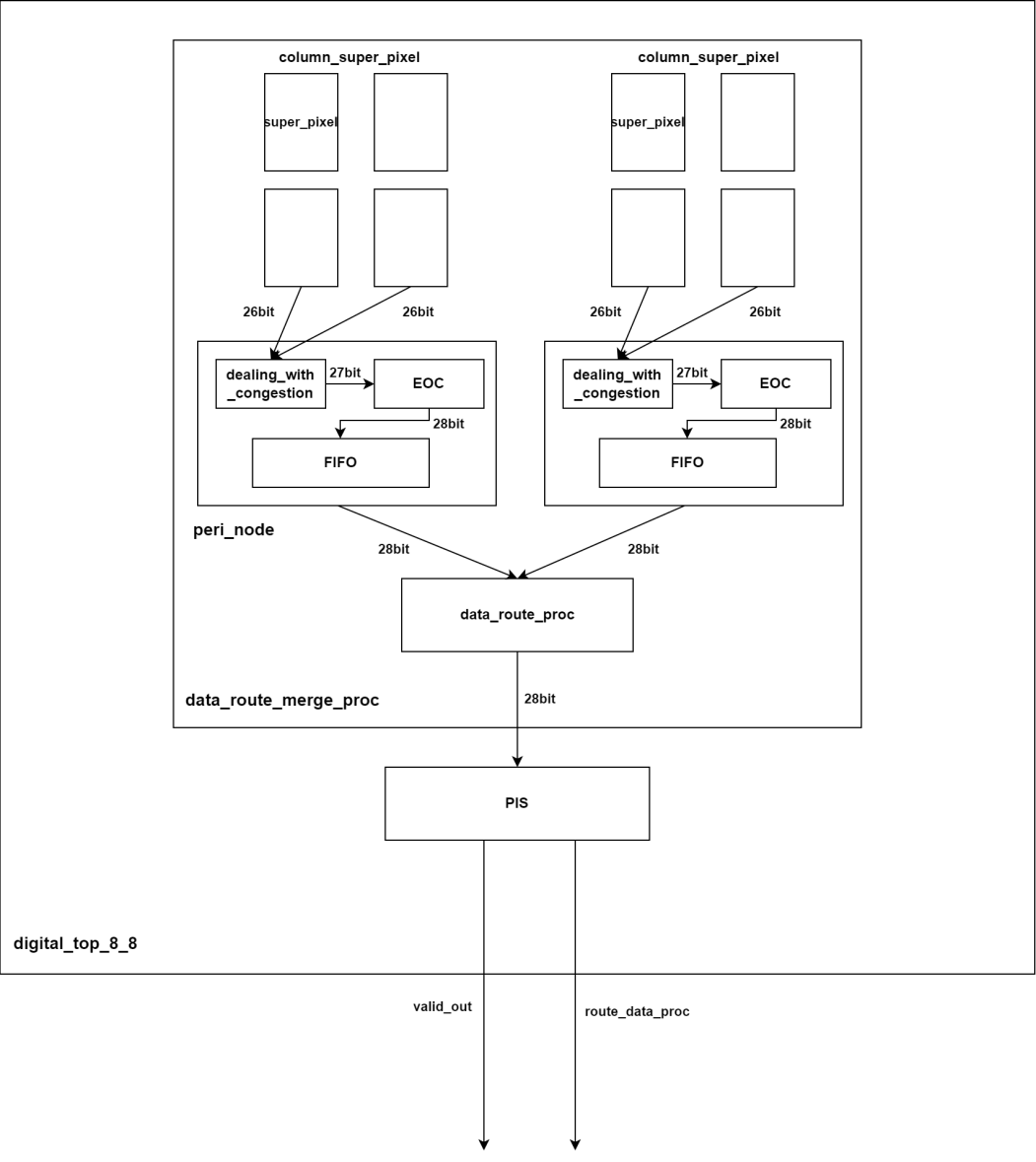


图4-11 数据传输示意图

4.2.2SPI配置电路

从主机输入的数据经过in\_out\_clear模块更新下时序输入到spi\_data\_trans中，该模块包含spi\_interface和ctrl\_register，spi\_interface是用来和主机进行通信的接口，从主机收到的信息经过接口写到ctrl\_register寄存器堆中，然后通过寄存器堆中的信息对电路进行配置。

像素配置有两种方式，一种是外部配置，一种是SPI配置。外部配置的能力十分有限，仅作为备用方式，如果想精确配置还是需要通过SPI进行。

表4-5 寄存器堆配置

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 名称 | 地址 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | | Bit2 | Bit1 | Bit0 |
| Reg0 | 3’d0 | Apulse\_en | shake\_ hands\_col | shutter | mode | rst\_n\_pixel | | 1’b0 | 1’b0 | 1’b0 |
| Reg1 | 3’d1 | 1’b0 | 1’b0 | L\_DAC[3] | L\_DAC[2] | L\_DAC[1] | | L\_DAC[0] | Mask | Dpulse\_en |
| Reg2 | 3’d2 | route\_data\_proc[7:0] | | | | | | | | |
| Reg3 | 3’d3 | route\_data\_proc[15:8] | | | | | | | | |
| Reg4 | 3’d4 | route\_data\_proc[23:16] | | | | | | | | |
| Reg5 | 3’d5 | 4’d0 | | | | | route\_data\_proc[27:24] | | | |
| Reg6 | 3’d6 | 8’d0 | | | | | | | | |
| Reg7 | 3’d7 | 8’d0 | | | | | | | | |

在进行SPI配置时，SPI向001号寄存器写一次配置，双列尾的寄存器就会左移6位，然后将刚写入的配置放入寄存器中，当写4次后双列寄存器满，向111号寄存器写数据产生push\_clk，4个双列寄存器整体向双列上移一位，重复这个过程直到将所有像素配置完全。

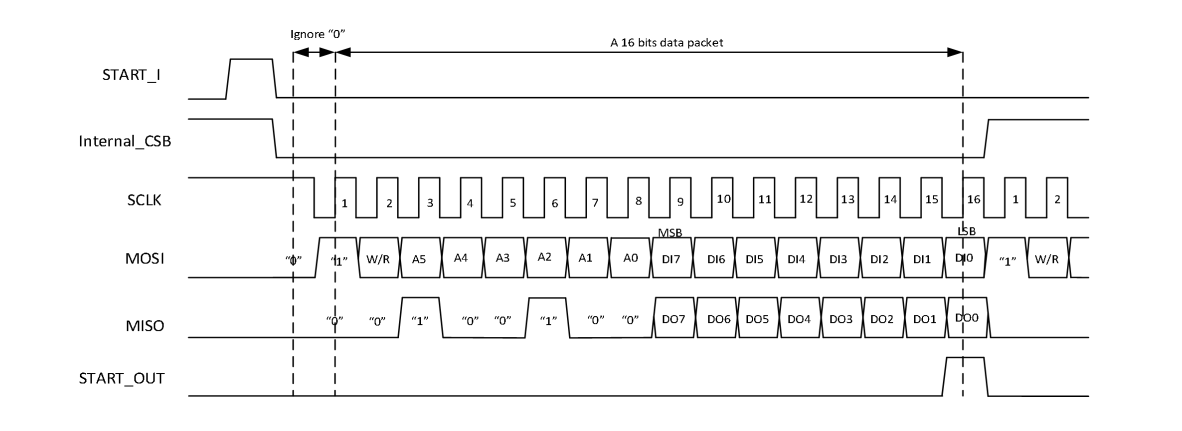


图4-12 SPI数据帧传输时序图

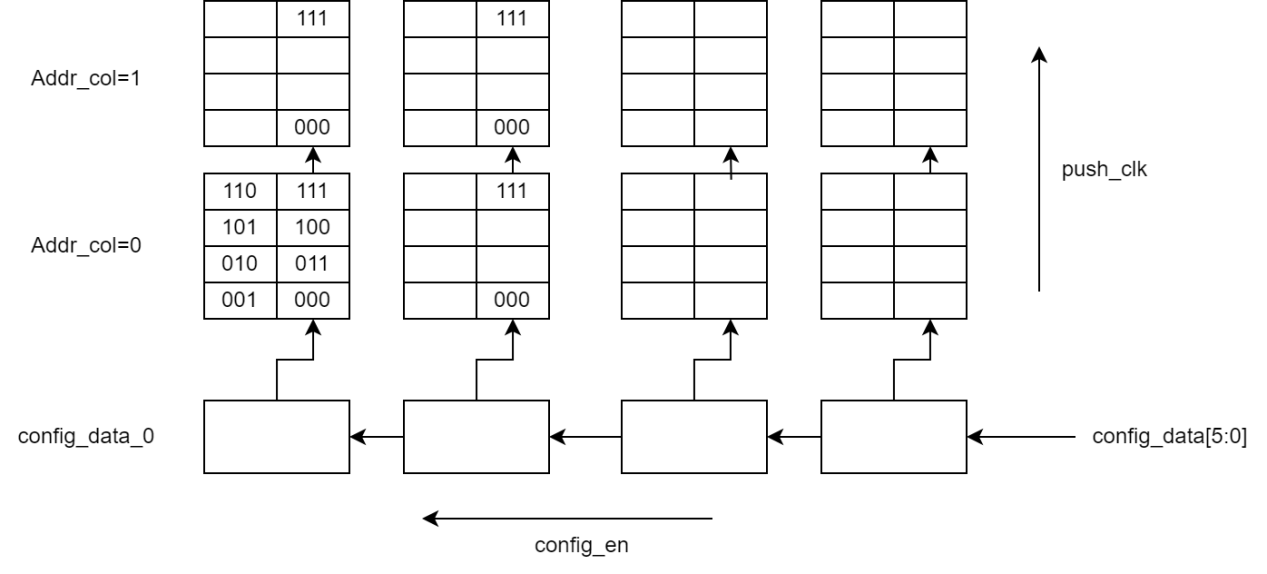


图4-13 SPI配置示意图